Announcements

- Adrian office hours: Tu 2:30-4:30
- Both texts should be on reserve by now
- Short lecture Thurs 11:40-12:30
  - Devoted to project discussion

This Lecture

- RPC Systems
  - [BN97] Ch 3
  - [ACKM04] Sec 2.2
- TP Monitors
  - [BN97] Ch 2,
  - [ACKM04] Sec 2.3
  - [BN97] Ch 9 (2PC)

Types of Middleware

- RPC-based systems
- TP Monitors
- Object Brokers
- Object Monitors
- Message-Oriented Middleware (MOM)
- Message Brokers

RPC Systems

- The foundation for most of the others
- We'll cover in some depth
- Requires support infrastructure
  - Interace Definition Language (IDL) compilers, stub generators
  - directory services for binding
  - etc.

TP Monitors

- Most established form of middleware
- TP-Lite
  - A 2-Tier architecture
  - Allow application logic as stored procedures in a database
- TP-Heavy
  - 3-Tier architecture
  - Implements *transactional RPC*
  - Coordinator for distributed transactions

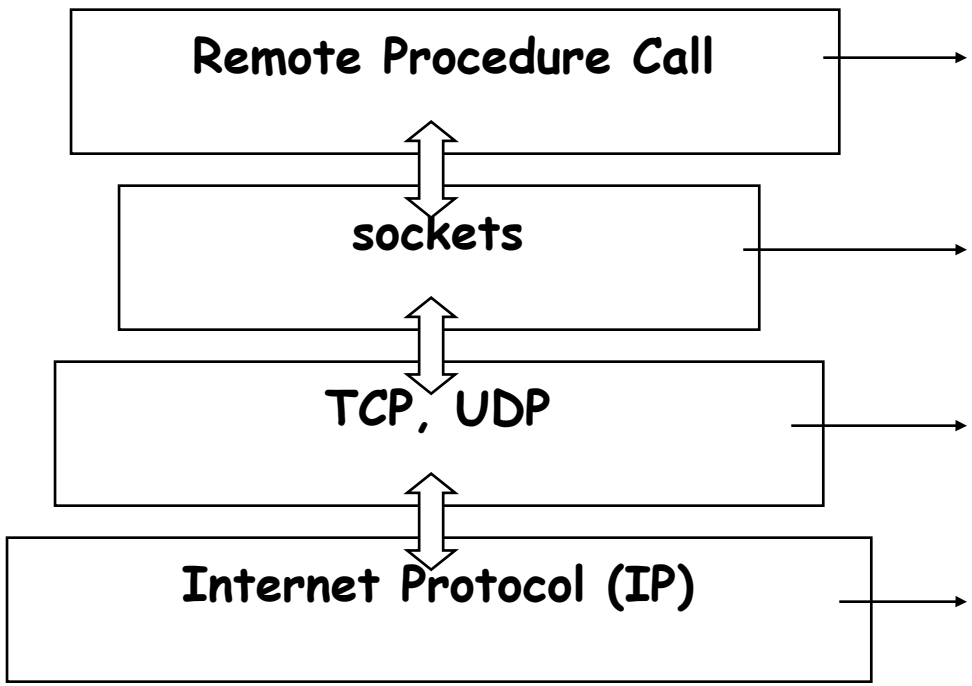
Objects

- Object Broker (OMG CORBA)
  - Like an object-oriented RPC system
- Object Monitor
  - Like an object-oriented TP Monitor

Queues

- Message-Oriented Middleware
  - Queues (and transactional queues) to support asynchronous messaging
- Message Broker
  - All of the above
  - Ability to run application logic for message routing

RPC as Layer of Abstraction



Remote Procedure Call:
hides communication details behind a procedure call and helps bridge heterogeneous platforms

sockets:
operating system level interface to the underlying communication protocols

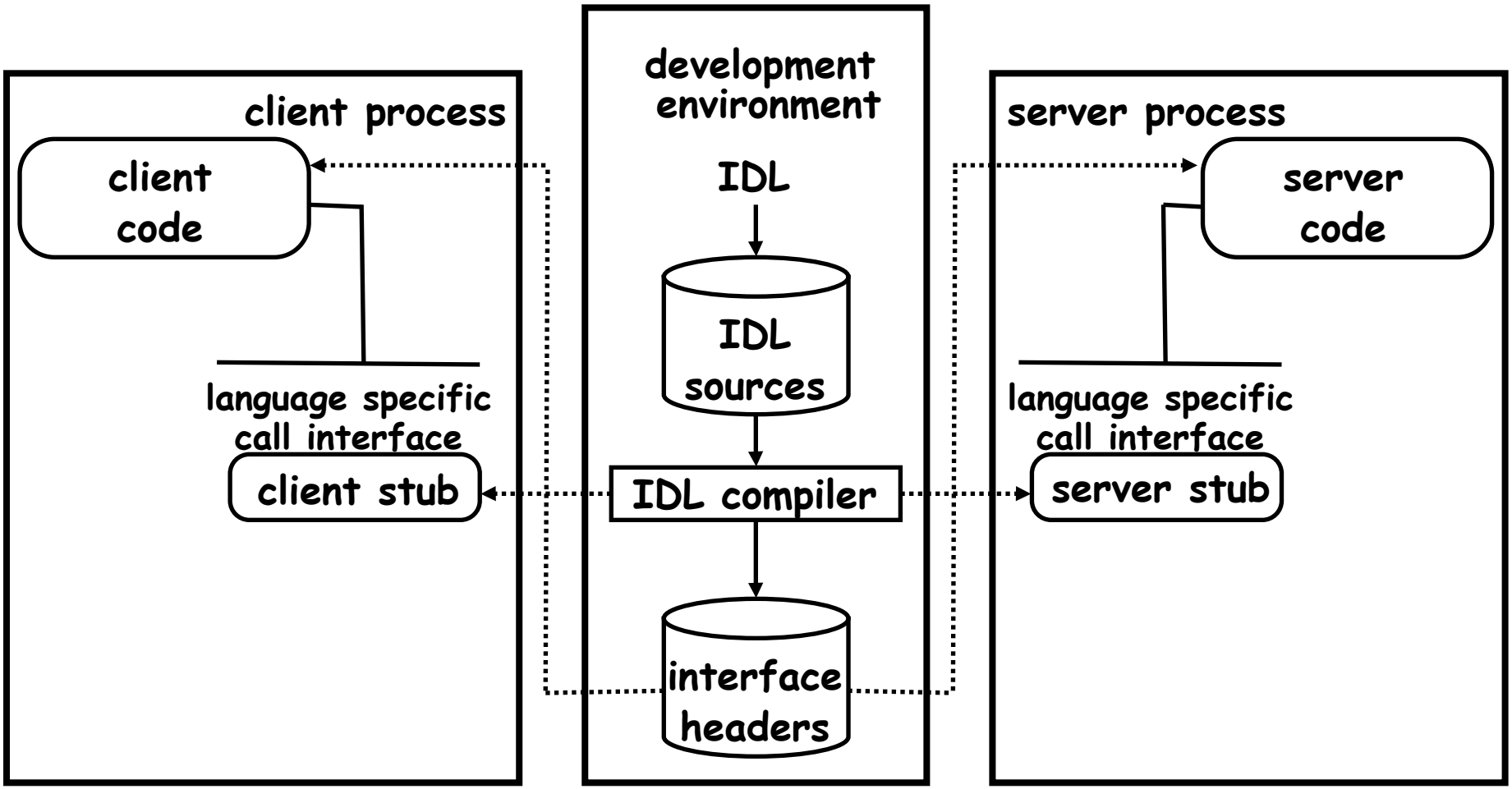
TCP, UDP:
User Datagram Protocol (UDP) transports data packets without guarantees
Transmission Control Protocol (TCP) verifies correct delivery of data streams

Internet Protocol (IP):
moves a packet of data from one node to another

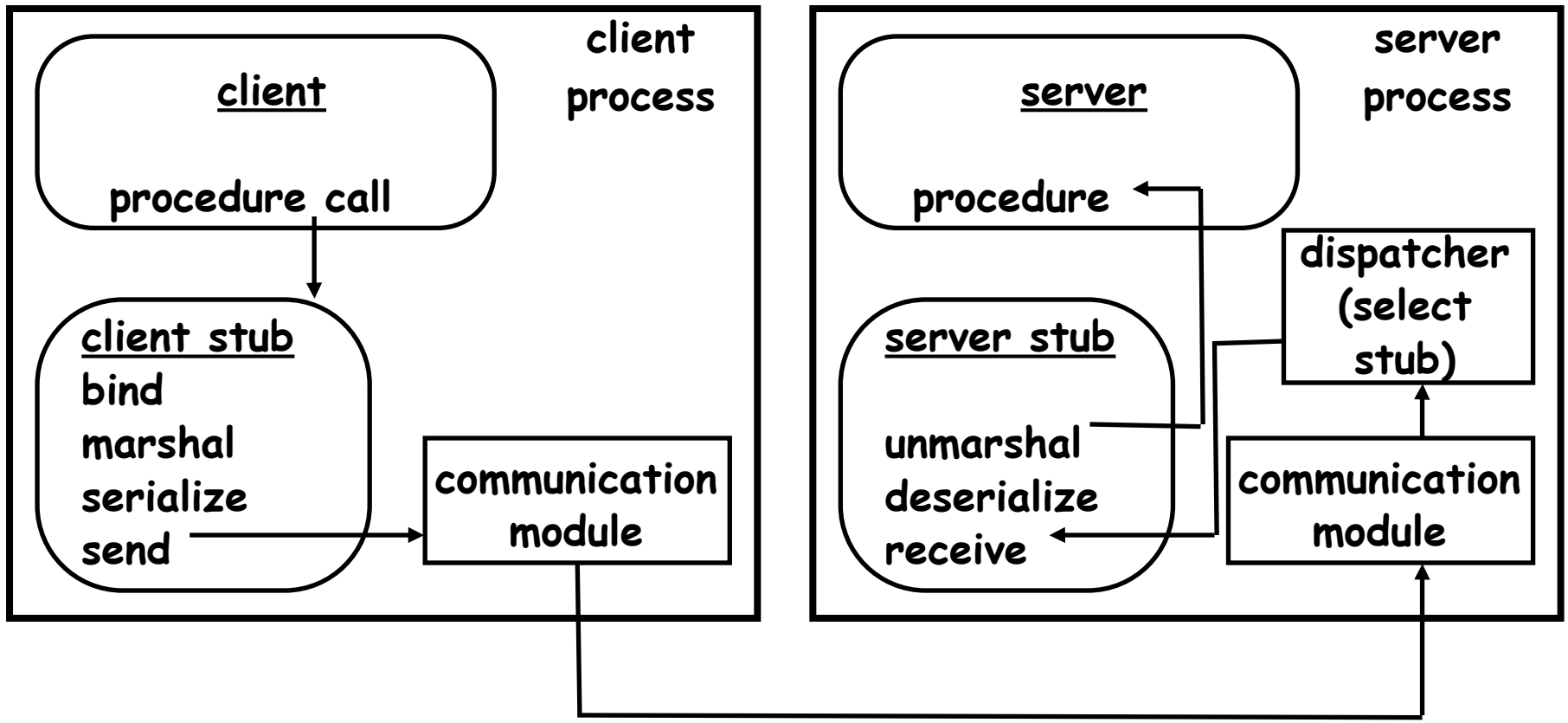
RPC Development

- Interface Description Language (IDL) in which signatures of procedures are described
- IDL Compiler generates caller & callee *stubs*
- Stubs are responsible for marshalling, transmitting and unmarshalling arguments and results

RPC Development



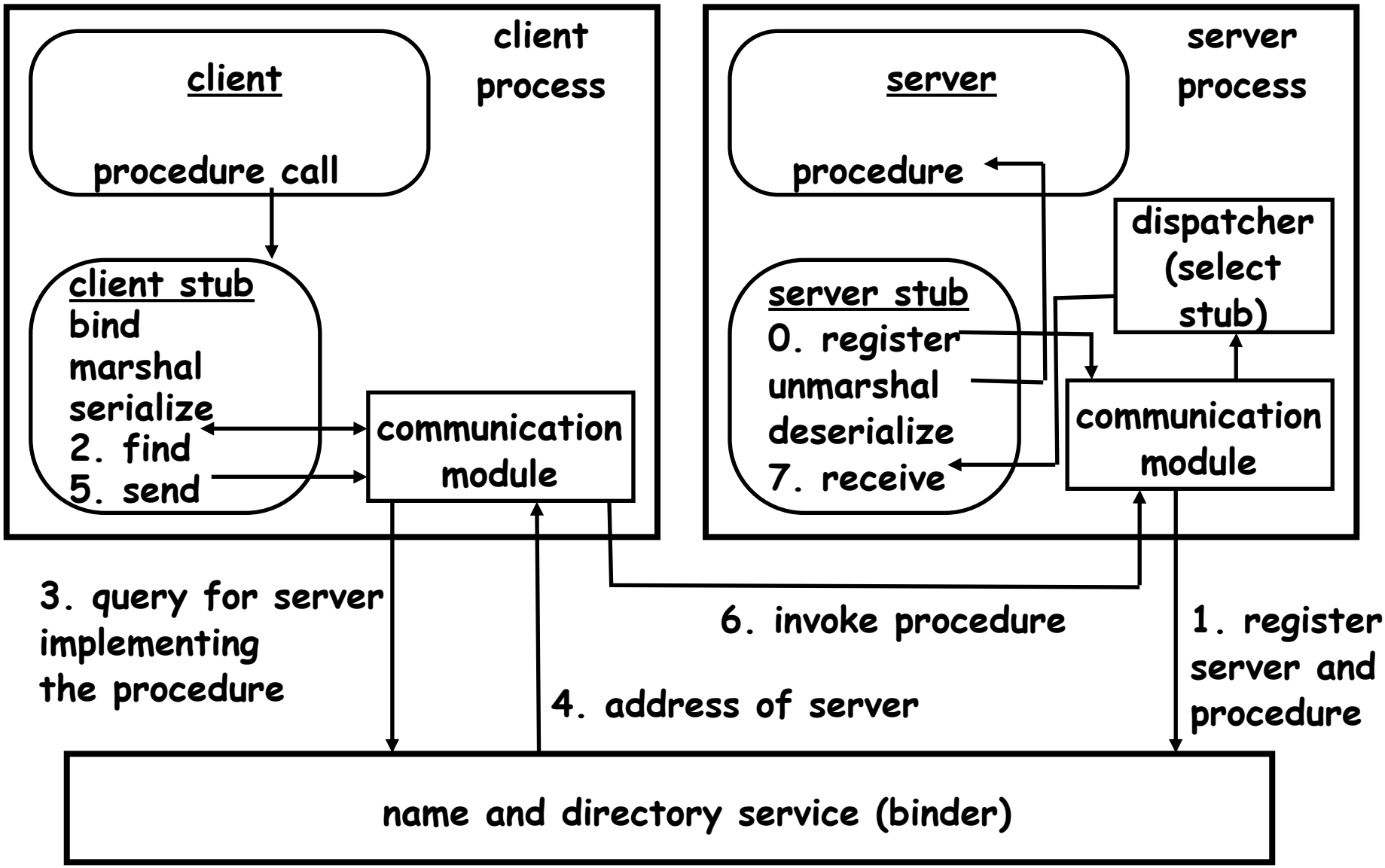
Basic RPC Runtime



Naming & Directory Service

- Callee registers with NDS
- Caller looks up callee by name & signature
- Possibly multiple matches
  - => *traders*
- Possibly multiple server instances
  - => potential for load balancing
- Possibly no active server instances
  - => start one?

Dynamic Binding for RPC



Parameter Translation

- Canonical encoding on wire
  - the n**2 problem
- Receiver-translates
  - best performance if homogeneous

Security

- Advantageous to build authentication into RPC infrastructure
- Discussion deferred until later

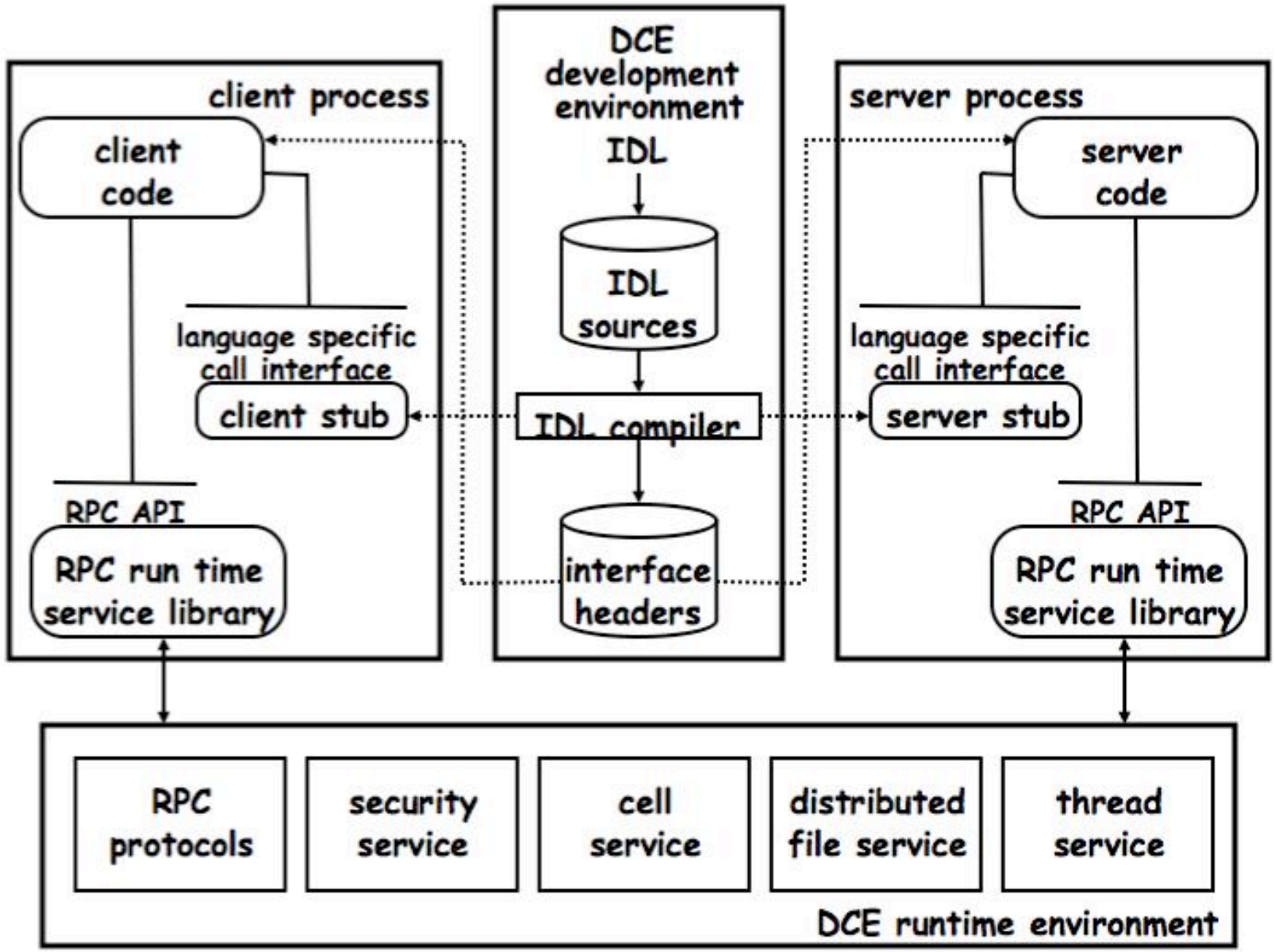
Fault Tolerance

- Retries
  - Reliable transport?
  - If idempotent => seek *at least once*
  - Not idempotent => *at most once*
  - See *Transactional RPC*

RPC Performance

- Procedure invocation overhead 100-1000 times greater for RPC than local call
- *Plus* the communication latency
- 15,000 machine instructions don't take very long these days ...

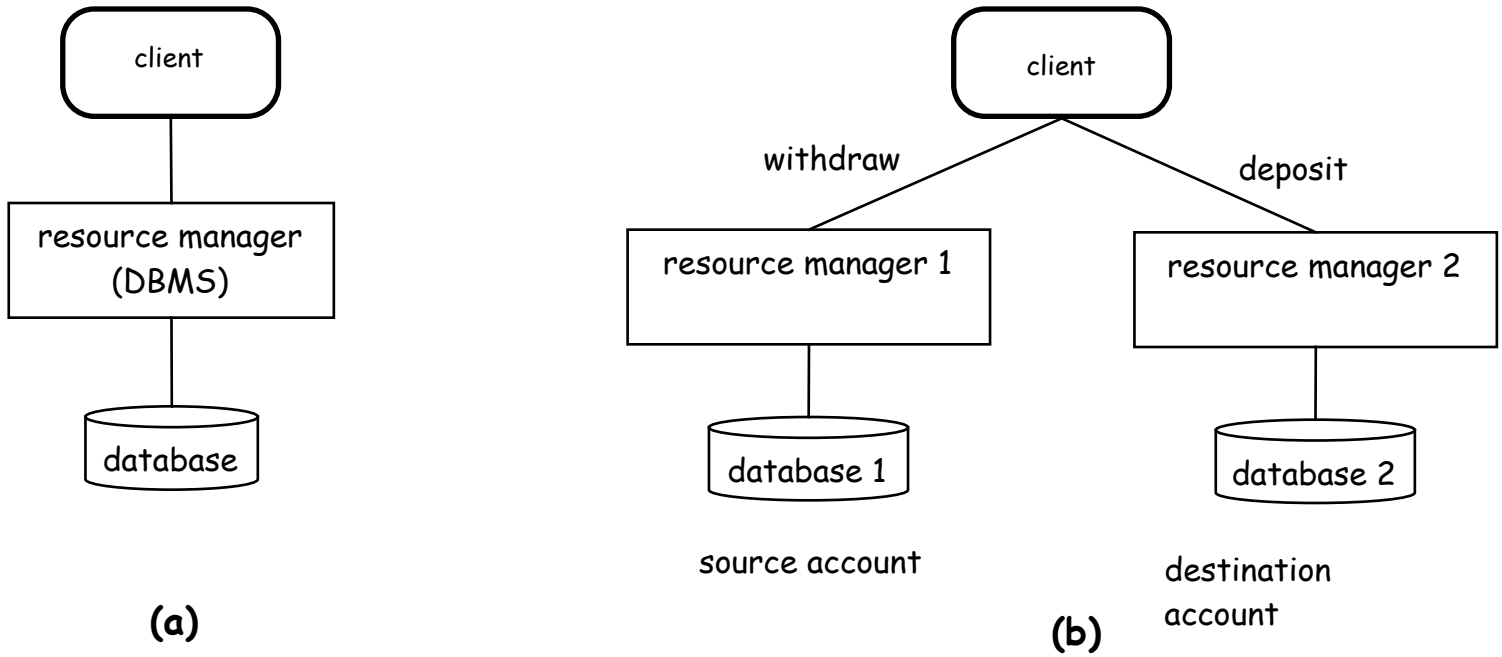
DCE Environment



Transactional RPC

- Suppose RPCs done in application that requires transactional ACID properties
- Distribution makes this difficult ...

Distributed Commit

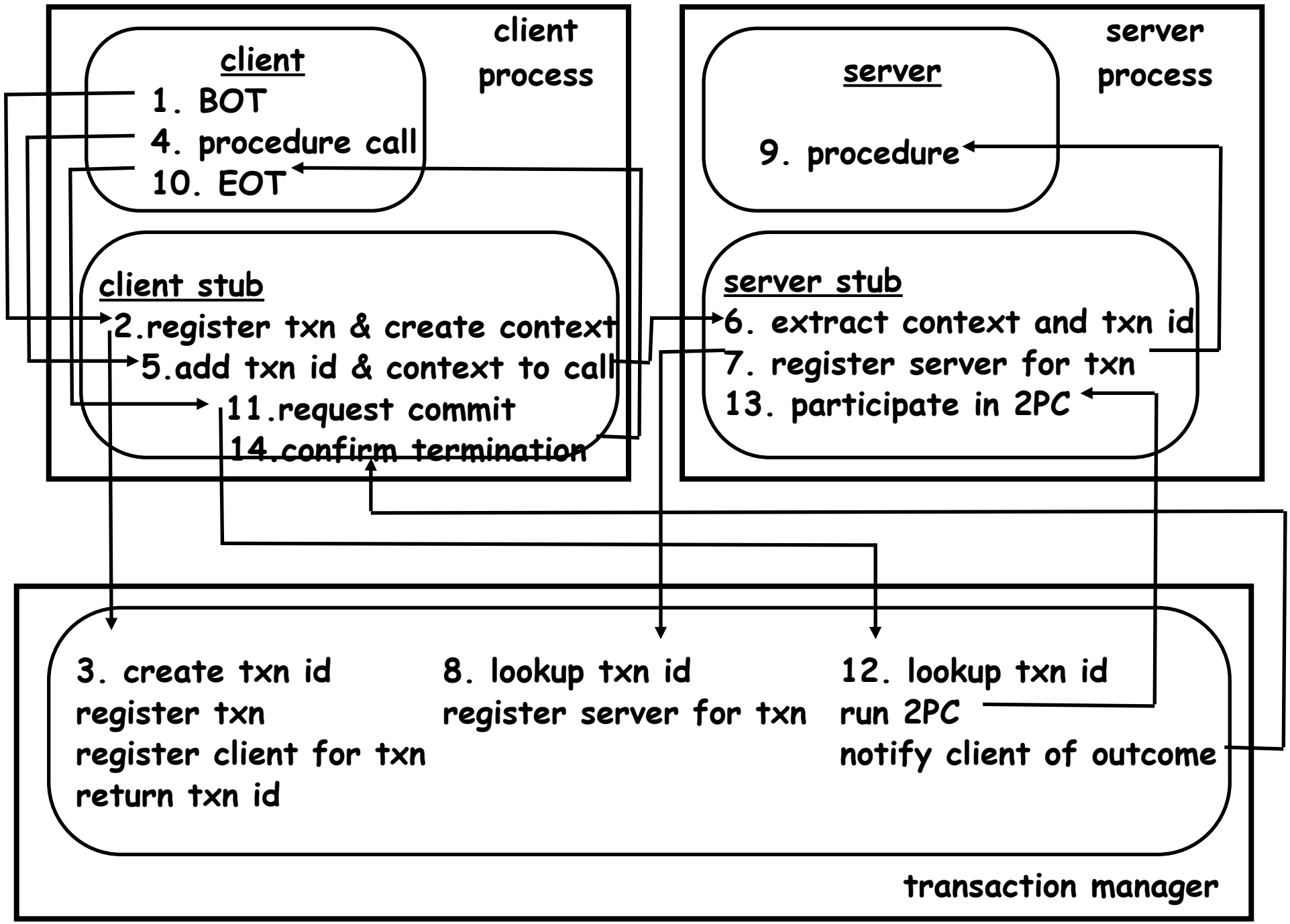


- RM1 and RM2 must *both* commit or *both* abort
- What if one of them fails during commit protocol?
- Failure model is not Byzantine or FailStop but *Crash-Recover*

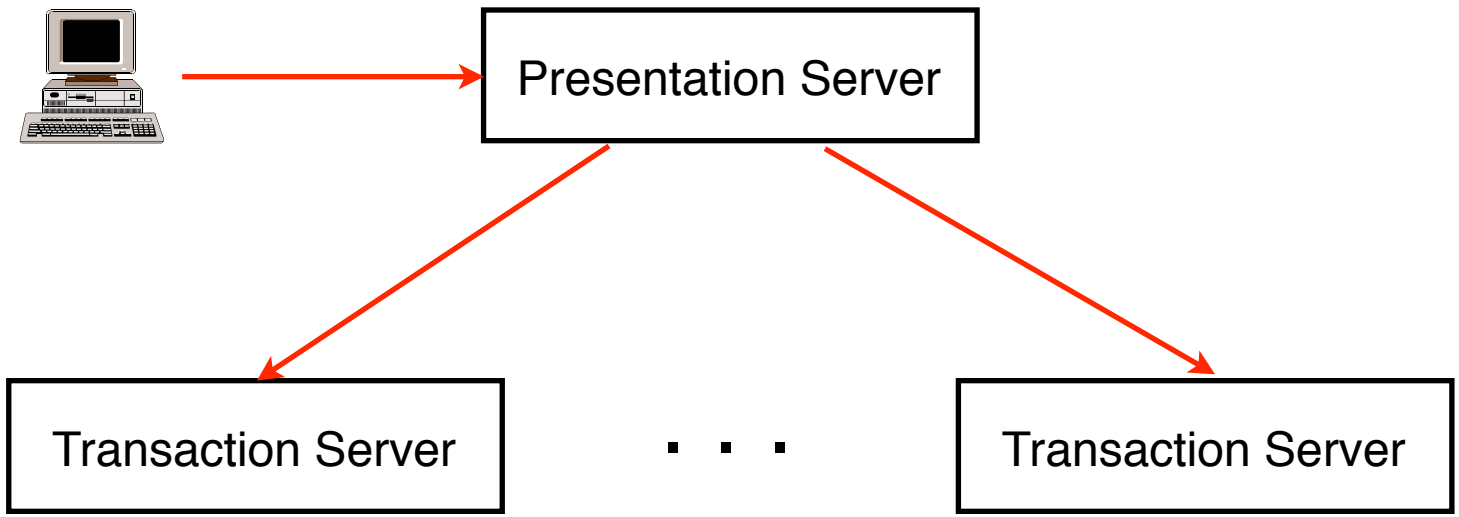
Solution: TP Monitor

- Transaction Manager
  - Multiple RPCs between BOT-EOT calls execute as one transaction
- Coordinator for distributed 2-Phase Commit
  - (Details of 2PC later ...)

Transactional RPC

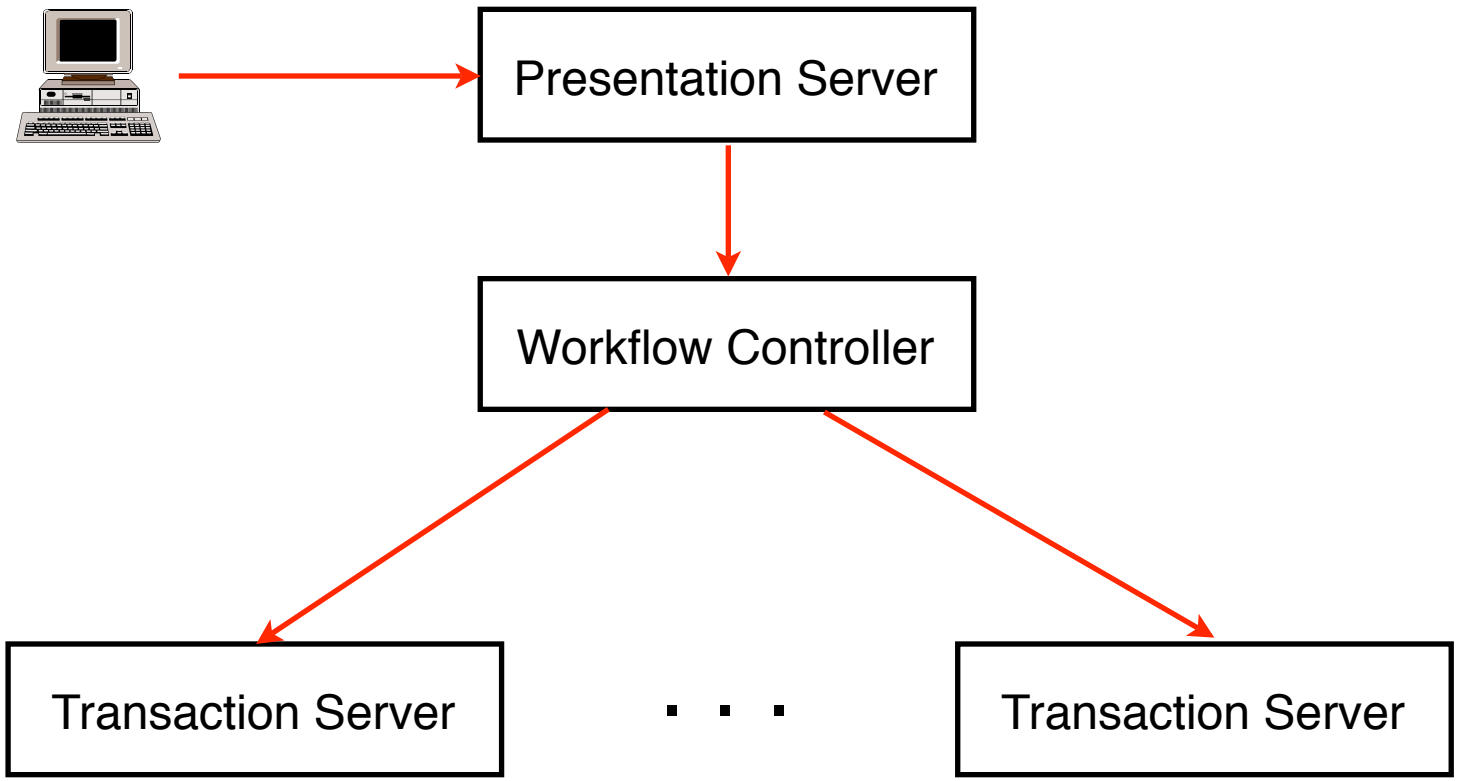


2-Tier TP Monitor Architecture



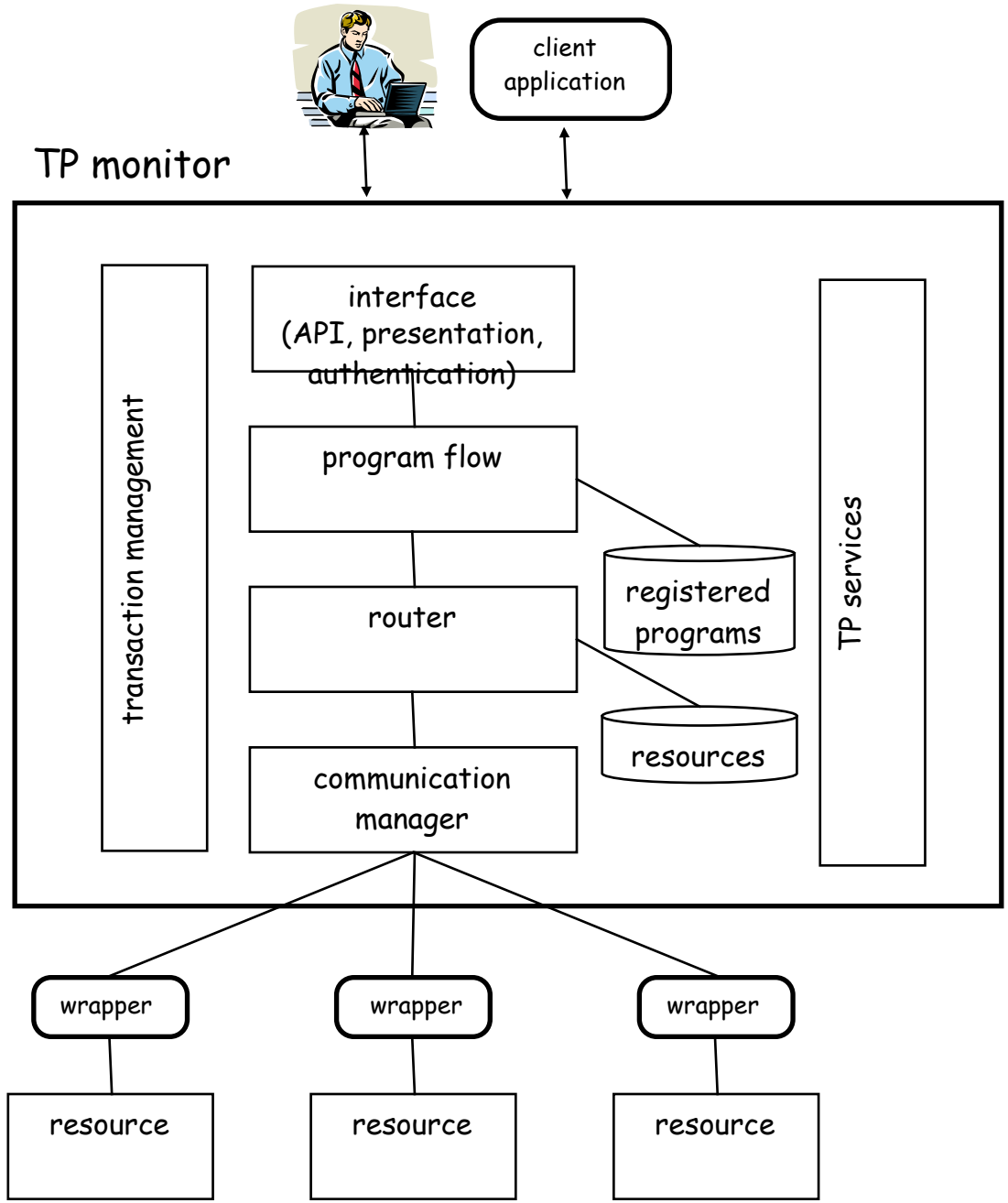
If there is more than one transaction server, the commit coordinator has to be in the upper tier ...

3-Tier TP Monitor Architecture



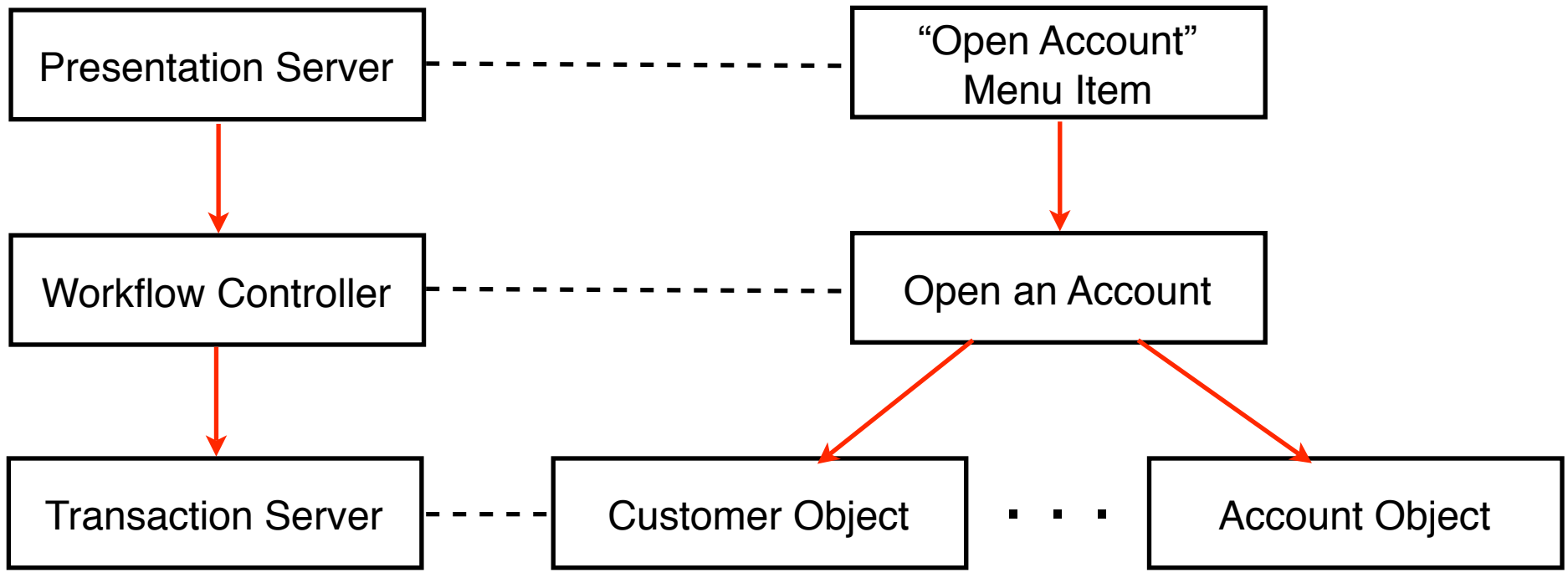
The 3 Tiers match the 3 application layers ...

More Detail ...

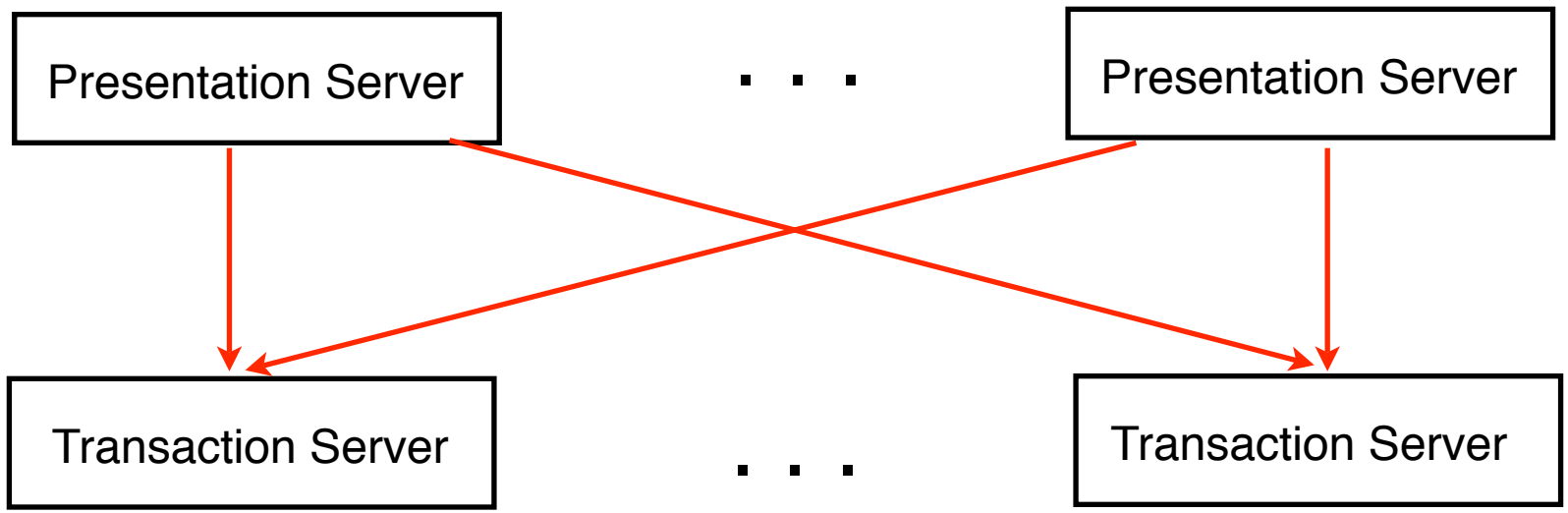

client application
TP monitor
transaction management
interface (API, presentation, authentication)
program flow
router
communication manager
registered programs
resources
TP services
wrapper
wrapper
wrapper
resource
resource
resource

3-Tier TP Monitor ...

- 3-Tier Model Maps to Object-Oriented Application ...

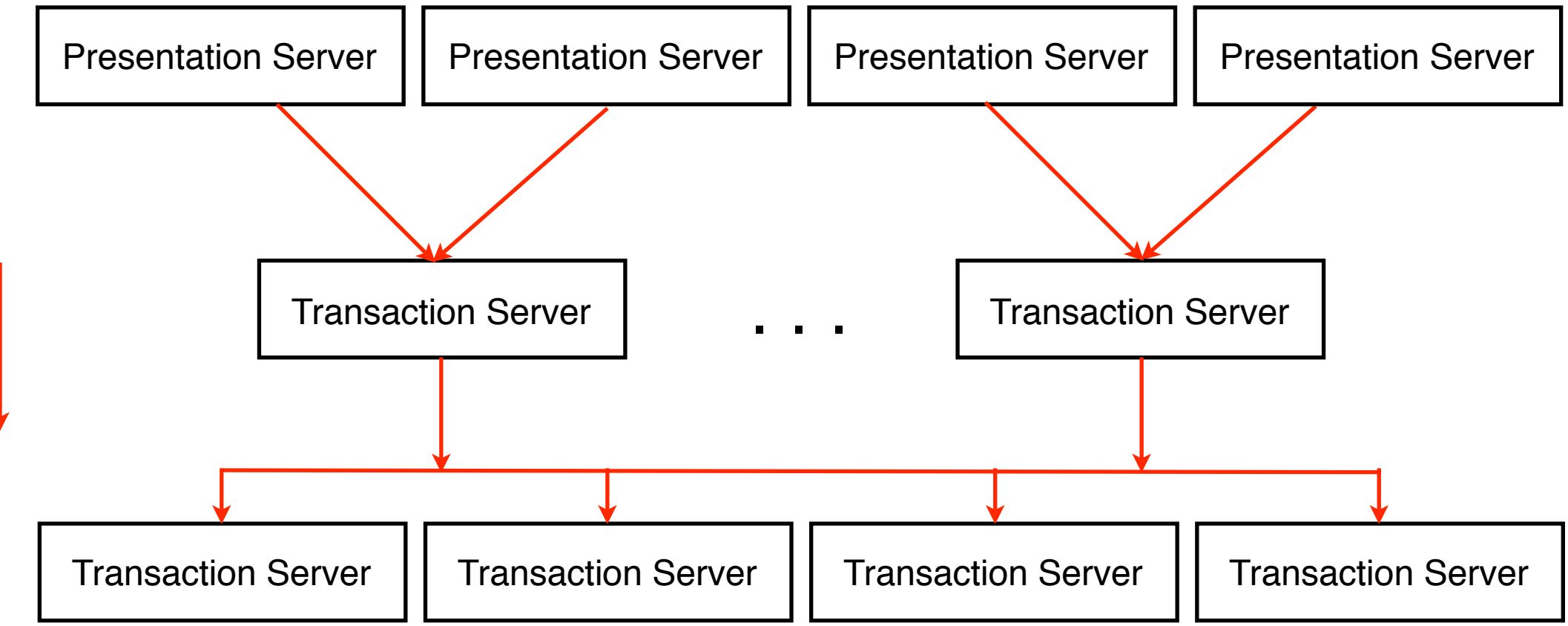


2-Tier Communication



2-Tier system requires quadratically many edges (sessions)

3-Tier Communication



3-Tier system requires only linearly many sessions

2-Phase Commit

- Phase 1:
  - Coordinator sends PREPARE to all participants and waits for responses
  - Participants reply YES or NO, or fail to reply

2-Phase Commit

- Phase 2:
  - Coordinator decides YES iff received YES votes from all participants
  - Coordinator sends decision to all participants
  - Participants reply DONE
  - Coordinator frees resources after receiving DONE from all participants

2-Phase Commit - Blocking

- Correctness:
  - After voting NO participant may abort
  - After voting YES participant may not commit or abort until receiving the coordinator decision -- *in doubt*
  - What if coordinator fails while some participants are in doubt? *Blocked!*

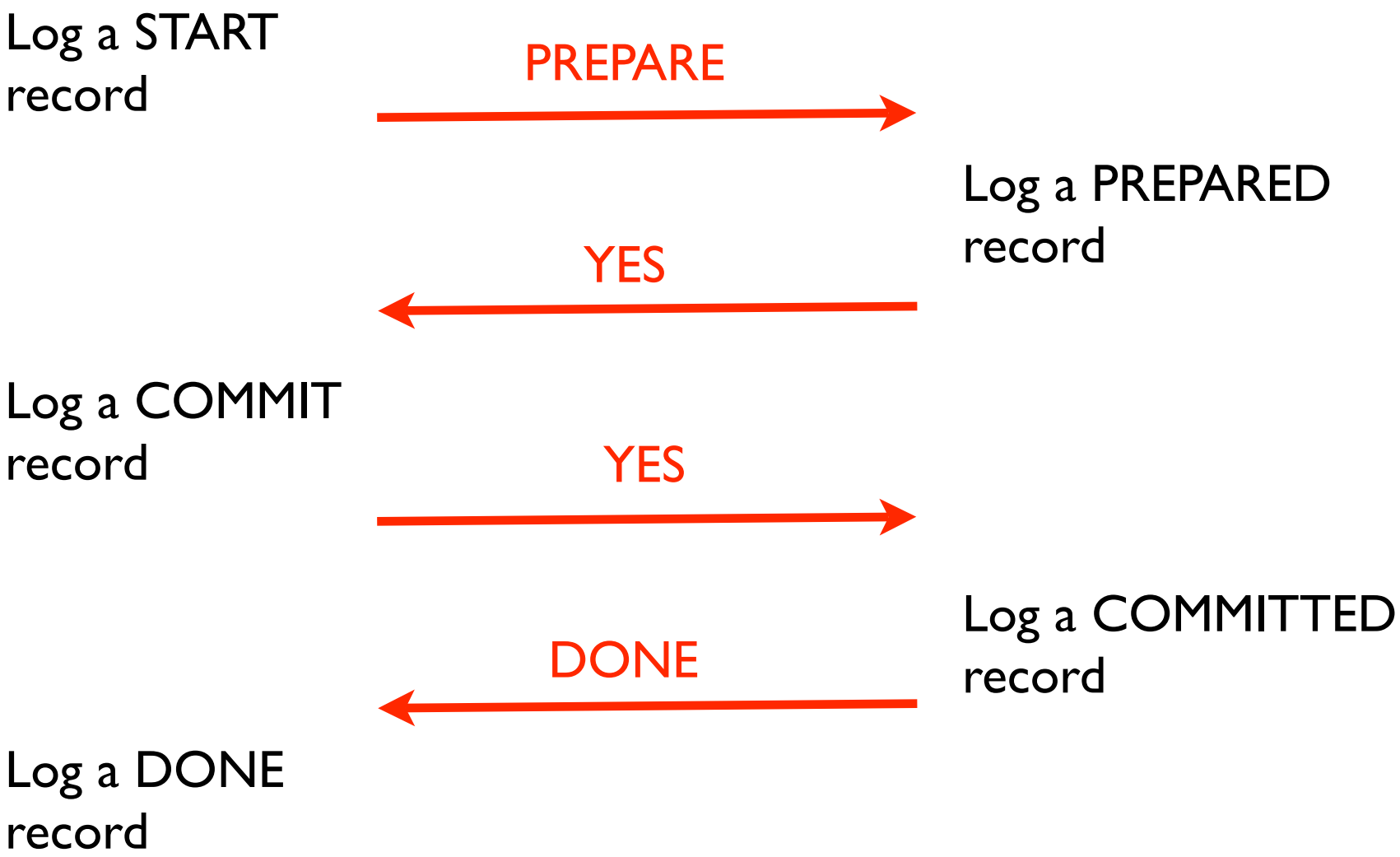
2-Phase Commit - Theorems

- For every possible distributed commit protocol, a communication failure can cause a participant to become blocked.
- No distributed commit protocol can guarantee *independent recovery* (recovery without cooperation from coordinator) of failed participants.

Logging in 2PC

- Coordinator and participants must log enough information to enable recovery if a failure occurs during execution of the 2PC protocol

Logging in 2PC



Logging in 2PC