Prof. Dr.-Ing. Stefan Deßloch
AG Heterogene Informationssysteme
Geb. 36, Raum 329
Tel. 0631/205 3275
dessloch@informatik.uni-kl.de

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Chapter 9 – Web Services Coordination and Transactions

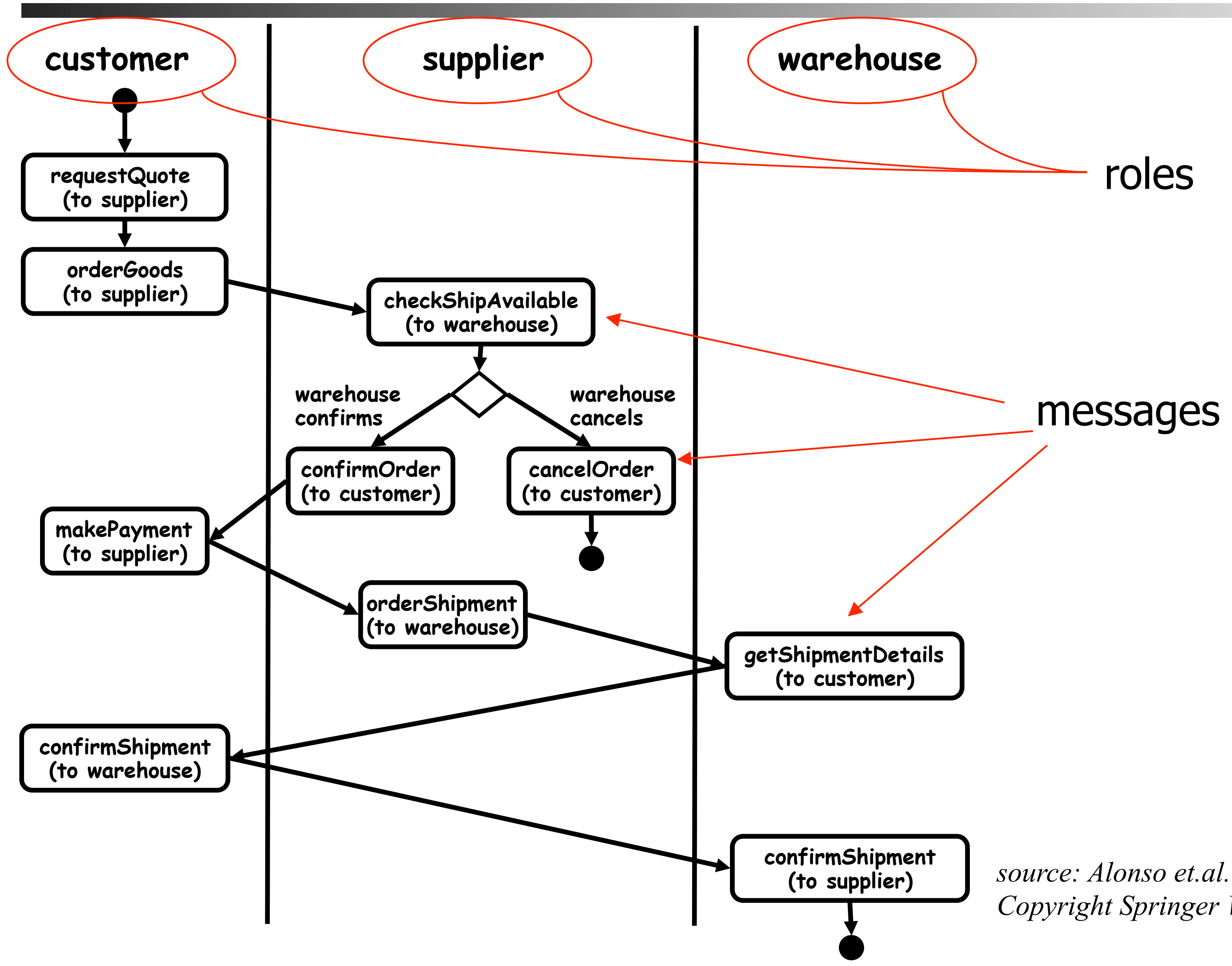
Coordination - Motivation

- Interactions are typically more complex than simple invocations
- Need to coordinate (sets of) activities or applications
  - Distributed
  - Running on different platforms using local coordinators
- Examples
  - Reach consistent agreement on the outcome of distributed transactions
    - Atomic transactions, 2PC
  - Coordinate auctioning activities
    - involves seller, auctioneer, buyers
  - Interactions between a customer and a supplier for ordering a product
    - request order, order goods, make payment

Conversations and Coordination Protocols

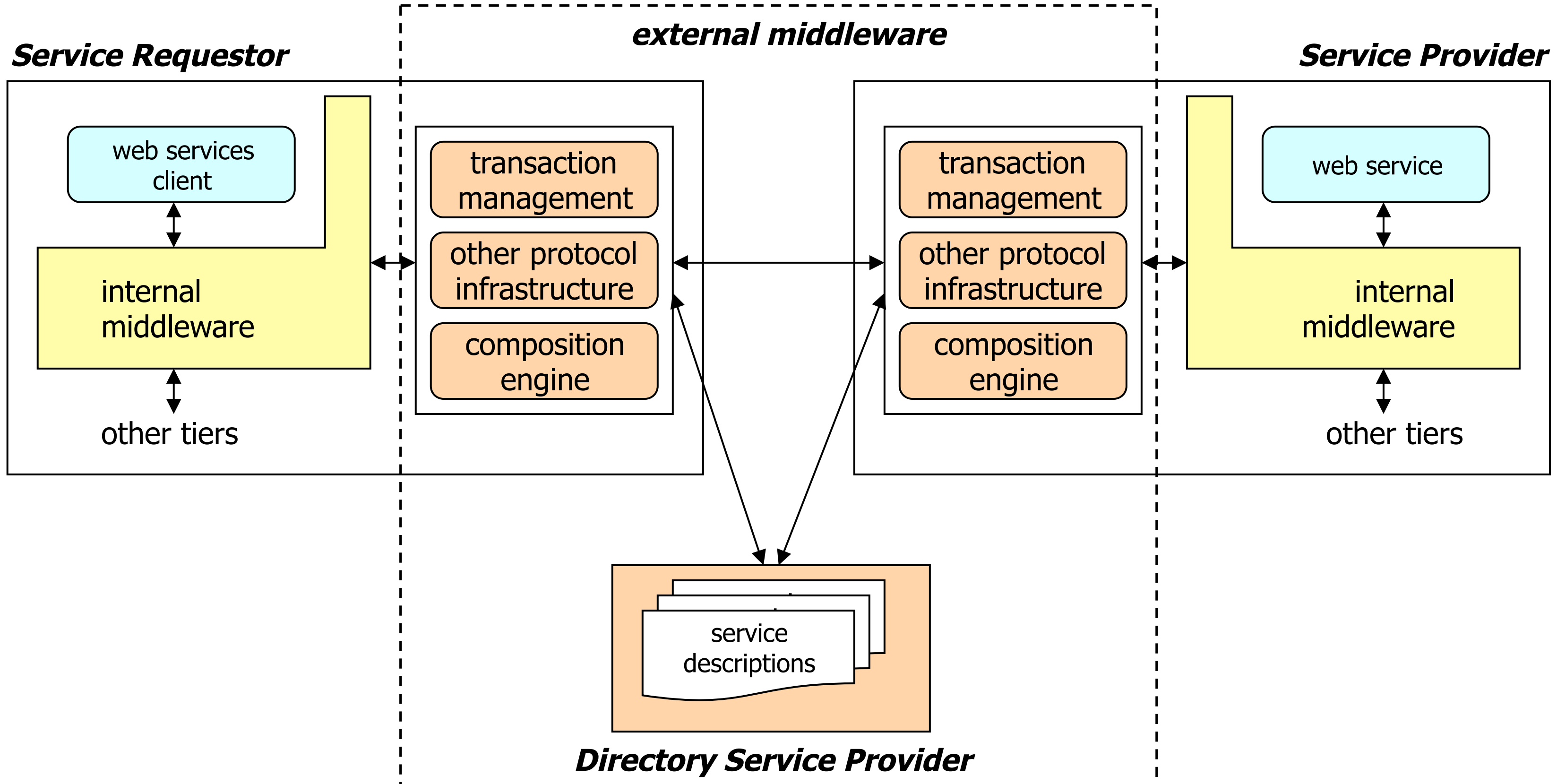
- Interactions form a ***conversation***
  - sequences of operations (message exchanges)
    - maintain context information across invocations
- Interactions adhere to a ***coordination protocol***
  - specifies a set of correct/accepted conversations
  - *vertical* protocols: specific to business area (e.g., product ordering protocol)
  - *horizontal* protocols: define common infrastructure (e.g., transactions)
- Different ways of modeling conversations
  - state machines
  - sequence diagrams
  - activity diagrams
- Middleware support can be provided, with various degrees of automation
  - conversation controllers
  - generic protocol handlers

Modeling Protocols - Activity Diagrams


customer
supplier
warehouse
roles
requestQuote (to supplier)
orderGoods (to supplier)
checkShipAvailable (to warehouse)
warehouse confirms
warehouse cancels
messages
confirmOrder (to customer)
cancelOrder (to customer)
makePayment (to supplier)
orderShipment (to warehouse)
getShipmentDetails (to customer)
confirmShipment (to warehouse)
confirmShipment (to supplier)

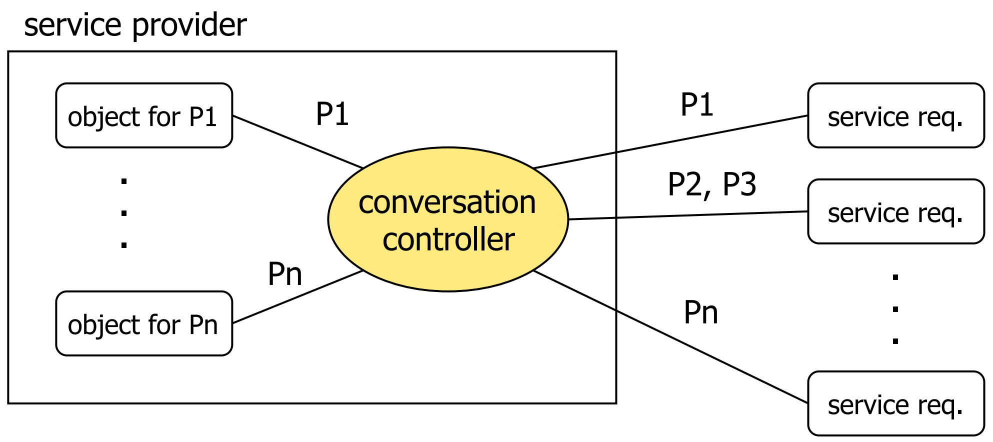
source: Alonso et.al.: Web Services, Springer, 2003
Copyright Springer Verlag Berlin Heidelberg 2003

External Web Services Architecture



Conversation Controller

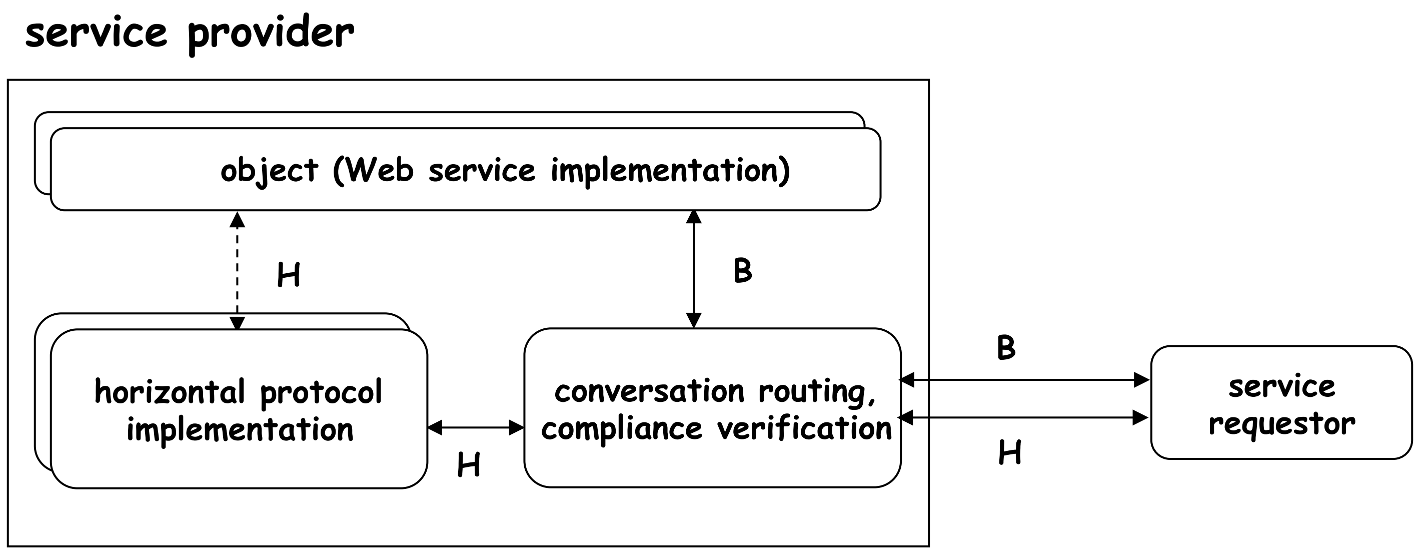
- Performs *conversation routing*
  - dispatch message to the appropriate "internal object"
    - one object for each instance of a conversation (e.g., an ordering session)
  - involves message correlation (conversation identifier), management of conversation context
    - example: session id
- Verifies *protocol compliance*
  - understand definition of the protocol (-> standardization of protocol descriptions)
  - check if all messages adhere to the protocol definition
- Can be implemented as a component of a SOAP router



Generic Protocol Handlers

- Module that implements a specific coordination protocol
  - includes protocol-specific logic
  - processes and generates messages in accordance with the protocol rules
- Mostly applicable to horizontal protocols
  - example: transactions
- Forms of protocol execution support
  - handler realizes complete support, no intervention from the web service
    - Example: reliable messaging
  - handler and web service jointly realize the support
    - Example: atomic, distributed TAs
      - infrastructure coordinates sending/receiving prepare/commit/abort messages
      - web services decide over commit/abort, implement operations

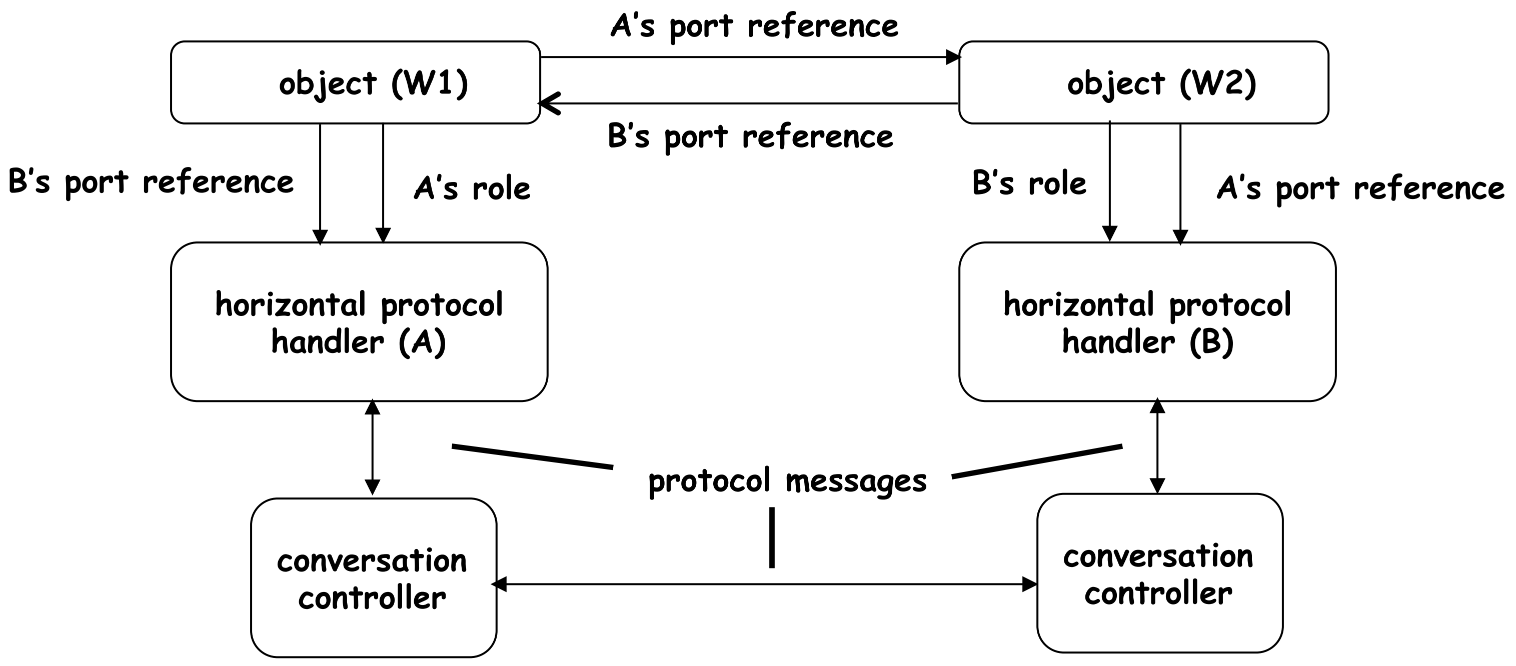
Implementing Horizontal Protocols


service provider
object (Web service implementation)
H
B
horizontal protocol implementation
H
conversation routing, compliance verification
B
H
service requestor

B: conversation compliant with a business protocol
H: conversation compliant with an horizontal protocol

source: Alonso et.al.: Web Services, Springer, 2003
Copyright Springer Verlag Berlin Heidelberg 2003

Communicating Roles and Port References



source: Alonso et.al.: Web Services, Springer, 2003
Copyright Springer Verlag Berlin Heidelberg 2003

Standardization

- Coordination infrastructure support for web services needs to be based on standards for
  1) generating and transporting unique conversation identifiers in SOAP headers
     - needed to map messages to conversations, and eventually to the objects handling them
  2) a framework and a set of (meta-) protocols for agreeing on which protocol is to be executed and how it is coordinated
  3) horizontal protocols
     - to separate horizontal protocol implementation from the inidividual web services
  4) protocol languages
     - to allow for protocol verification
- Web Services Coordination (WS-Coordination) Specification
  - standardizes 1), 2)
- Web Services Atomic Transaction (WS-AtomicTransaction) Specification
  - uses WS-Coordination framework to define coordination type for Atomic Transactions (i.e., it standardizes 3) for atomic TAs)
- Web Services Business Activity Framework (WS-BusinessActivity) Specification
  - same for (long-running) business transactions
- Standardized by the OASIS WS-TX technical committee
  - initial proposals by BEA, IBM, IONA, Microsoft

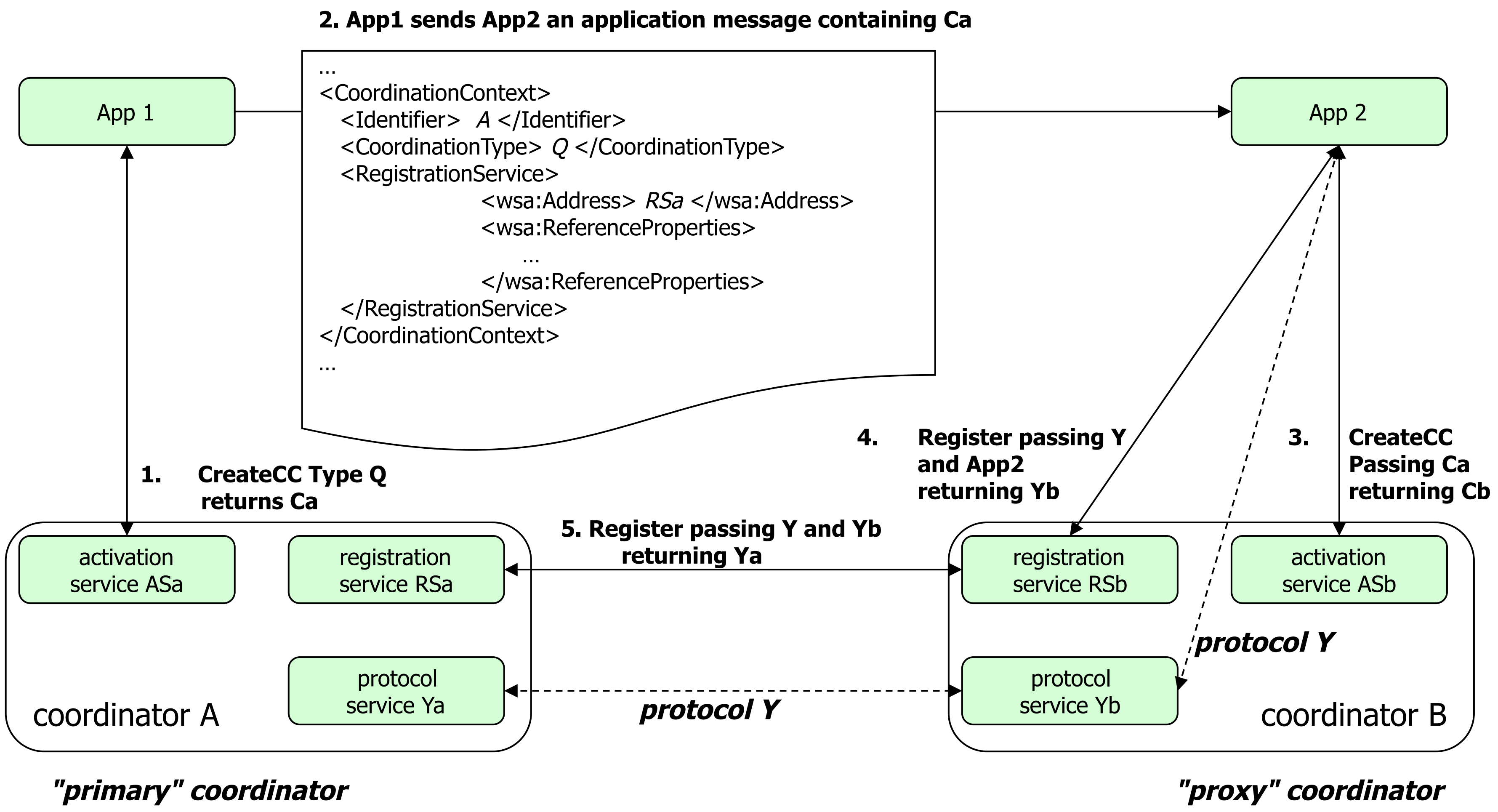
WS-Coordination

- Basic entities are **coordinators** and **participants** that wish to be coordinated
  - central coordination: all participants talk to a single coordinator
  - distributed coordination
    - each (or multiple) participant talks to its own coordinator
    - coordinators are chained together (subordinate coordinators act as participants)
- Abstractions to describe the interactions between coordinator and participants
  - coordination protocol
    - set of rules governing the conversation
      - example: 2PC
  - coordination type
    - set of logically related protocols
      - example: atomic transactions (completion, 2PC, volatile 2PC)
    - instance of a coordination type may involve several instances of the coordination protocols
- Coordination context
  - used to exchange coordination information among different parties
    - contains coordination type, identifier of the coordination type instance
    - placed within messages exchanged between parties (SOAP header)

Coordinator/Participant Interactions

- Coordination service (coordinator) consists of
  - **Activation** service (generic)
    - Used by a participant to create coordination context (initiate instance of protocol type)
    - WS Interfaces: ActivationCoordinator, ActivationRequester
  - **Registration** service (generic)
    - Enable application to register for coordination protocols
      - provide endpoint information, role
    - WS Interfaces: RegistrationCoordinator, RegistrationRequester
  - (set of) **coordination protocols** (protocol-specific)
    - Specific to coordination type
- Extensibility
  - Publication of new coordination protocols
  - Definition of extension elements that can be added to protocols and messages

Distributed Coordination - Interactions

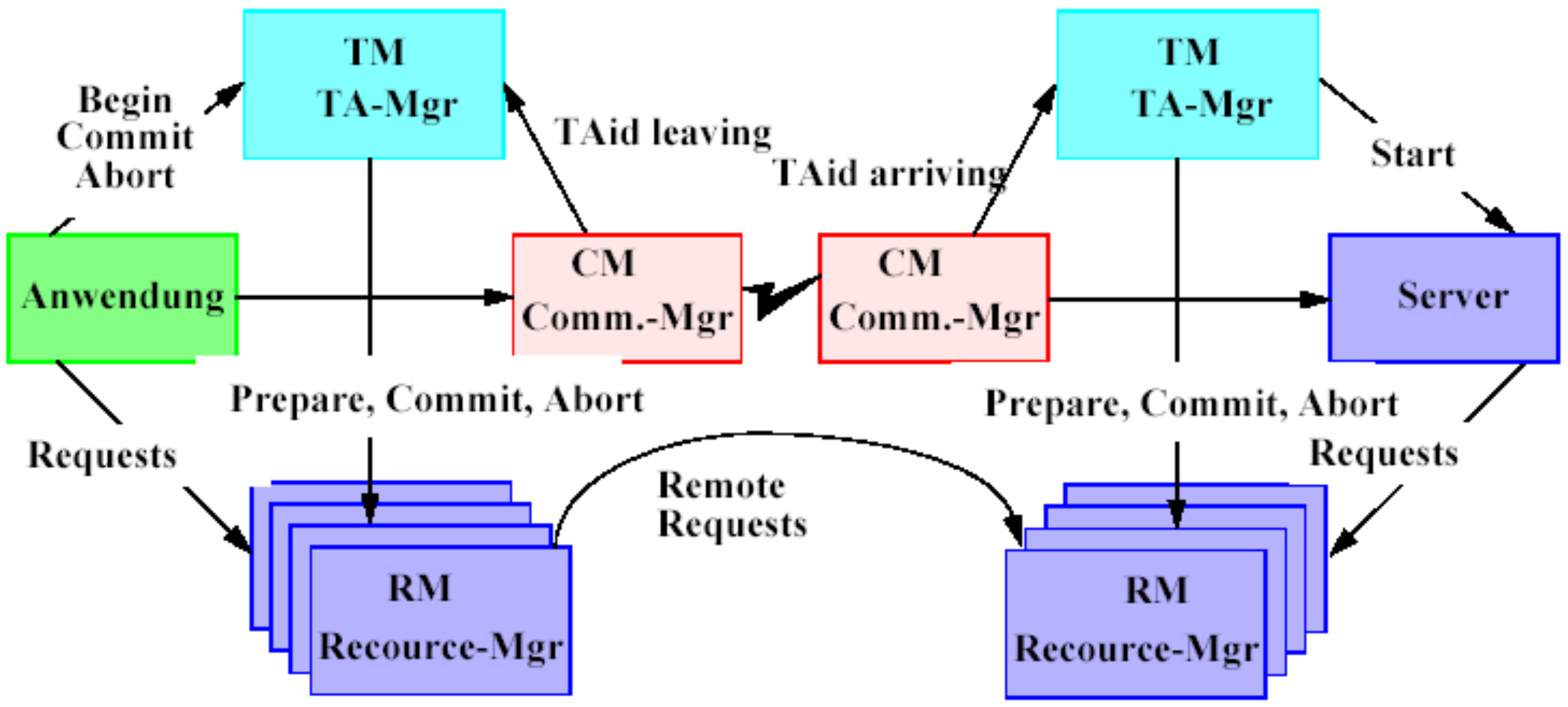


```
…
<CoordinationContext>
   <Identifier>  A </Identifier>
   <CoordinationType> Q </CoordinationType>
   <RegistrationService>
                  <wsa:Address> RSa </wsa:Address>
                  <wsa:ReferenceProperties>
                       …
                  </wsa:ReferenceProperties>
   </RegistrationService>
</CoordinationContext>
…
```

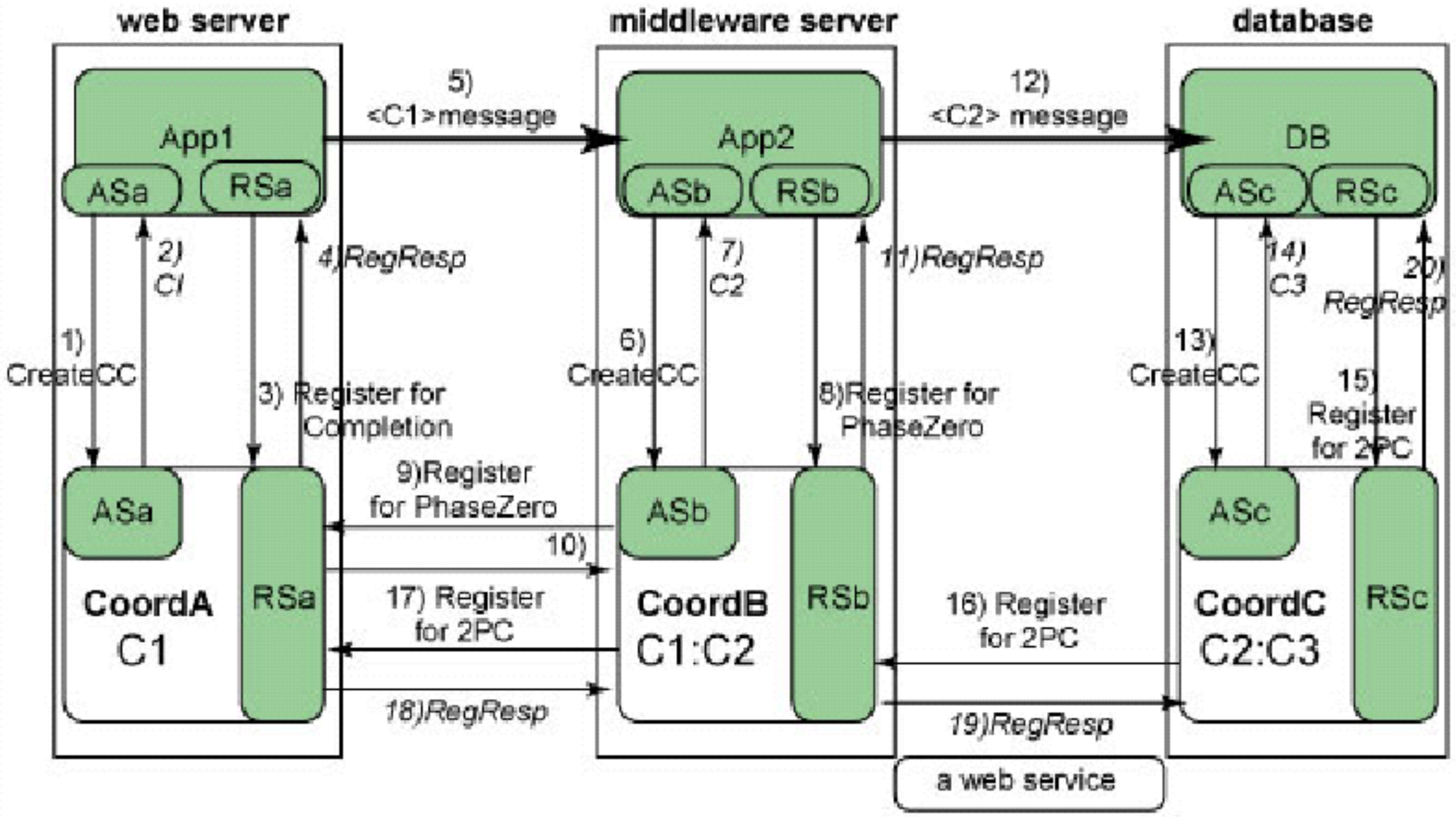
WS Atomic Transactions

- Atomic Transactions (TA) coordination type
  - Defines type-specific commit protocols
    - Completion: A participant (app creating the TA) registers so that it can tell the coordinator when/how to complete the TA (commit/abort)
    - 2PC: a resource manager (RM) registers for this protocol to be included in the commit/abort decision
      - Hierarchical 2PC (local coordinators can be interposed as subordinate coordinators)
    - Two variants of 2PC
      - volatile 2PC: a participant wants to be notified by the coordinator just before the 2PC begins
        - Example: participant caches, needs to communicate changes on cached data to DBMS before TA commits
      - durable 2PC: a participant (e.g., DBMS) manages durable resources
  - Completion must be registered with the root coordinator
  - Participants can register for more than one protocol
  - Extension elements
    - Example: communicate isolation levels

X/Open DTP revisited …



AT WS-Coordination Flow



AT WS-Coordination Flow (cont.)

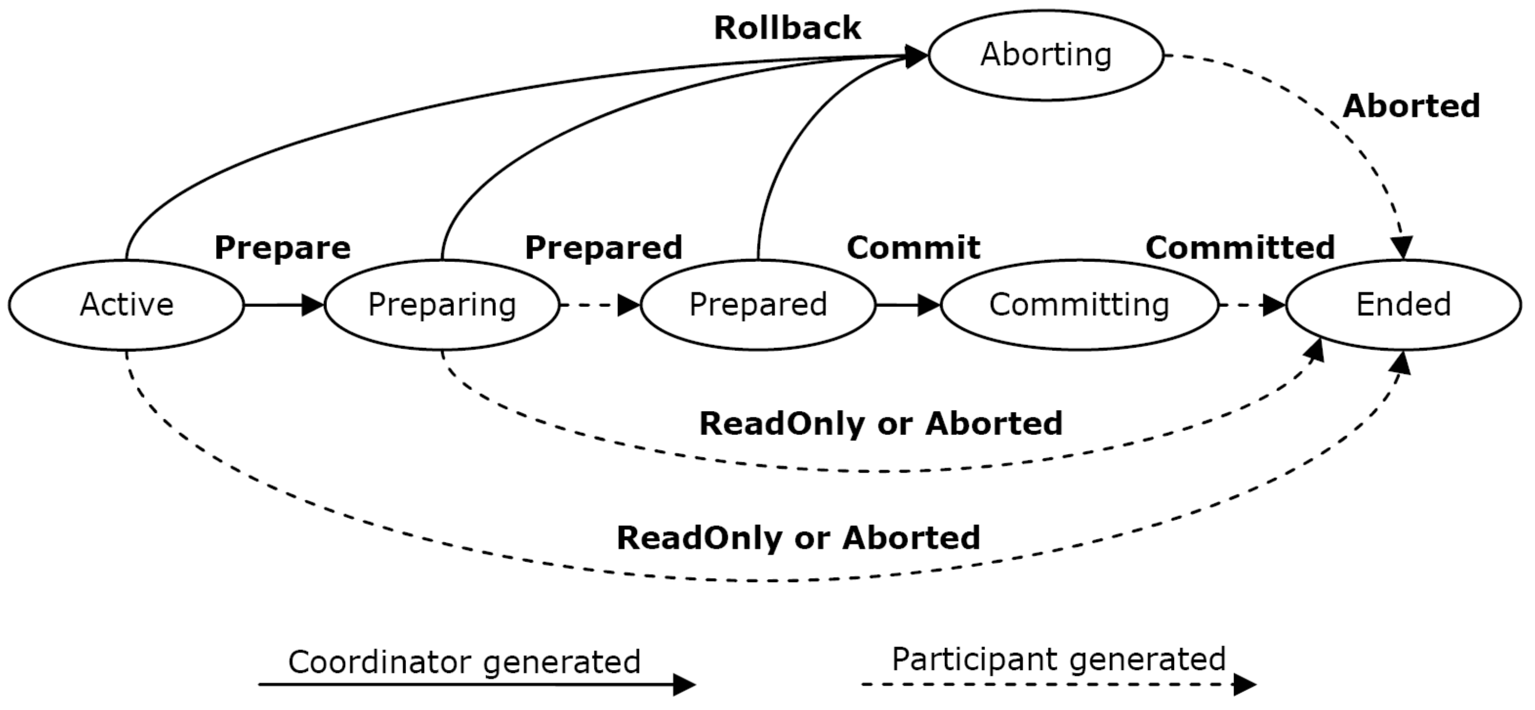
- App1:
  - sends a CreateCoordinationContext message (1) to its local coordinator's Activation service ASa
    - create an atomic transaction T1
    - gets back in a CreateCoordinationContextResponse message (2) a CoordinationContext C1 containing the transaction identifier T1, the atomic transaction coordination type and CoordA's registration address RSa
  - sends a Register message (3) to RSa to register for the Completion protocol
    - gets back a RegisterResponse message (4), exchanging protocol service addresses for the coordinator and participant sides of the two-way protocol
  - sends an application message to App2 (5)
    - propagating the CoordinationContext C1 as a header in the message.
- App2:
  - decides to interpose local coordinator CoordB in front of CoordA
    - acts as a proxy to CoordA for App2
    - CoordA is the superior and CoordB is the subordinate
  - does this by sending a CreateCoordinationContext message (6) to the Activation service of CoordB (ASb) with C1 as input
    - getting back (7) a new CoordinationContext C2 that contains the same transaction identifier (T1) and coordination type, but has CoordB's registration address RSb.
  - registers with CoordB for the PhaseZero (volatile 2PC) protocol (8 and 11)
    - CoordB registers with CoordA for the PhaseZero protocol (9 and 10)
  - sends a message to DB (12), propagating CoordinationContext C2

AT WS-Coordination Flow (cont.)

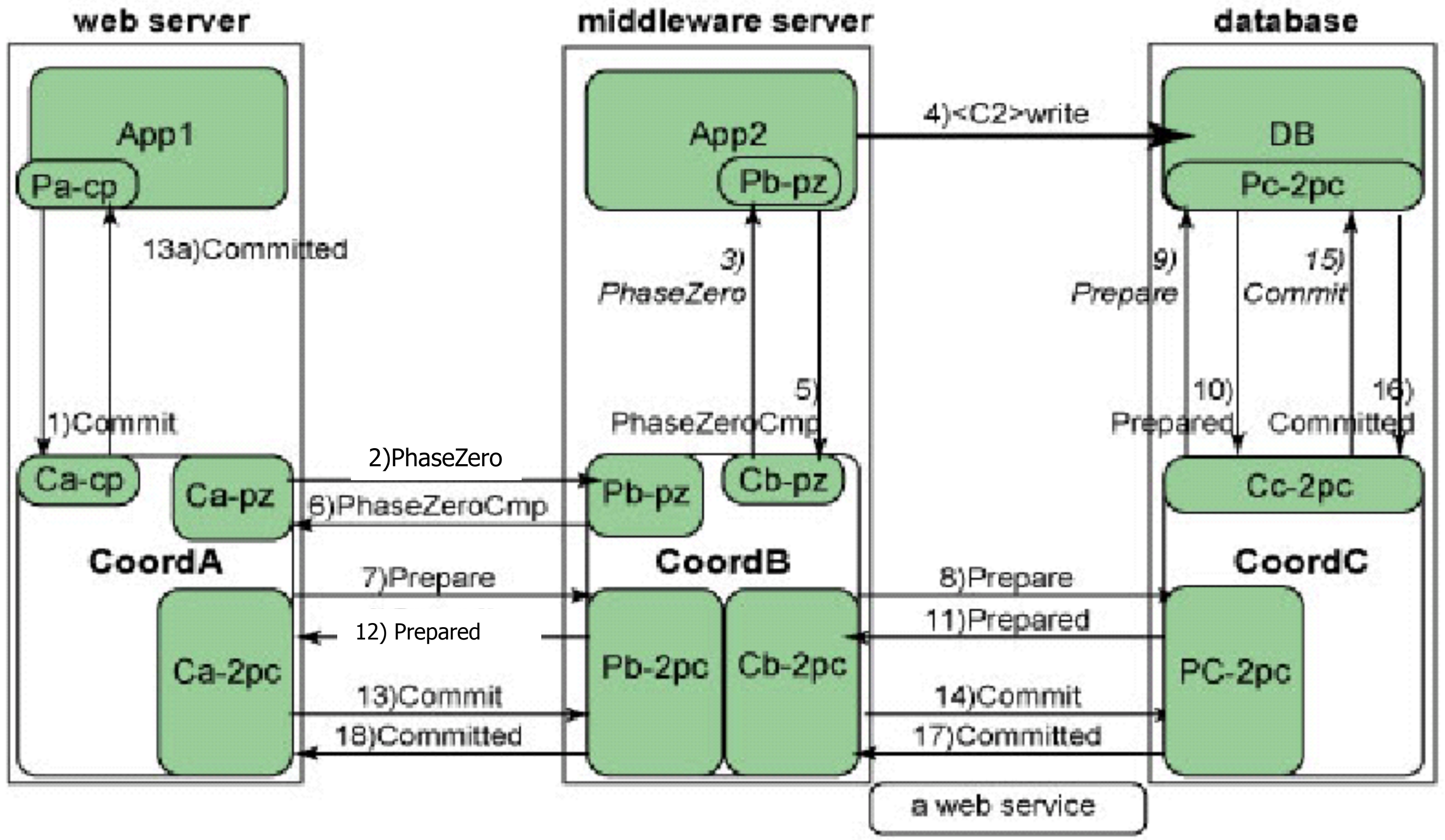
- DB:
  - decides to interpose its local coordinator CoordC by sending a CreateCoordinationContext message (13), further extending the superior-subordinate chain
    - gets back (14) a new CoordinationContext C3 that contains the same transaction identifier (T1) and coordination type, but CoordC's Registration service address RSc
  - registers with CoordC for the 2PC protocol because it is a resource manager (15 and 20)
  - causes CoordC to register with CoordB for the 2PC protocol (16 and 19)
  - causes CoordB to register with CoordA for the 2PC protocol (17 and 18)

AT – 2PC Protocol

- Two-way protocol
  - Exchange of messages between coordinator and participant
- State Diagram
  - State reflects common knowledge of both parties



AT Coordination Protocol Flows



AT Coordination Protocol Flows (cont.)

- App1:
  - tries to commit the transaction using the Completion protocol (1)
- CoordA executes prepare-phase of **Volatile 2PC protocol**
  - has 1 participant registered for PhaseZero (CoordB), sends a Prepare message (2) to CoordB's PhaseZero Participant protocol service Pb-pz
  - CoordB relays Prepare message to App2 (3)
  - App2 sends its cached updates to DB
    - application message (4) propagates the CoordinationContext C2
    - sends a Prepared message (5) to CoordB
- CoordA executes prepare-phase of **durable 2PC protocol**
  - sends a Prepare message (7) to CoordB's 2PC Participant protocol service Pb-2pc
  - CoordB sends Prepare message (8) to CoordC's 2PC Participant protocol service Pc-2pc
  - CoordC tells DB to Prepare (9)
- CoordA commits
  - sends Commit message (13) to CoordB
    - Committed notification to App1 (13a) can also be sent
  - CoordB sends Commit message (14) to CoordC
  - CoordC tells DB to commit T1
    - DB receives the Commit message (15) and commits
  - Committed message returns (16, 17 and 18)

WS-BA – Business Activities Framework

- Characteristics (see discussion in chapter on WfMS)
  - Usually long-running
    - Responding to a request may take a long time
  - May consume lots of resources, perform a lot of work
    - Early commit of atomic subactivities/transactions
    - Forward recovery, compensation
- Goal: define protocols that "wrap" proprietary business activity mechanisms to achieve interoperability
- Design points
  - State transitions need to be reliably recorded
  - All request messages are acknowledged
    - Detect problems early
  - Response to a request is a separate operation
    - Not the output of the request
    - Avoid problems with timeouts of message I/O implementations

Compensation

- An action used to logically undo the effects of another action is called compensation action
  - Extends to real world actions
    - drilling a hole: throw away part
  - Semantic Recovery: Recovery schema based on compensation
  - Compensation very likely one of today's most frequently exploited techniques in transaction processing
- Compensation action is often dependent on context
  - E.g. writing an offer and sending it via mail to a customer
    - If letter is still in outbasket, simply remove it from outbasket
    - If letter is already received by the customer, write and send a countermanding letter
- Compensation often cannot recreate the same state that existed before the proper action had been performed
  - E.g. canceling a flight might cost a cancellation fee
    - Even more complicated, the cancellation fee might depend on the point in time, i.e. it is higher the later the cancellation is requested
- Compensation action may fail!

Sagas – Transactions and Compensation

- Sagas support specification of compensation actions in advance and run them automatically on abort
  - Sequence of (Sub-)Transaction/compensating action pairs
  - DBMS guarantees LIFO execution of compensation actions during abort/rollback of Saga
  - ACID for each sub-TA

Definition:

A Saga is a sequence [(T1,C1),..., (Tn,Cn)] having the following properties:

1. T1,...,Tn and C1,...,Cn are two sets of transactions, such that Ci is the compensation function for Ti,
2. [(T1,C1),..., (Tn,Cn)] is executed as one of the following sequences:
   i. [T1,...,Tn], if all Ti committed, or
   ii. [T1,...,Ti, Ci-1,..., C1] if Ti aborts and T1,...,Ti-1 committed before.

Business Activities Model

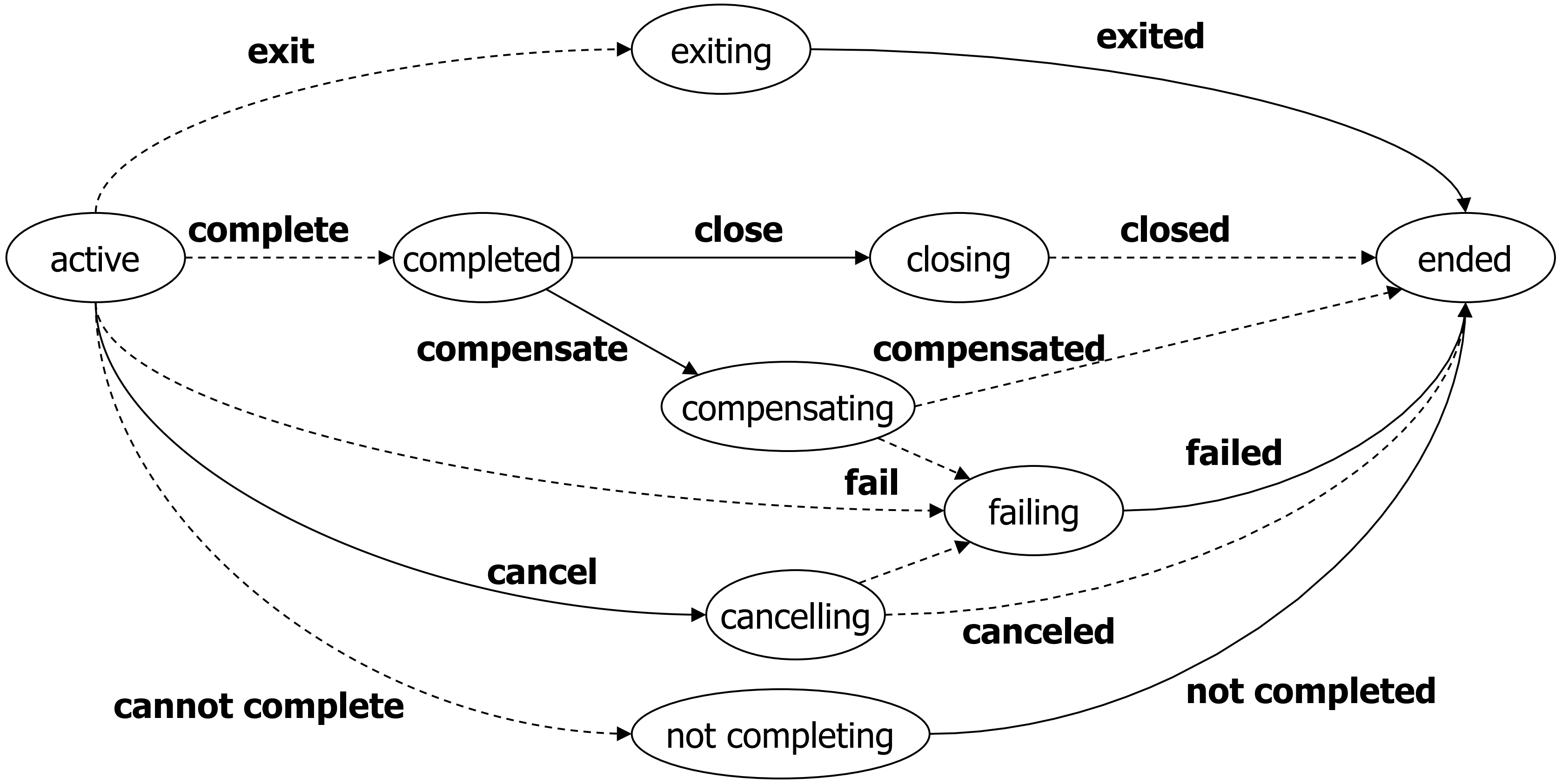
- Application is partitioned into business activity scopes
  - carries out business tasks using web services (participants)
  - mutually agreed outcome of all participants
- Participants registered with a coordinator of a BA
  - notify the coordinator about (successful) completion
  - may be asked by the coordinator to cancel an active task or to compensate a completed task
  - may indicate that it
    - cannot complete the task (and has cancelled it)
    - is leaving (exit) the BA (and has cancelled it)
    - has failed (during regular activities, when compensating or cancelling the task)
      - state of work is undetermined!
- Scopes may be arbitrarily nested

Business Activity (cont.)

- Business Activity (BA) coordination types
  - AtomicOutcome: coordinator directs all participants to either close or compensate
  - MixedOutcome: coordinator may direct some participants to close, others to compensate
- BA protocol types
  - BusinessAgreementWithParticipantCompletion protocol
    - participant must know when it has completed all the work for a business activity
  - BusinessAgreementWithCoordinatorCompletion protocol
    - participant relies on coordinator to tell it when it has received all requests for work in the business activity

Business Agreement Protocol

- BusinessAgreementWithParticipantCompletion – State Diagram



Summary

- Coordination protocols
  - protocol defines set of correct conversations (WS message exchanges)
    - involves multiple partners, roles implemented as web services
  - vertical vs. horizontal protocols
  - different modeling approaches (e.g., activity diagrams)
- Infrastructure
  - conversation controller for internal routing based on conversation identifier
  - generic protocol handlers for horizontal protocols
- Coordination protocol infrastructure
  - WS-Coordination as a framework for supporting coordination protocols
  - central vs. distributed coordination
- WS-Transaction
  - based on WS-Coordination infrastructure
  - atomic transactions vs. business activities