University of Aizu

COV888:Building Reliable Networked Applications

2016 Transactions and Web Services

References: CACM, October 2003, Vol. 46, No. 10 (Mark Little)

[I] Distributed System (Algorithms) [2]

- Centralized Algorithms

Coordinator Site $\longrightarrow$ sites connect; Submit request to access resources at distributed databases

- Distributed Algorithms

Resourc Coordinators $\longrightarrow$ Many resource Coordinators

$\longrightarrow$ Prevent the Cenral Coordinator failure problem.

- Peer-to-peer (P2P) Algoritms

Fully Distributed Algorithms

- Current Protocols on LAN based Systems $\longrightarrow$ are often Distributed Algorithms

[I] Distributed System (Algorithms) [3]

CASE 1 $\longrightarrow$ Distributed Deadlock Detection

+ Blocked Transaction $\longrightarrow$
Generate TWFG for its own detection (Distrubted approach)

+ Blocked Transaction $\longrightarrow$
Send out a Probe for its own detection (Distrubted approach)

- **Existing Algorithms do not consider P2P solutions**

$\longrightarrow$ Cost of P2P algorithms is high
(example - Distributed Mutual Exclusion).

[I] Distributed System [4]

- $\boxed{\text{CASE 1}}$ $\longrightarrow$ Distributed Commit Algorithm

- 2 Phase Commit

Transaction Aromicity $\longrightarrow$ All sites must commit a Transaction or reject it

- Coordinator

- Informs about a request + in Phase 2 obtains an agreement

Distributed Algorith does not use a P2P solution

[I] Distributed System [5]

- LECTURE 1 - Air Traffic Controls Application over Cloud Computing

- P2P Solution (else Network Partitioning ? 2 Coordinators ?)

- LECTURE 5 - Web Services for E-Commerce Applications

- - P2P Solution (except the directory services)

[I] Distributed System [6]

client

- Dynamic Discovery $\longrightarrow$

finds a service

service

- Dynamic Discovery $\longrightarrow$

finds a client

outline

- [A] Dynamically
- [B] Over the network
- [C] No prior knowledge of each other
- [D] Based on IP Multicast
- [E] Need ! $\longrightarrow$
- Automatic Self healing network
- [F] Show status of all available and running services

[I] Example - Jini [7]

- $\boxed{\text{Jini}}$ $\longrightarrow$ also called Apache River

$\longrightarrow$ a network architecture for the construction of distributed systems in the form of modular co-operating services.

$\longrightarrow$ Originally developed by Sun,

$\longrightarrow$ released under an open source license (Apache license

[I] Example - Jini [8]

$\longrightarrow$ Reference: Jini on Wikipedia

$\longrightarrow$ Services try to contact a lookup service (LUS), by dynamic multicast discovery.

$\longrightarrow$ The strategy is more convenient than Java remote method invocation, which requires the client to know the location of the remote service in advance.

[I] Example - Jini [9]

Jini uses a lookup service to broker communication between the client and service.

This uses a centralized model

It does not scale well to very large systems.

[I] Summary - Jini [10]

- Jini - Distributed System based on LAN

- Shows status of all available and running services

- No human administration
- If a service fails or stops $\longrightarrow$

automatic update of system information

- Jini - Maintains a location service (directory)
- Based on JAVA (minimum support + JAVA)

[I] Example - J2EE Ref: Wikipedia [11]

- Vendor Independence (use J2EE compliant server)
- company (use J2EE compatible clients)

J2EE

- [A] server supports :

Security

Transactions: Concurrency, logging, 2 phase commit

persistence

networking code implementation

- [B] Based on JAVA
- [C] Based on LAN
- [C] Focus on learn one API - work on Business logic

[I] Web Services [12]

- Standards $\longleftarrow$ being worked out
- Tools $\longleftarrow$ to be developed
- Aim $\longleftarrow$ Ineroperability
- $\boxed{\text{Busines Applicatiions}}$ $\longrightarrow$

Old legacy application [+ Web Services] $\longrightarrow$

XML interface with SOAP messages

- Security : (!) HTTPS for mutual authentication; XML ? text
- Transactions : No support
- Dynamic Discovery [OK]
- $\boxed{\text{Not JAVA based}}$ $\longrightarrow$

J2EE may support Web Services interface for Web (outside)

[I] Service ? [13]

Steps	• 1. Send a message from one software system to another may not have human involvement
result	• Starts with software but may move a Robot
example	• [A] Send text to high speed printer • [B] Huge calculations for Genetic Match service • [C] Play a game of GO • [D] Buy concert tickets; Reserve travel • [E] move a camera • [F] Dail a phone, send a FAX
definition	• [Aim] Same as Object-Oriented programs - Have least amount of coupling - client $\longleftrightarrow$ Server $\longrightarrow$ know least about one another

[I] Service - How to use [14]

- How do clients find a service ?
- How to know what can be done ?
- Which methods to call ?

- $\longrightarrow$ Service needs to **declare** itself

- Service Interface - Jini / J2EE / Web Services

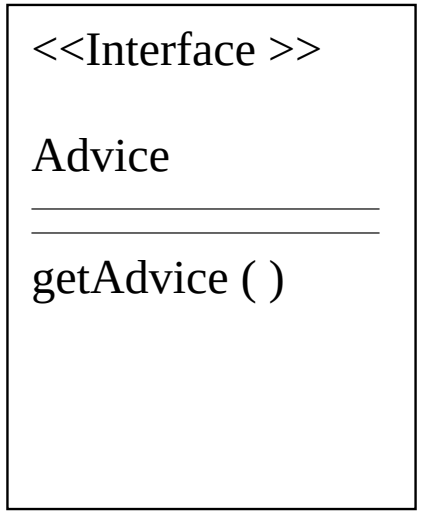
- use different ways

- Service - Methods !

- Arguments to Methods !

- return values

[I] Service Interface [15]



Interface : Advice

Method: one
: getAdvice()

Return Parameter: string
+ useful random advice

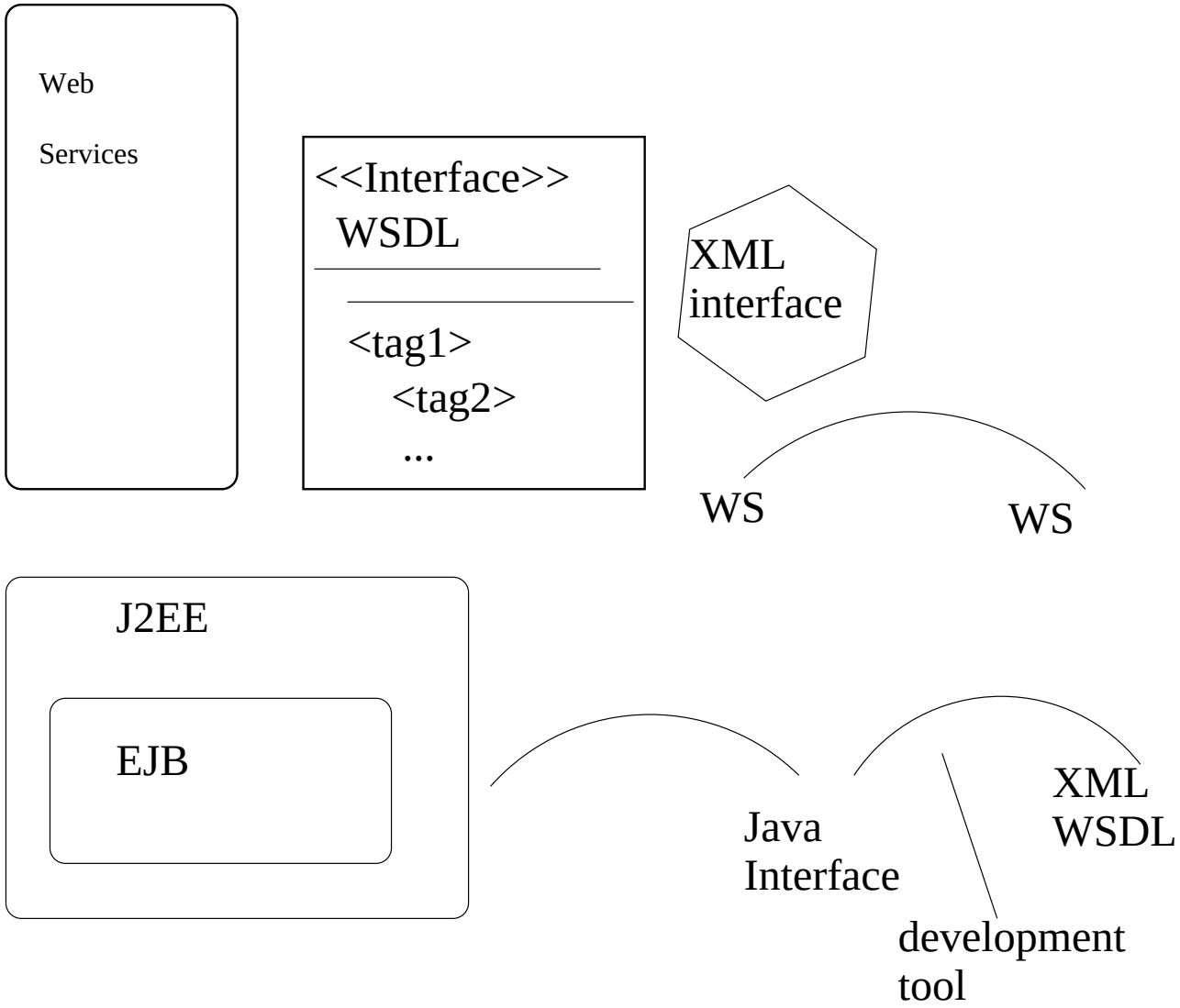
Input Parameter: null

- Jini and J2EE(EJB) : use JAVA Remote Interface (Based on RMI)

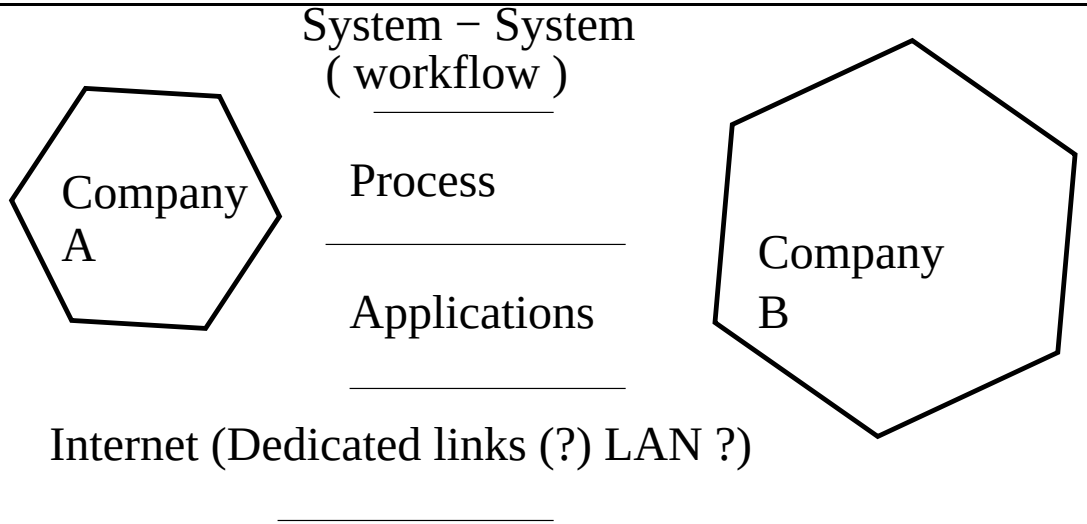
getAdvice() - declare a Remote Exception $\longrightarrow$ java.rmi.Remote

- [Jini $\longrightarrow$ Jini]; [J2EE $\longrightarrow$ J2EE]

[I] Web Service Model



[I] Inter-Enterprise Level [17]



- Car Maker (A) $\longleftrightarrow$ Part Supplier (B)
- Traditional Transaction Semantics [X]
- Traditional Transaction Protocols [X]

[I] Web Services Transactions [18]

- Long Duration of activity
- Commitment to be negotiated at run time
- Isolation Levels ?
- ACID properties ?
- Inter-Enterprise Level differences

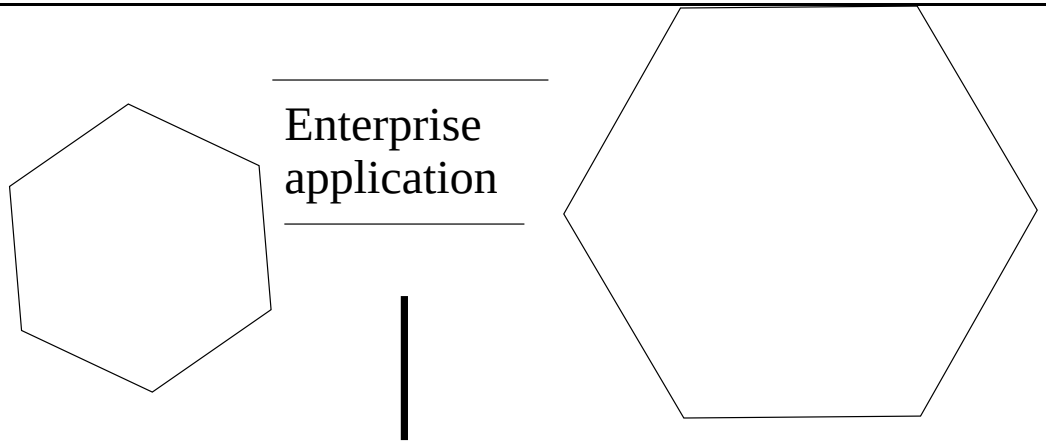
[I] Business Transactions (BT) [19]

BT	• Business Transactions (BT)
WST	• Web Services Transactions (WST)
Business Transaction	• [A] Need $\longrightarrow$ extended Transaction Model • [B] need $\longrightarrow$ defined Interoperable Transaction Protocol • [C] Message Flows $\longrightarrow$ help to negotiate transaction guarantees at inter-enterprise level

[I] Business Transaction Protocol (BTP) [20]

proposal	• Business Transaction Protocol (BTP)
OASIS	• Organization for Advanced Structured Information Systems
meets requirements	• [A] long running • [B] collaborative • [C] business applications • [D] allows relaxation of Isolation levels • [E] ACID property implementation in a controlled manner • [F] Suitable for Web Services Environment

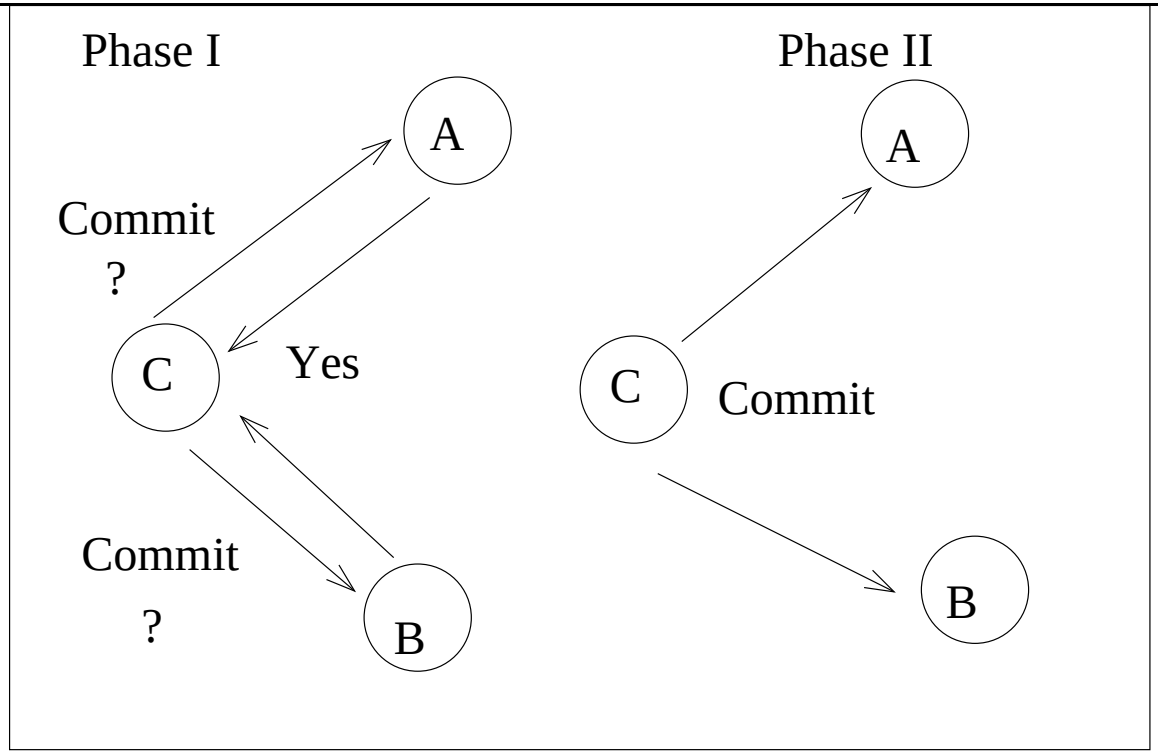
[I] Concept of Atomic Transactions [21]



Multi−party (multiple) business
operations −−−>>> Guaranteed
Consistent
Outcome

- Multiple participants with separate concerns
- final state $\longrightarrow$ Acceptable to all participants
- final state $\longrightarrow$ Consistent with business rules

[I] Concept of Atomic Transactions [22]



- Two Phase Commit Protocol

[I] Concept of Atomic Transactions [23]

State

- Consistent: Supported by rollback

Overall

- Final outcome - consistent with all

outline

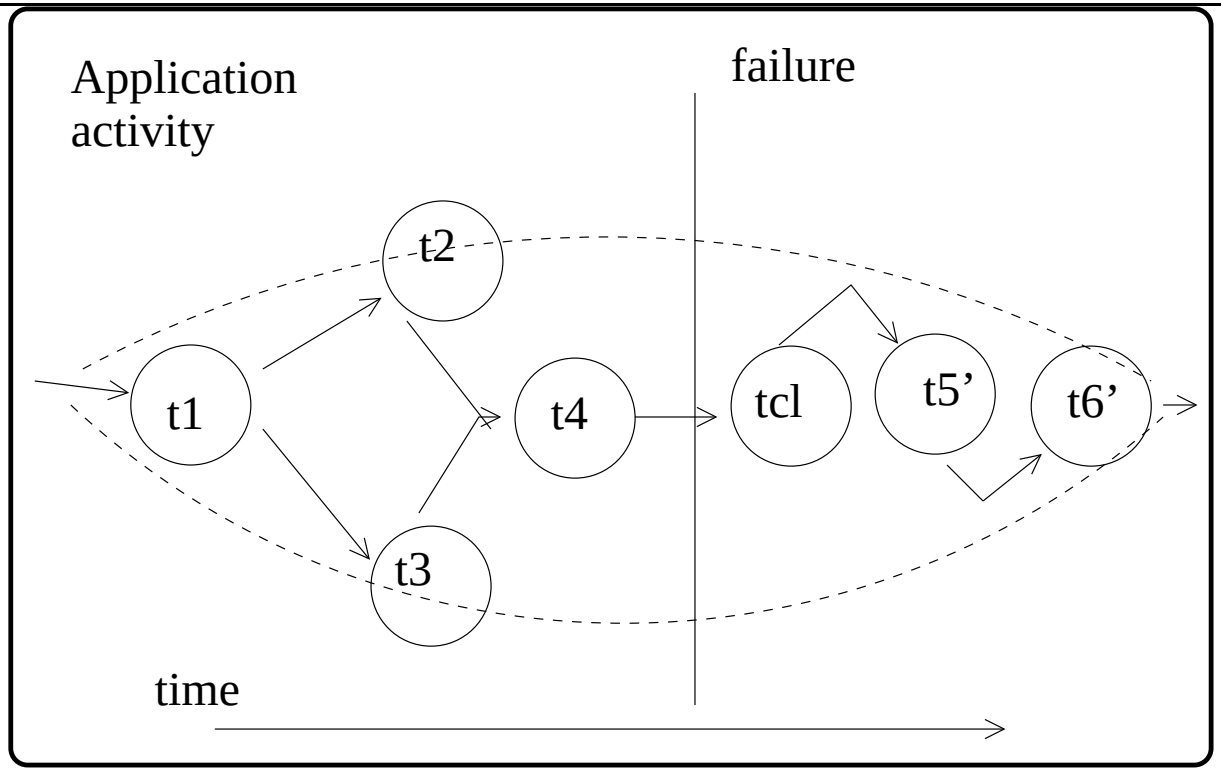
- [A] Concensus among parts
- [B] 1. State Changes $\longrightarrow$ temporary
$\longrightarrow$ LOG (used for redo/undo)
- [C] 2. Understand $\longrightarrow$ outcome at each level
- [D] 3. support $\longrightarrow$ completion (finish/cancel)
- [E] Steps (1.2.3) $\longrightarrow$ decided by workflow

[I] BT $\longrightarrow$ WS Transactions [24]

Long Transaction

- Internet - slow and asynchronous
- no guarantee
- Inter-enterprise business
- Tasks - send an order / wait for despatch
- More steps as verification , validation , quality control

[I] Concept of Atomic WS Transactions [25]



[I] ACID property

ACID

- too strong for Long Duration Transactions

Need

- Can guarantee consistency
in the presence of failures

Assume

- [A] Short transaction
- [B] Executes in closely coupled environment
- [C] Performs stable state changes

BTP

- Long lived Application Functions
- Copy ACID ideas
- [A] Phase I - make particpant changes durable (LOG)
$\longrightarrow$ rollback.commit at final steps
- [B] Protocol to get a concensus in Phase II
$\longrightarrow$ may overwrite earlier values
- [C] Problem $\boxed{\text{Blocking}}$ $\longrightarrow$ Long
- if coordinator fails before Phase 2

[I] Long-Running Activity [27]

- Long Transaction $\longrightarrow$ many independent short

duration transactions (t11 + t12 + t13 + ...

- Logical Long Running Transaction $\longrightarrow$ sum(t11+t12+ ...)

- purpose - structure of short duration transaction

... acquire and use resources for only the short required duration

- each ti $\longrightarrow$ a part$\longrightarrow$ a coordinated split part

[I] Example - Long running transaction [28]

2PC

- Not for all but only for sub-transaction

Overall

- A final check at completion time for activity

example

- [A] t1: Book a Taxi
- [B] t2: reserve a table at restaurant
- [C] t3: reserve a seat at a theater
- [D] t4:book room at a hotel
- [E] ...
- [Poit] if t2 - t6 do not need $\longrightarrow$

why block t1 resources until the end

- [Failure] (and if concurrant access) - run a compensating transaction [t4 aborts, tcl is run and t6 and t5 are changed]

[I] BTP

- BTP - Ensure $\longrightarrow$

atomicity between multiple participants

- 2 Phase Completion $\longrightarrow$

- prepare
- confirm
- cancel / finish

- Does not imply ACID

- Back-end Implementation $\longrightarrow$

+ details of prepare / confirm / cancel !
+ details of Consistency and Isolation

[I] Exercise [30]

- Use Traditional Models:

Ensure $\longrightarrow$ pair-wise communication (use atomicity between multiple participants)

- 1. Consider an interation based Deadlock detection procedure for web sevices environment using (a) Transaction wait for graph (TWFG) method, (b) Probe Method

- 2. Consider a pair-wise interation based transaction commit procedure for Web services environment