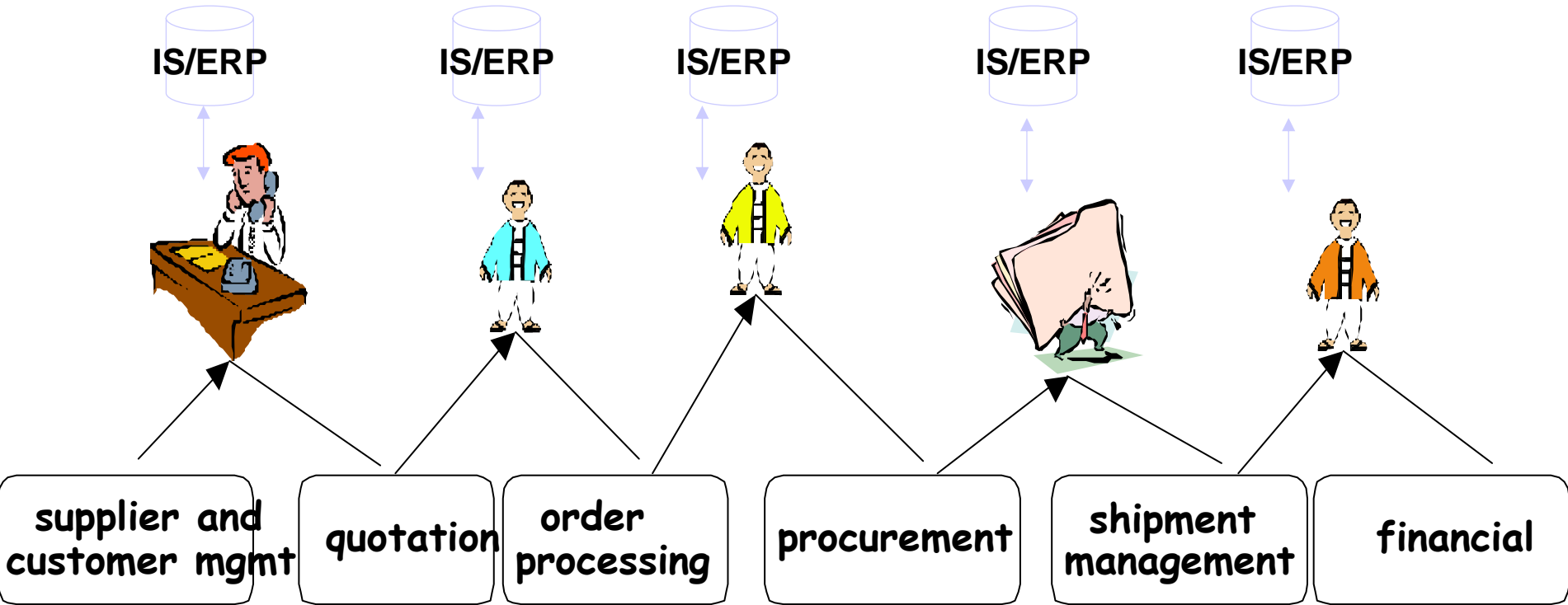


ERP Course: Enterprise Application Integration

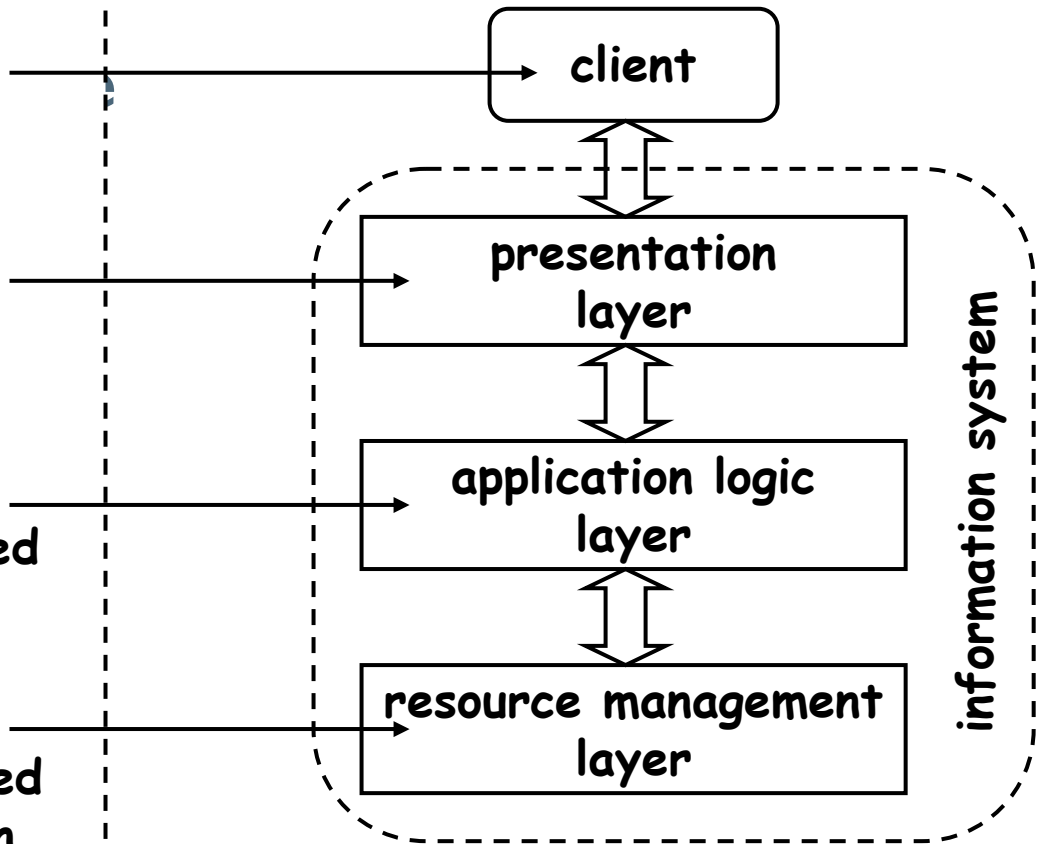
Readings: Chapter 3 from Gustavo Alonso et al

Peter Dolog
dolog [at] cs [dot] aau [dot] dk
E2-201
Information Systems
October 31, 2007

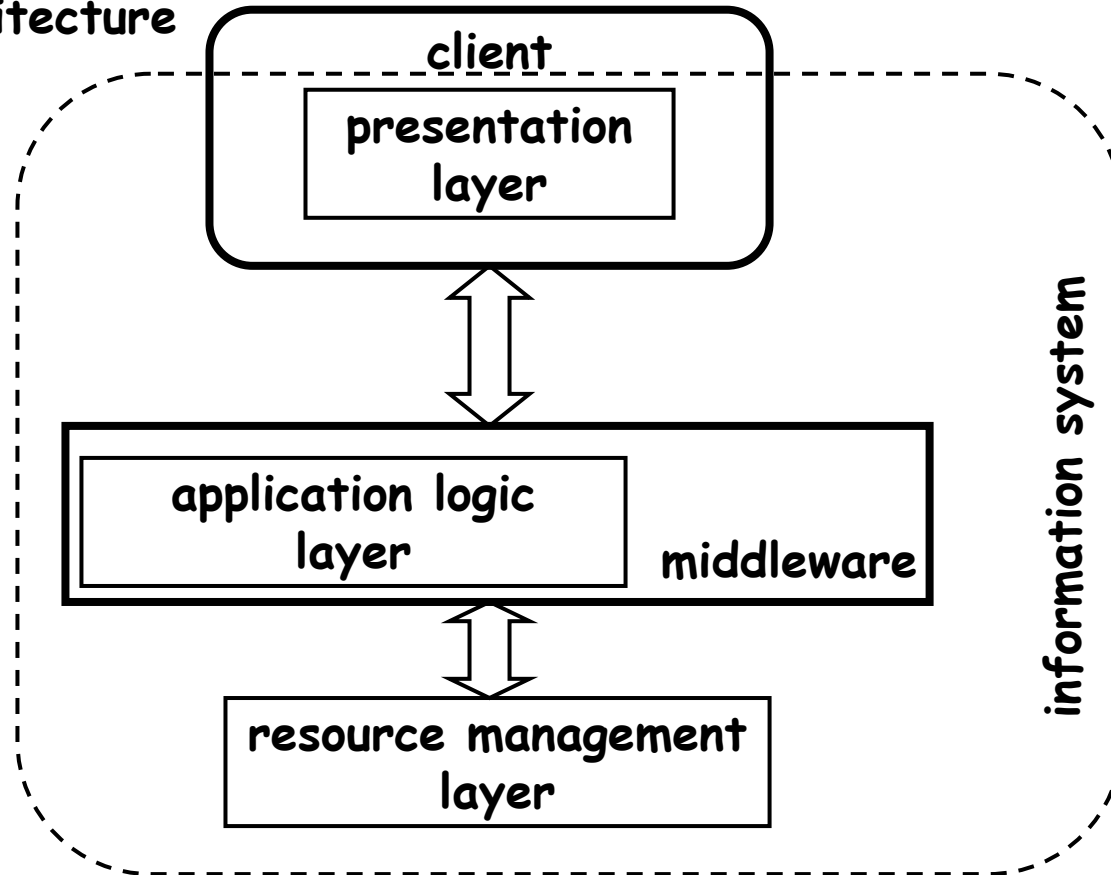


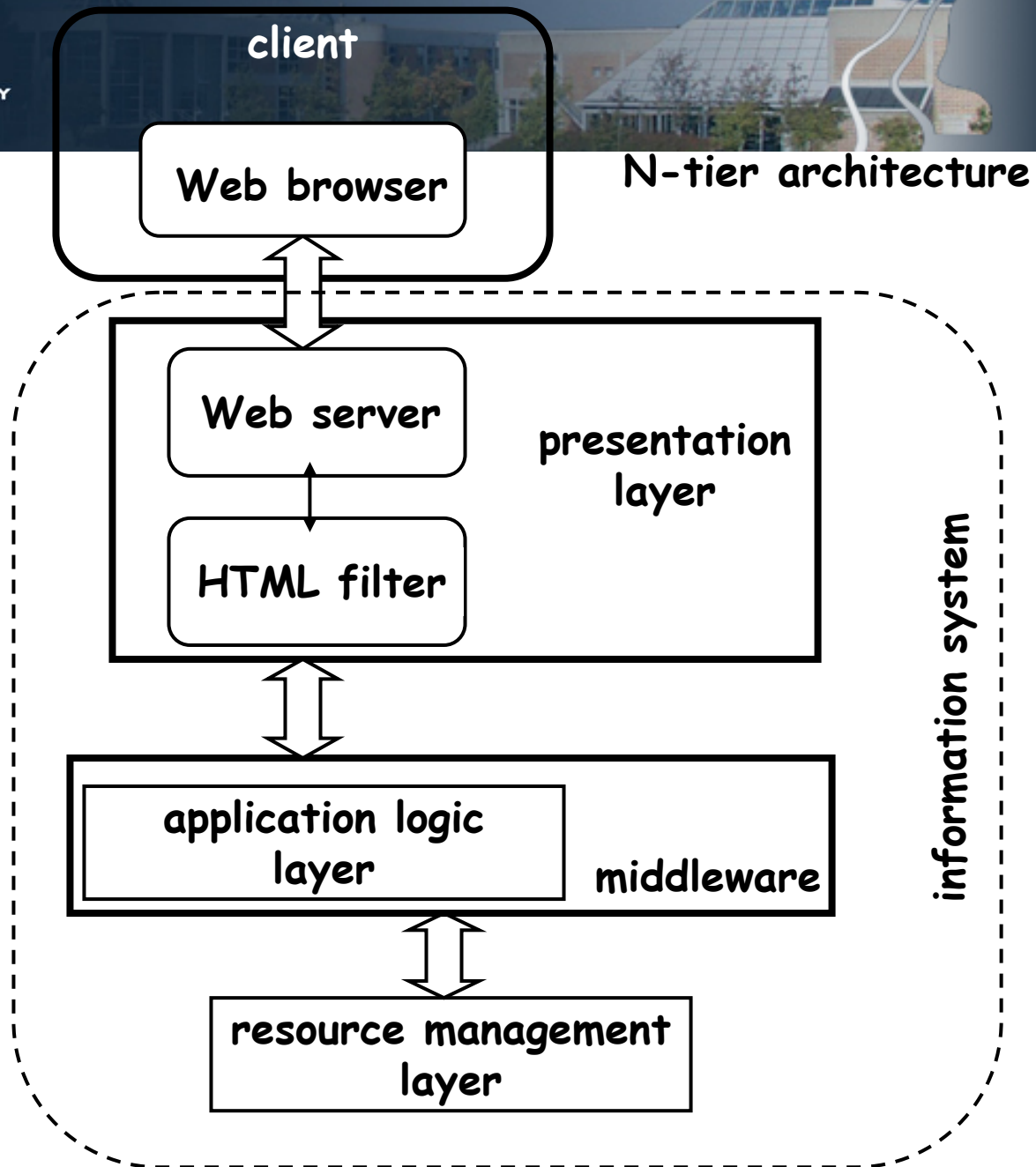
top-down design

1. define access channels and client platforms
2. define presentation formats and protocols for the selected clients and protocols
3. define the functionality necessary to deliver the contents and formats needed at the presentation layer
4. define the data sources and data organization needed to implement the application logic

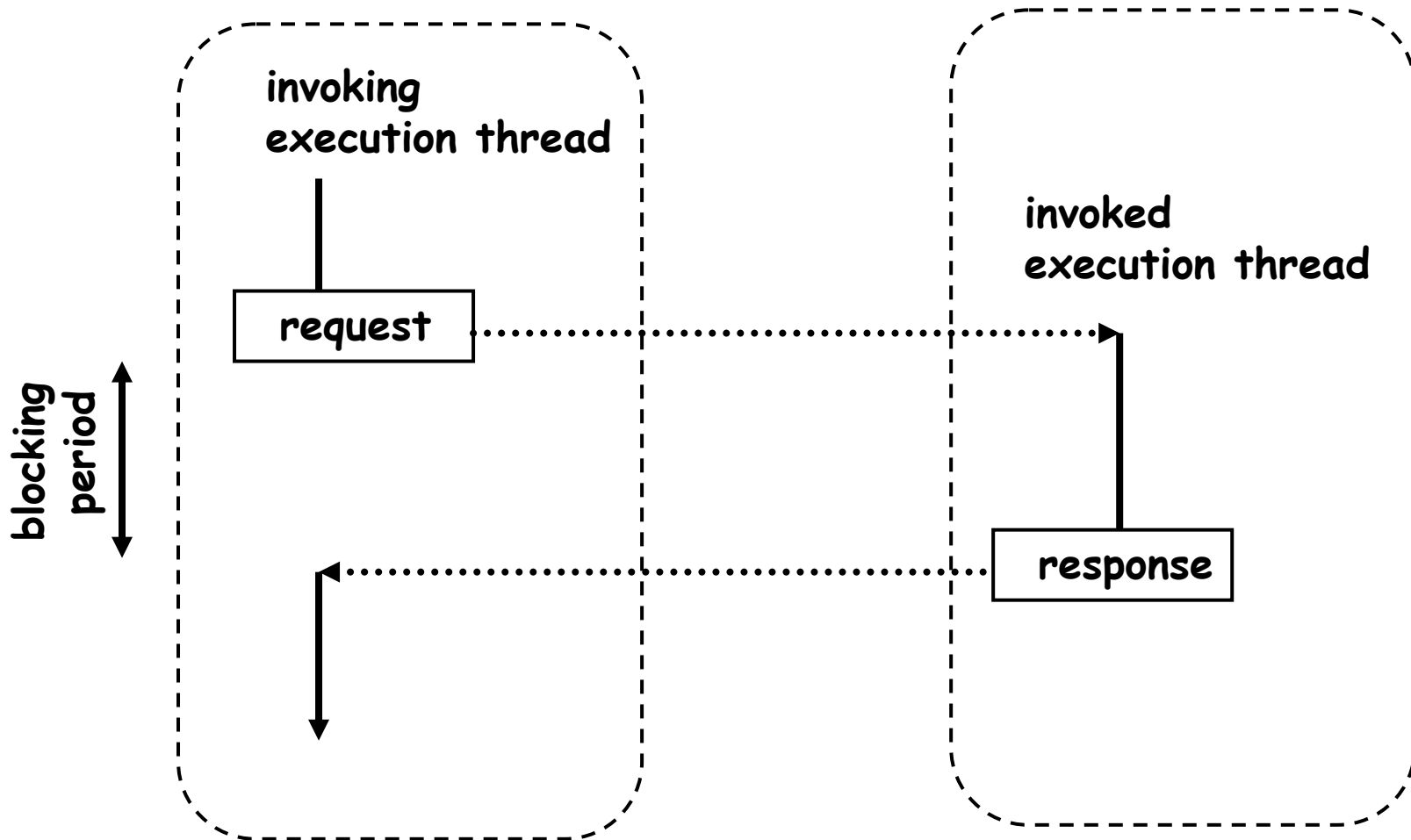


3-tier architecture

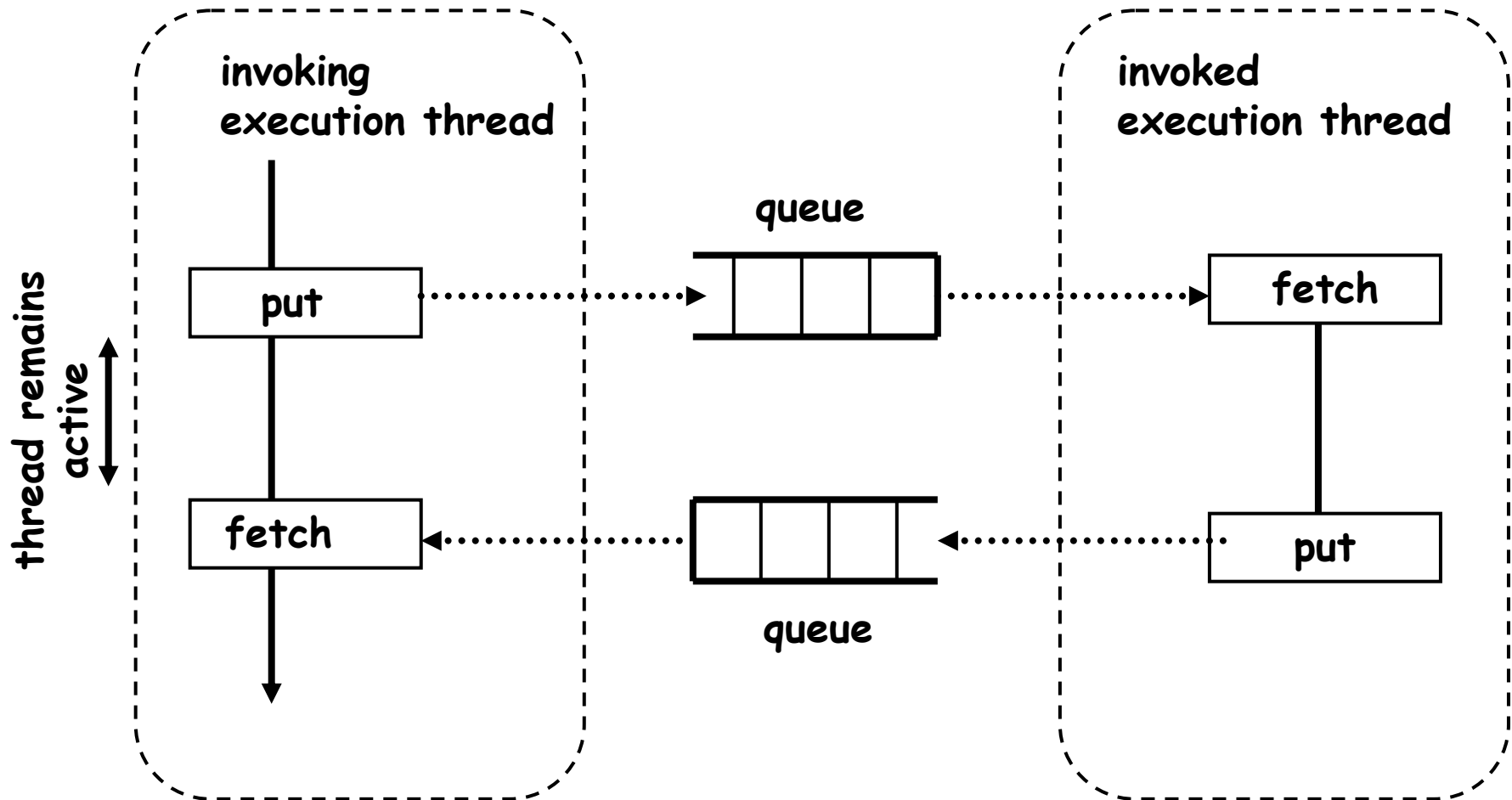




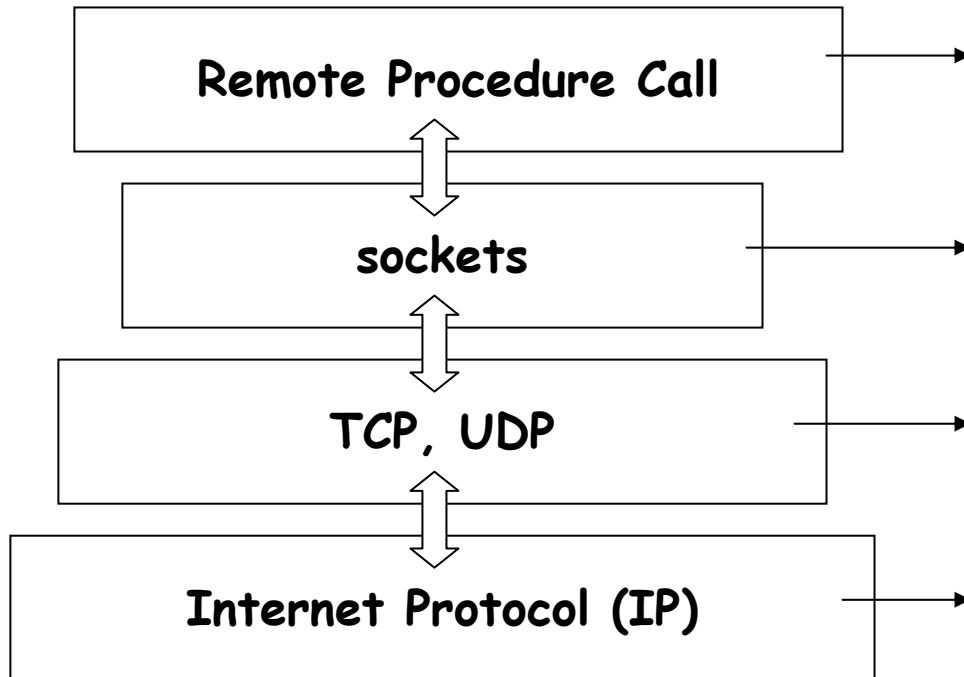
Blocking/Synchronous Communication



Nonblocking/Asynchronous Communication



RPC 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

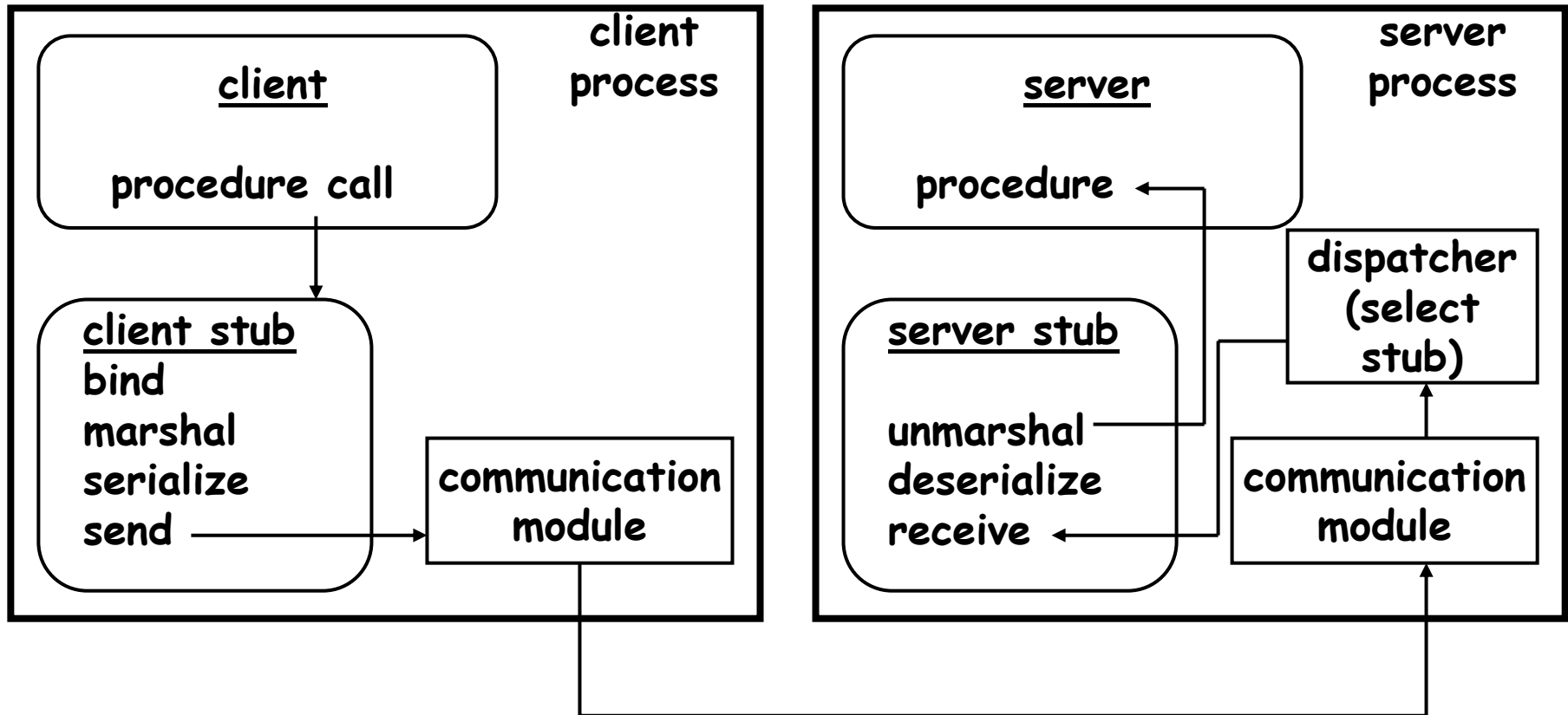
TCP, UDP:

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



client

client process

- 1. BOT
- 4. procedure call
- 10. EOT ←

client stub

- 2. register txn & create context
- 5. add txn id & context to call
- 11. request commit
- 14. confirm termination

server process

server

- 9. procedure ←

server stub

- 6. extract context and txn id
- 7. register server for txn
- 13. participate in 2PC ←

- 3. create txn id
- register txn
- register client for txn
- return txn id

- 8. lookup txn id
- register server for txn

- 12. lookup txn id
- run 2PC
- notify client of outcome

transaction manager

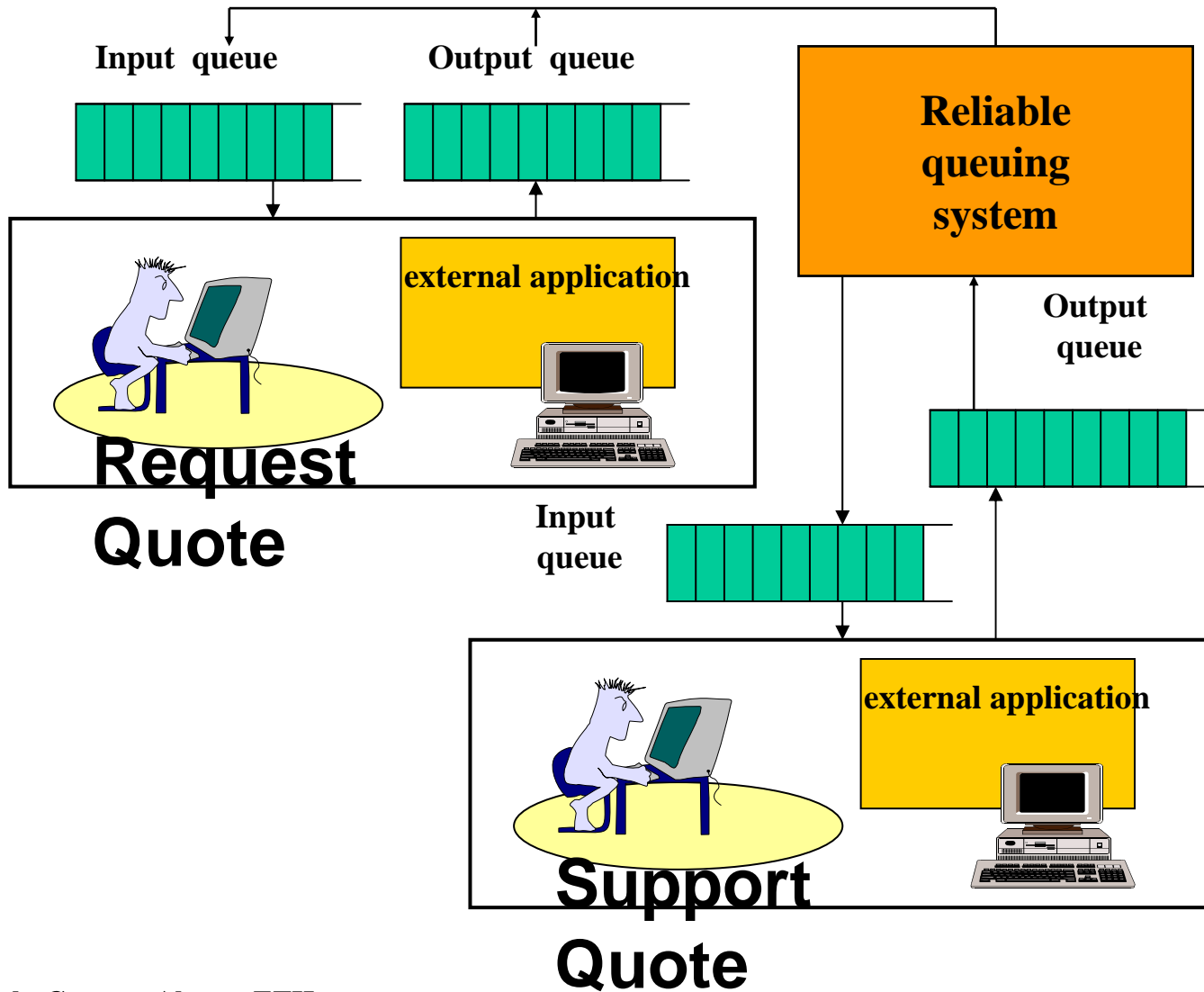
Other Middleware

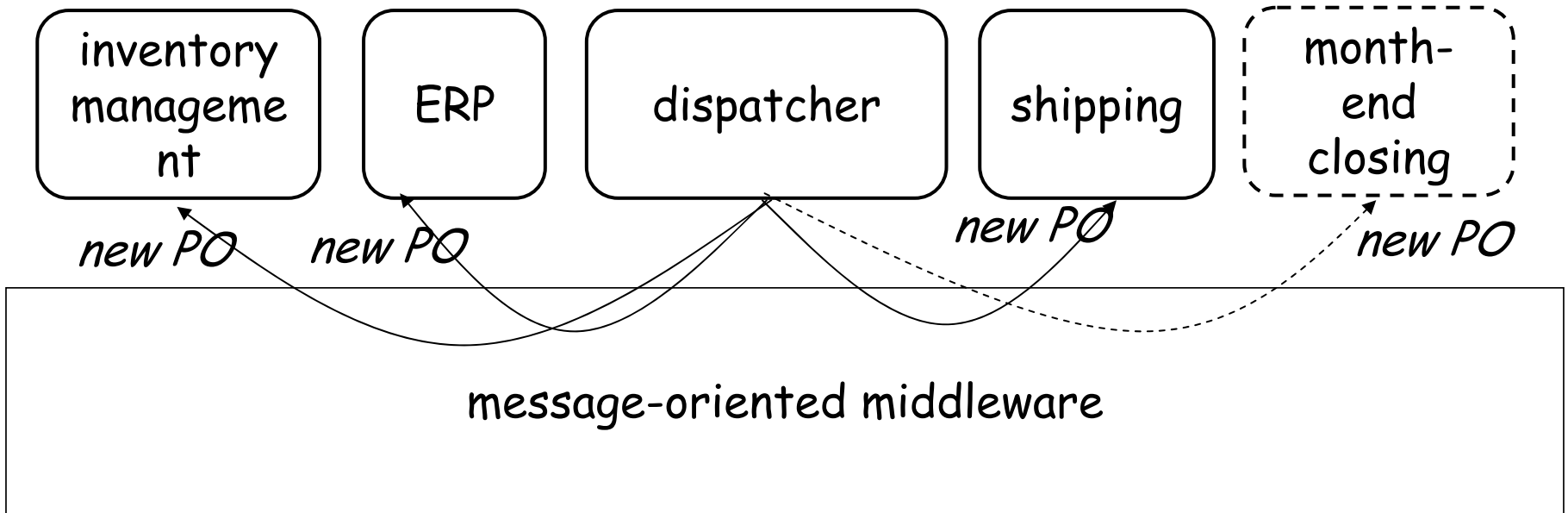
Object brokers

Object monitors

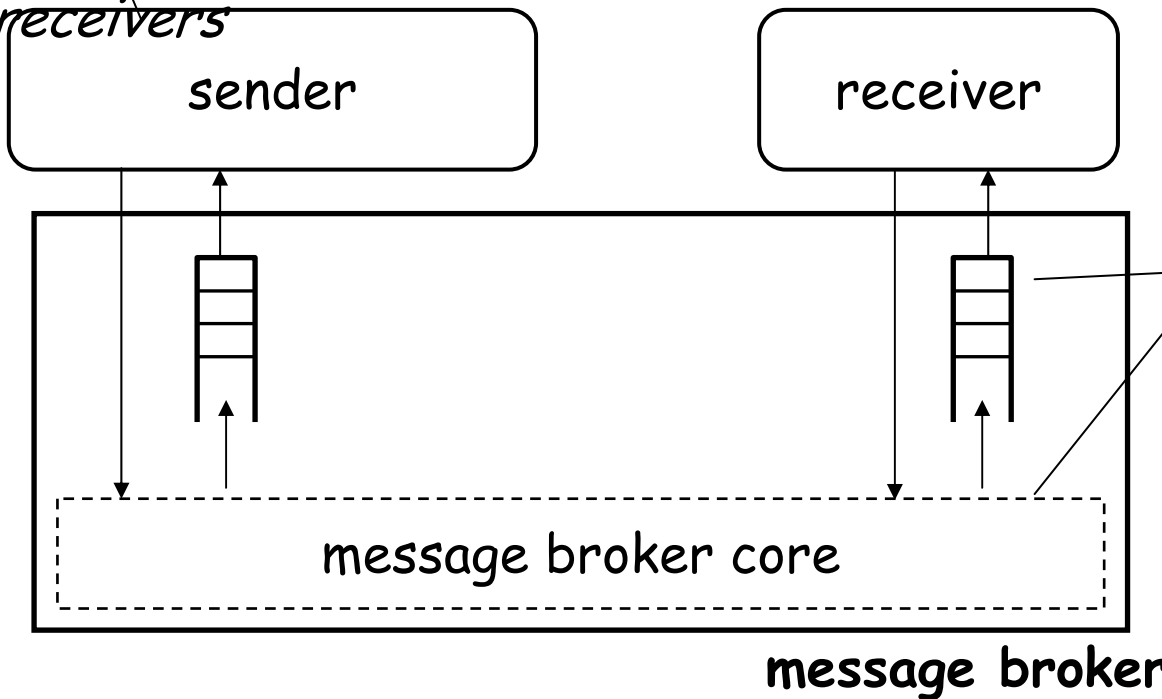
Message-oriented middleware

Message brokers

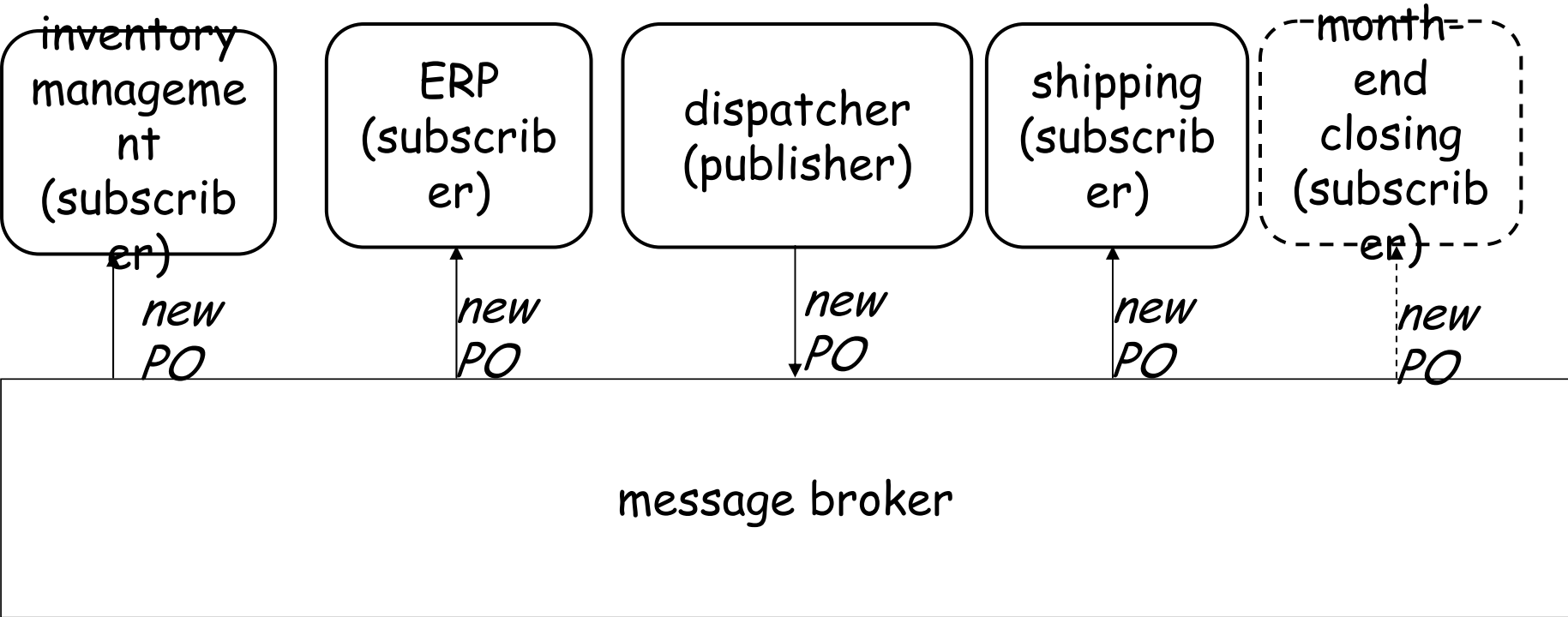


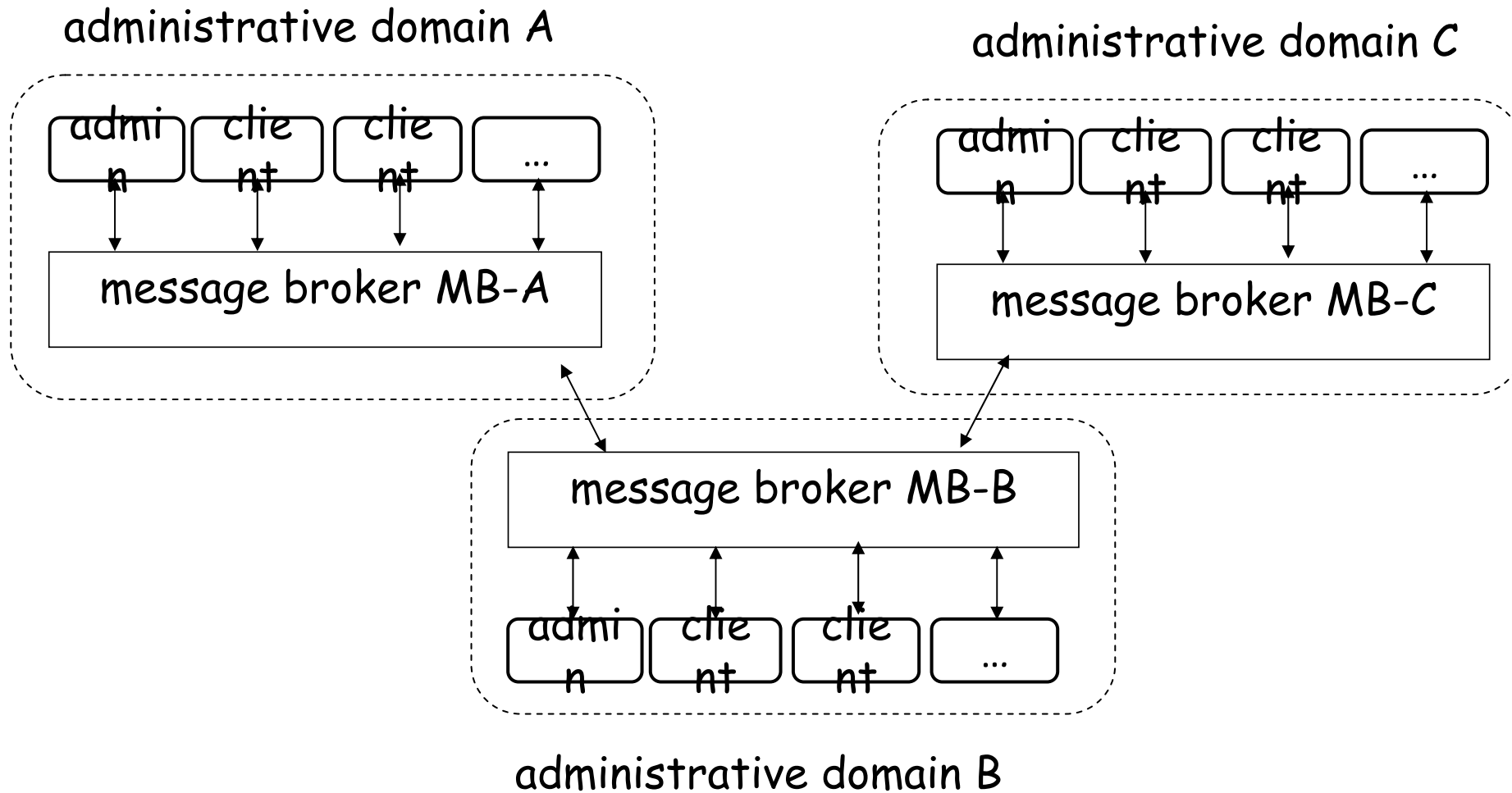


in basic MOM it is the sender who specifies the identity of the receivers



with message brokers, custom message routing logic can be defined at the message broker level or at the queue level





integrating application
(contains the composition logic)



message broker



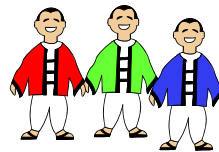
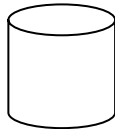
SmartQuotation
adapter

database
adapter

SmartForecasting
adapter

e-mail
adapter

XYZ
adapter

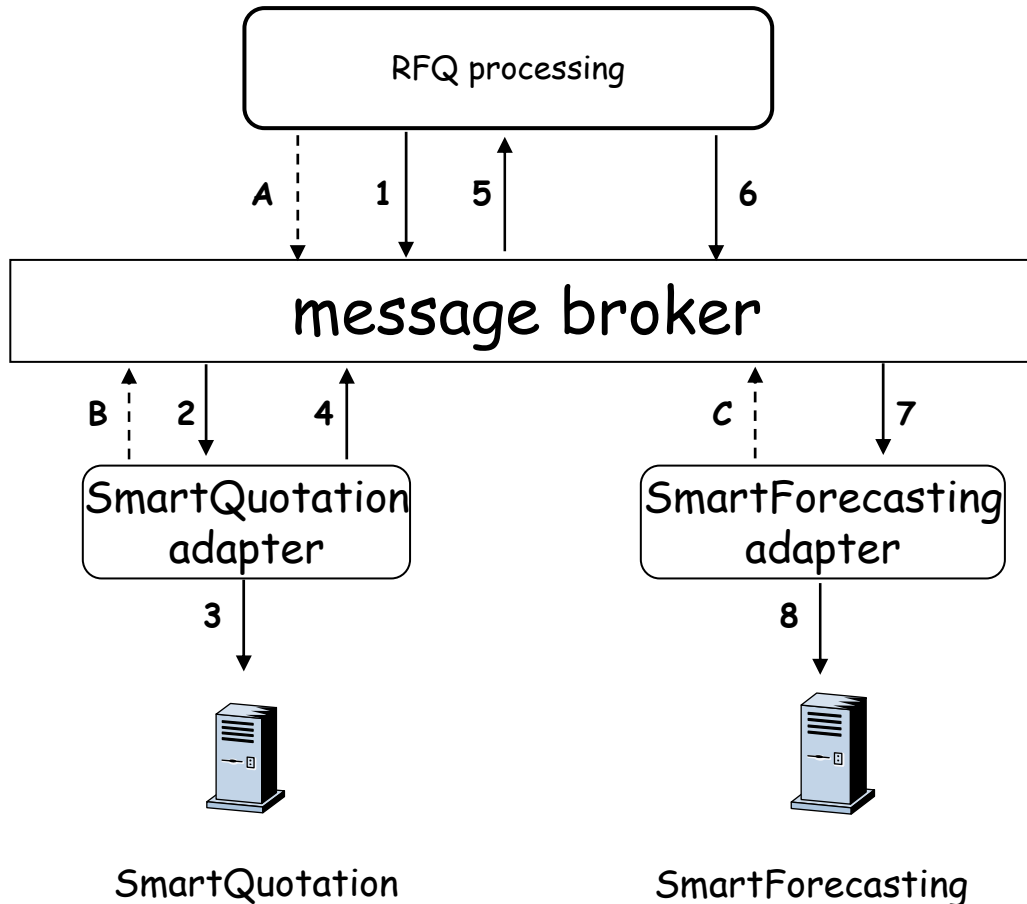


SmartQuotation

DBMS
applications

SmartForecasting

XYZ



at systems startup time (can occur in any order, but all must occur before RFQs are executed)

A: subscription to message *quote*

B: subscription to message *quoteRequest*

C: subscription to message *newQuote*

at run time: processing of a request for quote.

1: publication of a *quoteRequest* message

2: delivery of message *quoteRequest*

3: synchronous invocation of the *getQuote* function

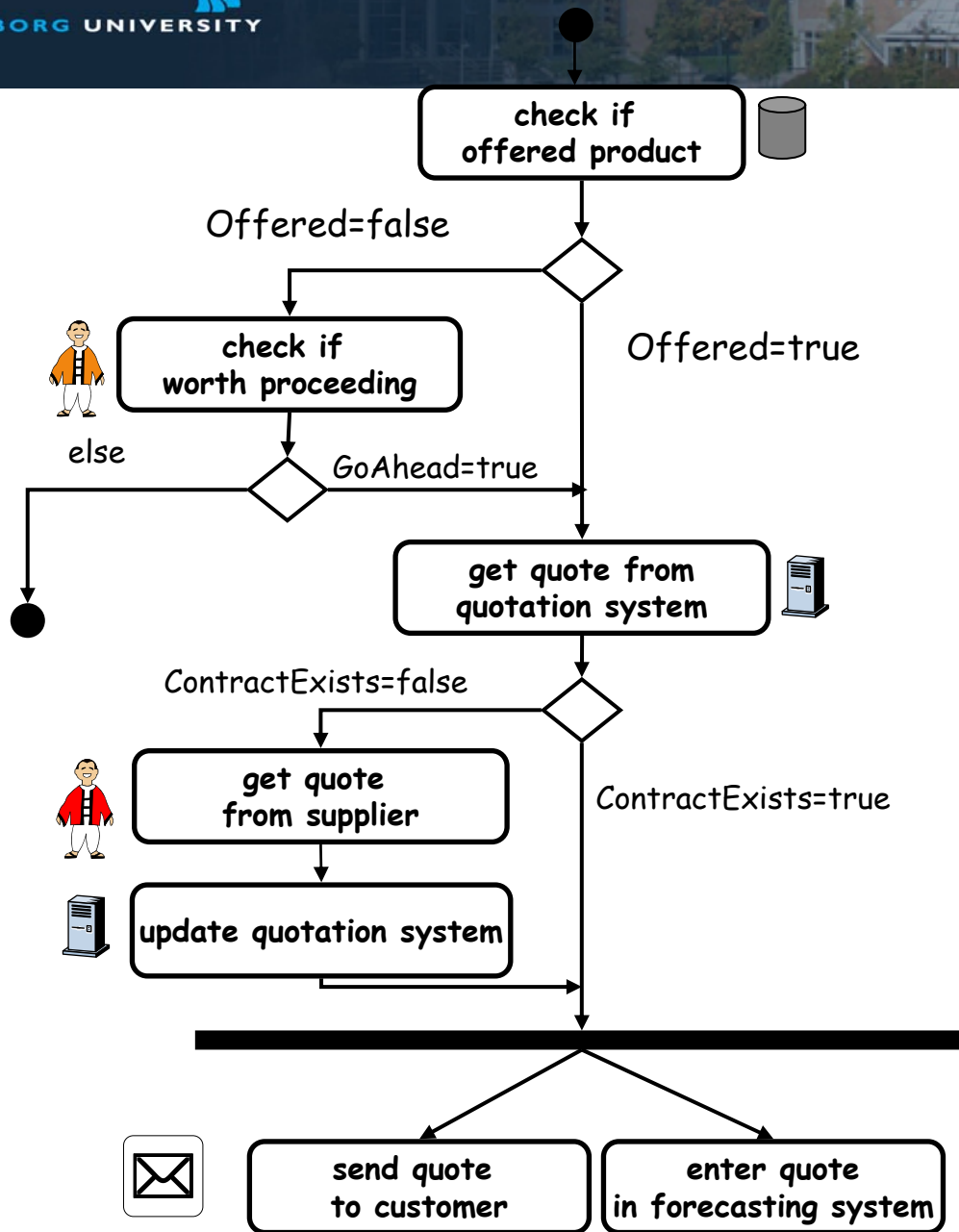
4: publication of a *quote* message

5: delivery of message *quote*

6: publication of a *newQuote* message

7: delivery of message *newQuote*

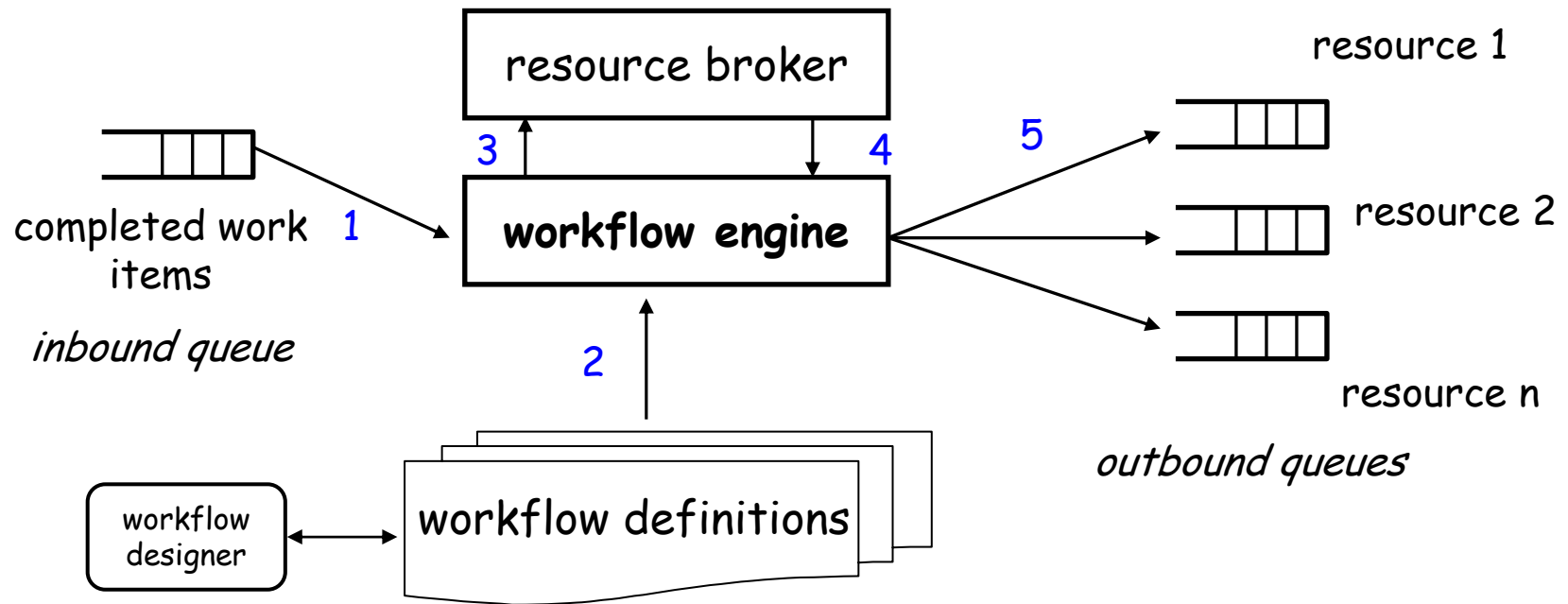
8: invocation of the *createForecastEntry* procedure



variables:

- QuoteReferenceNumber: int
- Customer: String
- Item: String
- Quantity: int
- RequestedDeliveryDate: Date
- DeliveryAddress: String
- GoAhead: Bool
- ContractExists: Bool
- Offered: Bool

Copyright Springer Verlag Berlin Heidelberg 2004



Workflow for Initiator Role

Workflow for Participant Role

