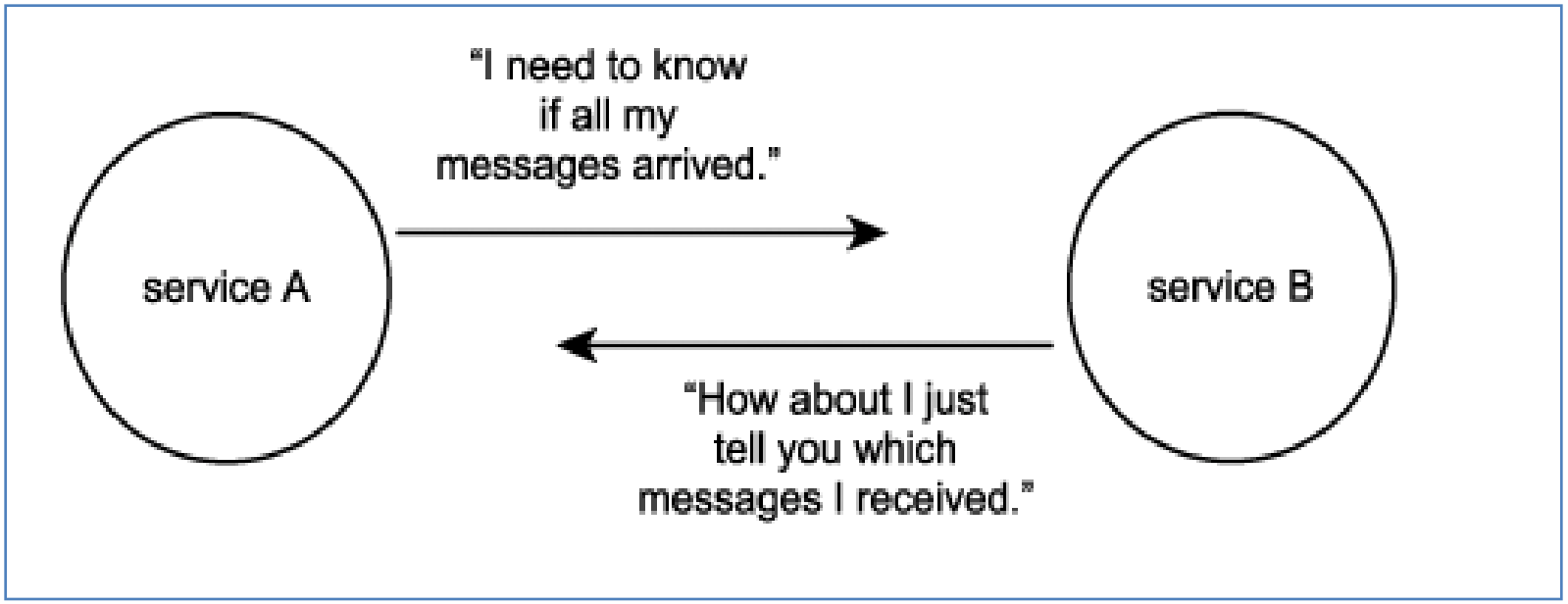
What is the side effect from benefits of loosely coupled messaging framework?

WS-Reliable Messaging

- The benefits of a loosely coupled messaging framework come at the cost of a loss of control over the actual communications process.
- After a Web service transmits a message, it has no immediate way of knowing:
  - whether the message successfully arrived at its intended destination
  - whether the message failed to arrive and therefore requires a retransmission
  - whether a series of messages arrived in the sequence they were intended to

WS-Reliable messaging

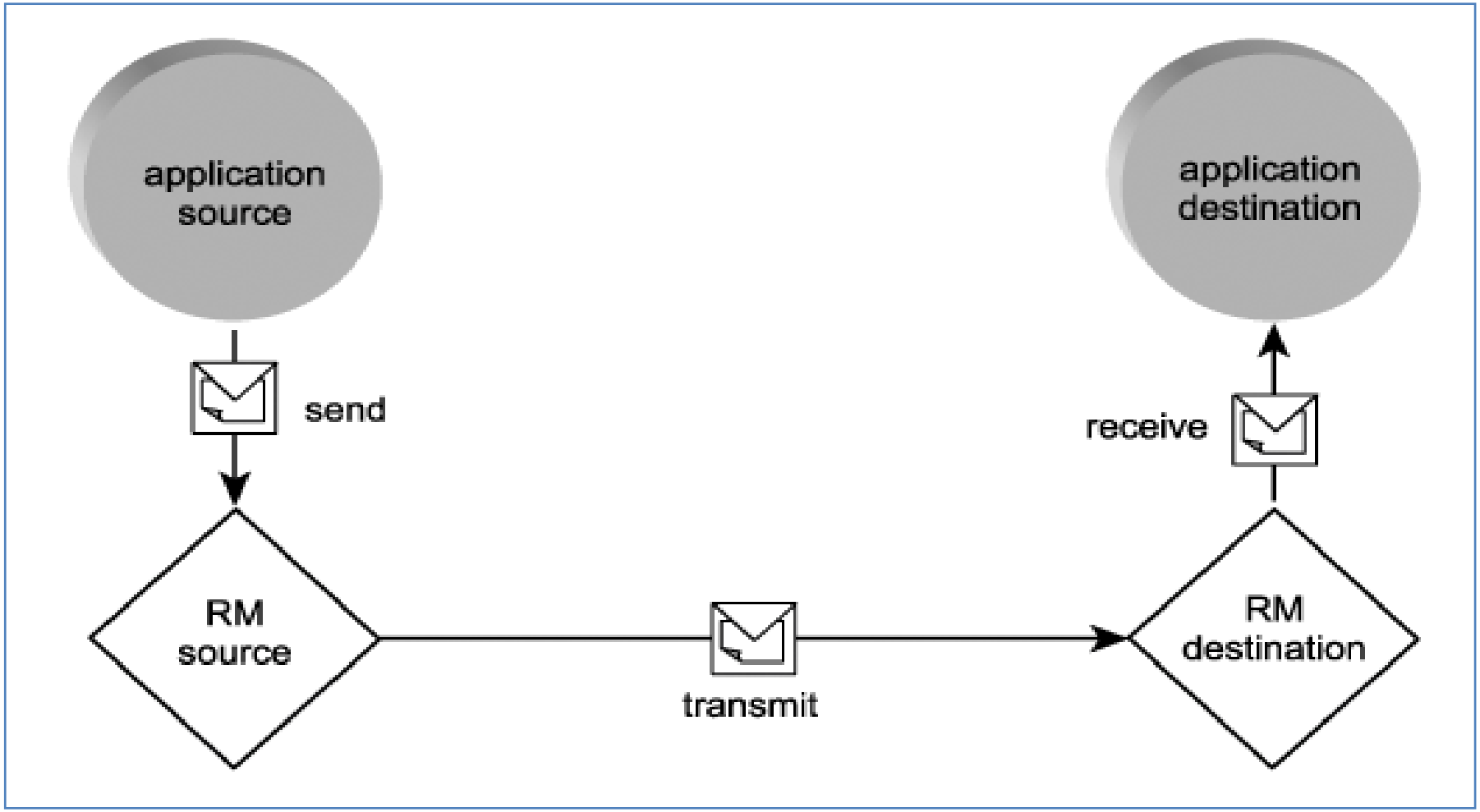
- Reliable messaging addresses certain concerns by establishing a measure of quality assurance that can be applied to other activity management frameworks .



WS-Reliable messaging

- WS-ReliableMessaging provides a framework capable of guaranteeing:
  - that service providers will be notified of the success or failure of message transmissions
  - that messages sent with specific sequence-related rules will arrive as intended (or generate a failure condition)
- Reliable messaging does not employ a coordinator service to keep track of the state of an activity; instead, all reliability rules are implemented as SOAP headers within the messages themselves.

RM Source, RM Destination, Application Source, and Application Destination



WS-Reliable messaging

- An application source is the service or application logic that sends the message to the RM source
- RM Source is the physical processor or node that performs the actual wire transmission.
- RM destination represents the target processor or node that receives the message and subsequently delivers it to the application destination

WS-Reliable messaging

- WS-ReliableMessaging makes a distinction between the parts of a solution that are responsible for initiating a message transmission and those that actually perform the transmission.

- It further assigns specific descriptions to the terms "send“, "transmit“, "receive" and "deliver" as they relate differently to these solution parts.

Sequence

- A **sequence** establishes the order in which messages should be delivered.
- Each **message** that is part of a sequence is labeled with a message number that identifies the position of the message within the sequence.
- The **final message** in a sequence is further tagged with a last message identifier.

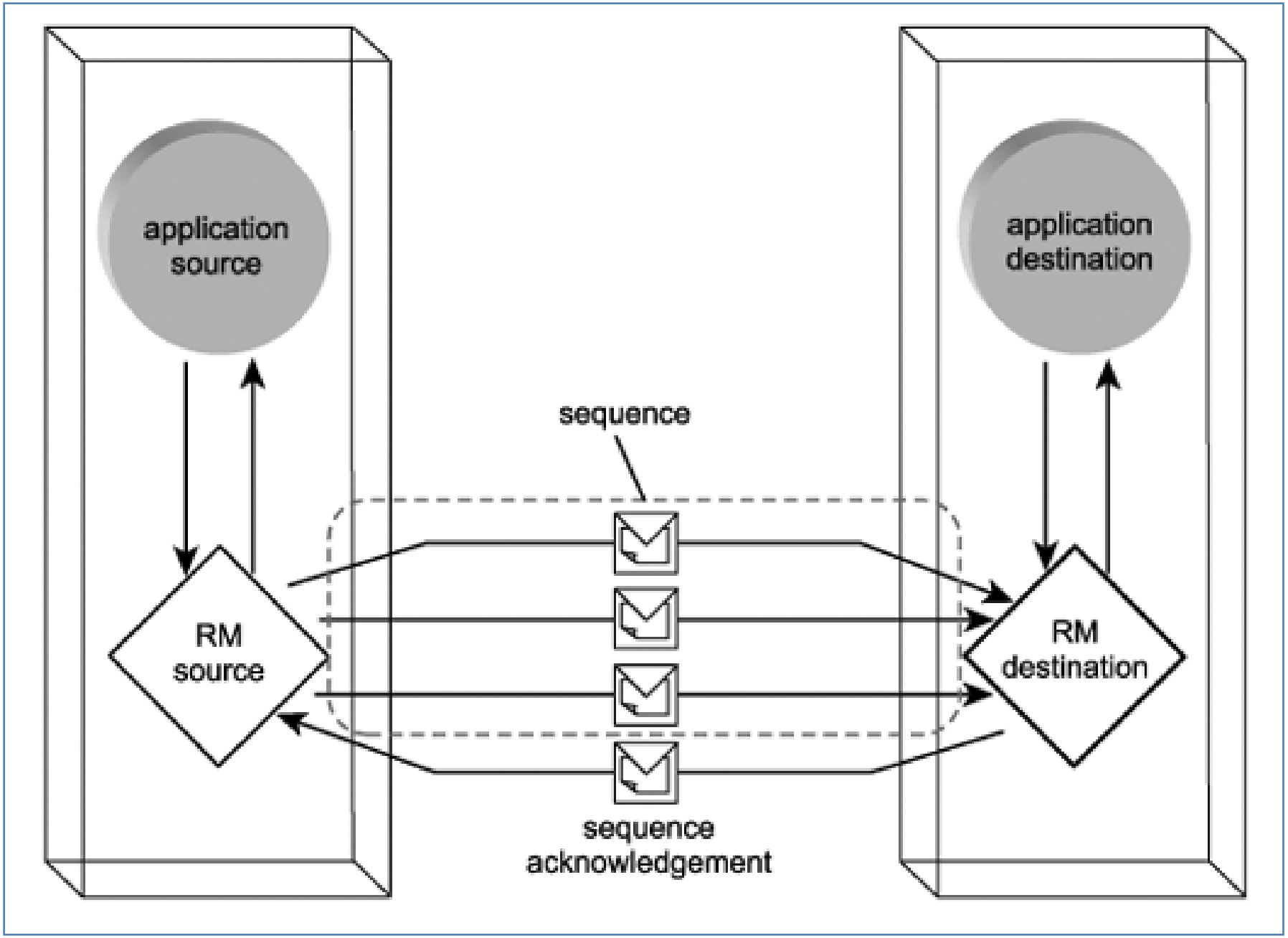
Sequence Acknowledgement

- A core part of the reliable messaging framework is a notification system used to communicate conditions from the RM destination to the RM source.
- Upon receipt of the message containing the last message identifier, the RM destination issues a sequence acknowledgement.
- The acknowledgement message indicates to the RM source which messages were received.

Acknowledgements

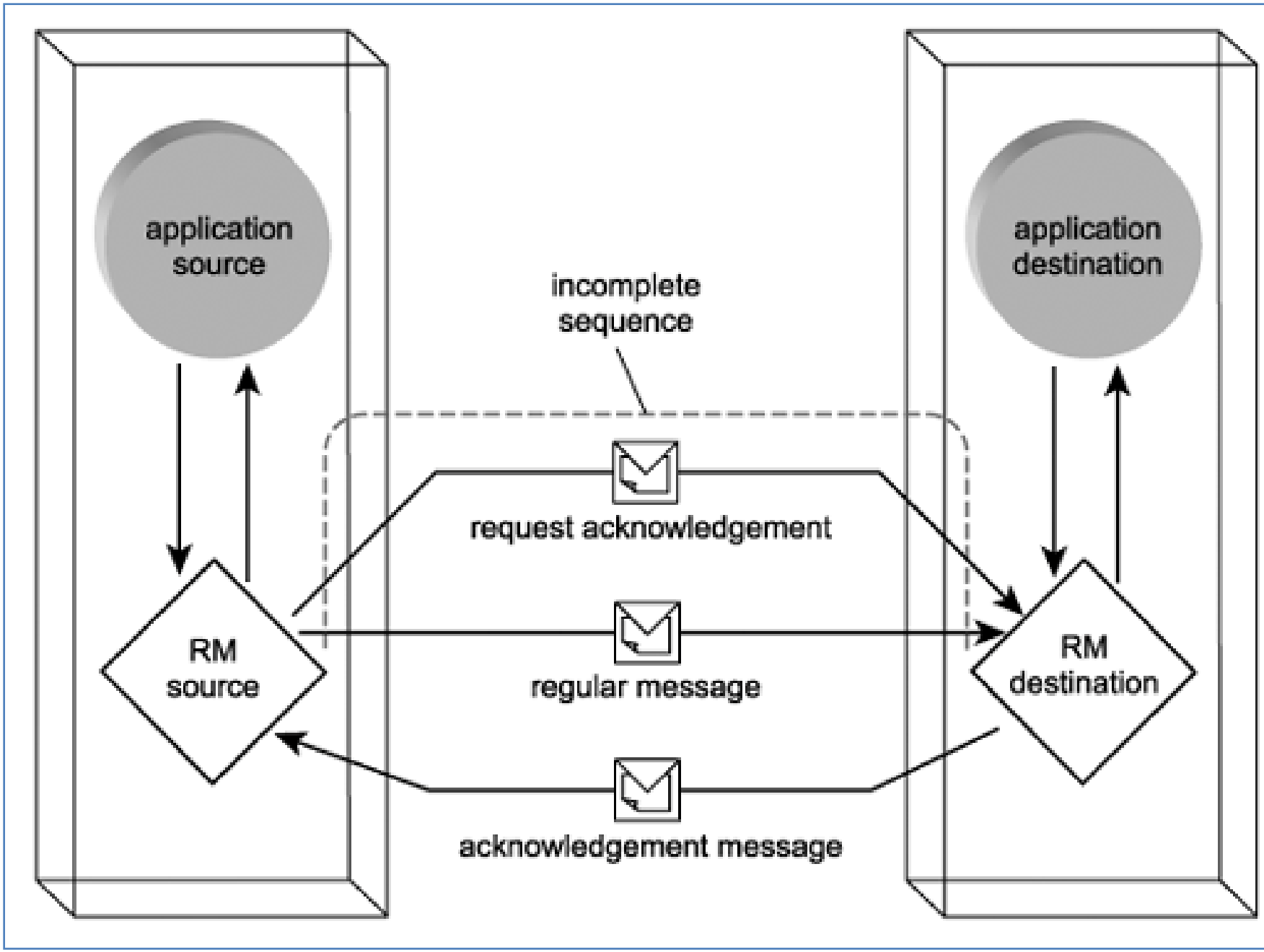
- It is up to the RM source to determine if the messages received are equal to the original messages transmitted.

- Missing messages can be retransmitted


application source
application destination
sequence
RM source
RM destination
sequence acknowledgement

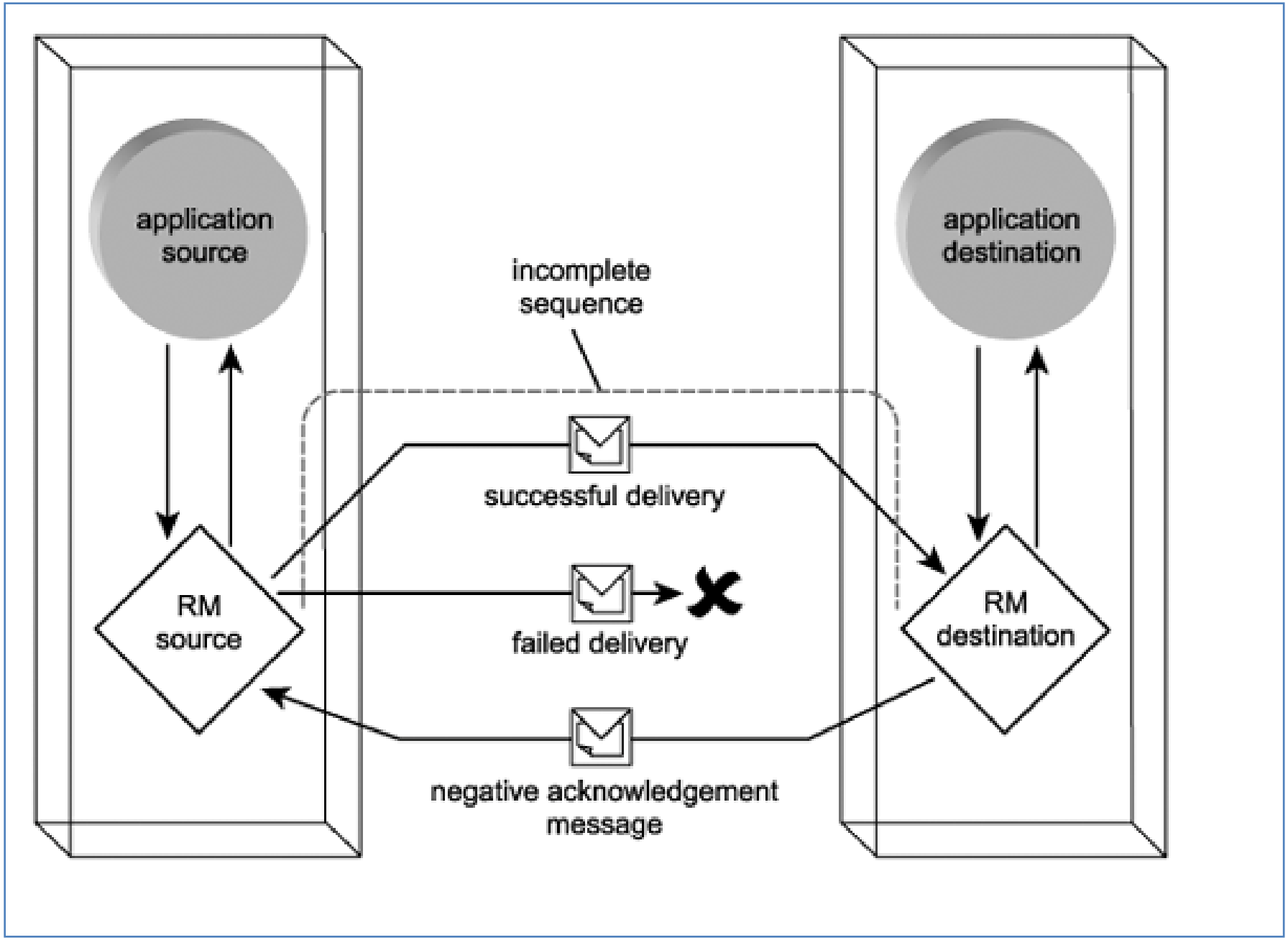
Request Acknowledgement

- RM source can request additional acknowledgement at any time by issuing request acknowledgement to RM destination


application source
application destination
incomplete sequence
request acknowledgement
RM source
regular message
RM destination
acknowledgement message

Negative Acknowledgement

- Negative Acknowledgement can be issued by RM destination to immediately indicate RM source that a failure has occurred

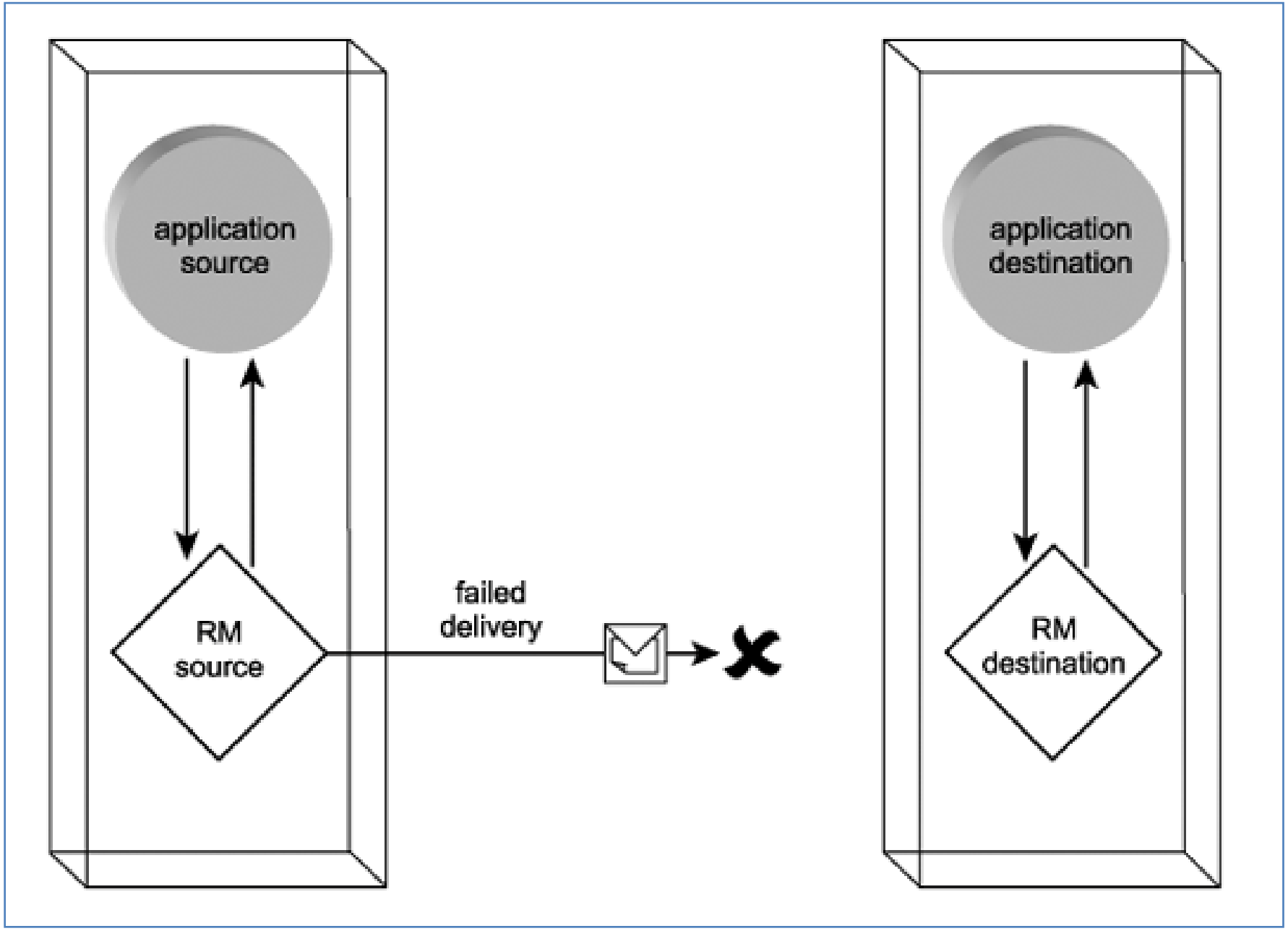

application source
application destination
incomplete sequence
successful delivery
RM source
failed delivery
RM destination
negative acknowledgement message

Delivery assurances

- The nature of a sequence is determined by a set of reliability rules known as delivery assurances.
- “Delivery assurances are predefined message delivery patterns that establish a set of reliability policies.”
  - AtMostOnce
  - AtLeastOnce
  - ExactlyOnce
  - Inorder

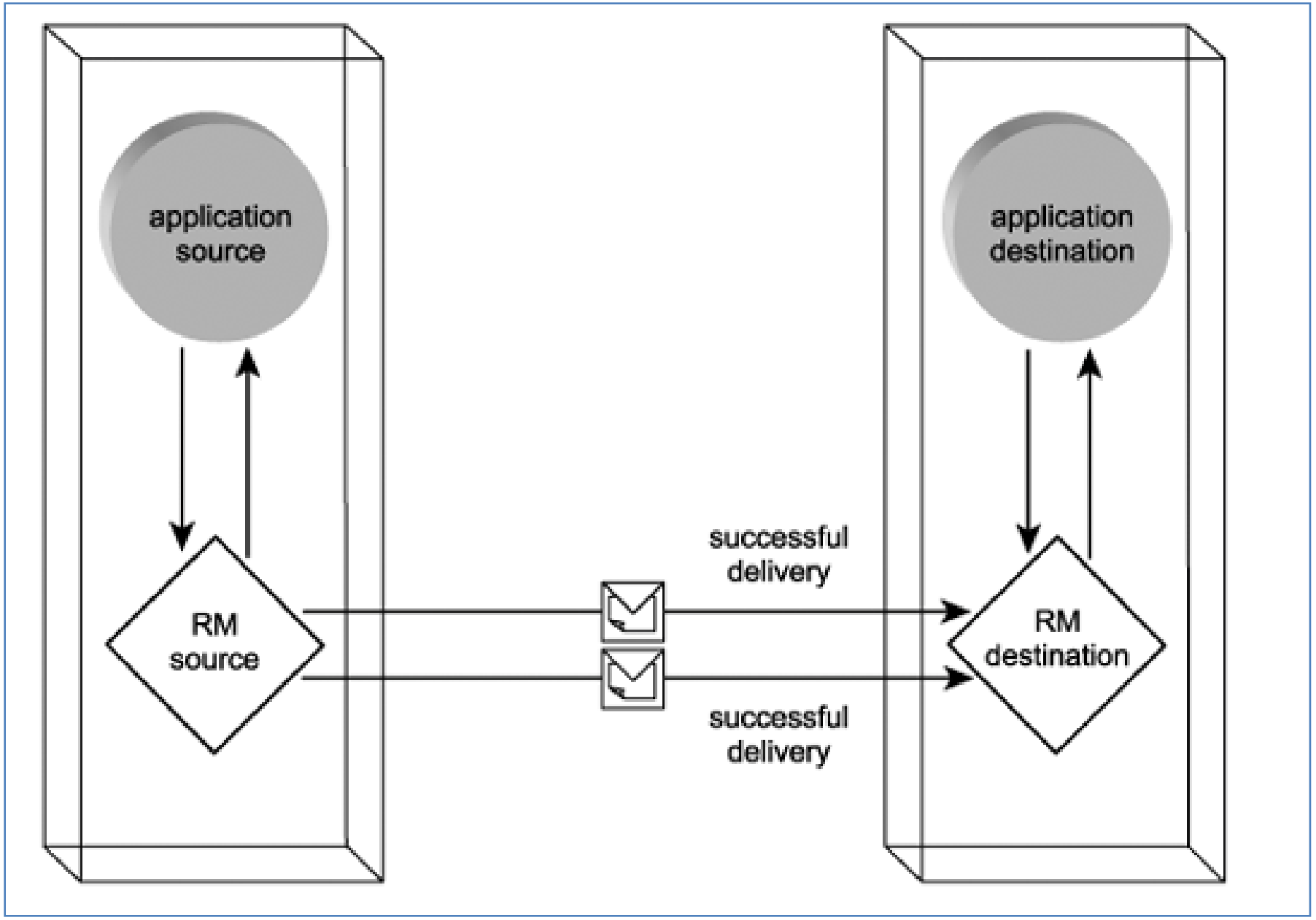
Delivery assurances

- **AtMostOnce:**
  - Ensures delivery of one or zero message
  - Error will be generated if more than one (duplicate) messages are delivered


application source
application destination
RM source
failed delivery
RM destination

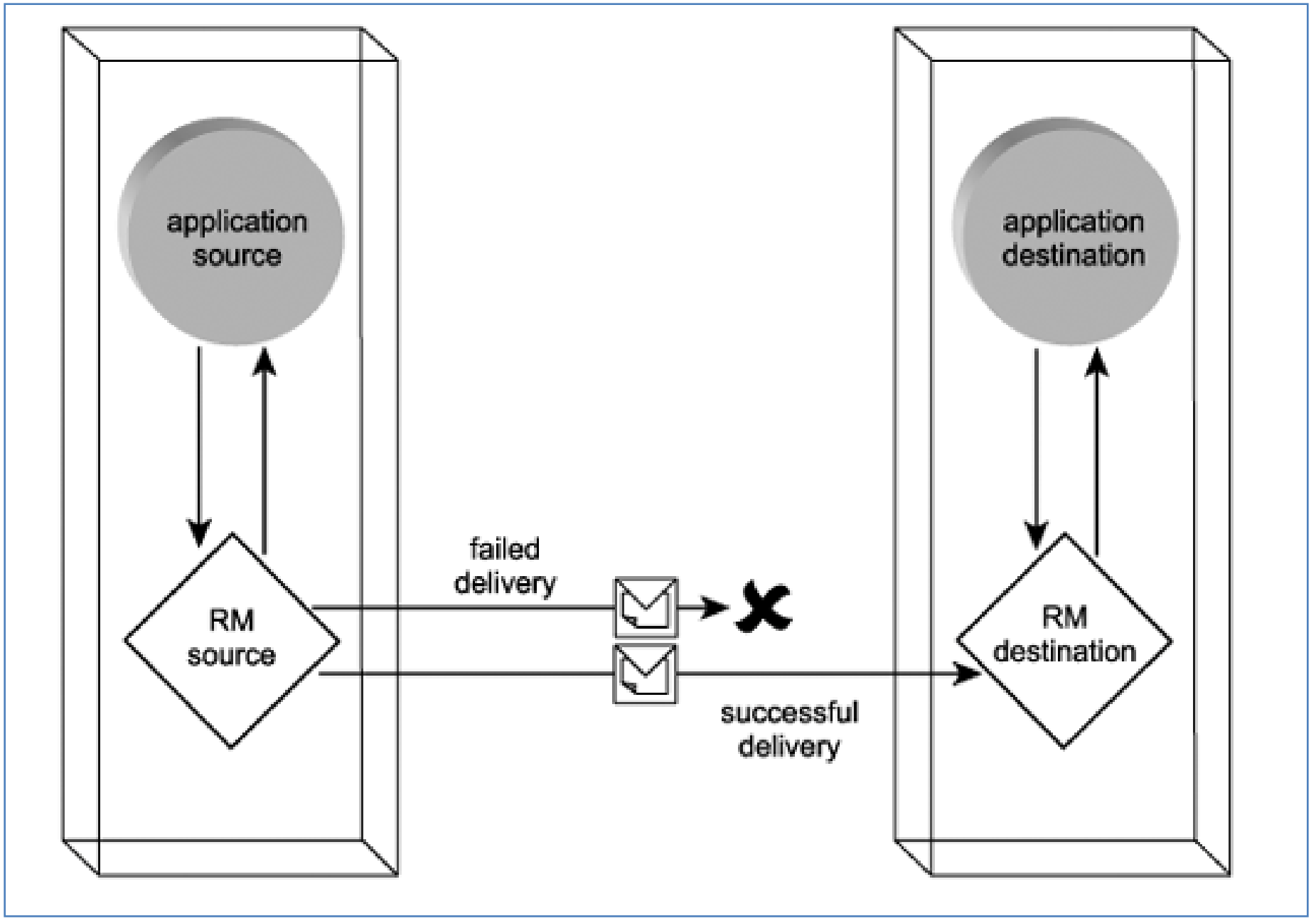
Delivery assurances

- **AtLeastOnce:**
  - Message can be delivered one or more time
  - Error will be generated if no messages are delivered


application source
application destination
RM source
RM destination
successful delivery
successful delivery

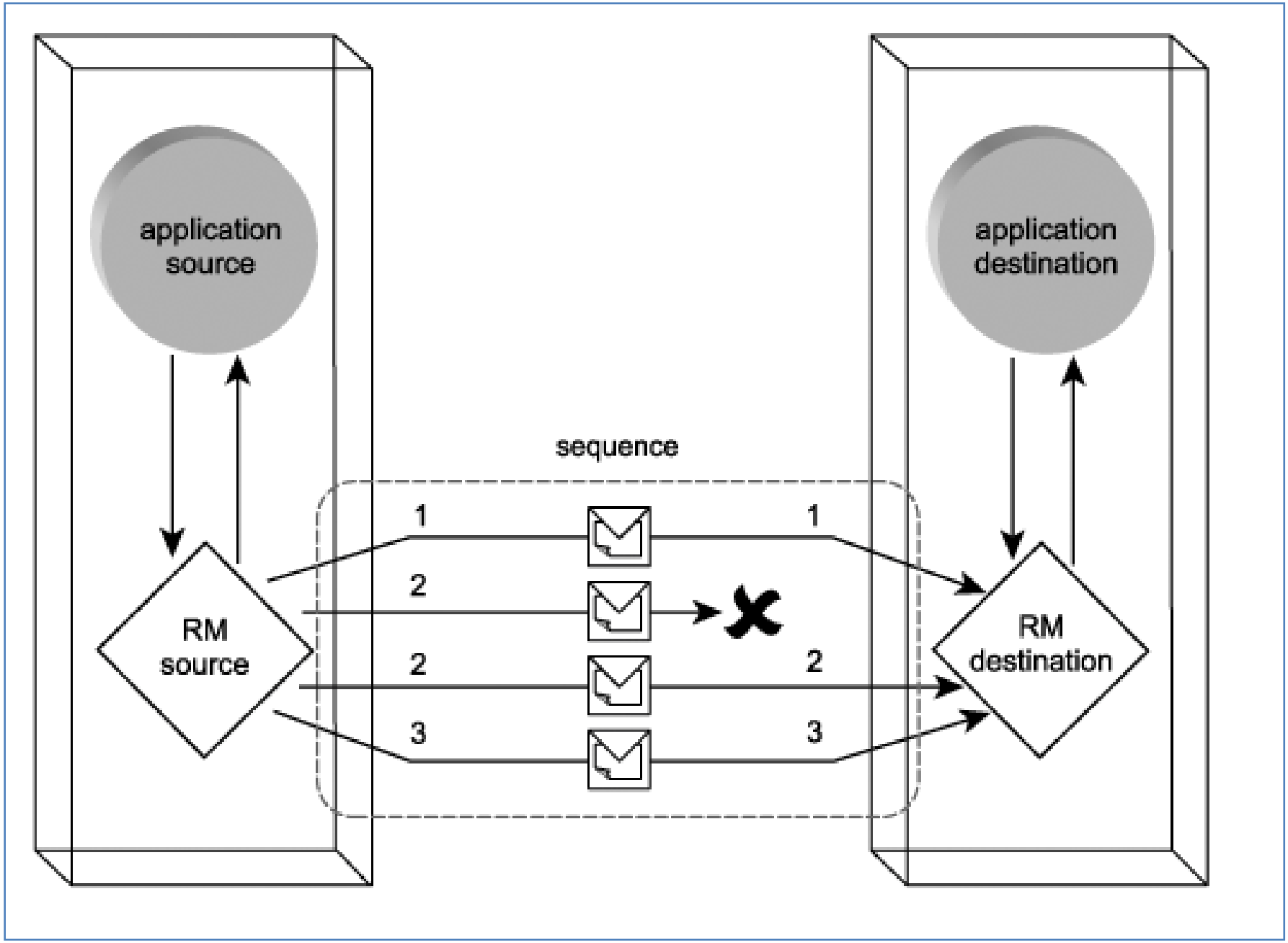
Delivery assurances

- **ExactlyOnce:**
  - Message will be delivered once
  - Error will be generated if zero or more than one (duplicate) messages are delivered


application source
application destination
RM source
RM destination
failed delivery
successful delivery

Delivery assurances

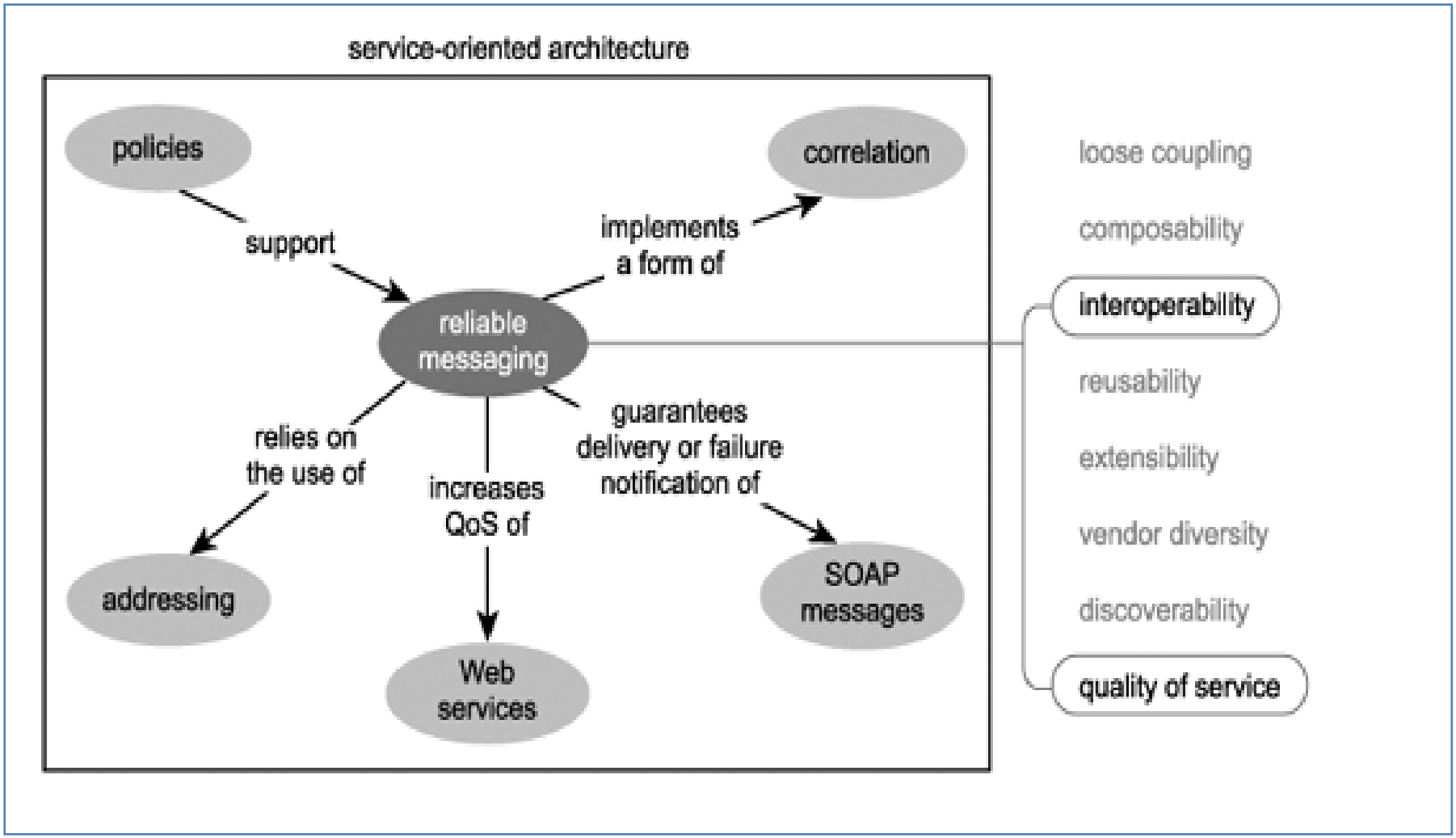
- **Inorder:**
  - Messages are delivered in specific sequence
  - Error will be generated if messages are delivered out of order

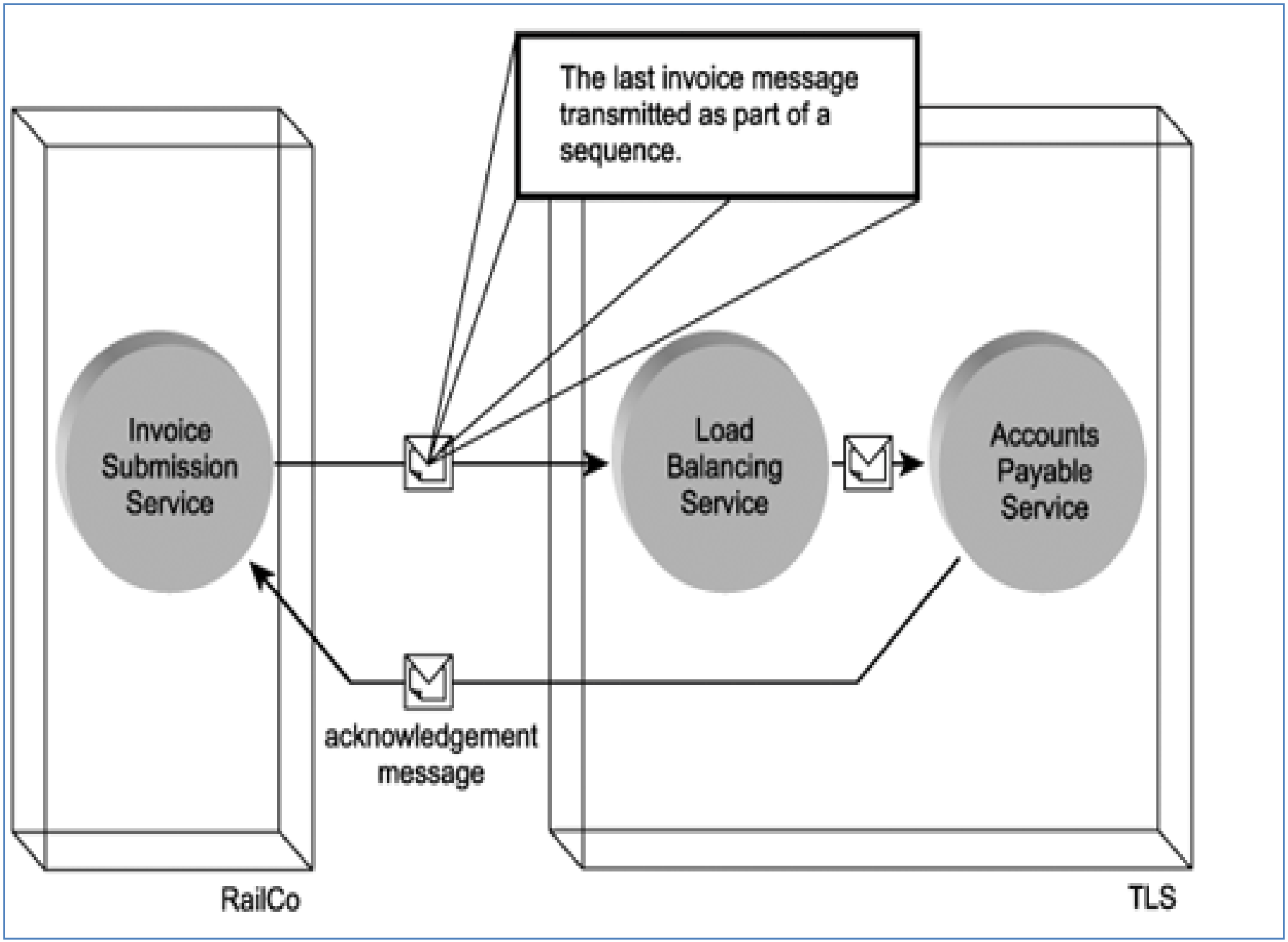

application source
application destination
sequence
1
1
2
2
2
3
3
RM source
RM destination

Reliable messaging and addressing

- WS-Addressing is closely tied to the WS-ReliableMessaging framework.
- In fact, it's interesting to note that the rules around the use of the WS-Addressing message id header were altered specifically to accommodate the WS-ReliableMessaging specification.
  - Message with same ID can be generated if retransmission is required

Reliable messaging and SOA




The last invoice message transmitted as part of a sequence.
Invoice Submission Service
Load Balancing Service
Accounts Payable Service
acknowledgement message
RailCo
TLS

So far, WS-Reliable messaging

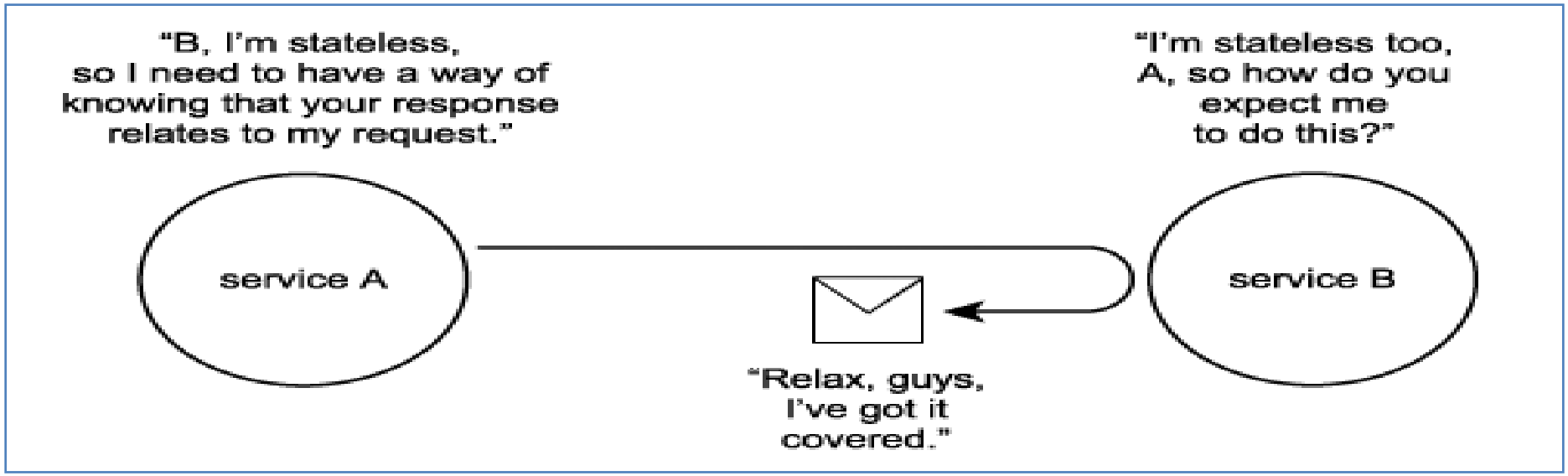
- WS-ReliableMessaging establishes a framework that guarantees the delivery of a SOAP message or the reporting of a failure condition.
- The key parts of this framework are a notification system based on the delivery of acknowledgement messages and a series of delivery assurances that provide policies comprised of reliability rules.
- WS-ReliableMessaging is closely associated with the WS-Addressing and WS-Policy specifications.
- Reliable messaging significantly increases SOA's quality of service level and broadens its interoperability potential.

Correlation

- One of the fundamental requirements for exchanging information via Web services is
  - the ability to persist context or state across the delivery of multiple messages.

Correlation

- Correlation addresses this issue by requiring that related messages contain some common value that services can identify to establish their relationship with each other or with the overall task they are participating in.



Correlation in abstract

- In tightly bound communication frameworks the issue of correlated units of communication (individual transmissions) rarely arose.
- It is up to the message to introduce the concept of correlation to provide services with the ability to associate a message with others.
- This is achieved by embedding a value in the message that is propagated to all related messages.

Correlation in MEPs and activities

- Because MEPs are generic and non-business-specific in nature, MEPs and activities have no predefined notion of correlation.
- They are simple, conceptual building blocks incorporated and assembled by either custom-developed solutions that employ custom correlation identifiers and related processing logic or by specifications that impose proprietary forms of correlation.

Correlation in Coordination

- The context management framework provided by WS-Coordination establishes a sophisticated mechanism for propagating identifiers and context information between services.
- A separate activation service is responsible for creating new context and subsequently distributing corresponding context data.
- Services can forward this information to others that can use it to register for participation in the activity.

Correlation in Orchestration

- WS-BPEL orchestrations need to concern themselves with the correlation of messages between process and partner services.
- This involves the added complexity of representing specific process instances within the correlation data.
- Further complicating this scenario is the fact that a single message may participate in multiple contexts, each identified by a separate correlation value.
- To facilitate these requirements, the WS-BPEL specification defines specific syntax that allows for the creation of extensible correlation sets.

Correlation in Addressing

- WS-Addressing's message id and relationship MI headers provide inherent correlation abilities, which can be leveraged by many composition and messaging extensions.

Correlation in Reliable Messaging

- Every message that participates in a WS-ReliableMessaging sequence carries sequence information with it.
- This data consists of a sequence identifier that represents the series of messages required to follow the messaging rules of the sequence, along with a message identifier that identifies how the current message fits into the overall sequence.
- As a whole, this information can be considered correlation-related.

Correlation and SOA

- Correlation is a key contributor to preserving service autonomy and statelessness.
- Though simple by nature, the ability to tie messages together without requiring that services somehow manage this association is an important function of correlation, primarily because of how common message associations are in enterprise SOAs.

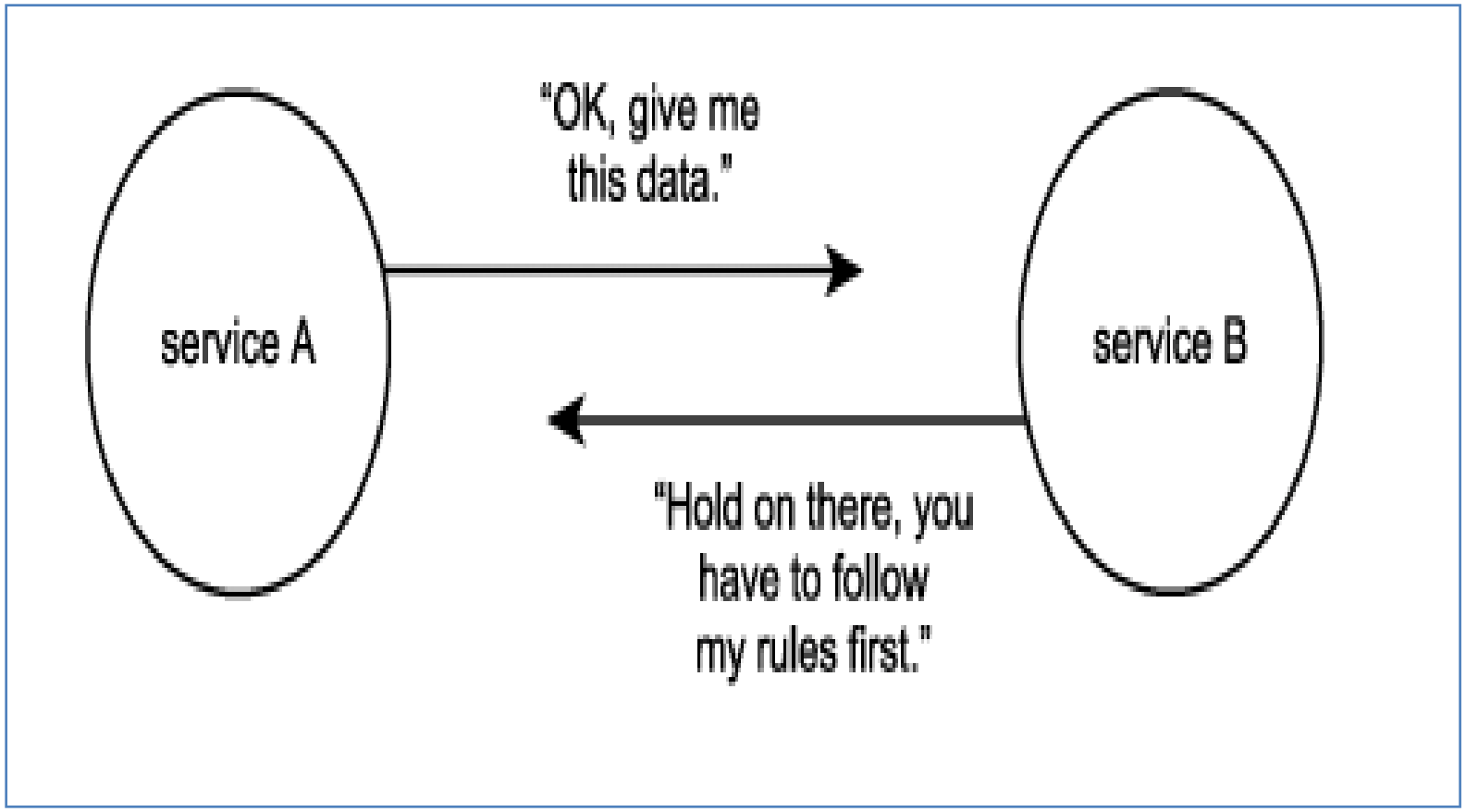
So far, Correlation

- Correlation is a required part of any SOA, as it enables the persistence of activity context across multiple message exchanges, while preserving the loosely coupled nature of service-oriented solutions.
- WS-* specifications implement correlation in different ways.
- Even though values from a message's content can be used for correlation purposes, SOAP headers are the most common location for correlation identifiers.
- Correlation is an essential part of messaging within SOA, as it preserves service statelessness and supports message autonomy.

Policies

- Every business task has rules and constraints:
  - Business Level Rules
  - Rules regarding the nature of data being exchanged
  - Rules regarding security measures

Policies


"OK, give me
this data."
service A
service B
"Hold on there, you
have to follow
my rules first."

Policies

- The use of policies allows a service to express various characteristics and preferences and keeps it from having to implement and enforce rules and constraints in a custom manner.

- It adds an important layer of abstraction that allows service properties to be independently managed.

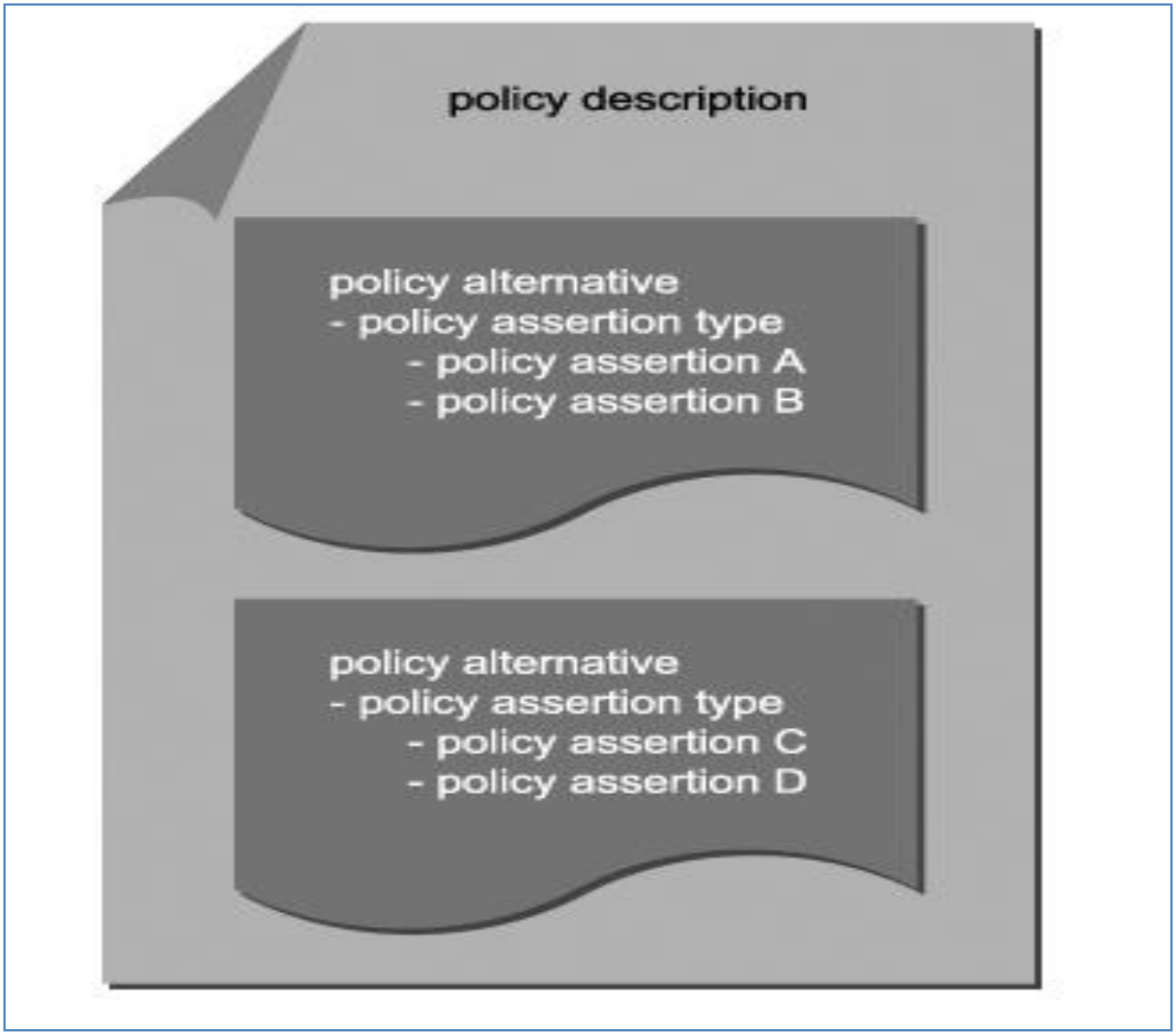
The WS-Policy framework

- The WS-Policy framework establishes extensions that govern the assembly and structure of policy description documents, as well as the association of policies to Web resources.
- The WS-Policy framework is comprised of the following three specifications:
  - WS-Policy
  - WS-PolicyAttachments
  - WS-PolicyAssertions

Use of Policies

- Policies can be programmatically accessed to provide service requestors with an understanding of the requirements and restrictions of service providers at runtime.

- Alternatively, policies can be studied by humans at design time to develop service requestors designed to interact with specific service providers.


policy description
policy alternative
- policy assertion type
- policy assertion A
- policy assertion B
policy alternative
- policy assertion type
- policy assertion C
- policy assertion D

Policy Assertions

- The service properties expressed by a policy description are represented individually by policy assertions.
- A policy description therefore is comprised of one or more policy assertions.
  - service characteristics
  - preferences
  - capabilities
  - requirements and rules

Policy Alternatives

- Policy assertions can be grouped into policy alternatives.
- Each policy alternative represents one acceptable (or allowable) combination of policy assertions.
- This gives a service provider the ability to offer service requestors a choice of policies.

Policy assertion types

- Policy assertions can be further categorized through policy assertion types.
- Policy assertion types associate policy assertions with specific XSD schemas.

Policy Subject and Policy Scope

- A policy can be associated with a Web service, a message, or another resource.
- Whatever a policy is intended for is called a **policy subject**.

- Because a single policy can have more than one subject, the collection of a policy's subjects is referred to as the **policy scope**.

Policy Expressions

- Policy assertions that are physically implemented using the WS-Policy language are referred to as policy expressions.
- In other words, a policy expression is simply the XML statement used to express a policy assertion in a manner so that it can be programmatically processed.

Policy attachments

- Policy expressions are physically bound to policy scopes using policy attachments.

Policies in coordination

- When the WS-Coordination context coordination service generates context information for participating services,
  - Security information and other related information is distributed to them

Policies in orchestration and choreography

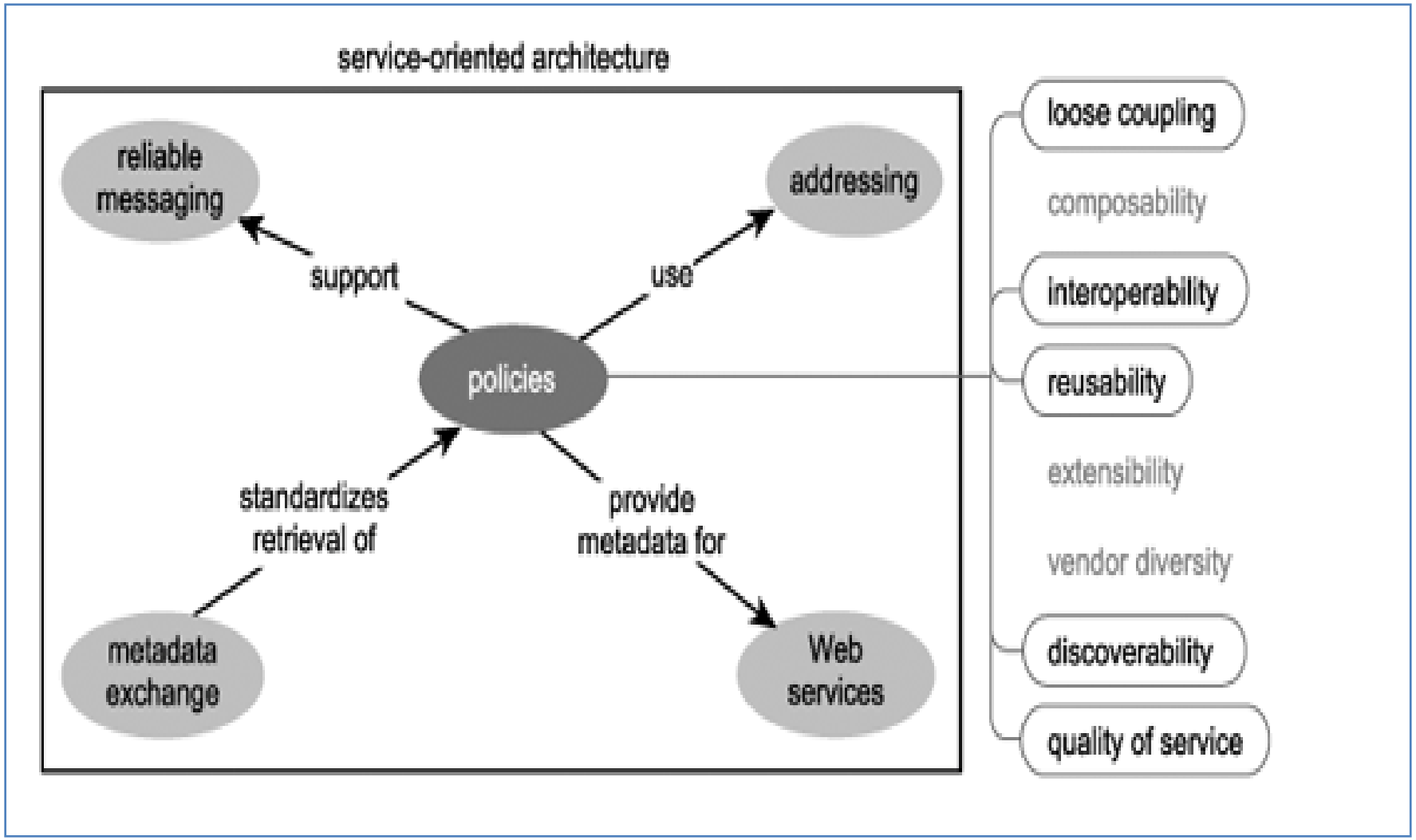
- Policies can be applied to just about any subjects that are part of orchestrations or choreographies.
- For example, a policy can establish various requirements for orchestration partner services and choreography participants to interact.

Policies in reliable messaging

- The WS-ReliableMessaging specification depends on the use of the WS-Policy framework to enable some of its most fundamental features.

- Policy information is attached in the form of delivery assurances to the messages that take part in Reliable messaging

Policies and SOA



So far, WS-Policies

- The WS-Policy framework provides a means of attaching properties to Web resources, most notably Web services.
- Individual properties are represented by policy assertions, which can be marked as optional or required.
- WS-Policy can be incorporated within the majority of WS-* extensions.
- Polices add an important layer of metadata to SOAs that increases the interoperability and discovery potential for services, while also elevating the overall quality of messaging within SOA.