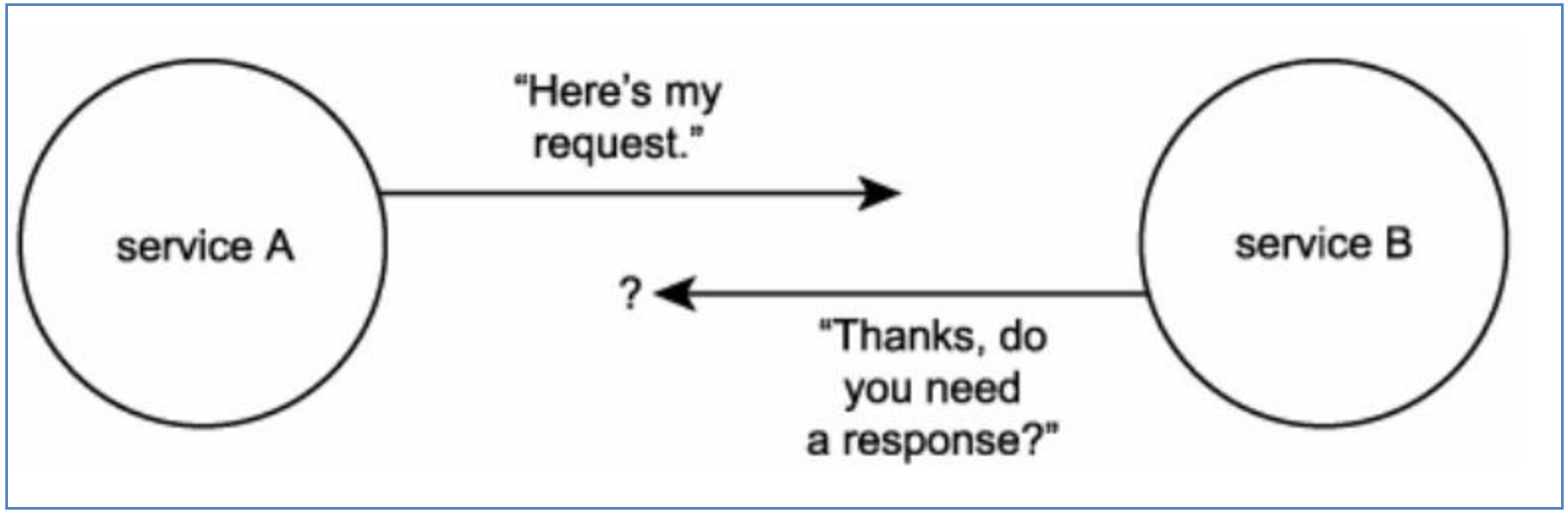
WS-* and Contemporary SOA

Message Exchange Patterns

- Web service tasks are considered different based on the application logic and role played by the service.
  - Tasks are executed by the transmission of multiple messages
    - Coordination is important

- Do all message exchanges require both requests and responses?

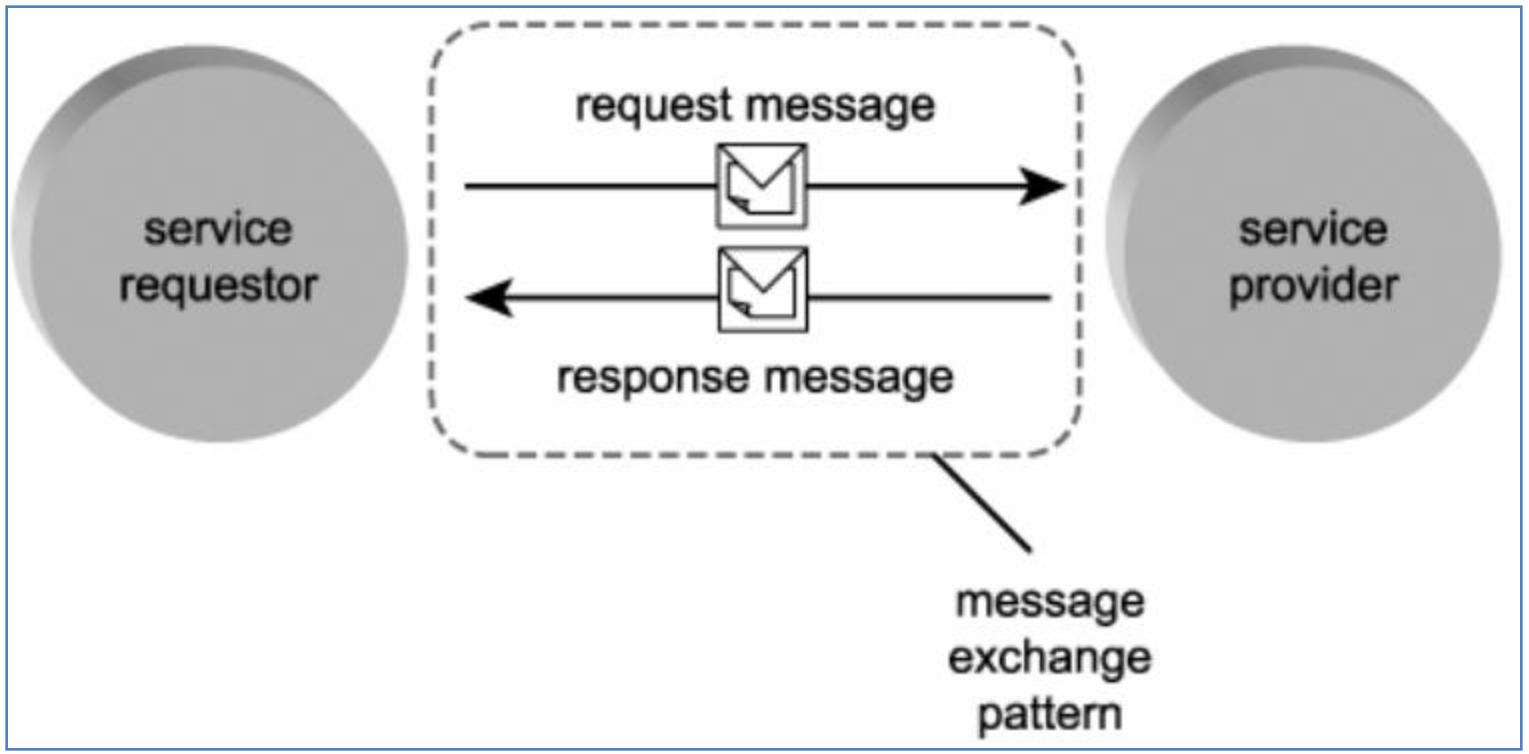


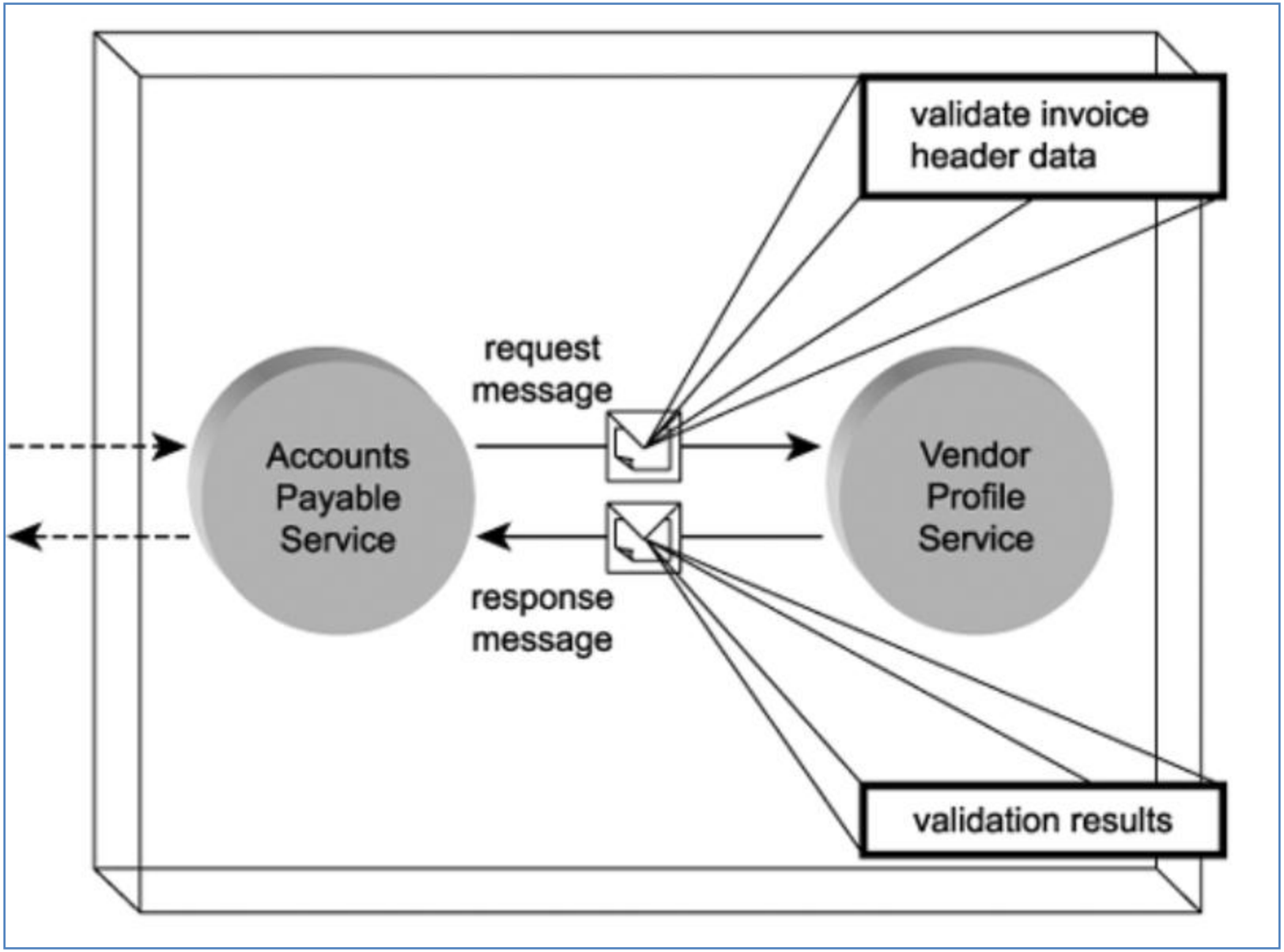
"Message exchange patterns (MEPs) represent a set of templates that provide a group of already mapped out sequences for the exchange of messages."

1. Primitive MEPs
2. Complex MEPs

Primitive MEPs

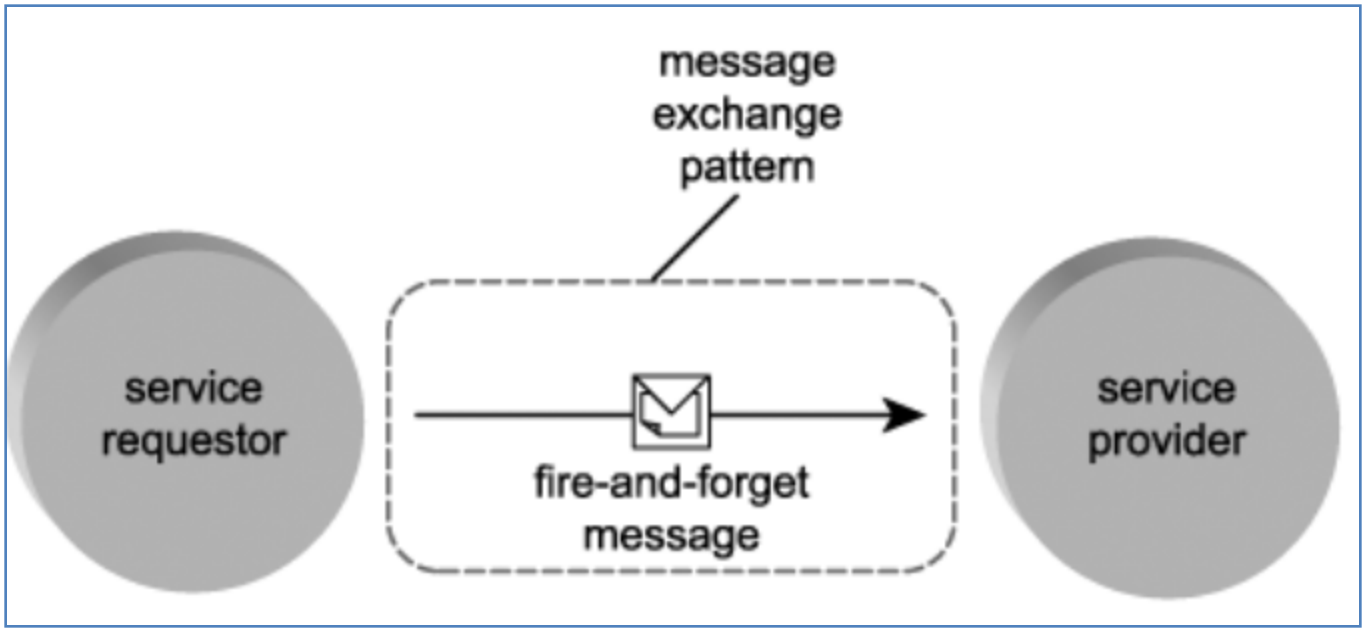
- **Request-response**
  - The one pattern that defines synchronous communication




validate invoice
header data
request
message
Accounts
Payable
Service
Vendor
Profile
Service
response
message
validation results

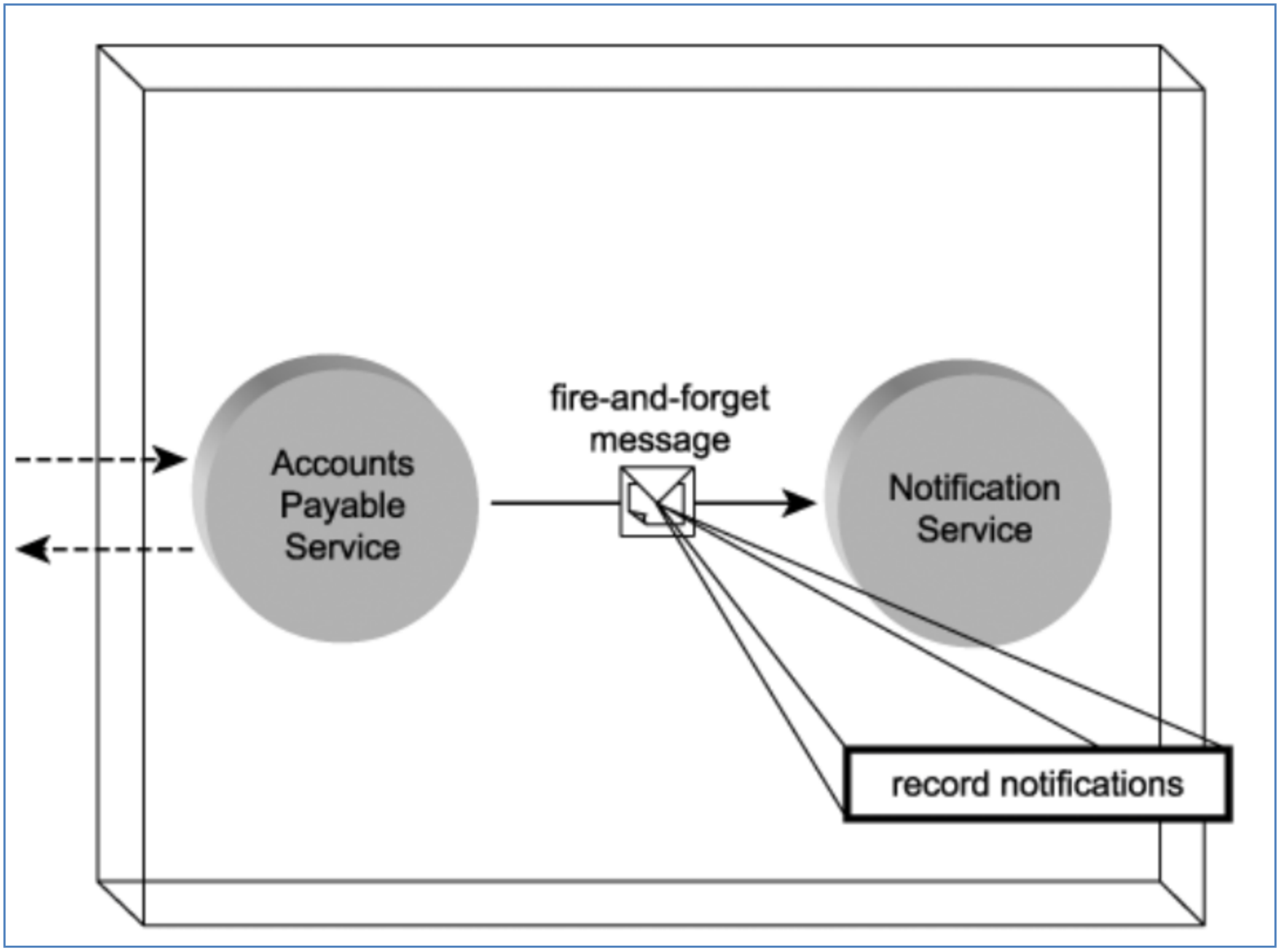
Primitive MEPs

- **Fire-and-forget**
  - This simple asynchronous pattern is based on the unidirectional transmission of messages from a source to one or more destinations



Primitive MEPs

- **Fire-and-forget Variations:**
  - The single-destination pattern, where a source sends a message to one destination only.
  - The multi-cast pattern, where a source sends messages to a predefined set of destinations.
  - The broadcast pattern, which is similar to the multi-cast pattern, except that the message is sent out to a broader range of recipient destinations.


Accounts
Payable
Service
fire-and-forget
message
Notification
Service
record notifications

Complex MEPs

- Primitive MEPs can be assembled in various configurations to create different types of messaging models, sometimes called complex MEPs.

- A classic example is the publish-and-subscribe model.

Complex MEPs

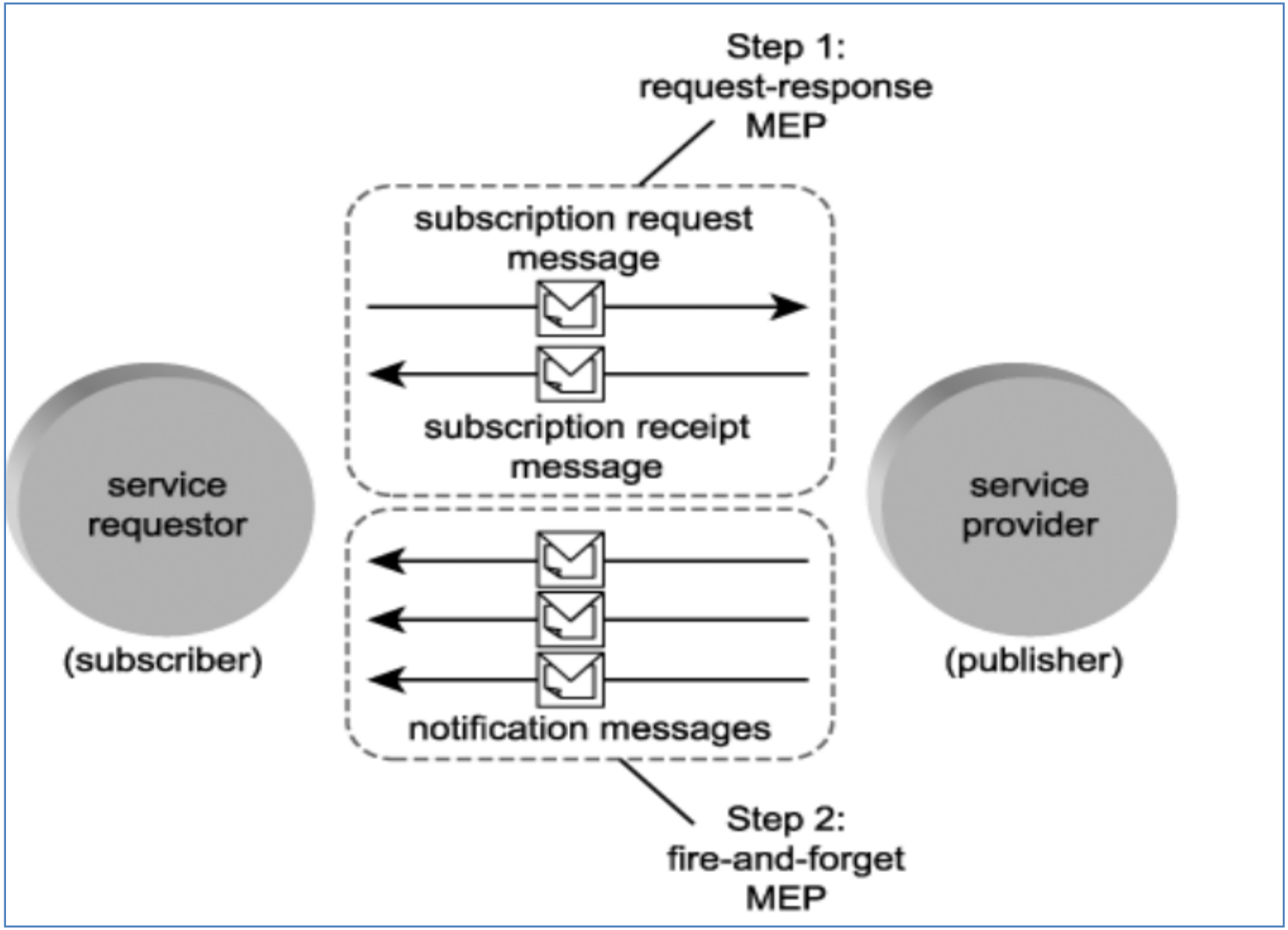
- **The publish-and-subscribe pattern**
  - It introduces new roles for the services involved with the message exchange.
    - Publishers and subscribers
  - This asynchronous MEP accommodates a requirement for a publisher to make its messages available to a number of subscribers interested in receiving them.

Complex MEPs

- **The publish-and-subscribe pattern**
  - **Step 1.**

    The subscriber sends a message to notify the publisher that it wants to receive messages on a particular topic.
  - **Step 2.**

    Upon the availability of the requested information, the publisher broadcasts messages on the particular topic to all of that topic's subscribers.


Step 1:
request-response
MEP
subscription request
message
subscription receipt
message
service
requestor
(subscriber)
service
provider
(publisher)
notification messages
Step 2:
fire-and-forget
MEP

MEPs and SOAP

- The SOAP allows countless messaging characteristics and behaviors to be implemented via SOAP header blocks.
- Hence, SOAP provides framework for Message Exchange

MEPs and WSDL

- Operations defined within service descriptions are comprised, in part, of message definitions.
- The exchange of these messages constitutes the execution of a task represented by an operation.
- MEPs play a larger role in WSDL service descriptions as they can coordinate the input and output messages associated with an operation.
- The association of MEPs to WSDL operations thereby embeds expected conversational behavior into the interface definition.

MEPs and WSDL

- WSDL operations support different configurations of incoming, outgoing, and fault messages.
- These configurations are equivalent to message exchange patterns, but within the WSDL specification, they often are referred to simply as patterns.

WSDL 1.1 MEPs

- Release 1.1 of the WSDL specification provides support for four message exchange patterns

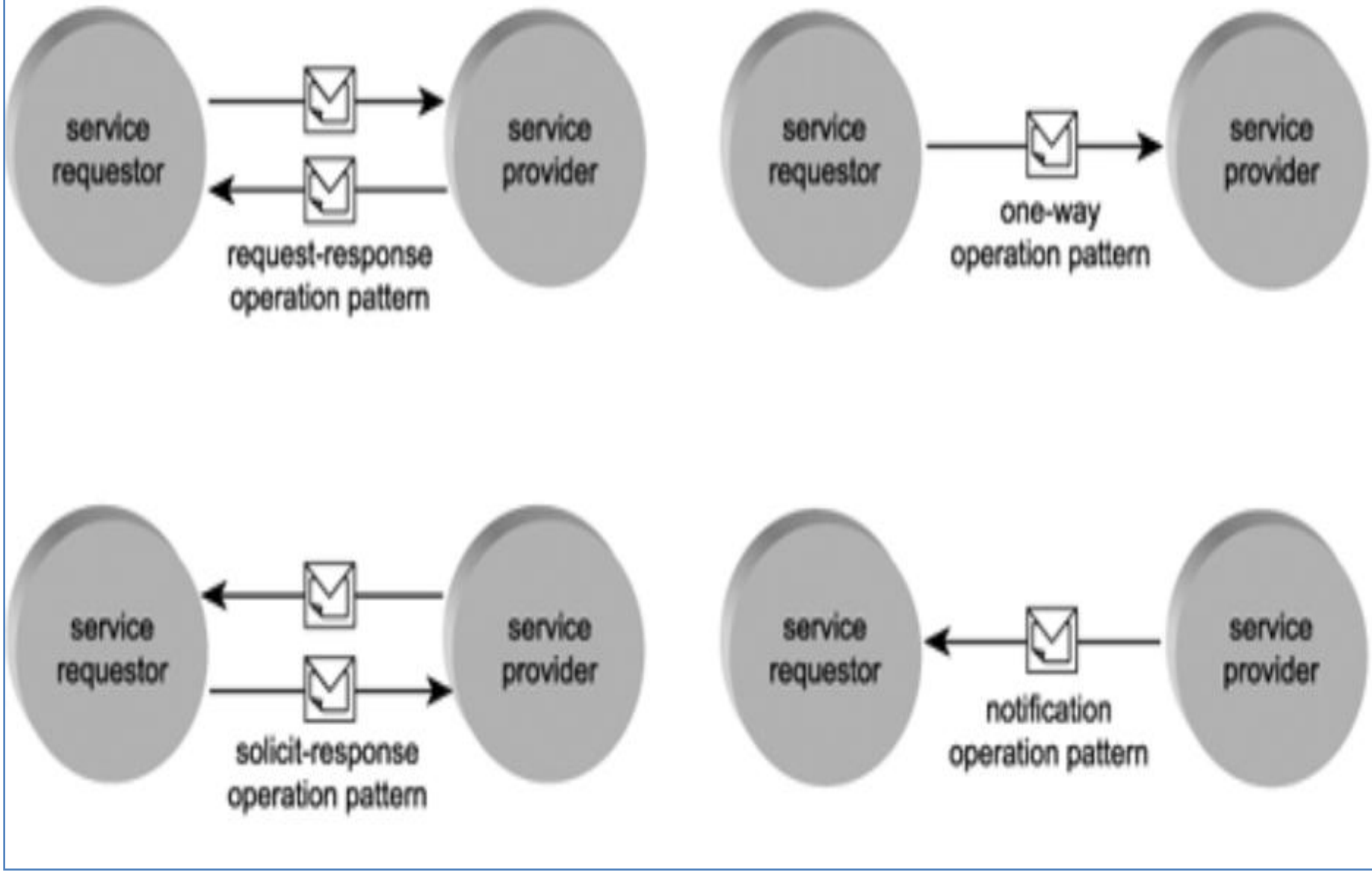
1. Request-response operation: Upon receiving a message, the service must respond with a standard message or a fault message.

2. Solicit-response operation: Upon submitting a message to a service requestor, the service expects a standard response message or a fault message.

WSDL 1.1 MEPs

3. One-way operation: The service expects a single message and is not obligated to respond.

4. Notification operation: The service sends a message and expects no response.


service requestor
service provider
request-response operation pattern
service requestor
service provider
one-way operation pattern
service requestor
service provider
solicit-response operation pattern
service requestor
service provider
notification operation pattern

WSDL 2.0 MEPs

- Release 2.0 of the WSDL specification extends MEP support to eight patterns.

1. The in-out pattern, comparable to the request-response MEP (equivalent to the WSDL 1.1 request-response operation).
2. The out-in pattern, which is the reverse of the previous pattern where the service provider initiates the exchange by transmitting the request. (Equivalent to the WSDL 1.1 solicit-response operation.)

MEPs and WSDL

3. The in-only pattern, which essentially supports the standard fire-and-forget MEP. (Equivalent to the WSDL 1.1 one-way operation.)

4. The out-only pattern, which is the reverse of the in-only pattern. It is used primarily in support of event notification. (Equivalent to the WSDL 1.1 notification operation.)

MEPs and WSDL

5. The robust in-only pattern, a variation of the in-only pattern that provides the option of launching a fault response message as a result of a transmission or processing error.

6. The robust out-only pattern, which, like the out-only pattern, has an outbound message initiating the transmission. The difference here is that a fault message can be issued in response to the receipt of this message.

MEPs and WSDL

7. The in-optional-out pattern, which is similar to the in-out pattern with one exception. This variation introduces a rule stating that the delivery of a response message is optional. This pattern also supports the generation of a fault message.
8. The out-optional-in pattern is the reverse of the in-optional-out pattern, where the incoming message is optional. Fault message generation is again supported.

MEPs and SOA

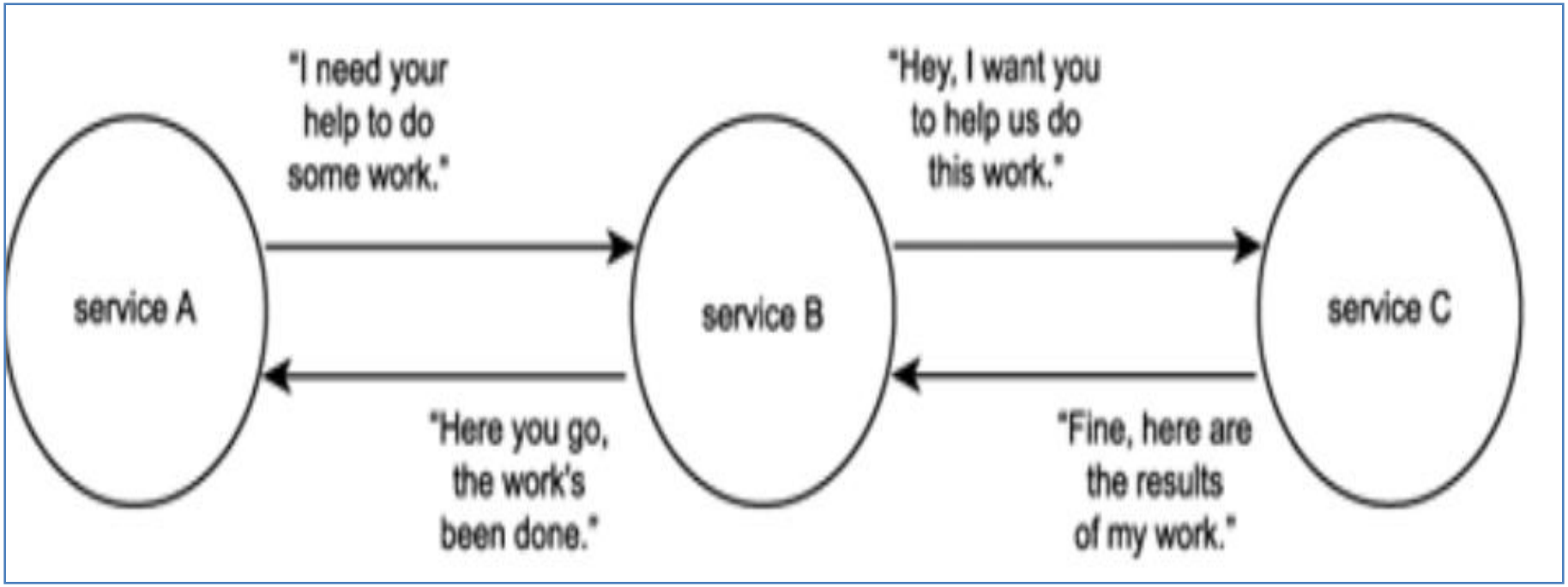
- MEPs are highly generic and abstract in nature.
- Individually, they simply relate to an interaction between two services.
- They are therefore a fundamental and essential part of any Web services-based environment, including SOA.

So far, MEPs

- An MEP is a generic interaction pattern that defines the message exchange between two services.
- MEPs have been around for as long as messaging-based middleware products have been used.
  - As a result, some common patterns have emerged.
- MEPs can be composed to support the creation of larger, more complex patterns.
- The WSDL and SOAP specifications support specific variations of common MEPs.

Service activity

- "The interaction of a group of services working together to complete a task can be referred to as a **service activity**."



Examples

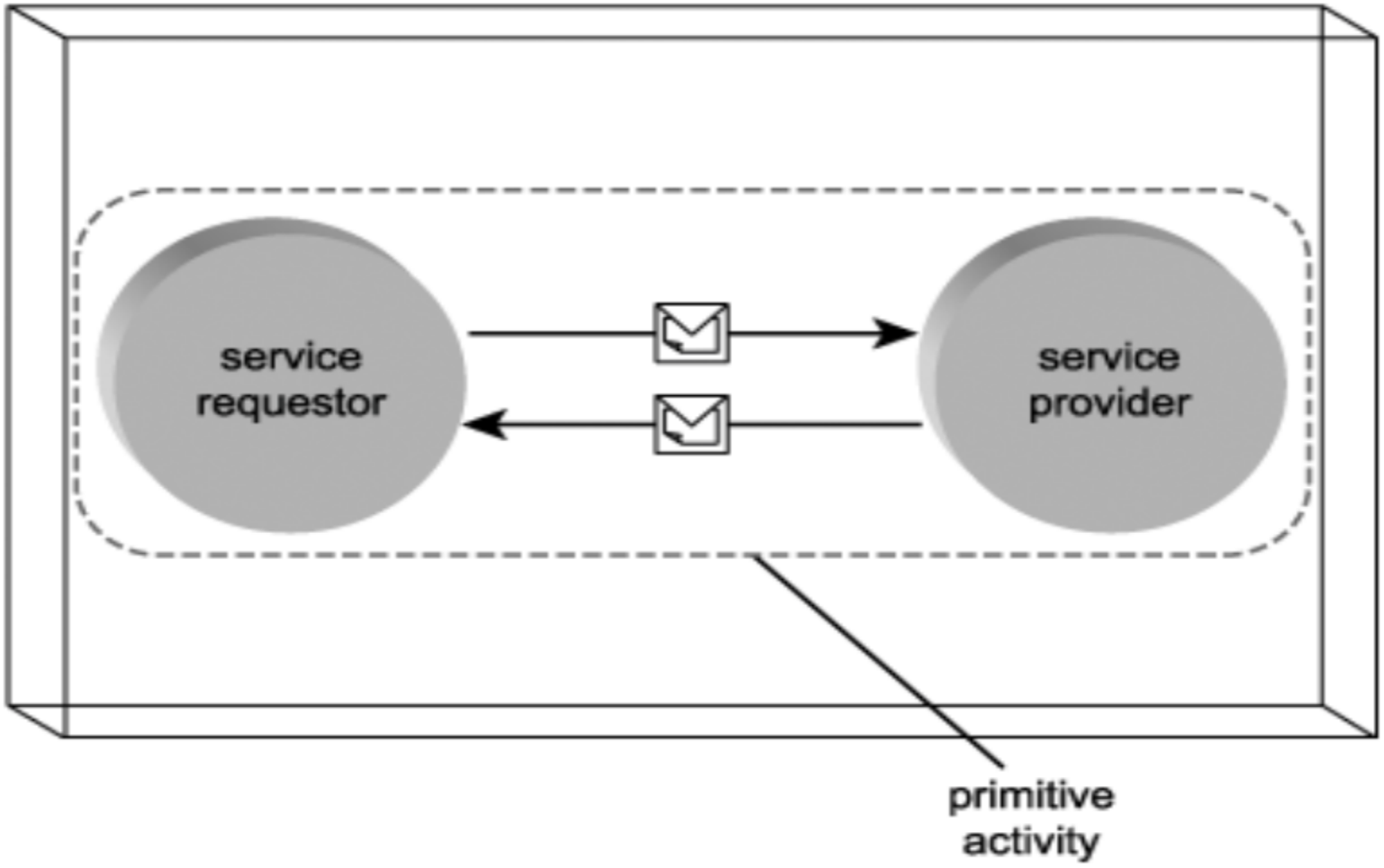
- Turning on TV
  - **1.**Pick up the remote control. **2.**Press the "Power" button.
- Washing a car
  - **1.**Locate bucket. **2.**Locate sponge. **3.**Locate hose. **4.**Fill bucket with warm water. **5.**Add soap to water. **6.**Soak sponge in water. **7.**Rub sponge on car….. OR
  - **1.**Gather required equipment. **2.**Prepare water. **3.**Wash car.

Primitive service activities

- A **simple or primitive activity** is typified by synchronous communication and therefore often consists of two services exchanging information using a standard request-response MEP

- Primitive activities are almost always short-lived; the execution of a single MEP generally constitutes the lifespan of a primitive activity.

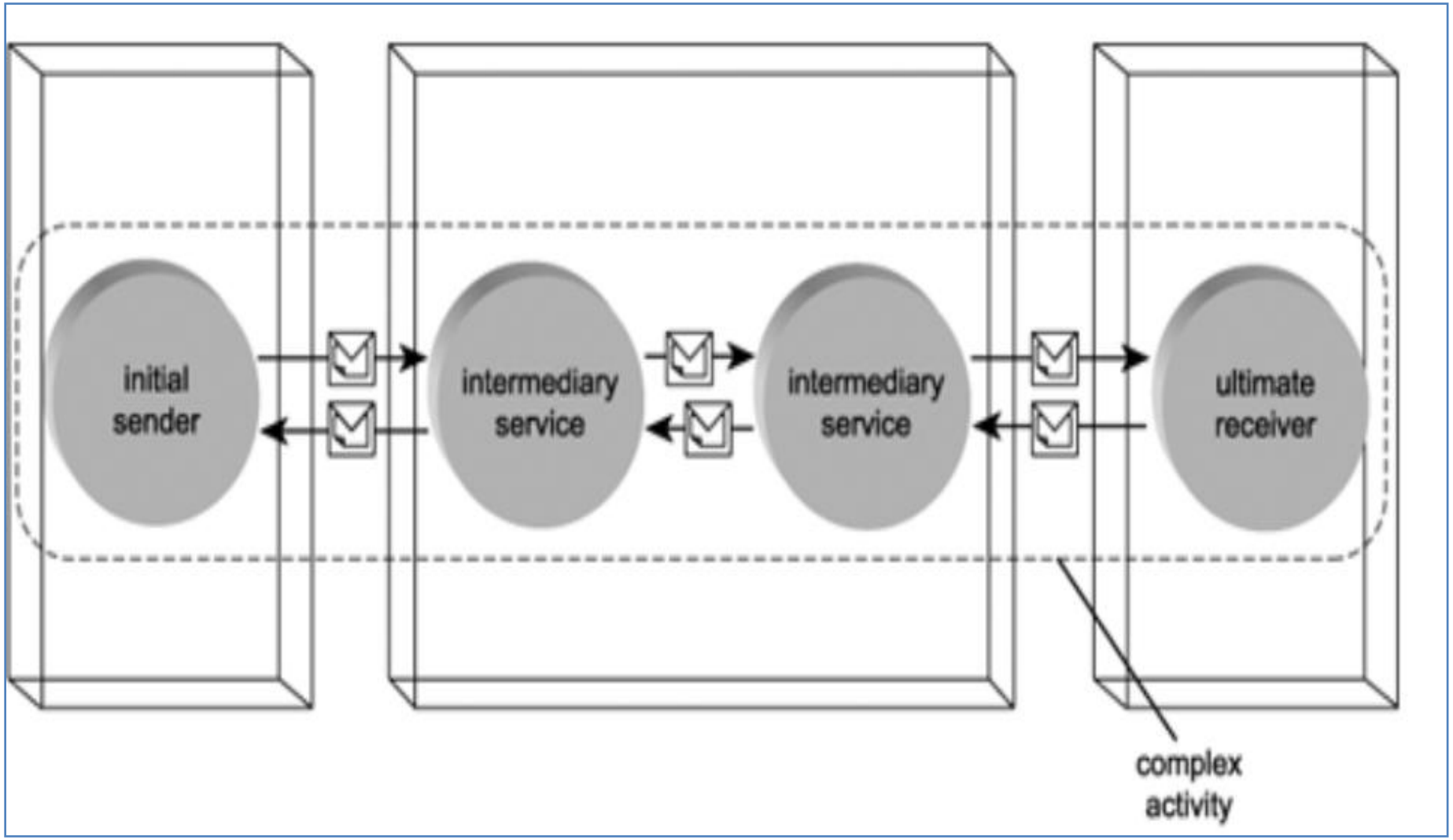
Primitive service activities

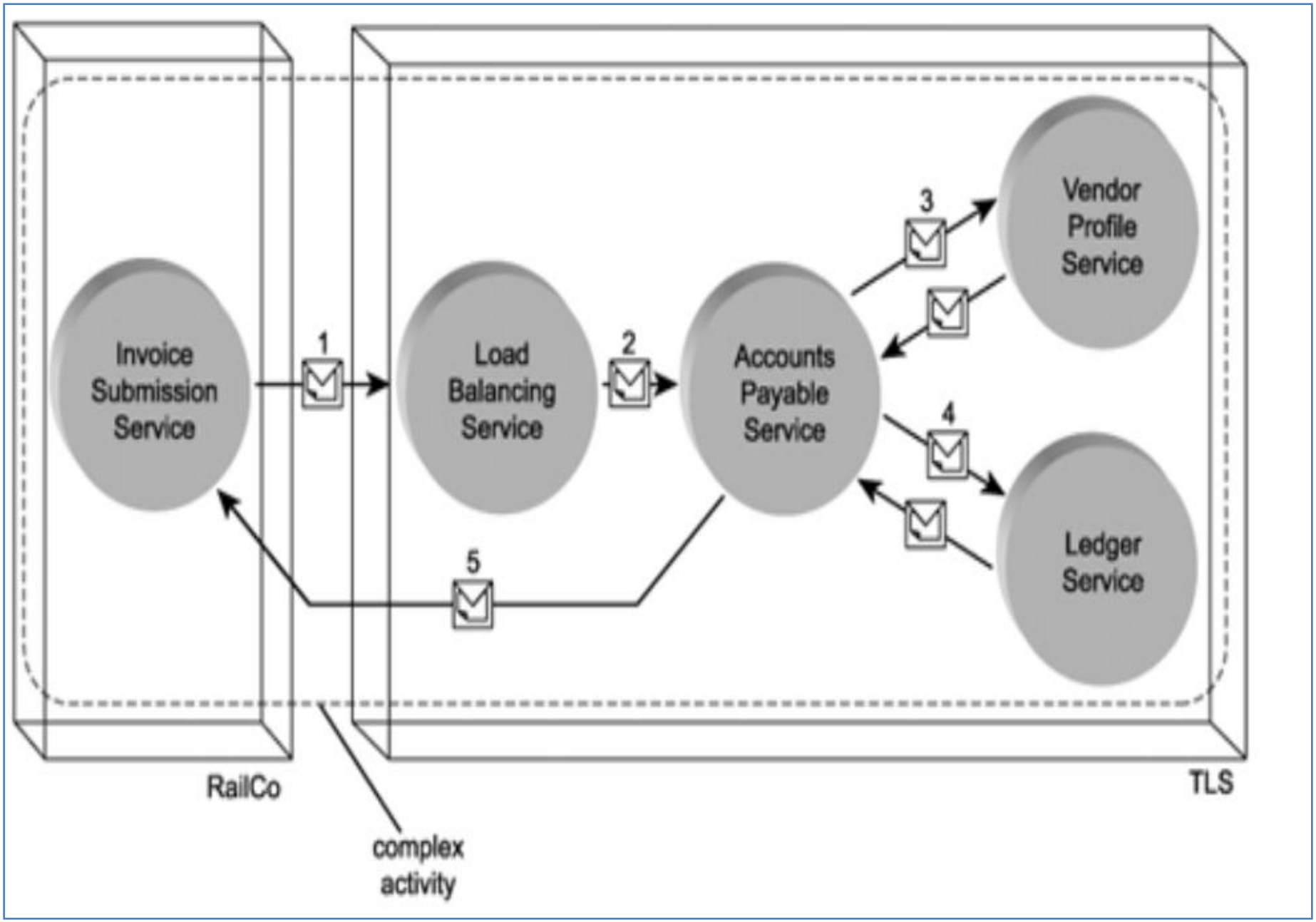


Complex service activities

- **Complex activities**, on the other hand, can involve many services (and MEPs) that collaborate to complete multiple processing steps over a long period of time.
- These more elaborate types of activities are generally structured around extension-driven and composition-oriented concepts, such as choreography and orchestration.

Complex service activities




Invoice Submission Service
1
Load Balancing Service
2
Accounts Payable Service
3
Vendor Profile Service
4
Ledger Service
5
RailCo
TLS
complex activity

So far, service activity

- An activity is a generic concept used to represent a task or a unit of work performed by a set of services.
- The scope of primitive activities can be limited to the completion of simple MEPs.
- Complex activities are common within SOAs and exist as part of any non-trivial service-oriented application.

Coordination

- Every activity introduces a level of context into an application runtime environment.

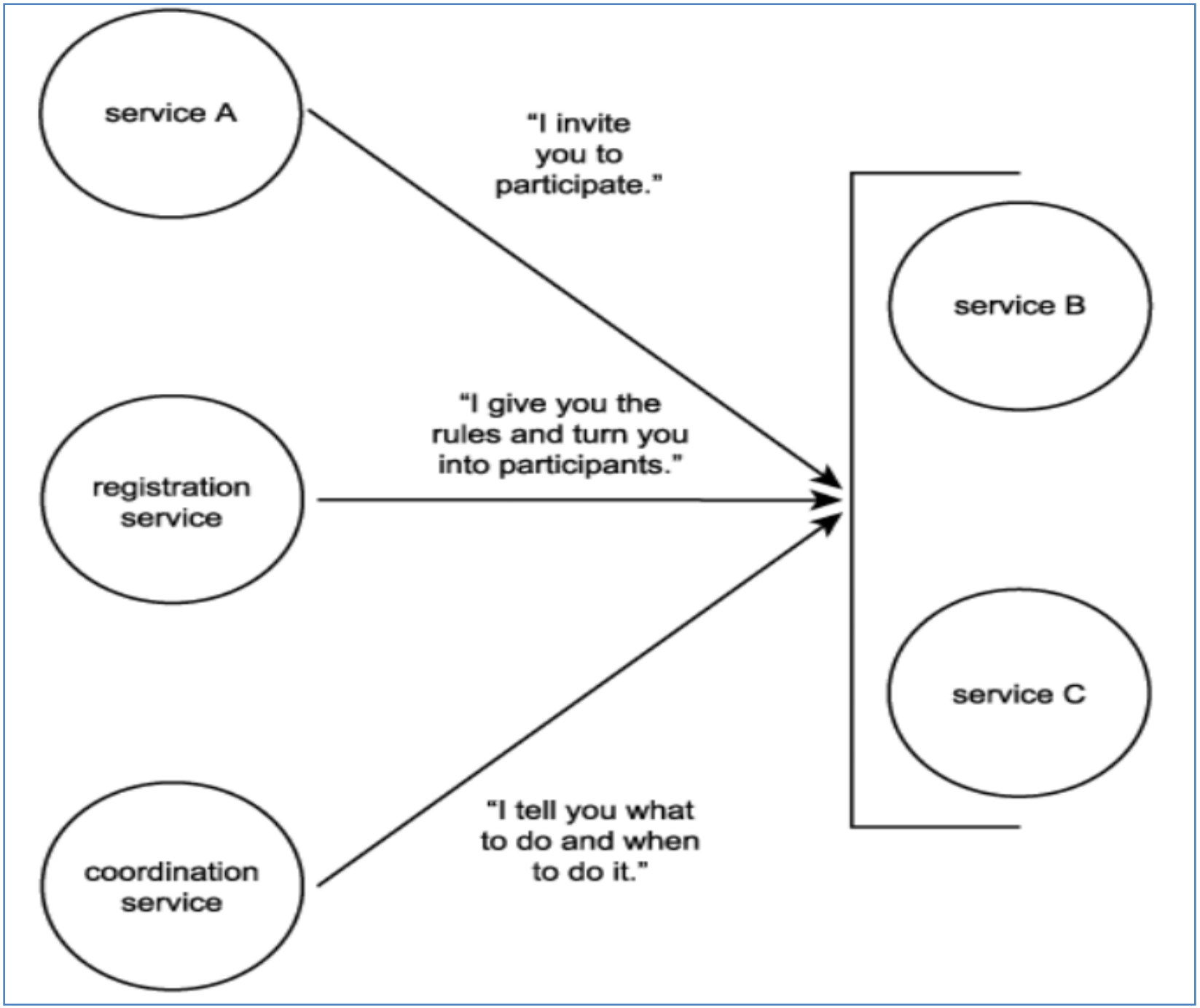
- *"The more complex an activity, the more context information it tends to bring with it."*

Coordination

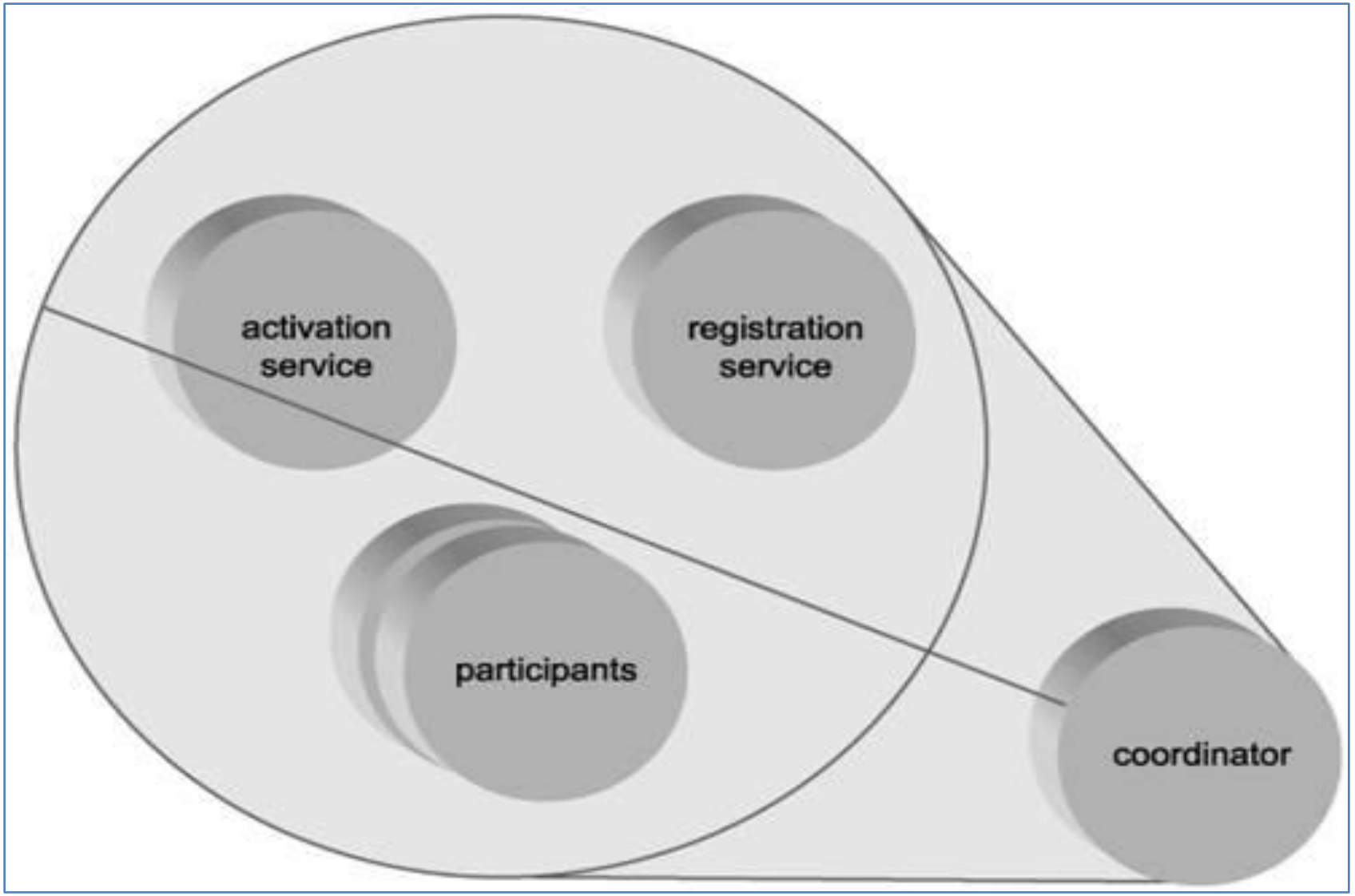
- The complexity of an activity can relate to a number of factors, including:
  - the amount of services that participate in the activity
  - the duration of the activity
  - the frequency with which the nature of the activity changes
  - whether or not multiple instances of the activity can concurrently exist

Coordination

- "Coordination is a framework
  - to provide a means for context information in complex activities
  - to be managed, preserved and/or updated, and distributed to activity participants."


service A
"I invite you to participate."
service B
registration service
"I give you the rules and turn you into participants."
service C
coordination service
"I tell you what to do and when to do it."

Coordinator composition



Coordinator composition

- The coordinator composition consists of the following services:
  - Activation service responsible for the creation of a new context and for associating this context to a particular activity.
  - Registration service allows participating services to use context information received from the activation service to register for a supported context protocol.
  - Coordinator is the controller service of this composition, also known as the coordination service.

Coordination types and coordination protocols

- Each coordinator is based on a coordination type, which specifies the nature and underlying logic of an activity for which context information is being managed.
  - WS-AtomicTransaction and WS-BusinessActivity.

- Coordination type extensions provide a set of coordination protocols.

Coordination contexts

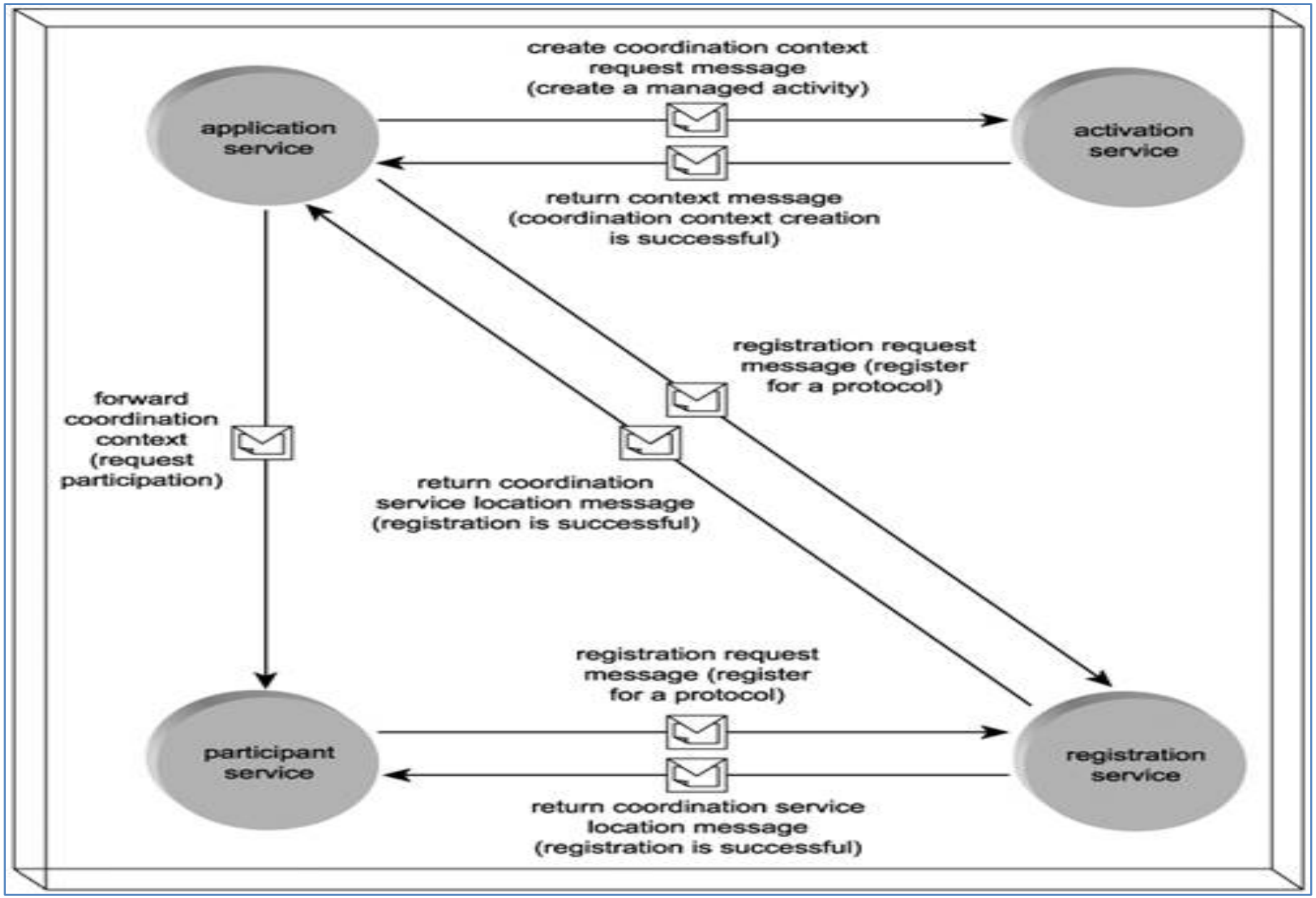
- “A context created by the activation service is referred to as a coordination context.”

- Examples of the type of data held within a coordination context include:
  - a unique identifier that represents the activity
  - an expiration value
  - coordination type information

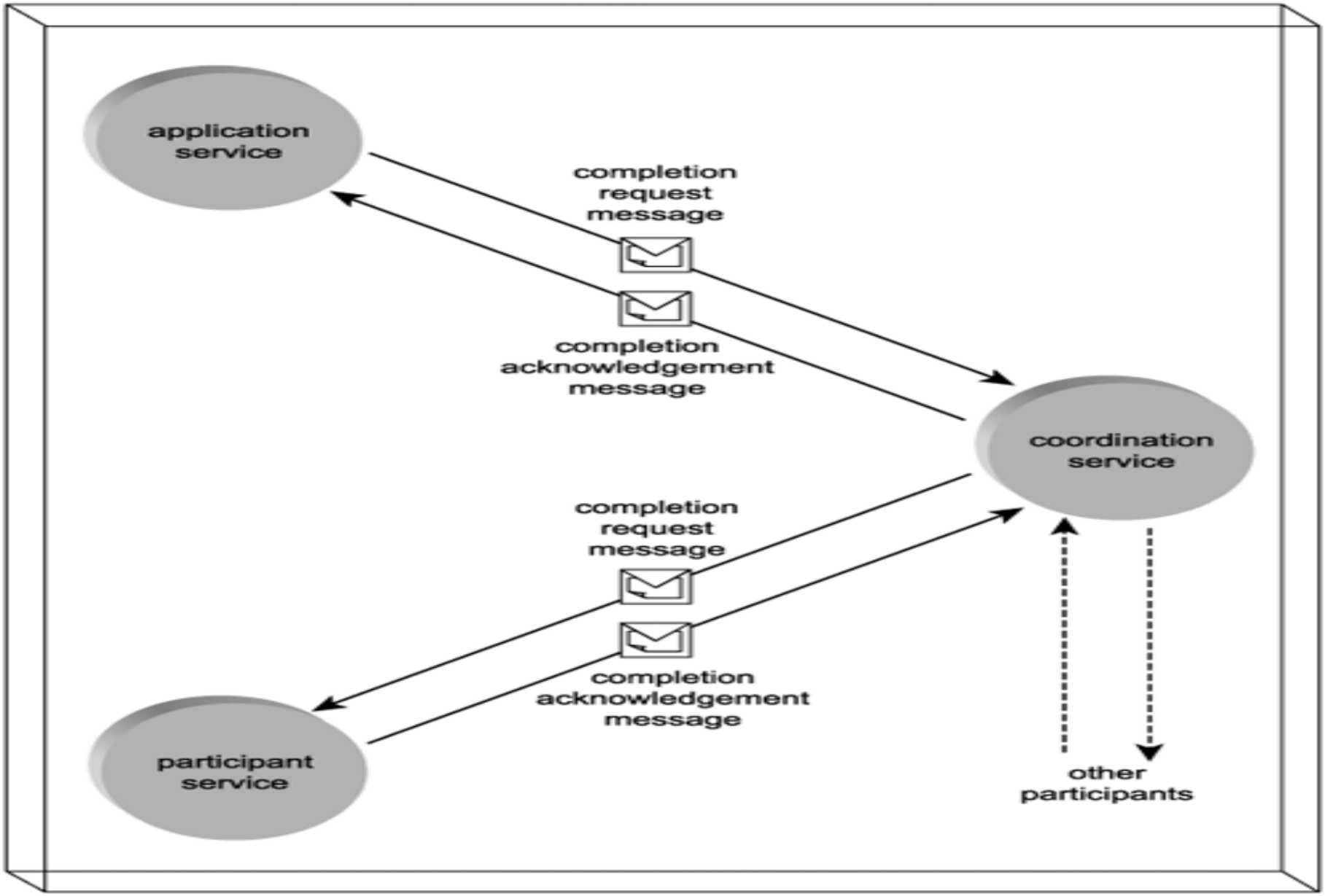
Coordination participants

- A service that wants to take part in an activity managed by WS-Coordination must request the coordination context from the activation service.
- It then can use this context information to register for one or more coordination protocols.
- A service that has received a context and has completed registration is considered a participant in the coordinated activity.

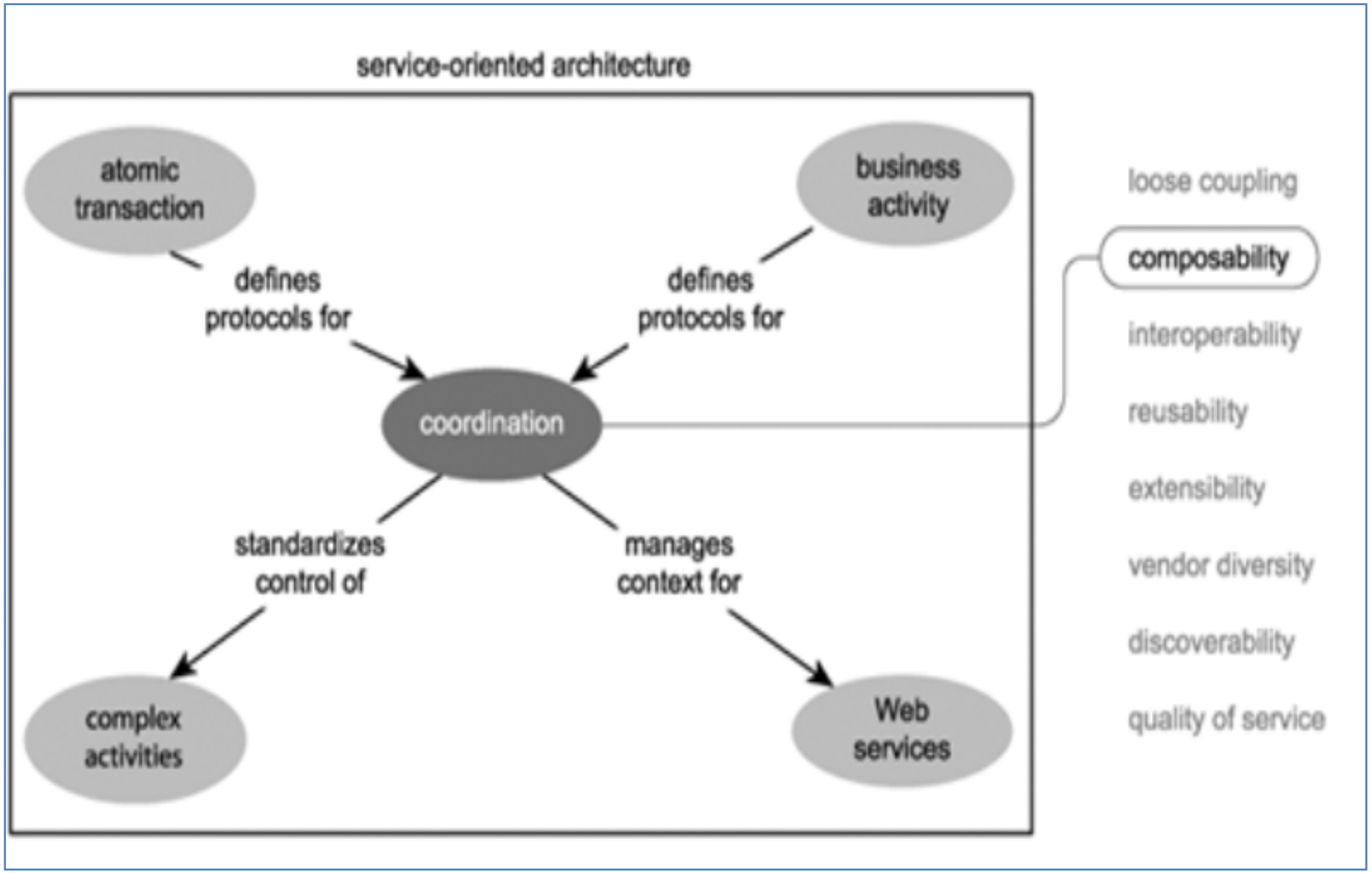
The activation and registration process

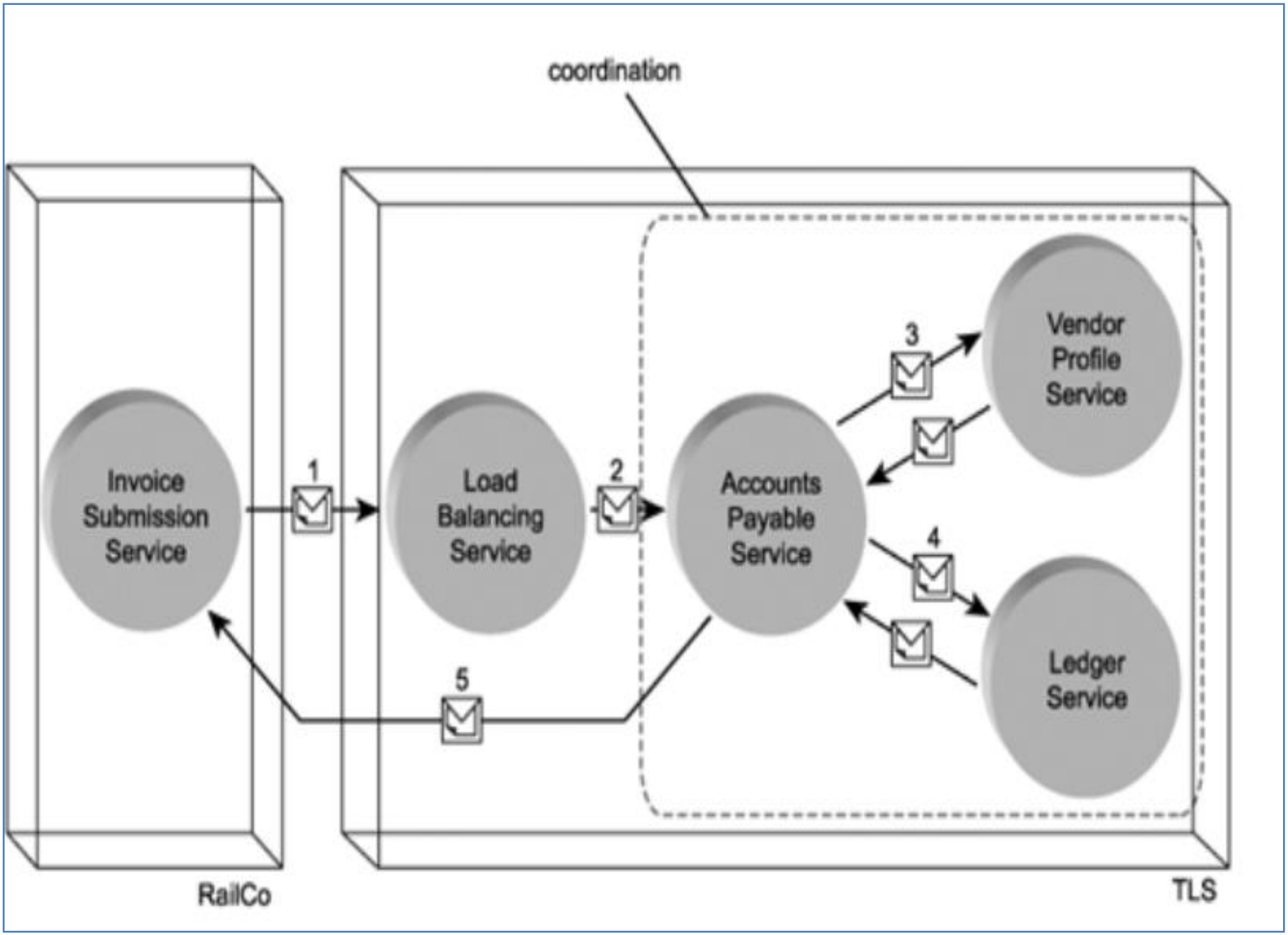


The completion process



Coordination and SOA



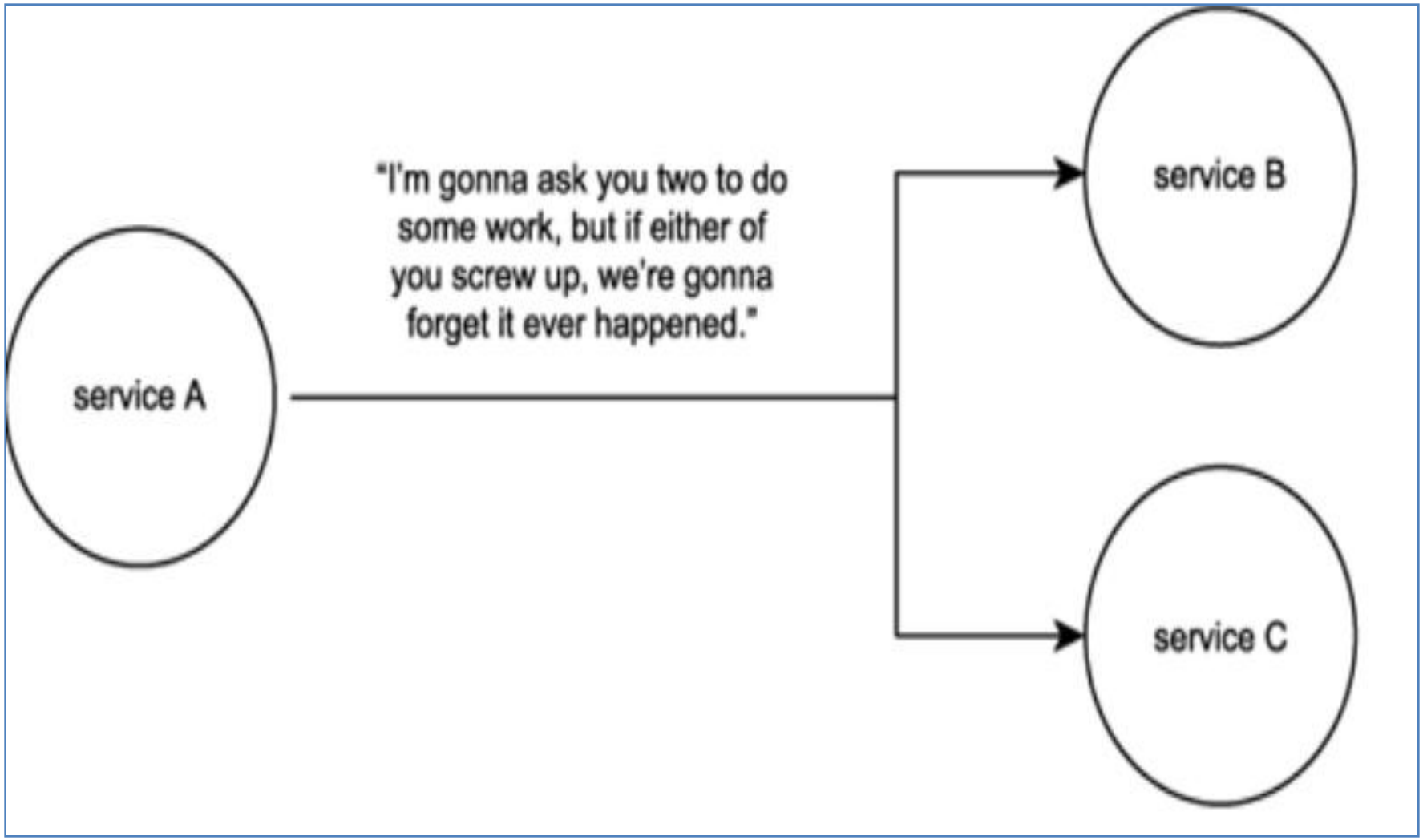

coordination
Invoice Submission Service
1
Load Balancing Service
2
Accounts Payable Service
3
Vendor Profile Service
4
Ledger Service
5
RailCo
TLS

So far,

- Complex activities tend to introduce the requirement for context data and the subsequent need for this data to be managed and coordinated at runtime.
- WS-Coordination provides a context management framework using a standardized service composition spearheaded by a coordinator service.
- Specialized implementations of this framework are realized through the use of coordination types, such as

  WS-AtomicTransaction and WS-BusinessActivity.
- By introducing an activity management layer to SOA, coordination promotes service statelessness and supports the controlled composition of complex activities.

Atomic transactions



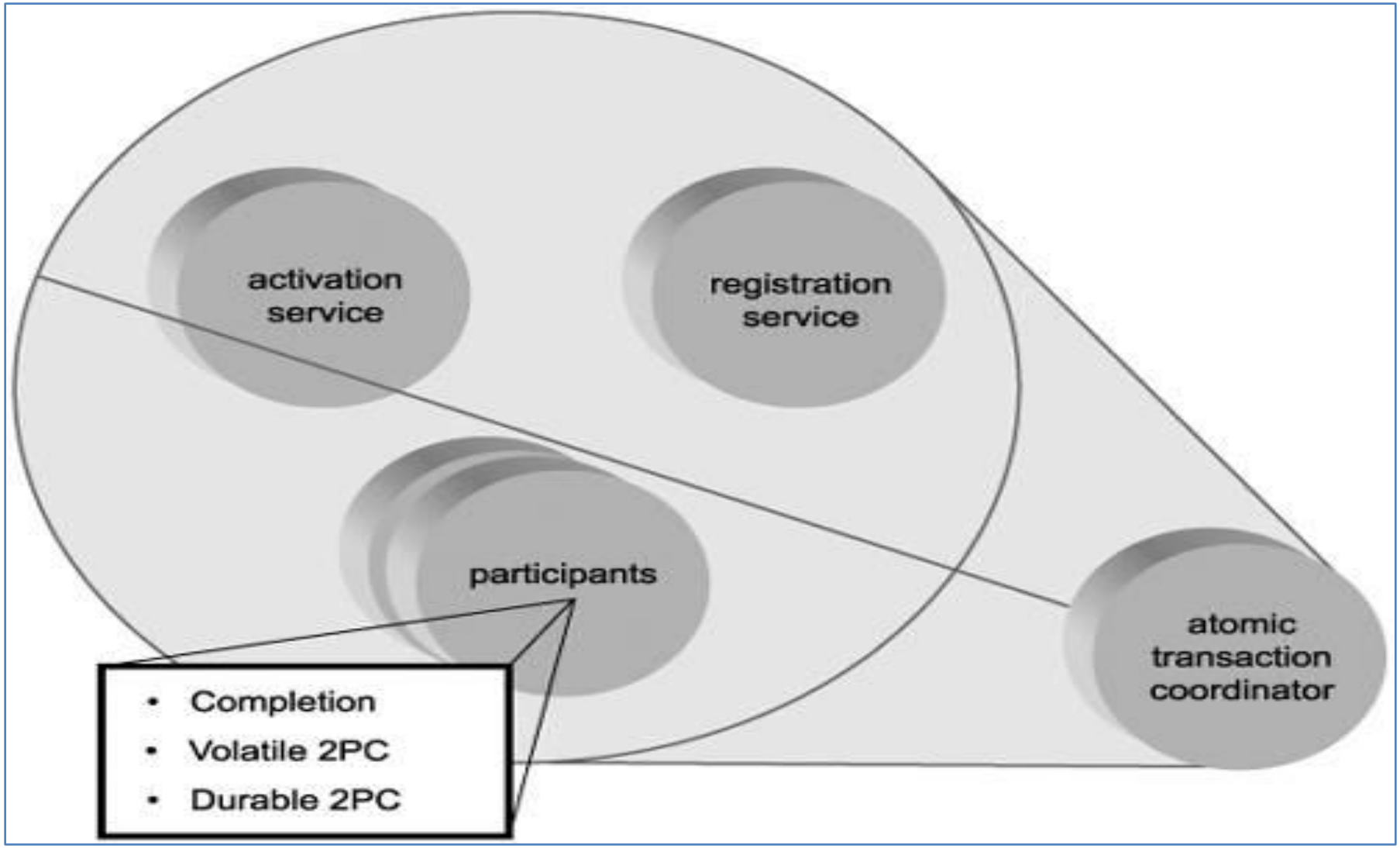
ACID transactions

- **Atomic**
  - All or nothing
- **Consistent**
  - No data change can violate the validity of any associated data models.
- **Isolate**
  - If multiple transactions occur concurrently, they may not interfere with each other.
- **Durable**
  - Upon the completion of a successful transaction, changes made as a result of the transaction can survive subsequent failures.

Atomic Transaction Protocols

- WS-AtomicTransaction is a coordination type
- Primary transaction protocols
  - A Completion protocol, which is typically used to initiate the commit or abort states of the transaction.
  - The Durable 2PC protocol for which services representing permanent data repositories should register.
  - The Volatile 2PC protocol to be used by services managing non-persistent data.

The Atomic Transaction Coordinator



The Atomic Transaction Process

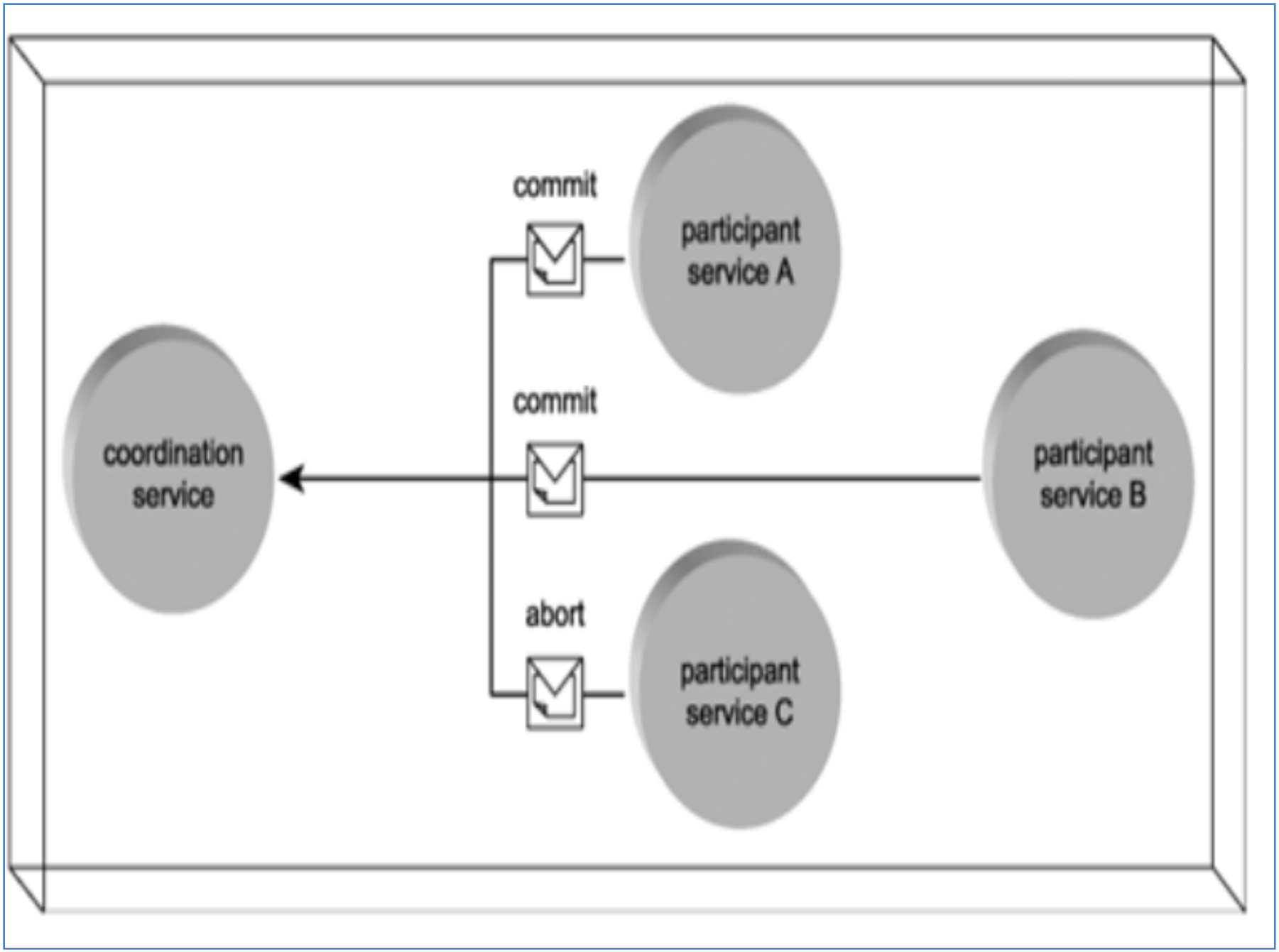
- The atomic transaction coordinator is tasked with the responsibility of deciding the outcome of a transaction.
  - It bases this decision on feedback it receives from all of the transaction participants.

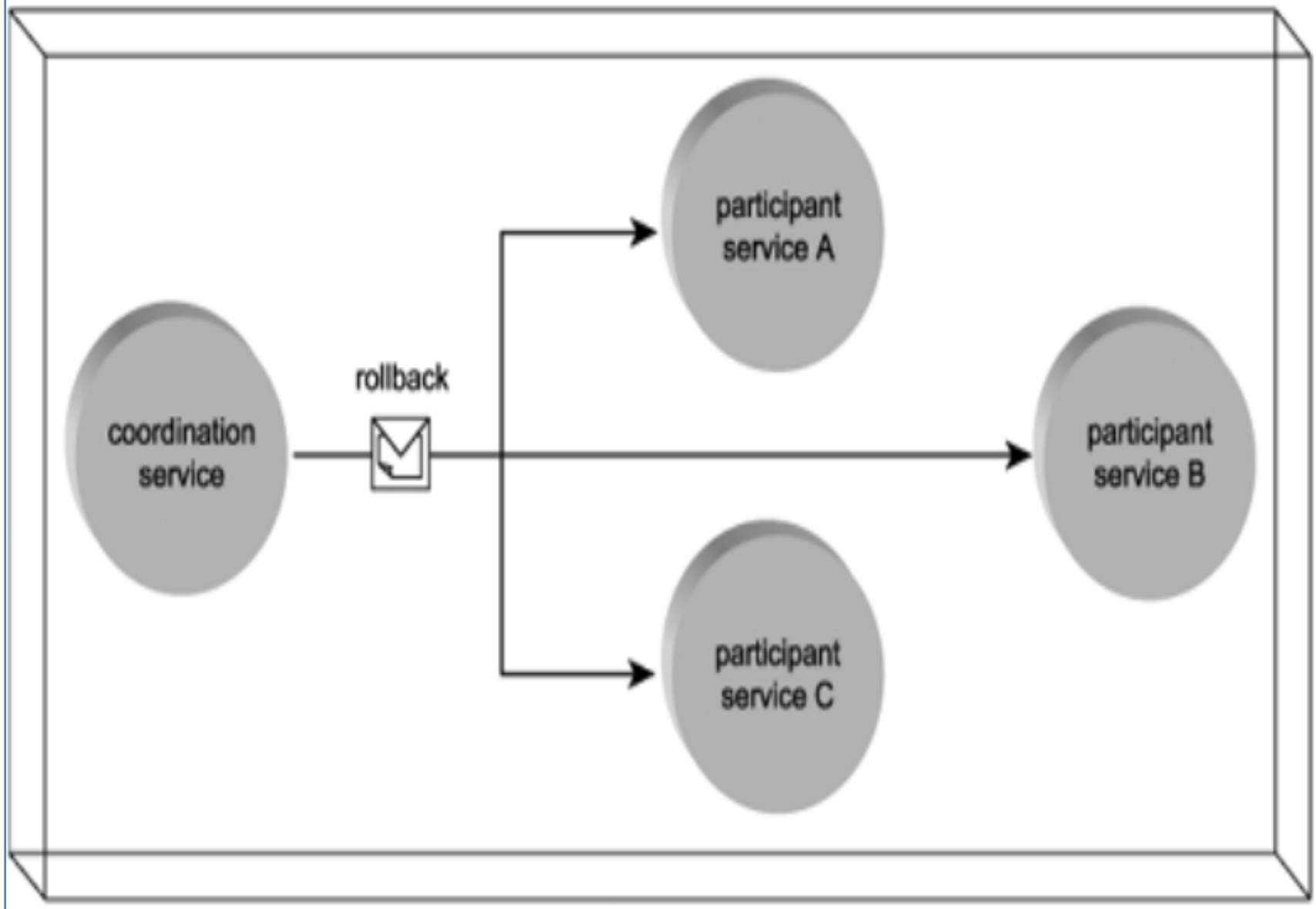
- Prepare phase and Commit phase

The Atomic Transaction Process

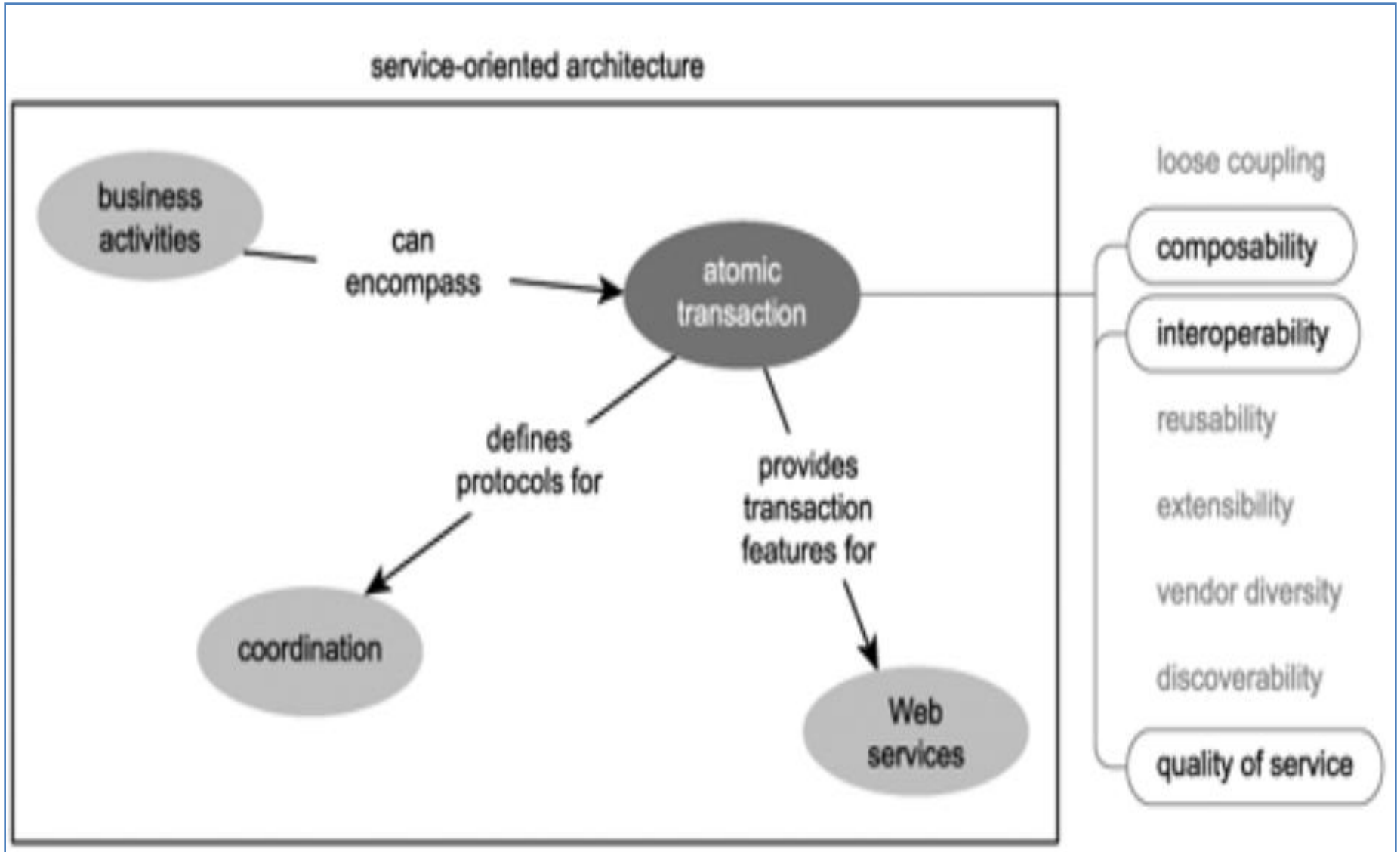
- **Prepare:**
  - All participants are asked to prepare and issue a vote (Commit or abort request)

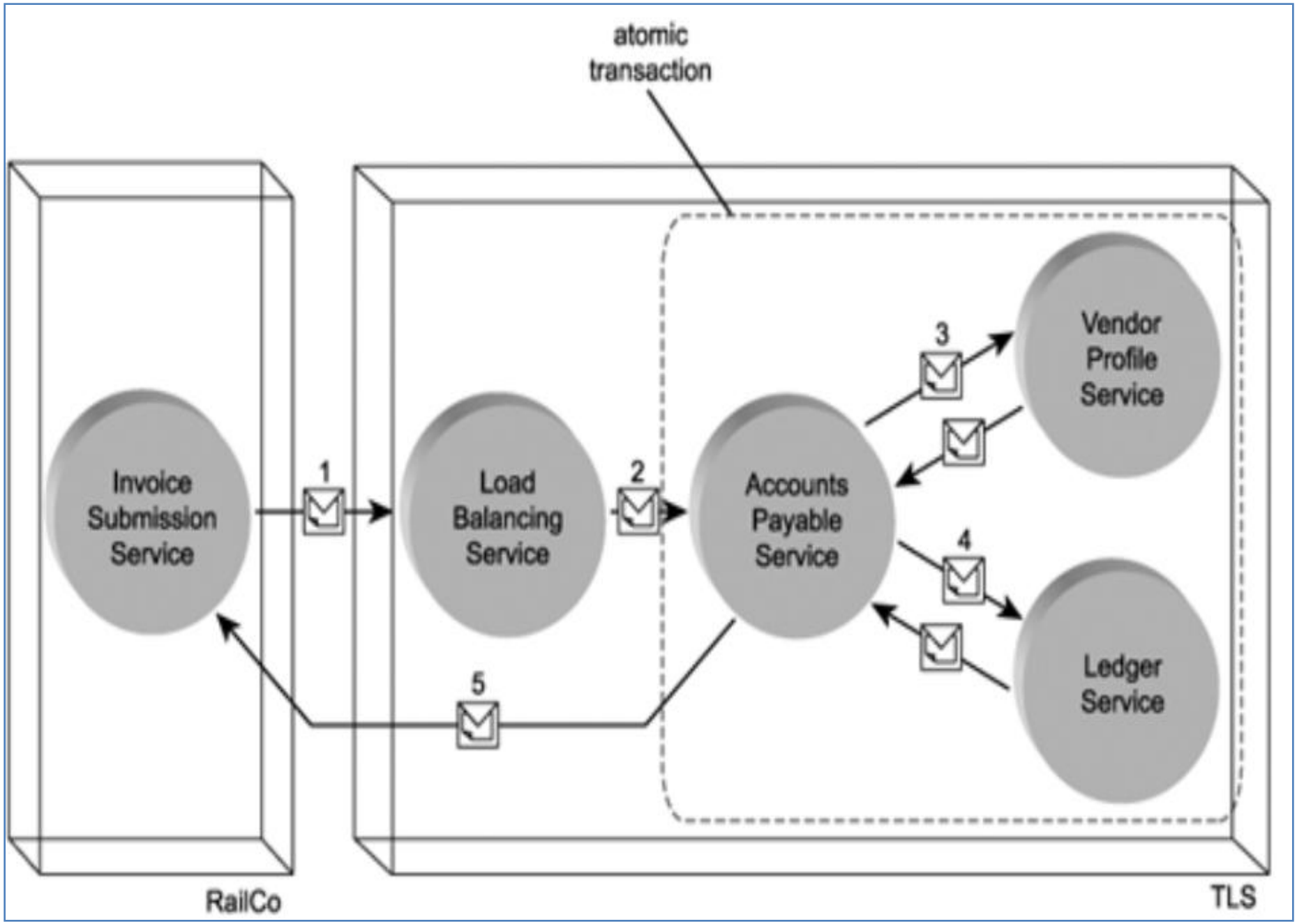
- **Commit:**
  - All votes are reviewed and decision to commit or rollback is taken


commit
participant service A
commit
coordination service
participant service B
abort
participant service C


participant
service A
coordination
service
rollback
participant
service B
participant
service C

Atomic Transactions and SOA



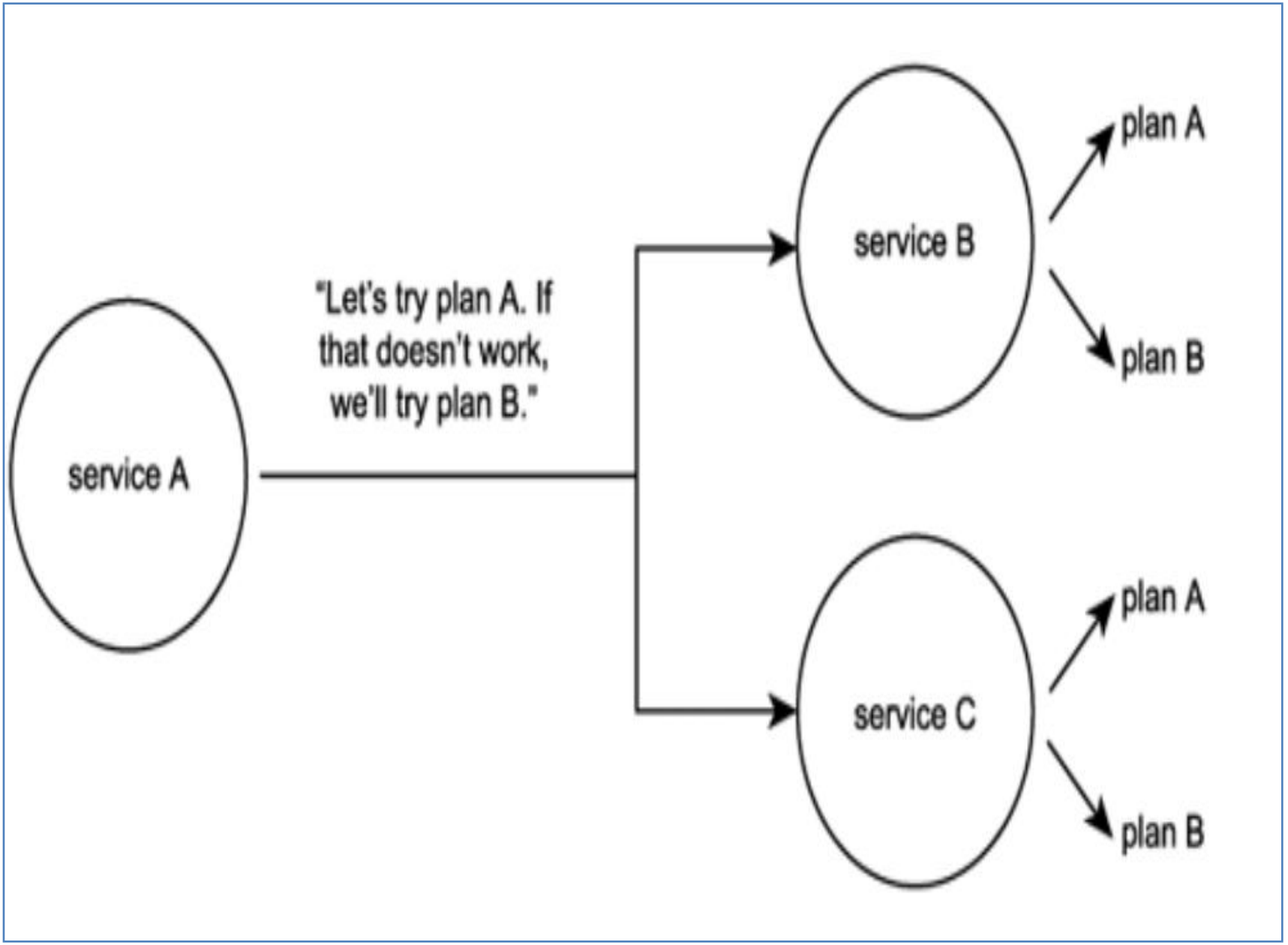

atomic
transaction
Invoice
Submission
Service
1
Load
Balancing
Service
2
Accounts
Payable
Service
3
Vendor
Profile
Service
4
Ledger
Service
5
RailCo
TLS

So far,

- WS-AtomicTransaction is a coordination type that supplies three coordination protocols that can be used to achieve two-phase commit transactions across multiple service participants.
- The atomic transaction coordinator makes the ultimate decision to commit or rollback a transaction. This decision is based on votes collected from participants.
- Contemporary SOAs can incorporate cross-service, ACID-type transaction features by using WS-AtomicTransaction.

Business Activities

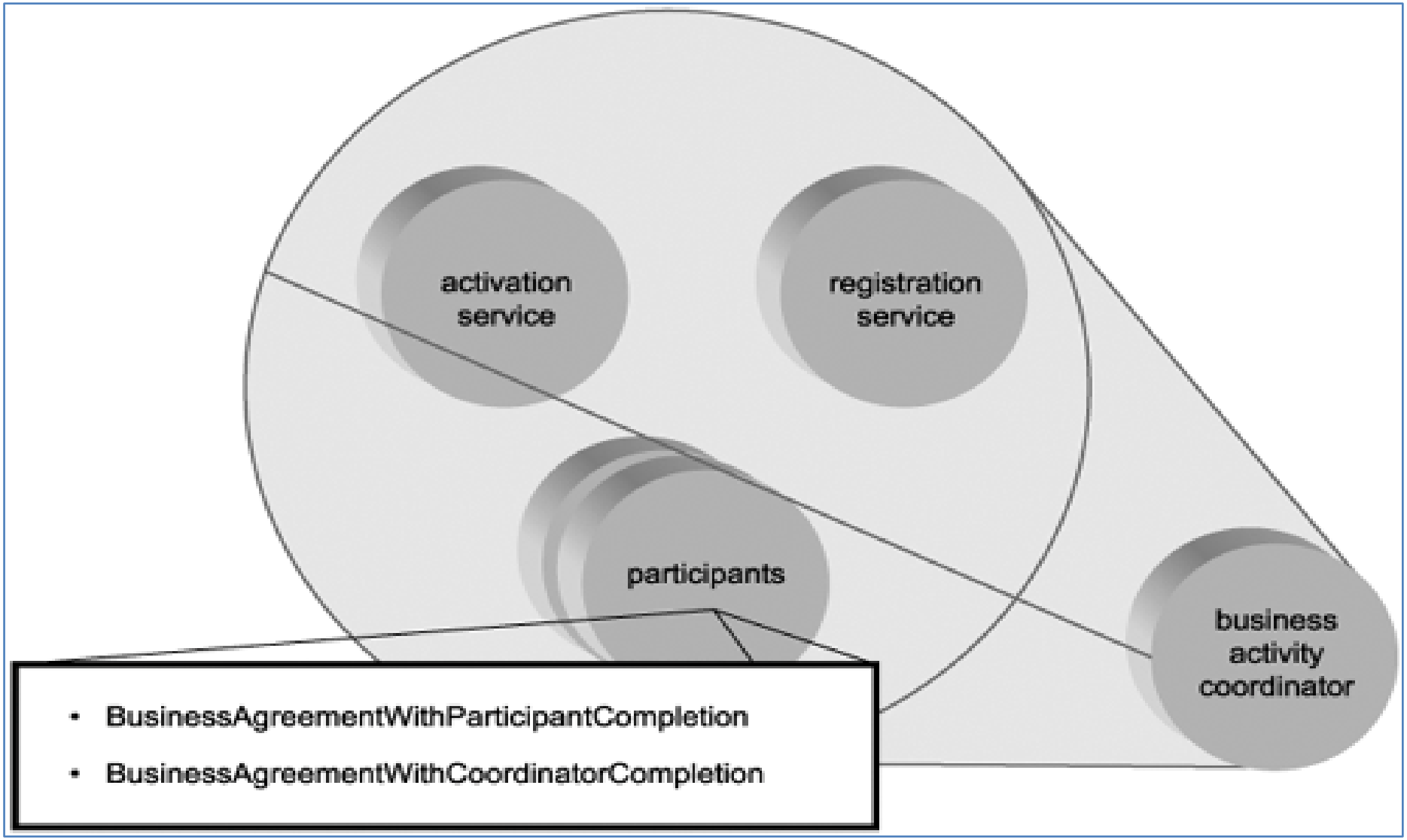
- Business activities govern long-running, complex service activities.
- What distinguishes a business activity from a regular complex activity is that its participants are required to follow specific rules defined by protocols.
- Business activities primarily differ from the also protocol-based atomic transactions in how they deal with exceptions and in the nature of the constraints introduced by the protocol rules.


service A
"Let's try plan A. If that doesn't work, we'll try plan B."
service B
plan A
plan B
service C
plan A
plan B

Business Activity Protocols

- WS-BusinessActivity is a coordination type supporting following protocols:
- The BusinessAgreementWithParticipantCompletion protocol
  - Allows participant to determine when it has completed its part in the business activity
- The BusinessAgreementWithCoordinatorCompletion protocol
  - Participants rely on coordinator to notify that it has no further processing responsibilities

The Business Activity Coordinator



Business Activity States

- During the lifecycle of a business activity, the business activity coordinator and the activity participants transition through a series of states.

- Active and Completed States:
  - A participant can indicate completion of activity by issuing a completed notification
  - This moves participant from Active to Completed state

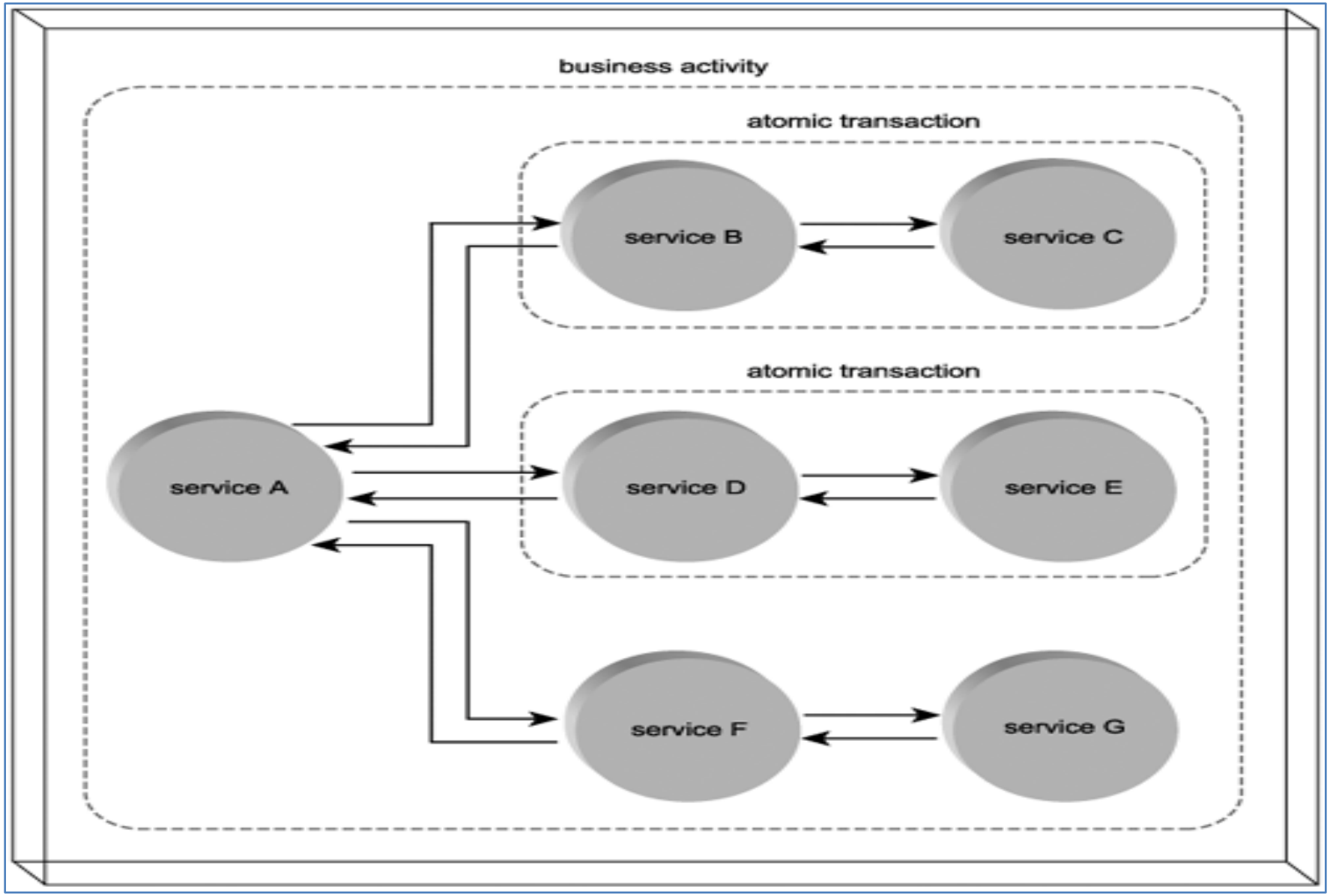
Business Activity States

- However, if things don't go as planned during the course of a business activity, one of a number of options are available.
  - Participants can enter a compensation state during which they attempt to perform some measure of exception handling.
  - It does not have rollback, but it can execute Plan B
  - Alternatively, a cancelled state can be entered.
    - Cancellation notifications are distributed

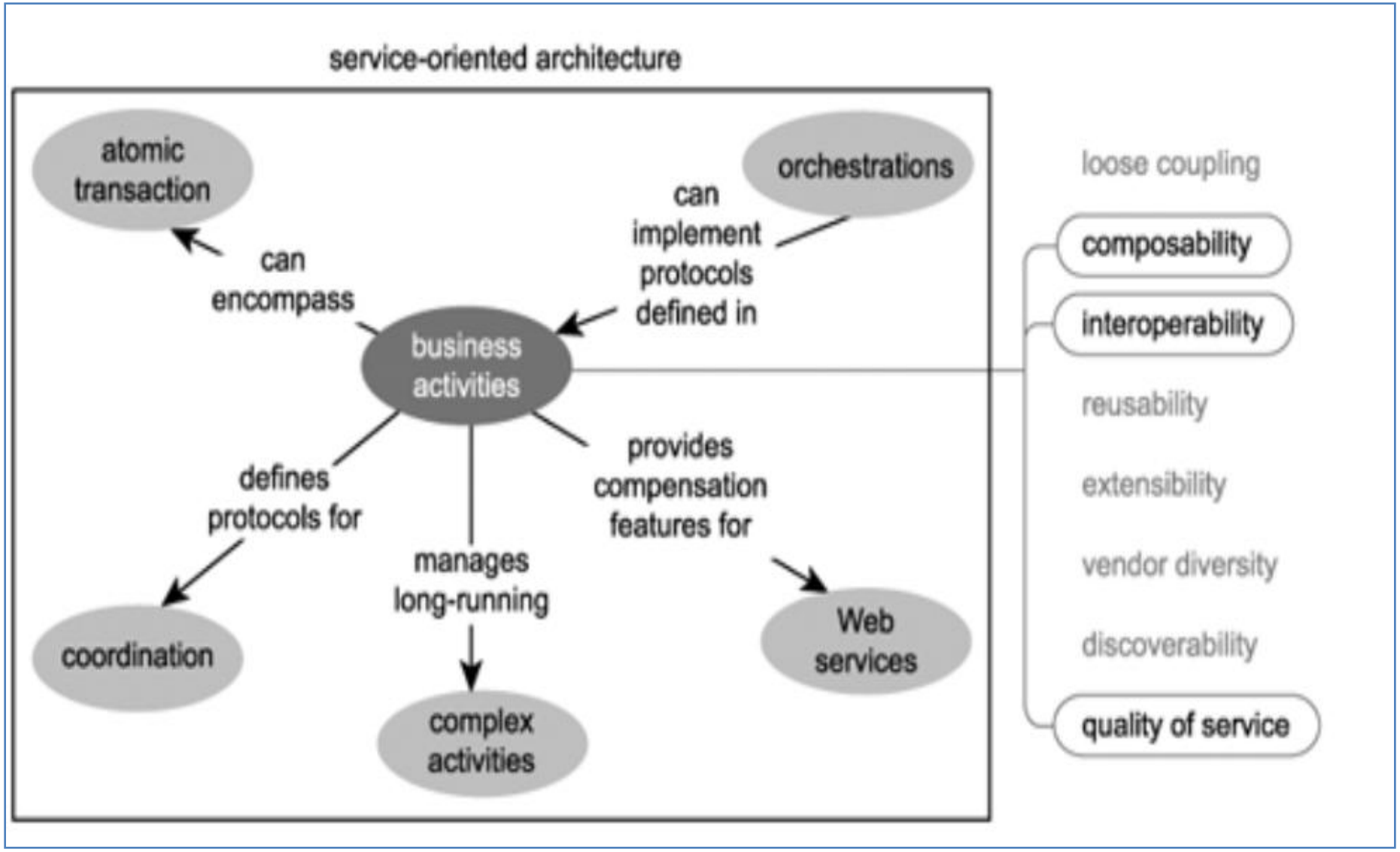
Business Activity States

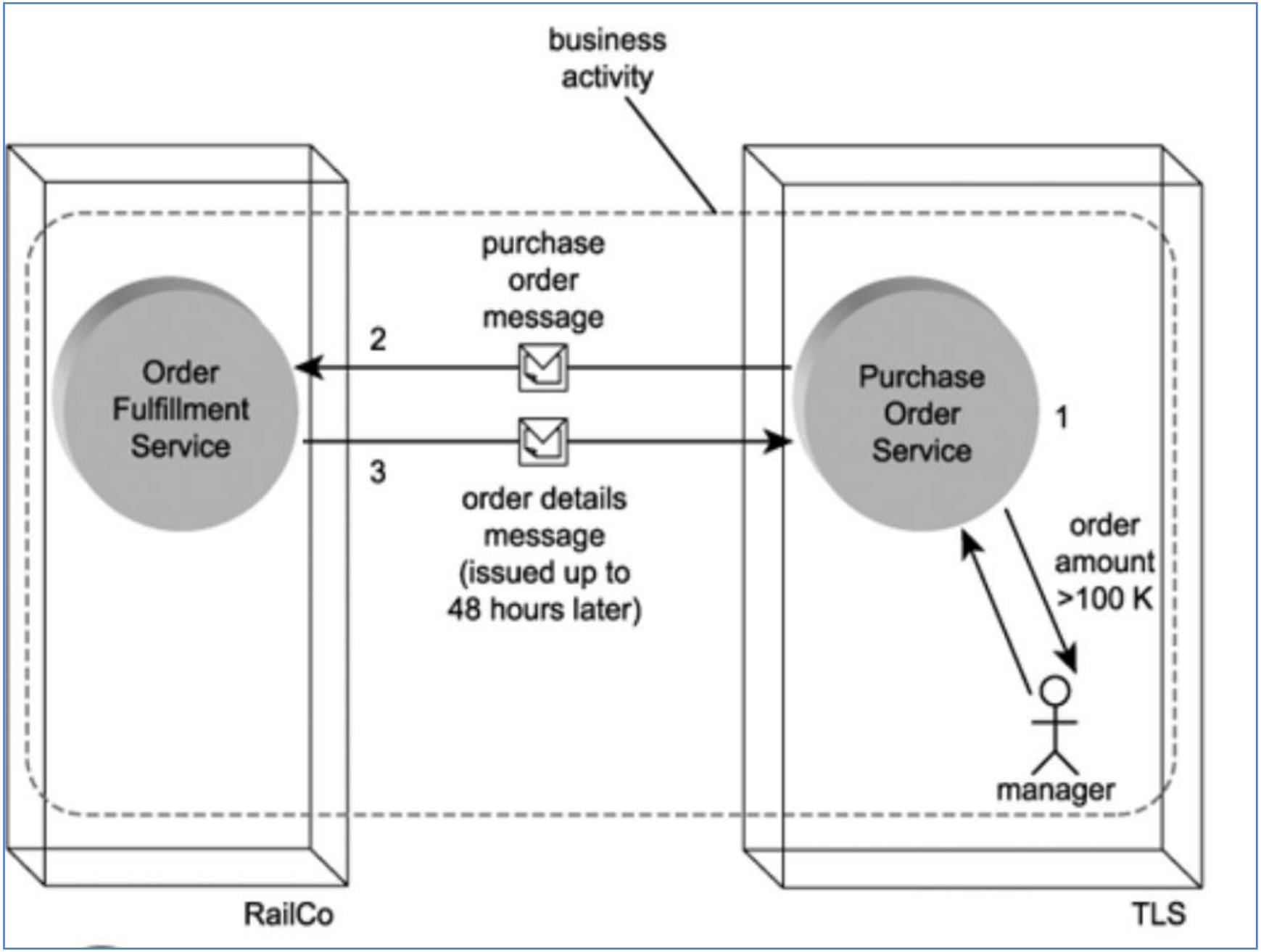
- Participating services are not required to remain participants for the duration of the activity.
  - Exit state
    - Difference between WS- Business Activity and WS- Atomic Transaction

Business Activities and Atomic Transactions



Business Activities and SOA




business
activity
purchase
order
message
2
Order
Fulfillment
Service
3
order details
message
(issued up to
48 hours later)
Purchase
Order
Service
1
order
amount
>100 K
manager
RailCo
TLS

So far,

- Business activities manage complex, long-running activities that can vary in scope and in the amount of participating services.
- WS-BusinessActivity builds on the WS-Coordination context management framework by providing two protocols for which activity participants can register.
- Participants and the business activity coordinator progress through a series of states during the lifespan of a business activity.
- Long-running activities are commonplace in contemporary SOAs, which positions WS-BusinessActivity as an important specification for the controlled management of logic that underlies these types of complex activities.