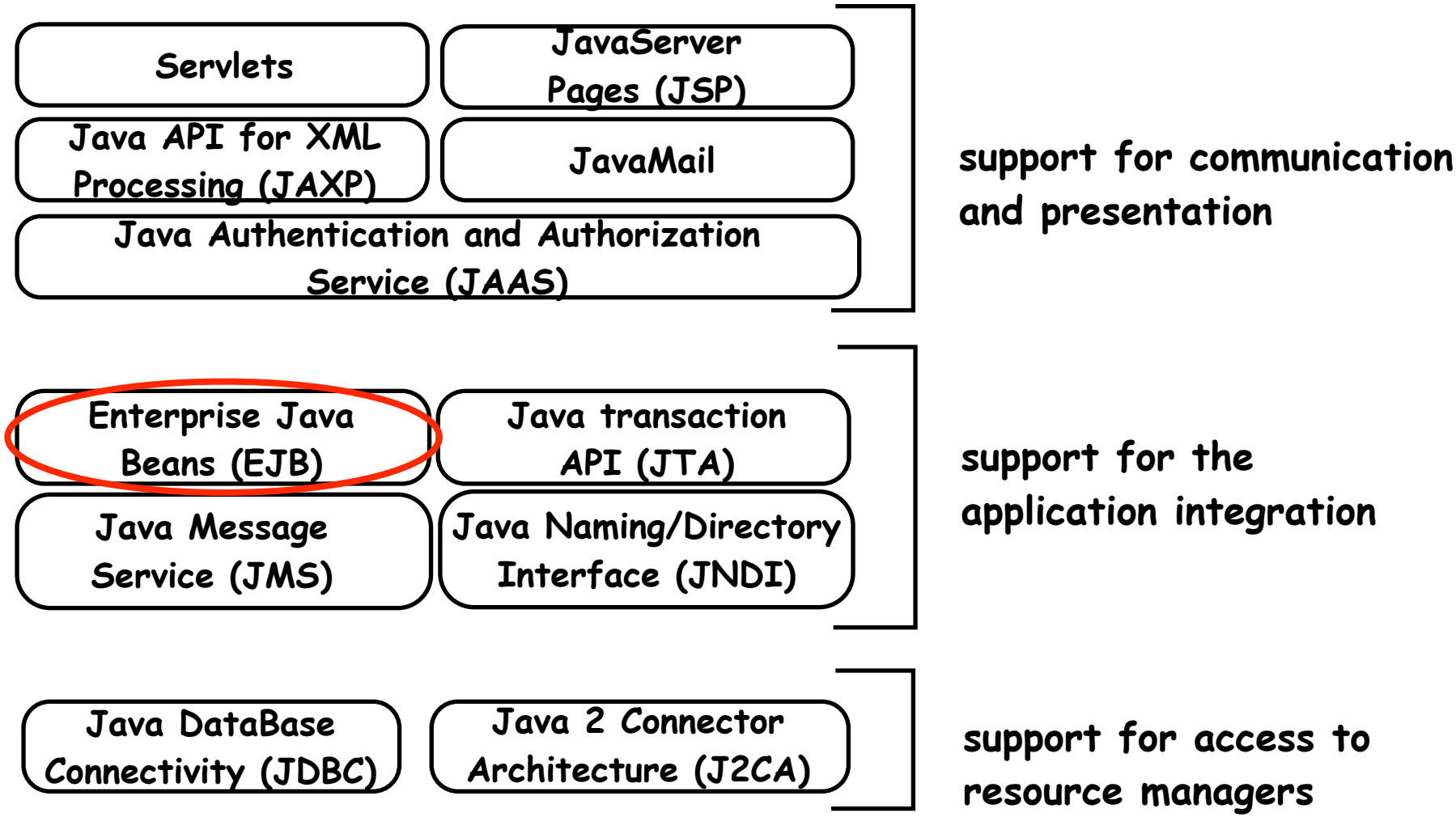
# Announcements

- Nearly all comments on project proposals have been emailed ...

# J2EE Application Server Architecture



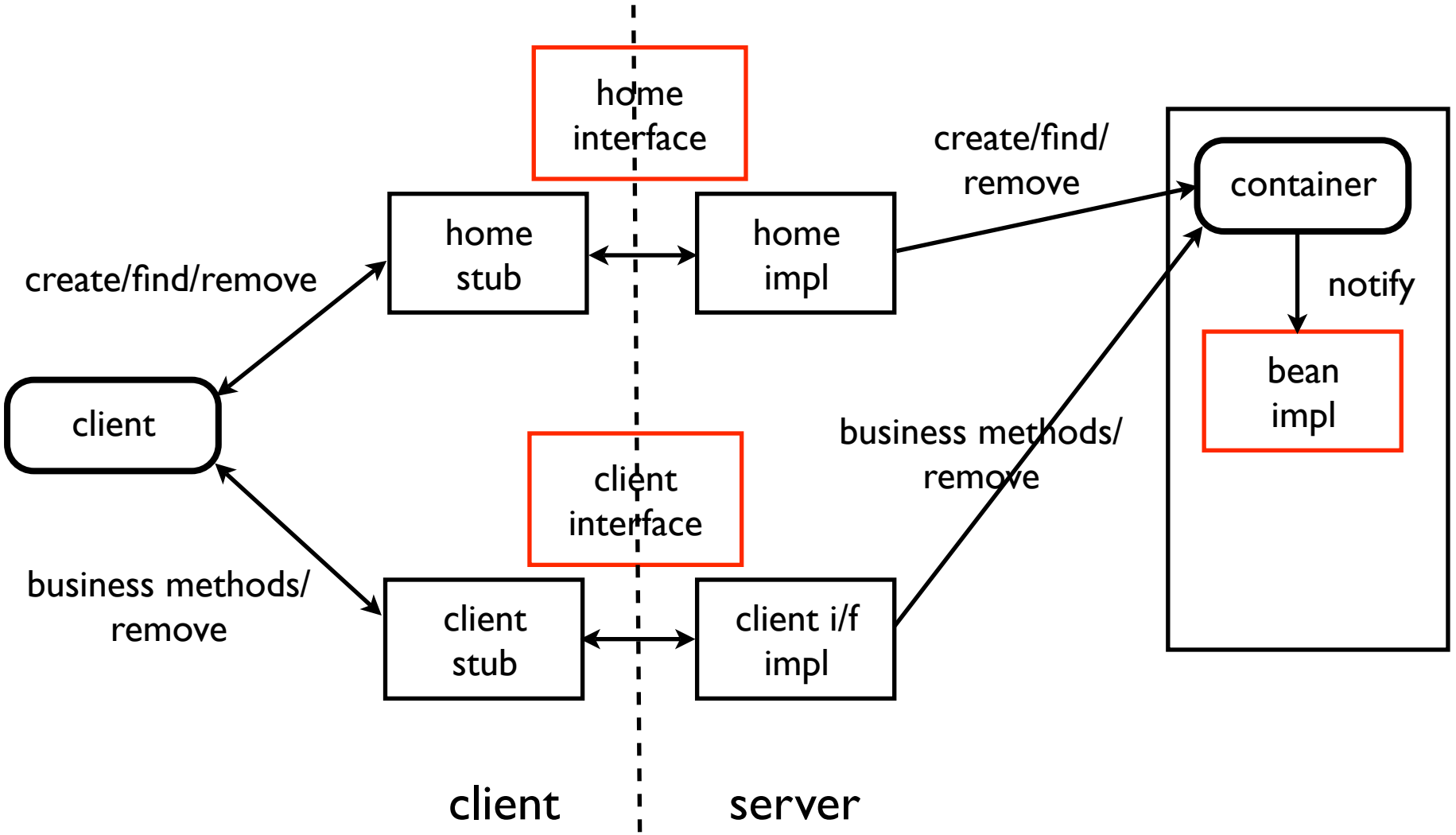
# What is a bean?

- Java Object
- Container provides many standard services
  - location / binding
  - life cycle
  - persistence
  - transactions
  - etc

# Classes of Beans

- Session
  - embodies business logic associated with a session / business txn
- Entity
  - represents a real-world object
- Message-driven
  - asynchronous processing using JMS queues

# EJB Classes and Stubs



# Deployment

- XML deployment descriptors
  - identify programmer-provided code
  - dependencies
  - transactional behavior
  - security properties
  - etc
- Container & tools generate stubs
- Modify code / create wrappers based on properties specified in deployment descriptors

# Session Beans

- embodies business logic associated with a session's / business txn
- always given a session context
- not persistent
- stateful
  - invoked only thru creating session
- stateless
  - use by concurrent sessions allowed

# Entity Beans

- represents a real-world entity
- may be shared by multiple sessions
- persistent beyond session or EJB container lifetime
- persistence may be *bean-managed* (JDBC/JTA) or *container-managed*

# Bean-Managed Persistence

- Container notifies bean on activation/ passivation
- Bean implementation responsible for serializing state to database
- Bean implementation responsible for transaction behavior (JTA)
  - distributed txns may be supported

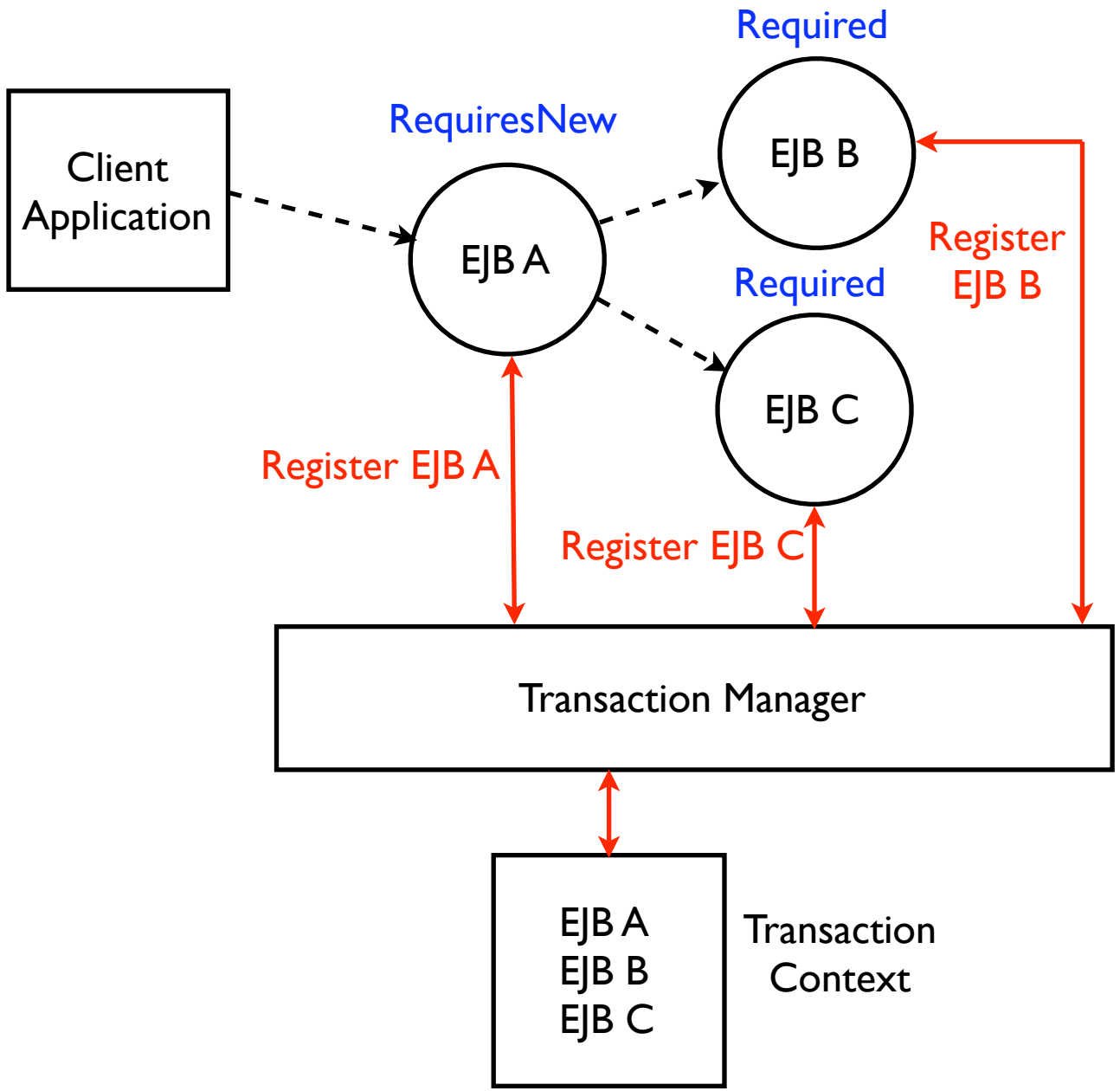
# Container-Managed Persistence

- Container manages serializing state to database and retrieving
- Container manages transactional behavior
  - declarative

# Declared Transaction Attributes

- NotSupported
- Supports
  - use txn if present
- Required
  - create txn if none
- RequiresNew
  - create txn
- Mandatory
  - error if no txn
- Never

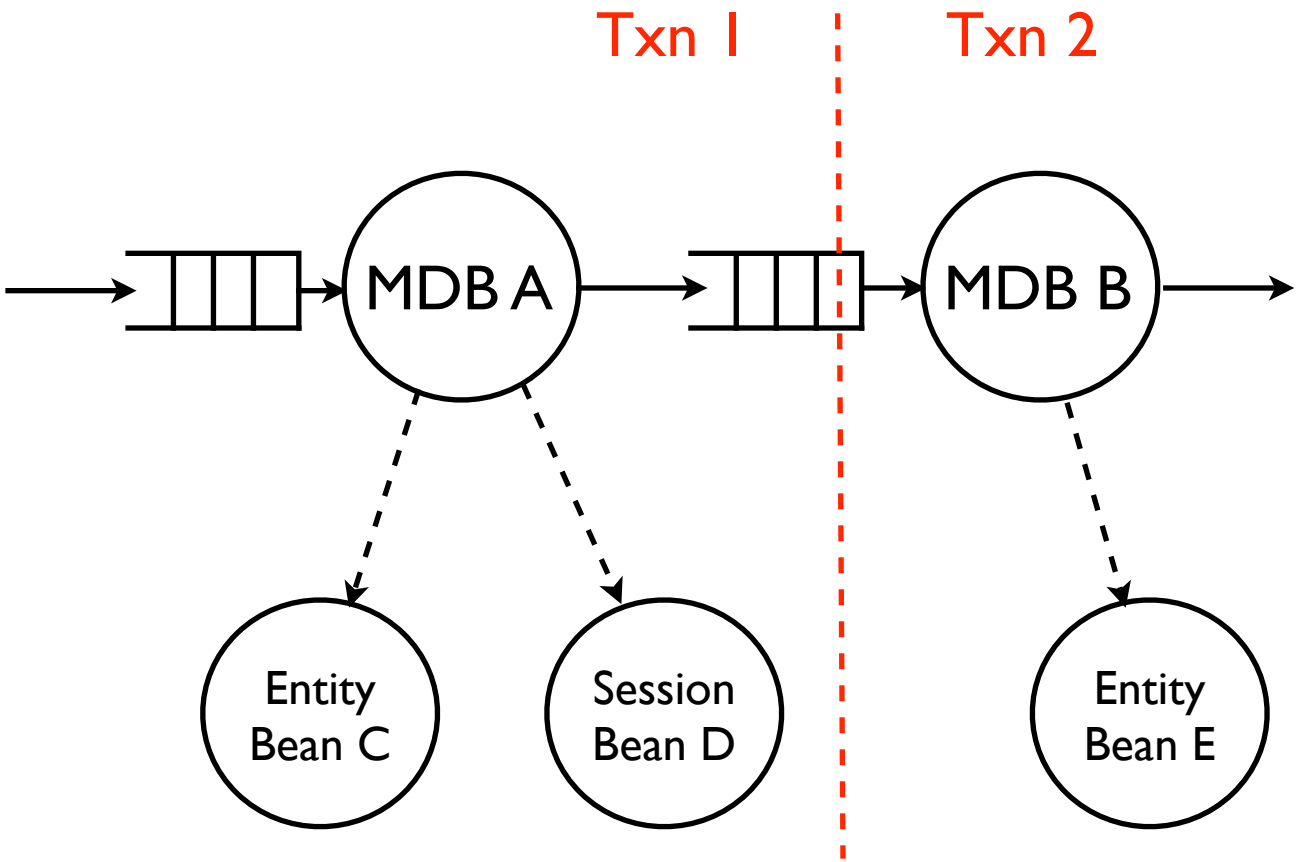
# Managing a Transaction Context



# Message-Driven Beans

- Communication by queueing (JMI) rather than RPC
- Container manages
  - creation of MDB when message arrives
  - invoke onMessage method with each arriving message
  - remove idle beans
- Equivalent to dequeue-process loop

# Message Driven Beans



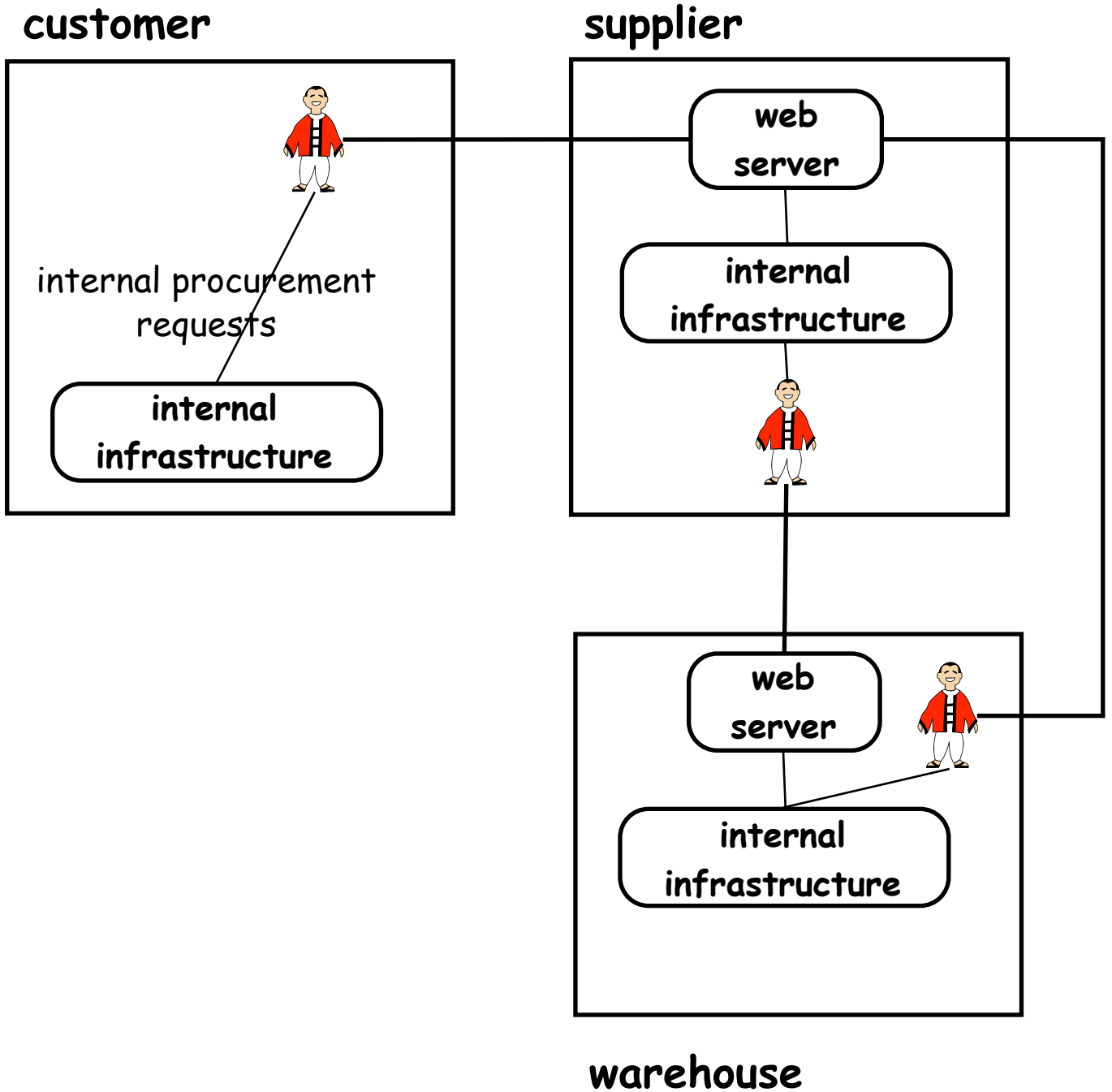
# Introducing Web Services

- [ACKM04] Ch 5, 6

# Definition

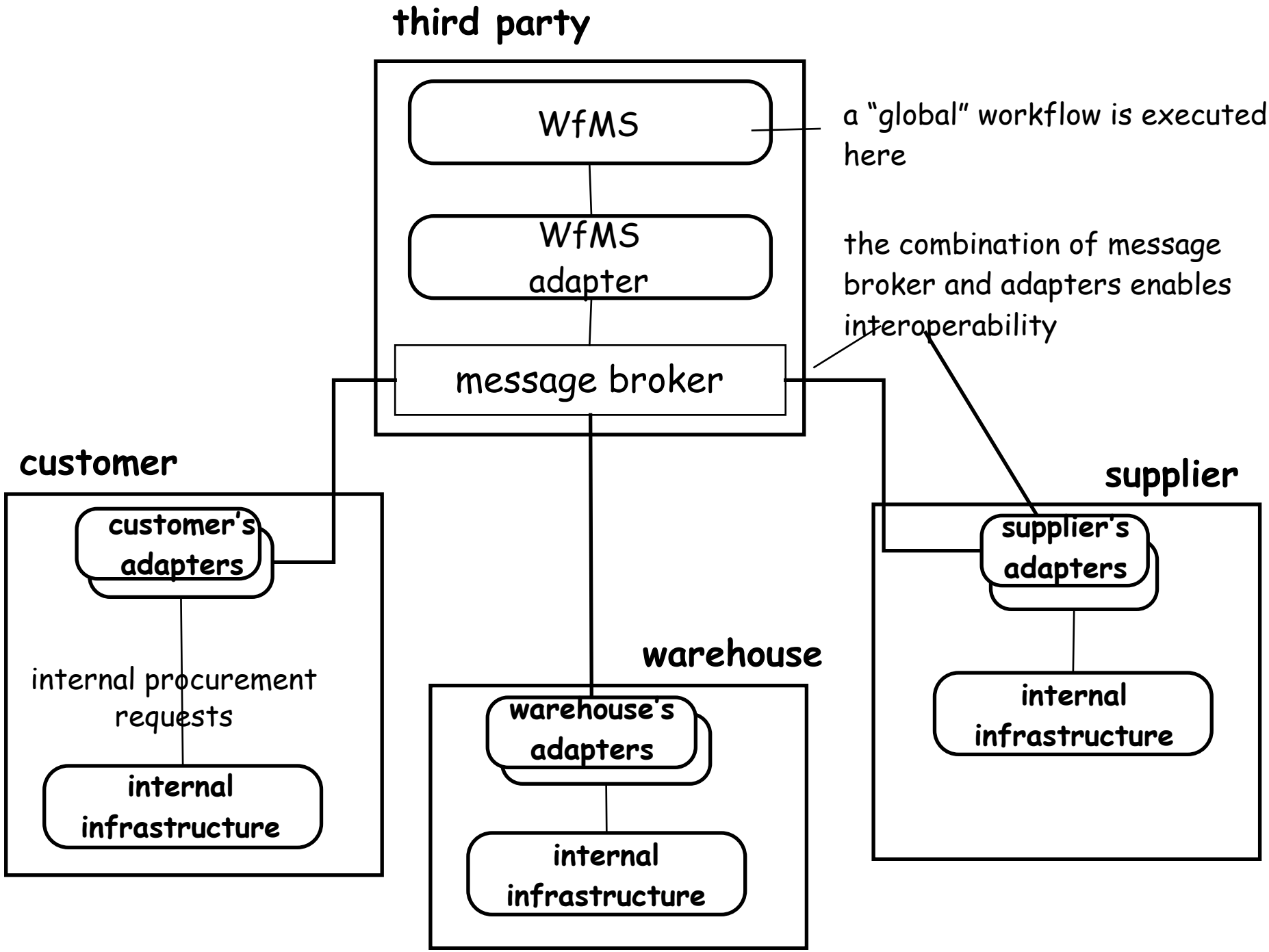
- A standardized way of integrating Web-based applications, using
  - XML to tag data
  - SOAP to transport data
    - Simple Object Access Protocol
  - WSDL to describe available services
    - Web Services Description Language
  - UDDI to list available services
    - Universal Description, Discovery and Integration

# Manual Integration Between Companies




B2B interactions occur by accessing Web pages, filling Web forms, or via email.

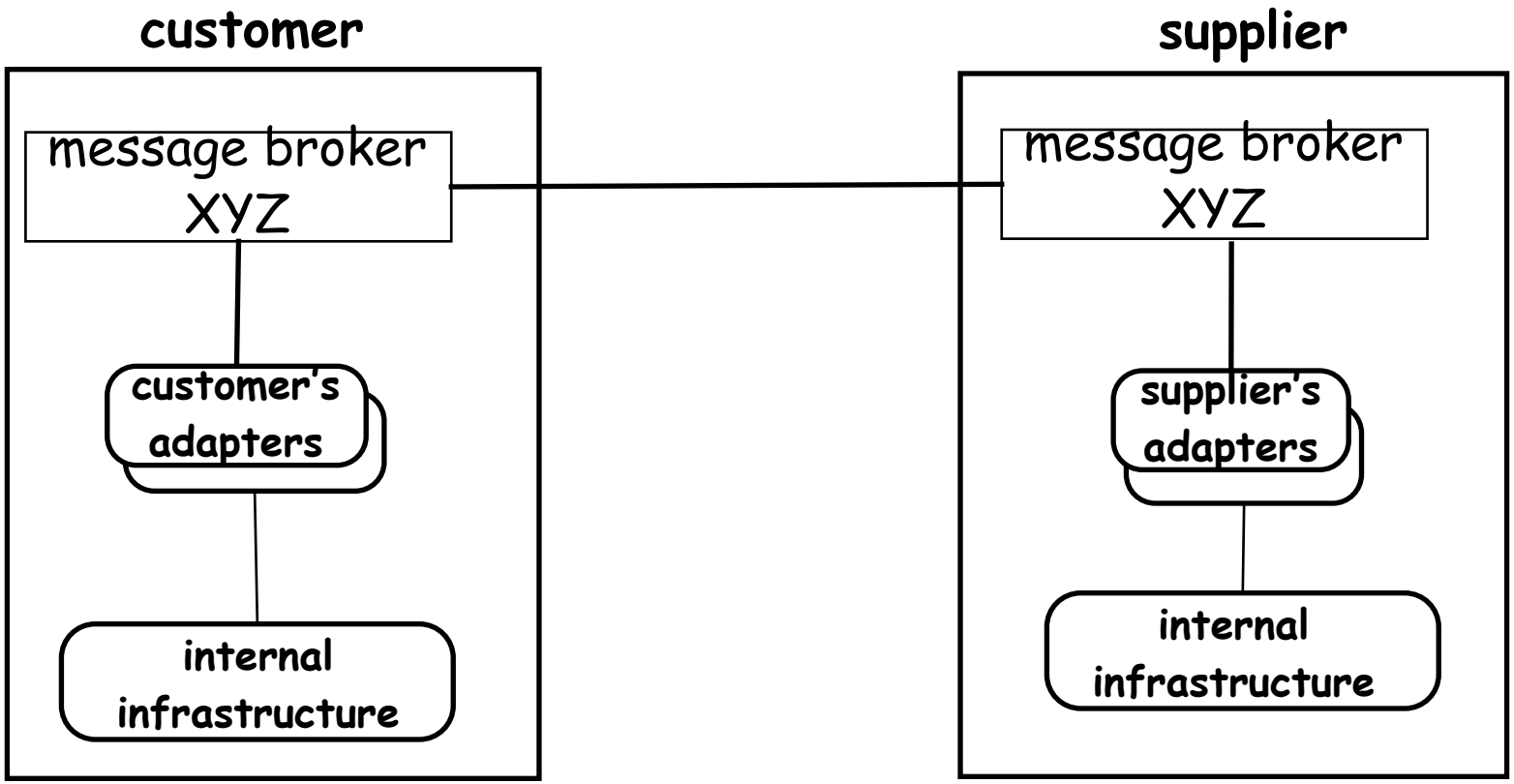
# In Principle - Middleware a Service



# Why not?

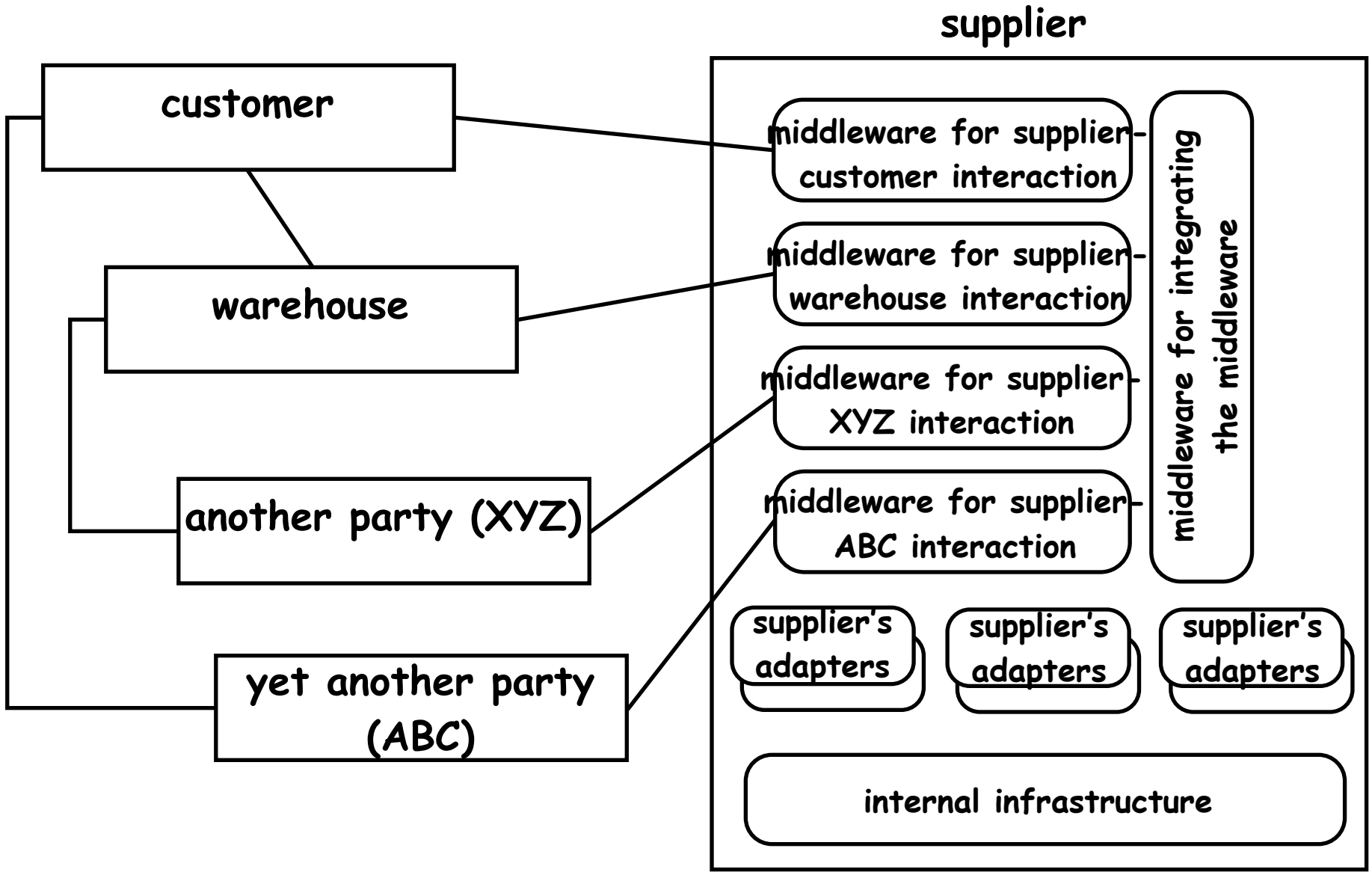
- Autonomy?
- Trust?
- Confidentiality?

# How About Point-to-Point?




- Note middleware must interoperate

# Middleware Explosion ...



# Goals for B2B Integration

- Service-Oriented Architecture
- Redesign of middleware protocols
- Standardization

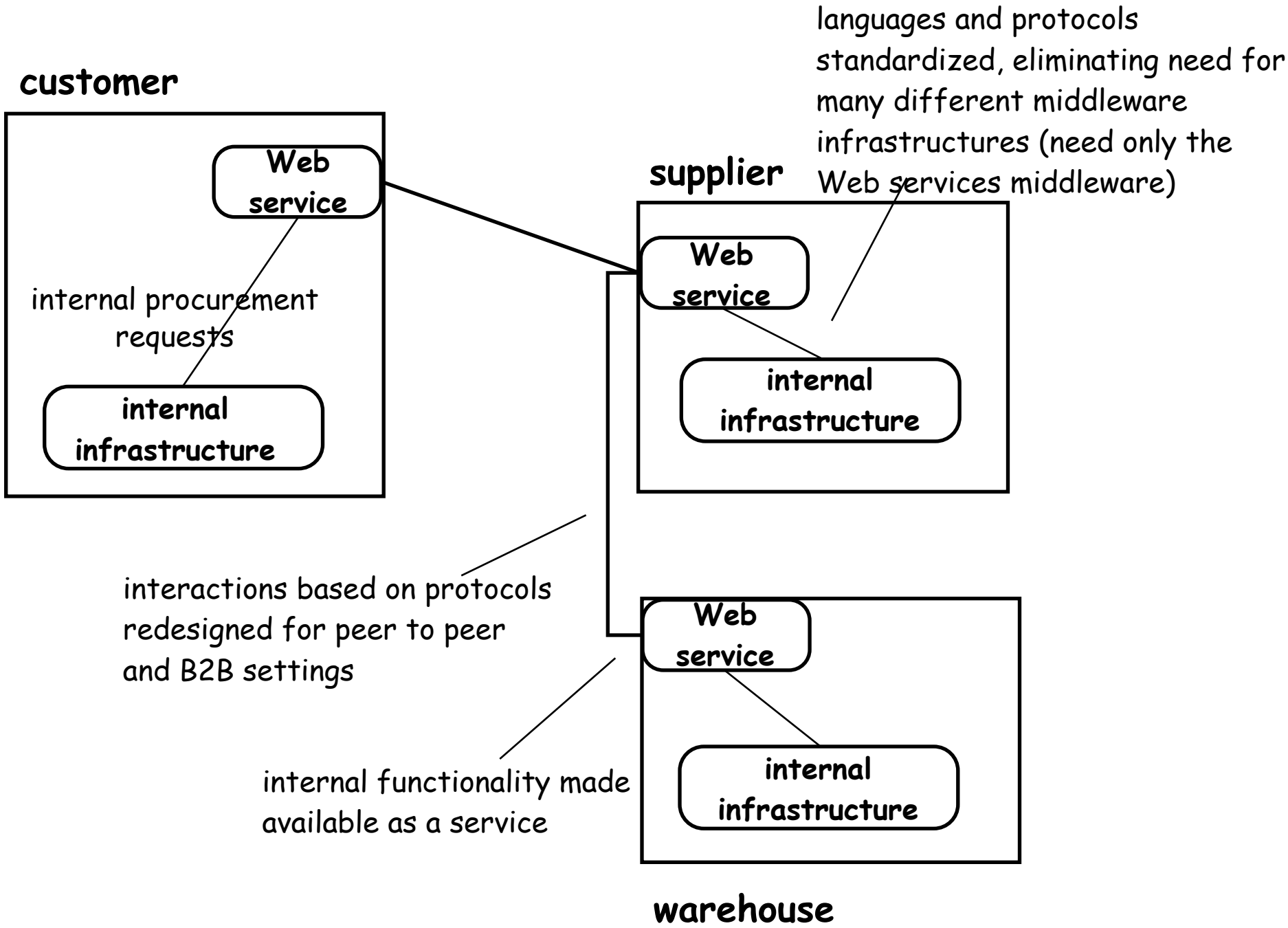
# Service-Oriented

- Functionality always exposed as services
- Loosely coupled
- Invoked by programs as well as users

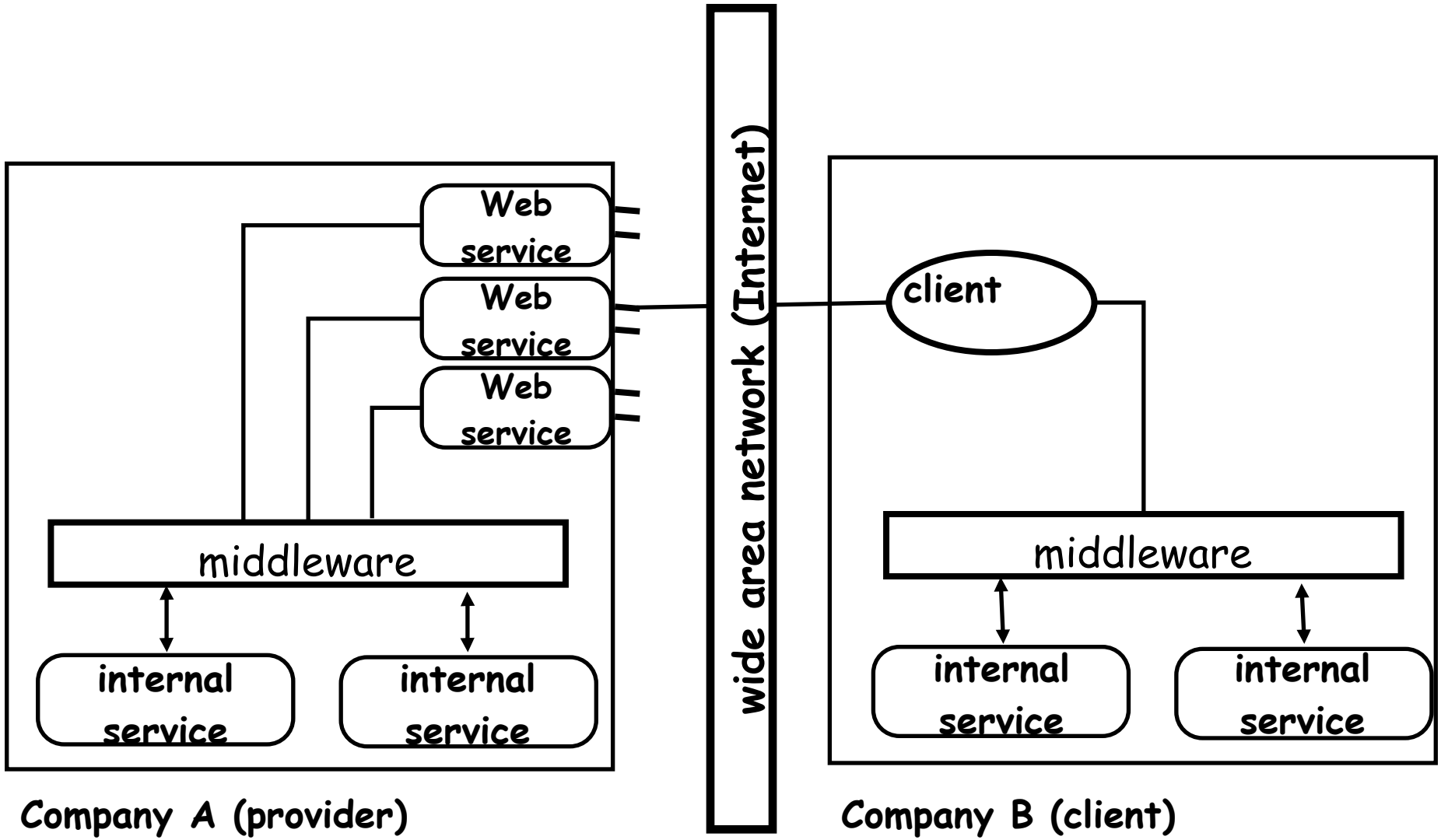
# Middleware Redesign

- Peer-to-Peer
- Between trust domains
- Compatibility with Internet

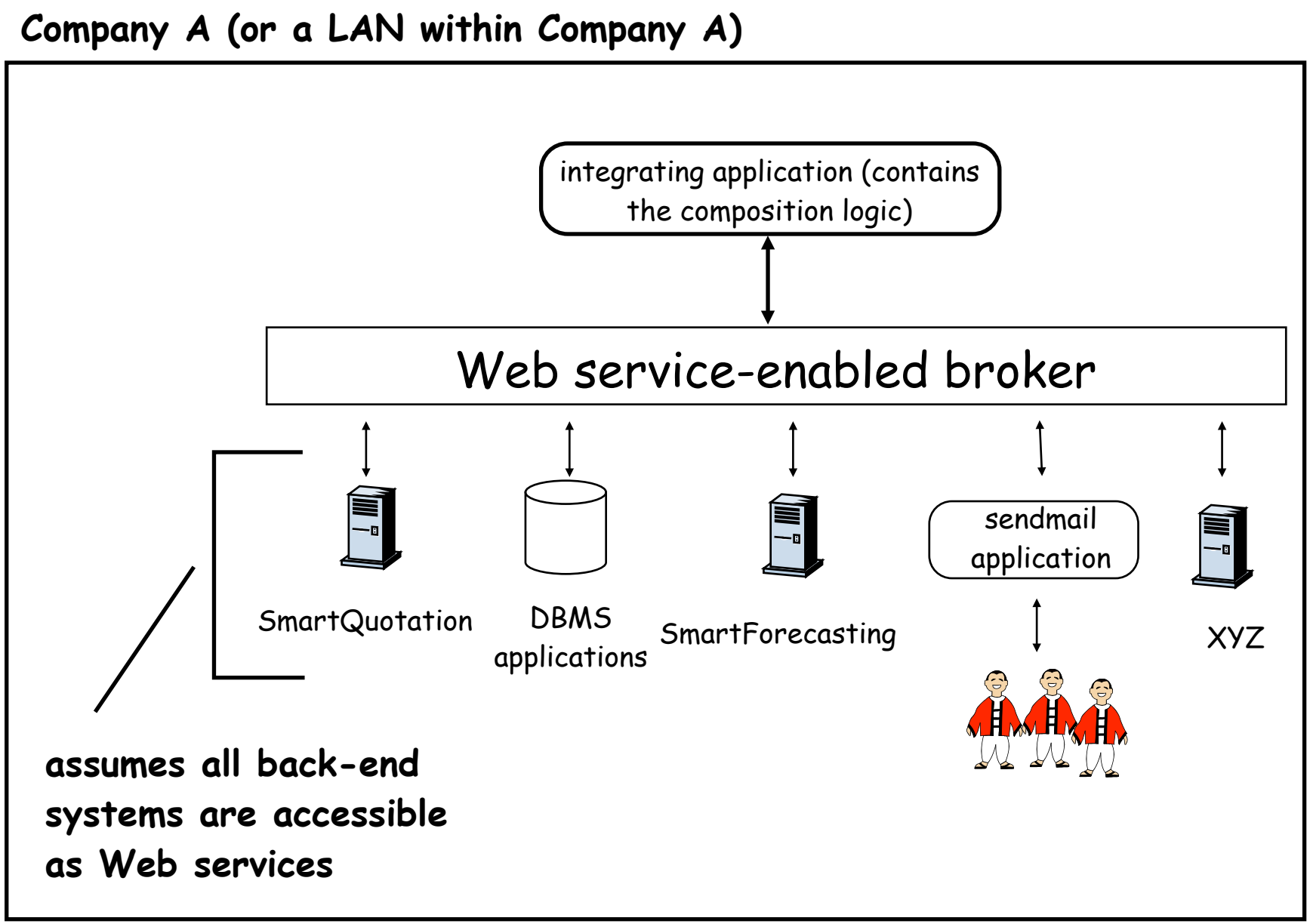
# Web Services Integrating Between Companies



# A Sophisticated Wrapper - Expose Apps on Web



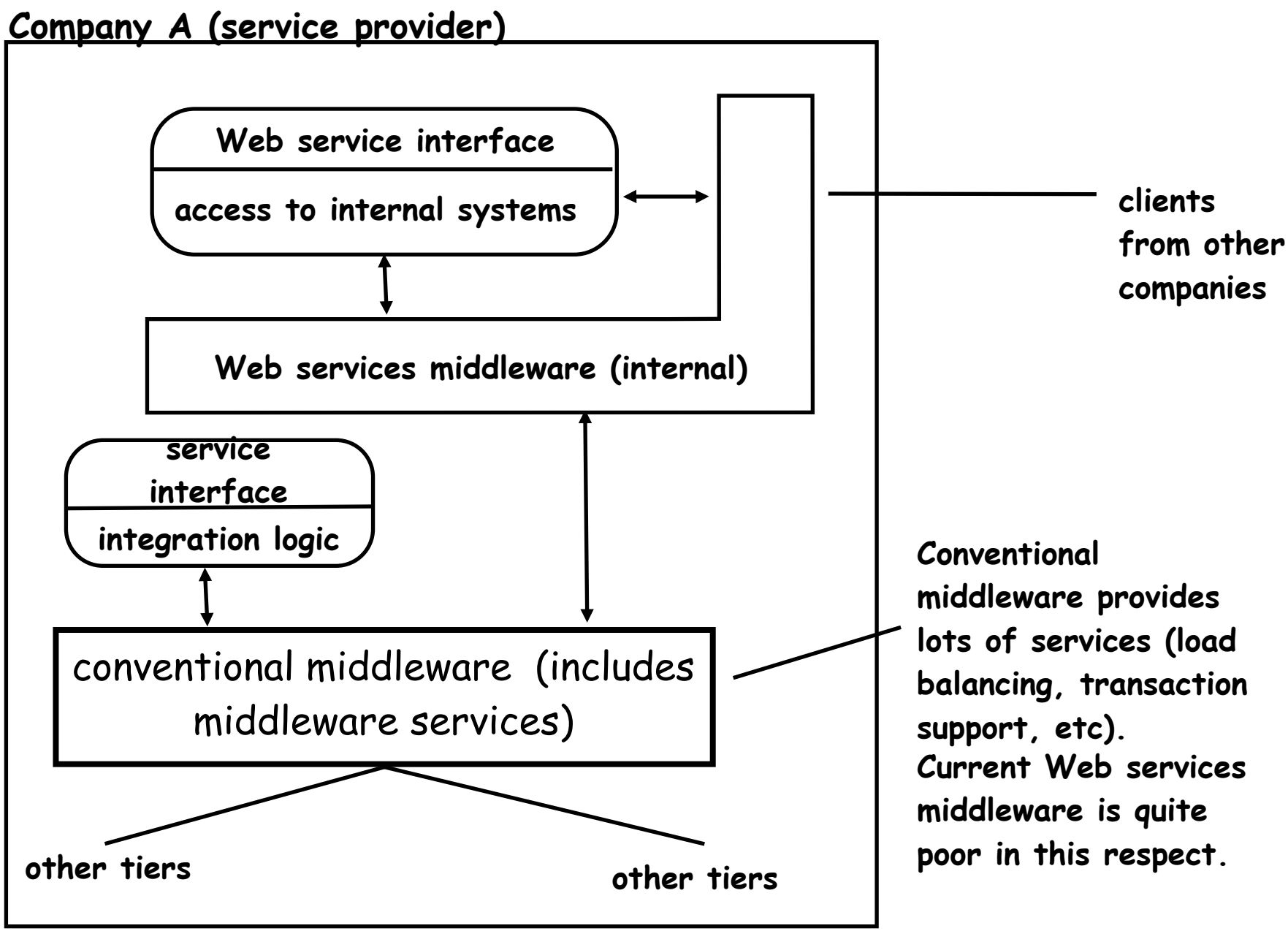
# Can Use WS Protocols Internally



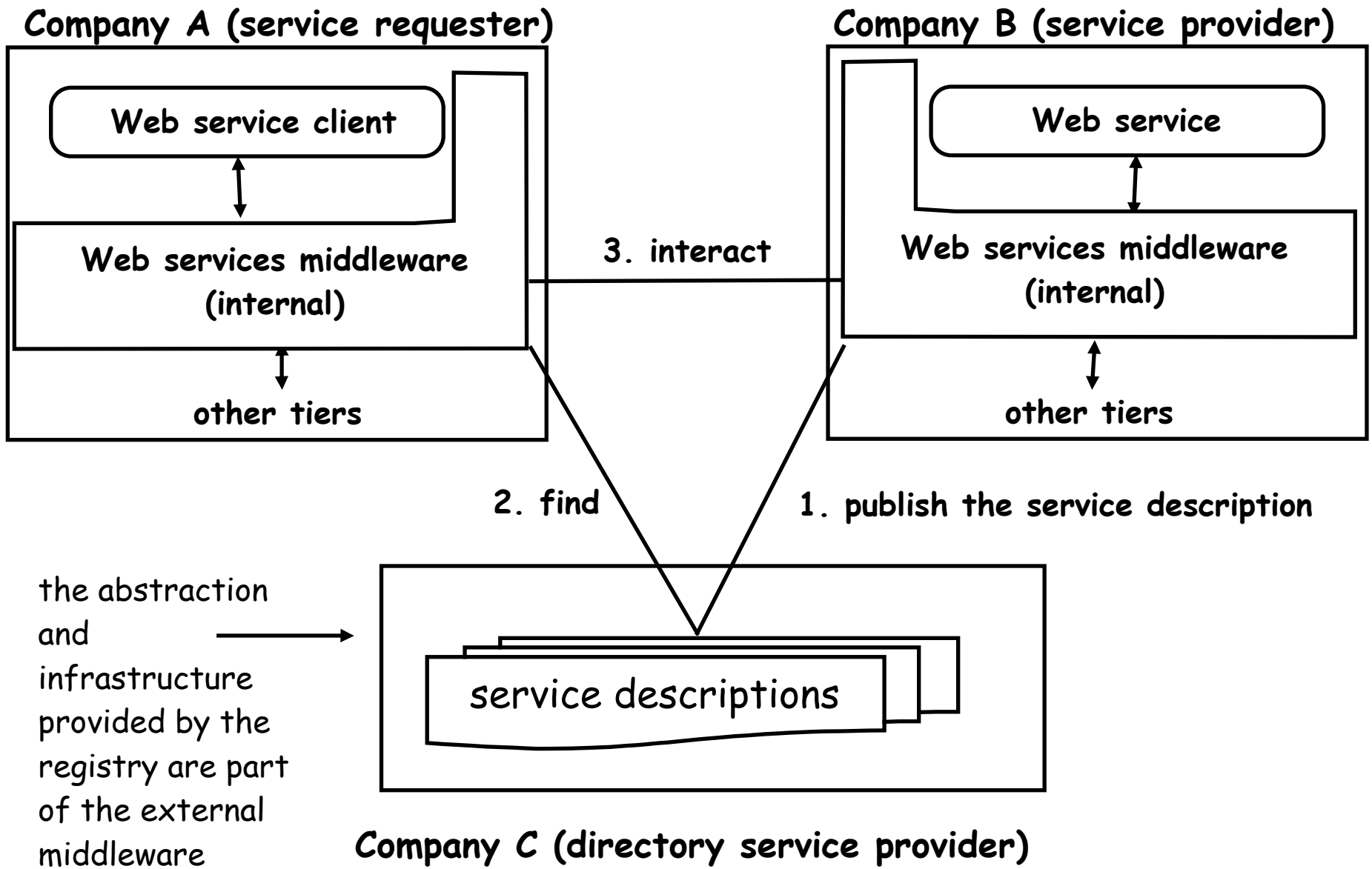
# Two Facets of Web Services Architecture

- Internal
  - run conventional apps
  - expose them as web services
- External
  - global services (like DNS )

# Internal Architecture



# External Architecture

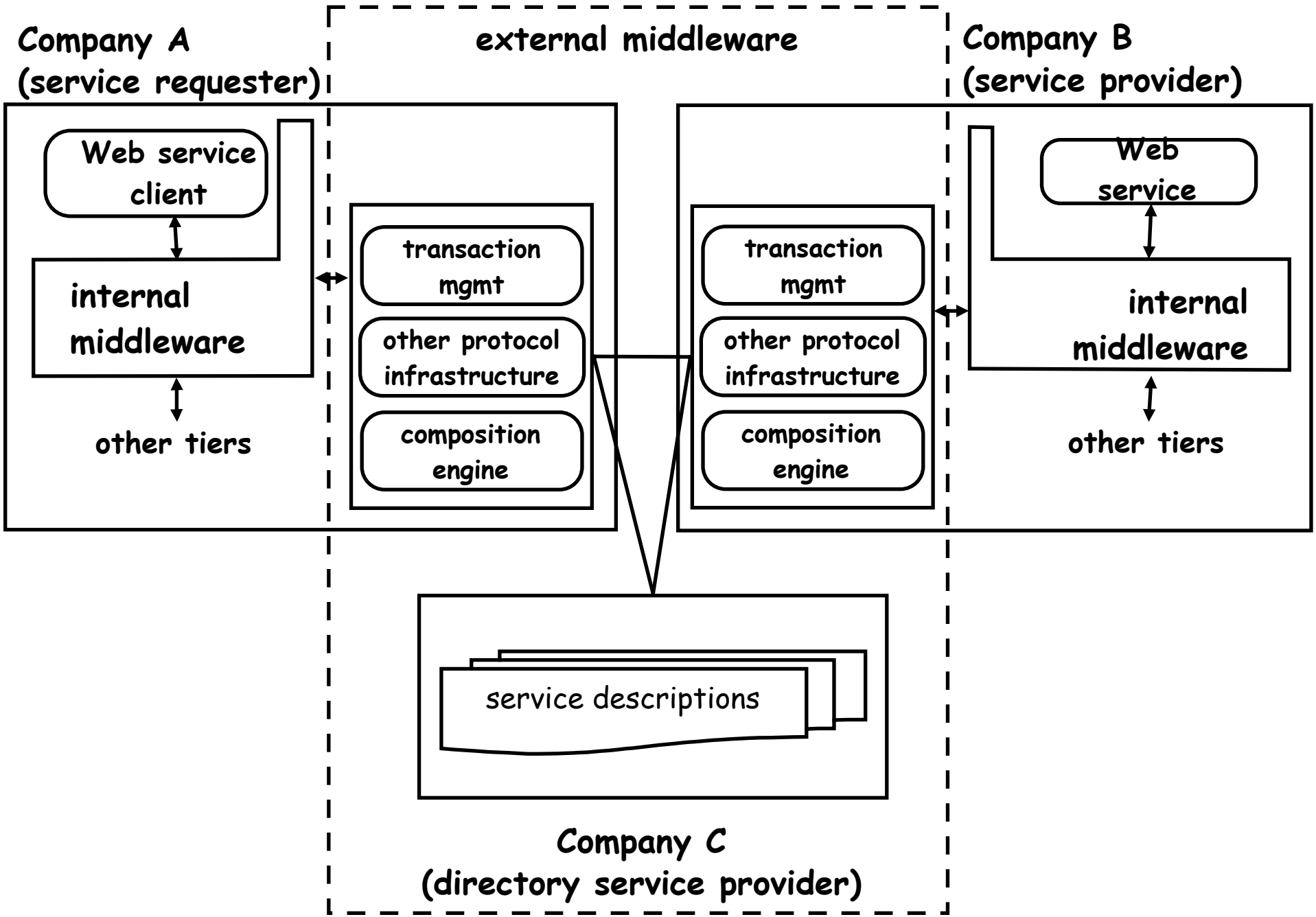



the abstraction and infrastructure provided by the registry are part of the external middleware

# External Architecture

- Is service discovery the only component of Web Services middleware?
- No but external middleware needs to run in peer-to-peer fashion with minimal trust requirements ...

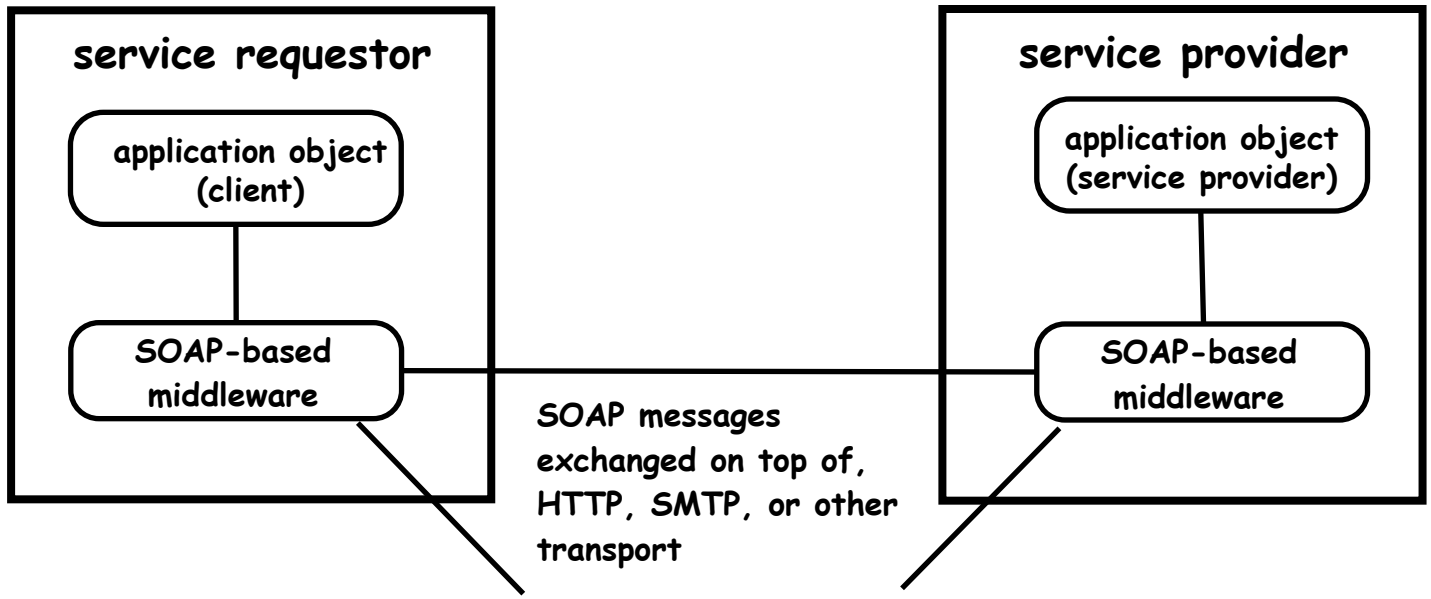
# External Architecture With Middleware



# Ch 6 - Basic Web Services

- Minimalist requirements:
  - communicate (SOAP)
  - describe services - IDL (WSDL)
  - directory service (UDDI)

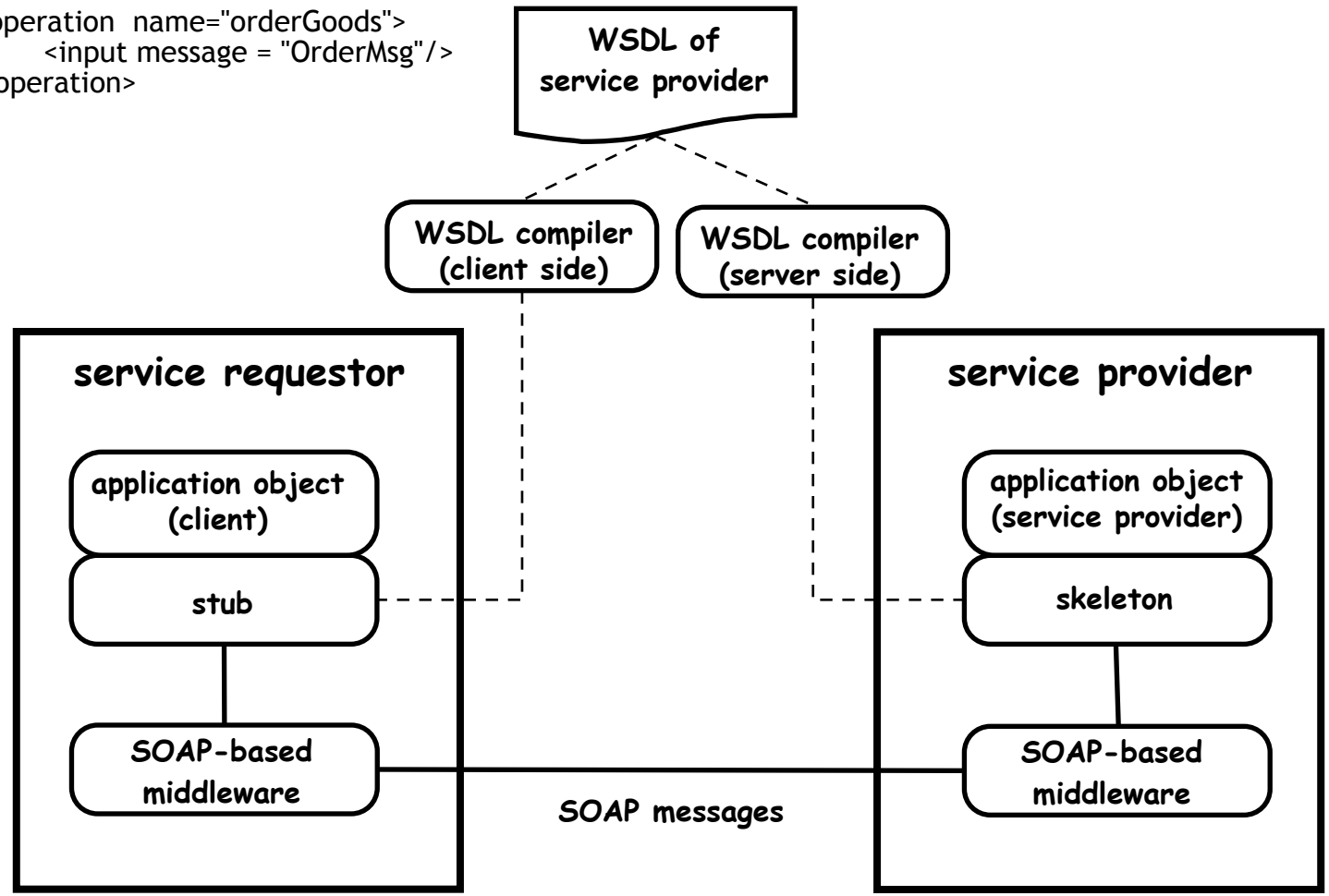
# Minimalist Infrastructure ...




converts procedure calls to/from XML messages sent through HTTP or other protocols.

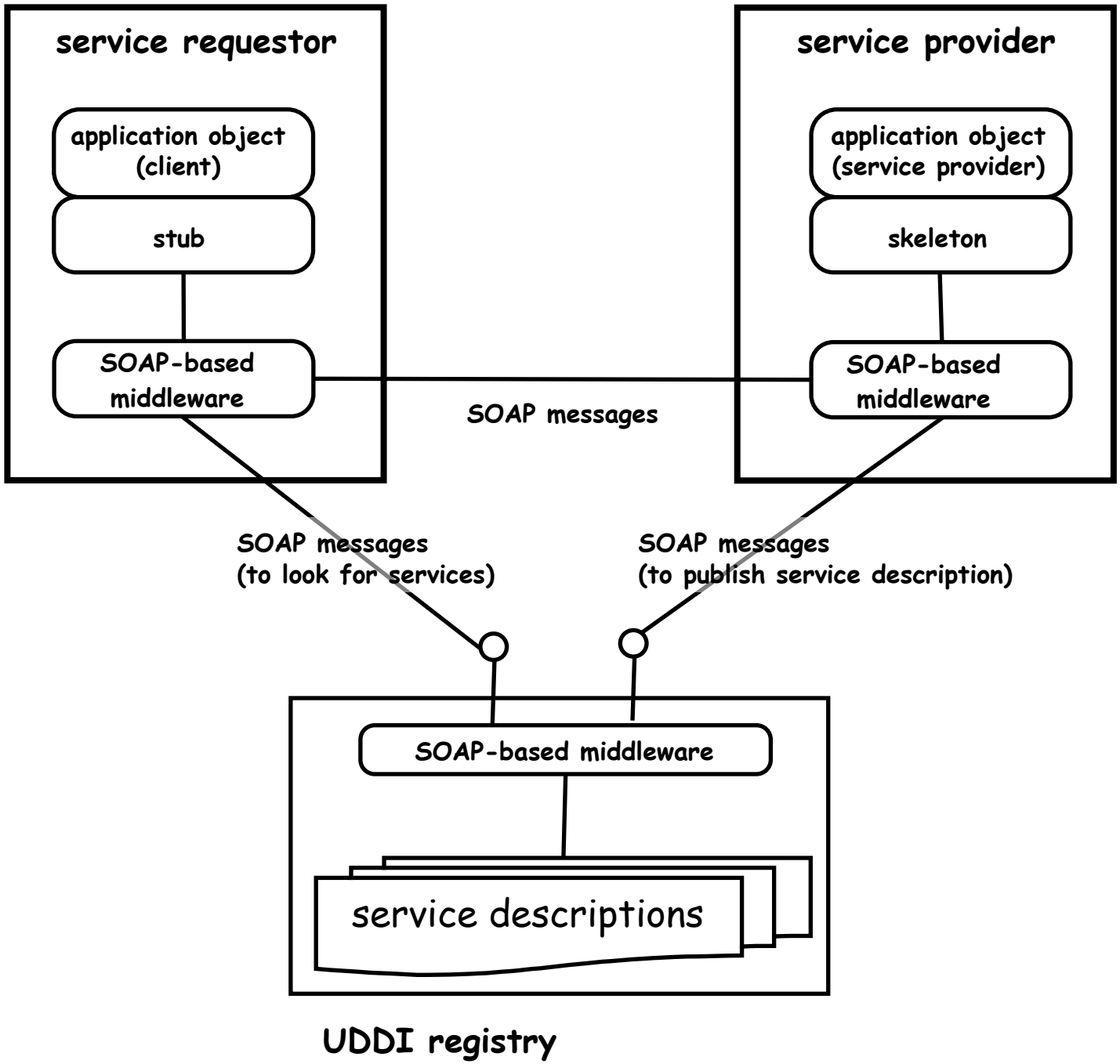
# Using WSDL Specification

```
<operation  name="orderGoods">
      <input message = "OrderMsg"/>
</operation>
```




Note all WSDL "processing" happens at development time.

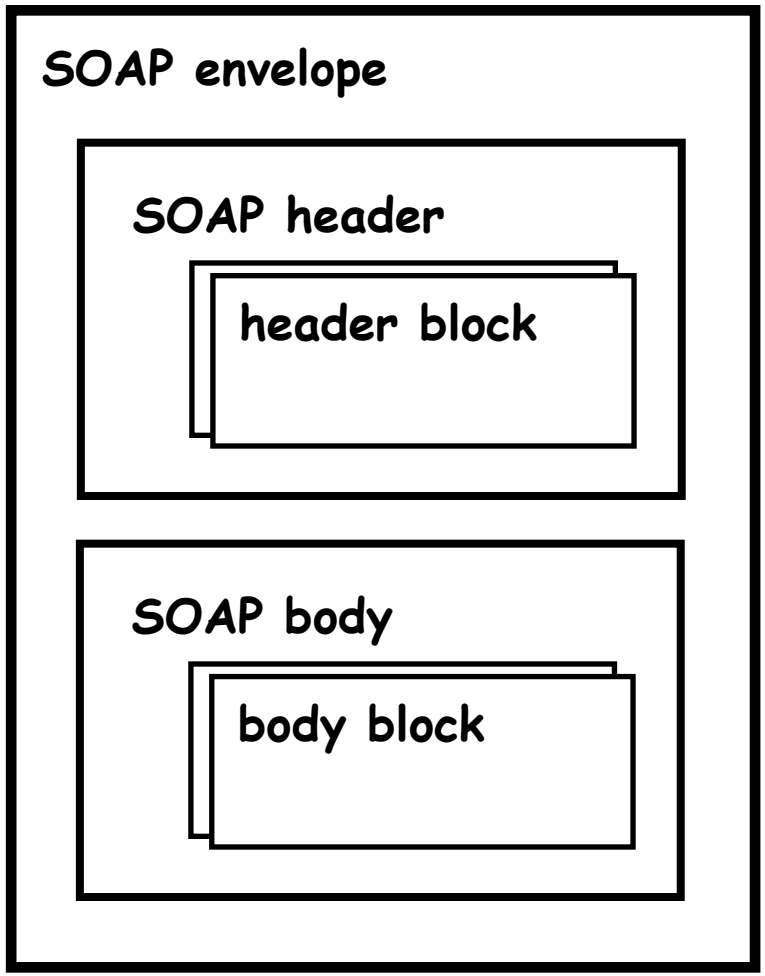
# Using UDDI Registry



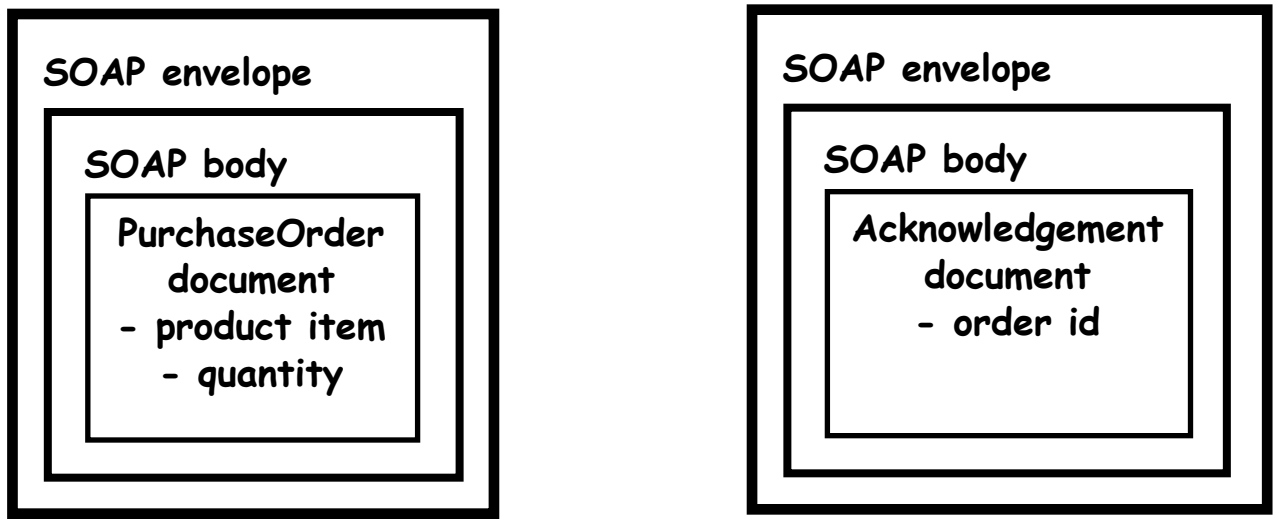
# Simple Object Access Protocol

- Specifies:
  - message format for 1-way comms
  - specification for SOAP RPC
  - rules for processing SOAP messages
  - rules for transport - HTTP and SMTP

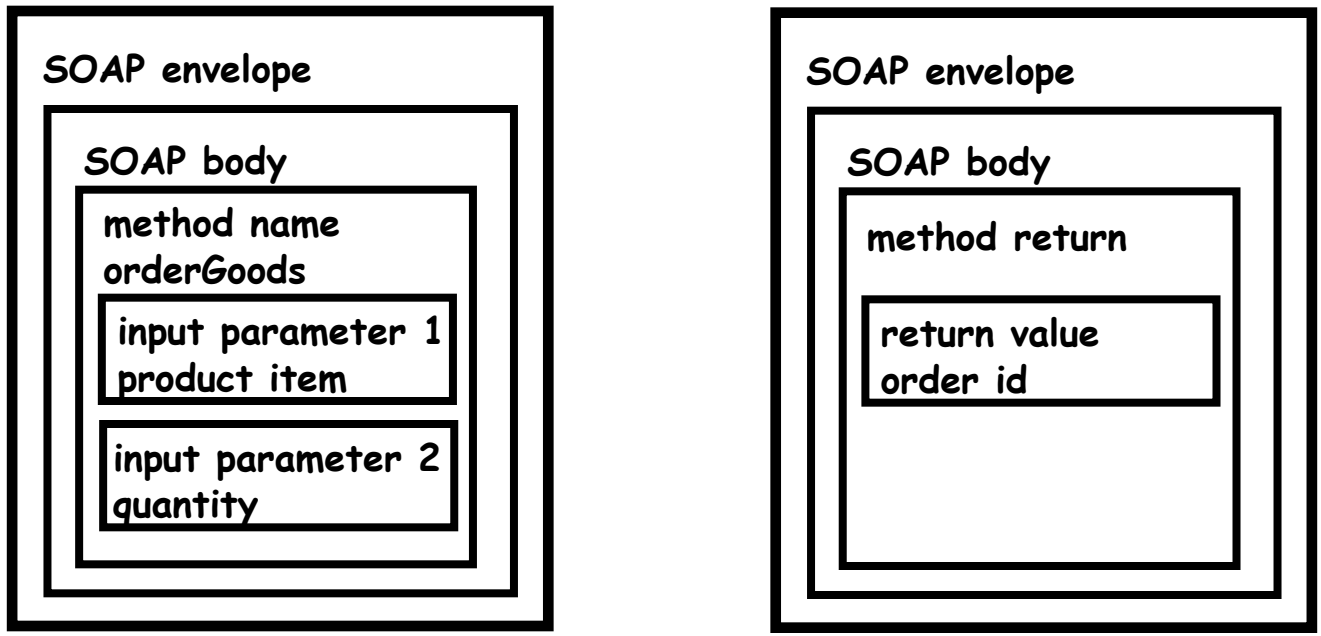
# A SOAP Message



# Document vs RPC




(a) Document-style interaction




(b) RPC-style interaction

# Intermediate Processing - Roles

- Header blocks only ...
- none: no node processes this block
- next: every node processes the block
- ultimateReceiver: only last node in path

# Intermediate Processing - Roles

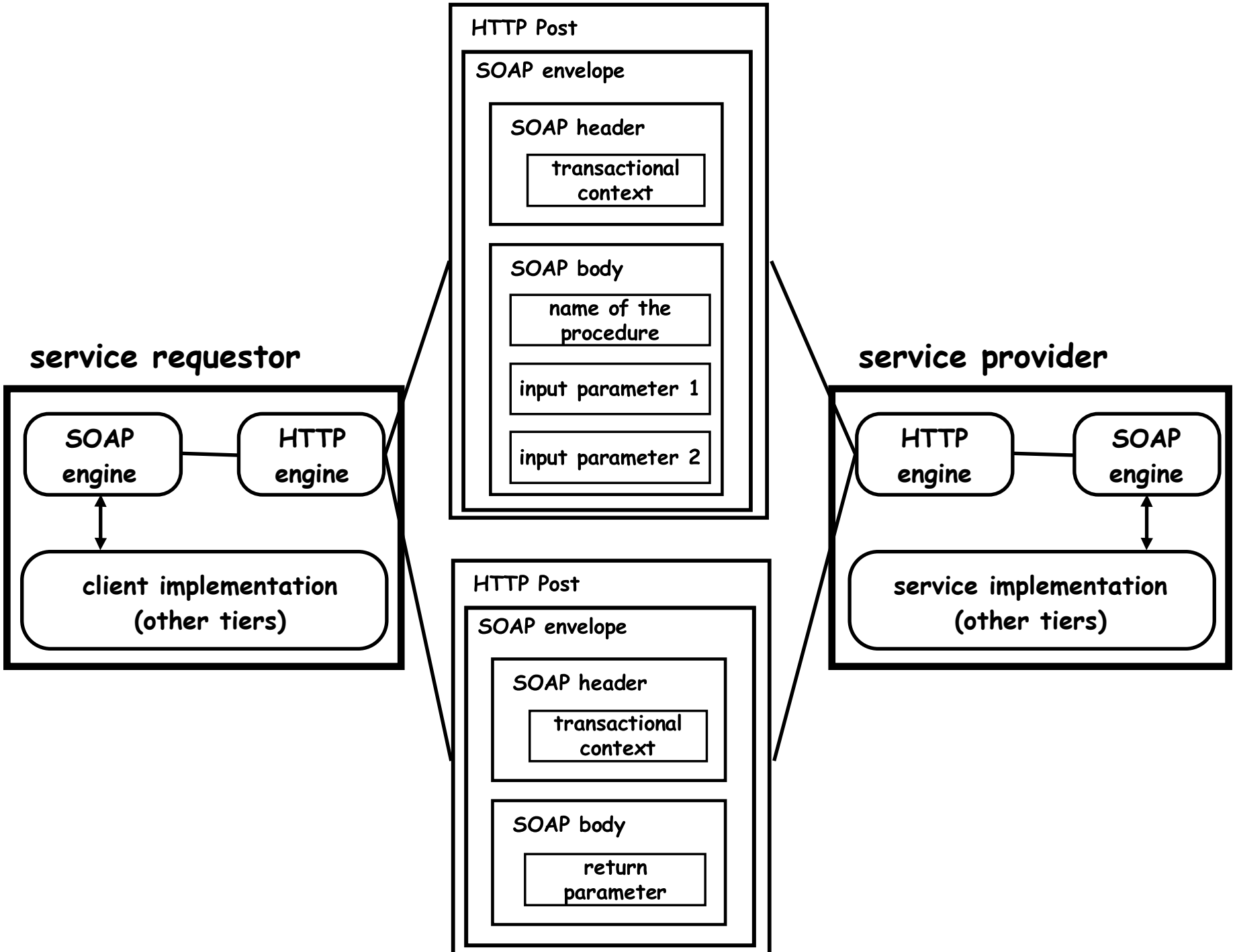

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2002/06/soap-envelope" >
 <env:Header>
   <t:transactionID
            xmlns:t="http://intermediary.example.com/procurement"
            env:role="http://www.w3.org/2002/06/soap-envelope/role/next"
            env:mustUnderstand="true" >
            57539
   </t:transactionID>
 </env:Header>

 <env:Body>
   <m:orderGoods
       env:encodingStyle="http://www.w3.org/2002/06/soap-encoding"
       xmlns:m="http://example.com/procurement">
    <m:productItem>
          <name>ACME Softener</name>
    </m:productItem>
    <m:quantity>
       35
    </m:quantity>
   </m:orderGoods>
 </env:Body>

</env:Envelope>
```

- Note "next" role in header

# SOAP over HTTP



# Simple Implementation