A Component Approach to Transactions

Initial Results from Zhixue Wu

Douglas Donaldson

APM Ltd.

9 January 1998

Internet Applications and Transactions

- Typically Three Tier Architecture
  - Presentation, Business Logic, Data Storage
- Transaction Processing (TP) Monitors offer a middleware environment oriented to handling transactions
  - Message-oriented: Microsoft Transaction System, BEA TUXEDO, JavaSoft JTS
  - Record-oriented: Remote ODBC, JavaSoft JDBC
- Problems
  - Weak support for middle tier development
  - No integration of transactions with application objects (file/record rather than object semantics)

Goals of our Open Approach

- High Transparency to Application Developers
- Separate business logic and system issues
  - for easy integration of business logic to a transaction framework
- Separate sequential behaviour (regular code) and concurrency semantics (declarative)
- Choose concurrency control strategies statically or dynamically as policies
- Integrate with FlexiNet

Strategy for our Open Approach

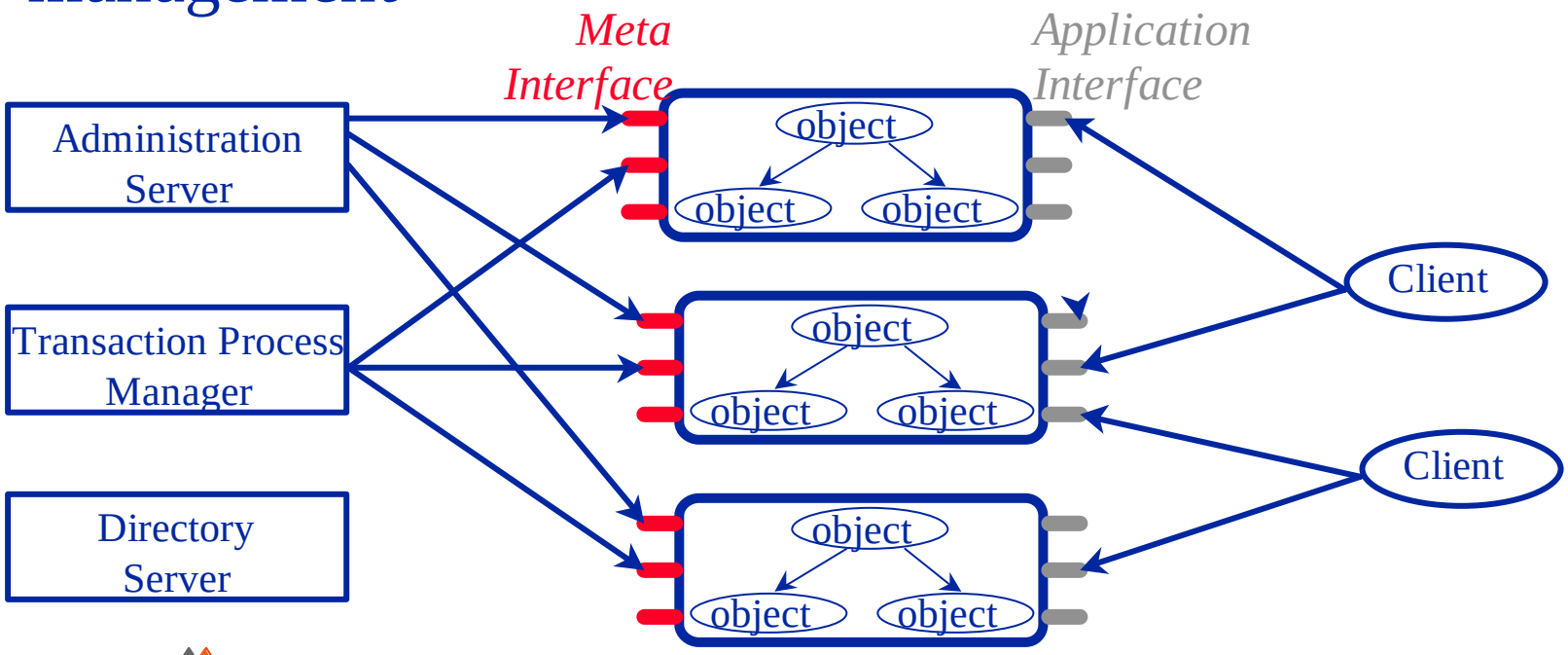
- Focus on middle tier
  - application objects representing work in progress
  - messages and data = objects!
- Component based
  - wrapping application objects inside Transaction Frames
  - compositional
  - making use of Java Beans
- Concurrency semantics specified declaratively
  - at the granularity of method invocation

Java Beans ?

- Software Component Model for Java
  - e.g. GUI components, Document components
- Beans are object classes amenable to introspection by software tools, aka Beanbox
  - e.g. GUI builder, Interactive Development Environment
  - achieved dynamically using Java Reflection
- Design time and Run time behaviours
- Run time behaviour includes arbitrary method calls
- Beans needn’t have a GUI representation

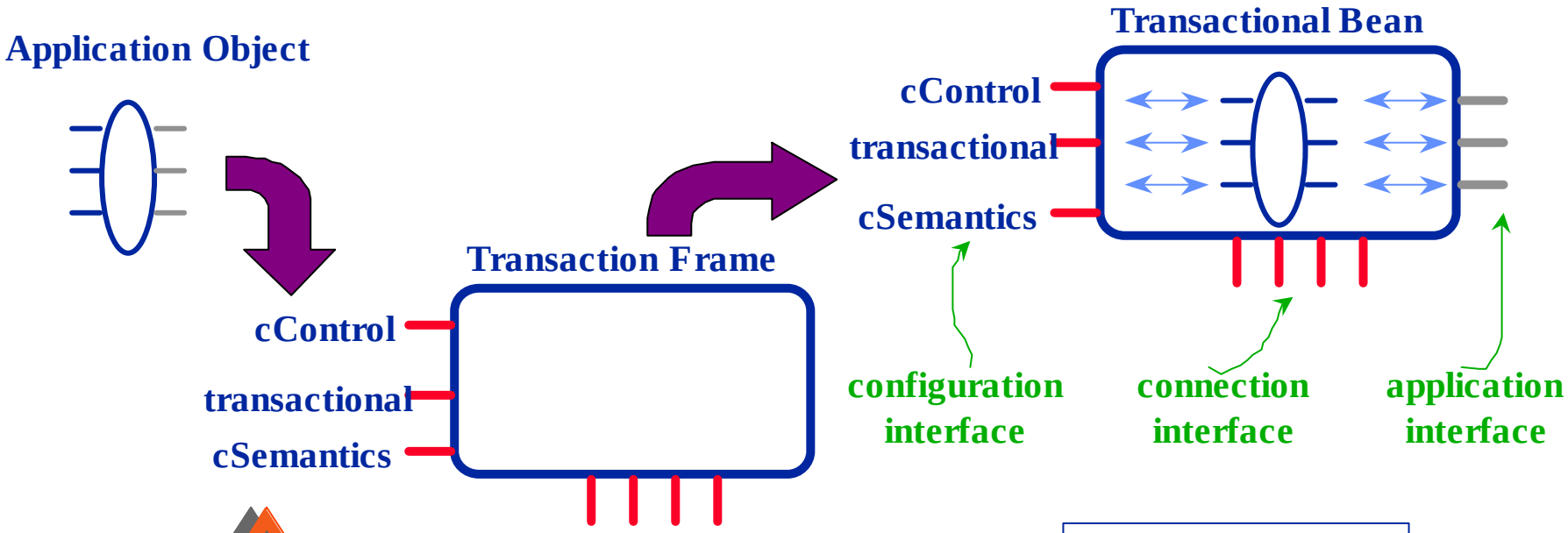
Transactional Beans

- A Transaction Frame plus a set of objects
- A unit of concurrency control, deployment and management



Construction of Transactional Beans

1. An arbitrary application object is placed in a Transaction Frame...
   - … automatically generating the *application interface* and default concurrency control semantics
2. The *configuration interface* is used to customise the Transactional Bean
3. The *connection interface* is used for connecting to a TP Manager

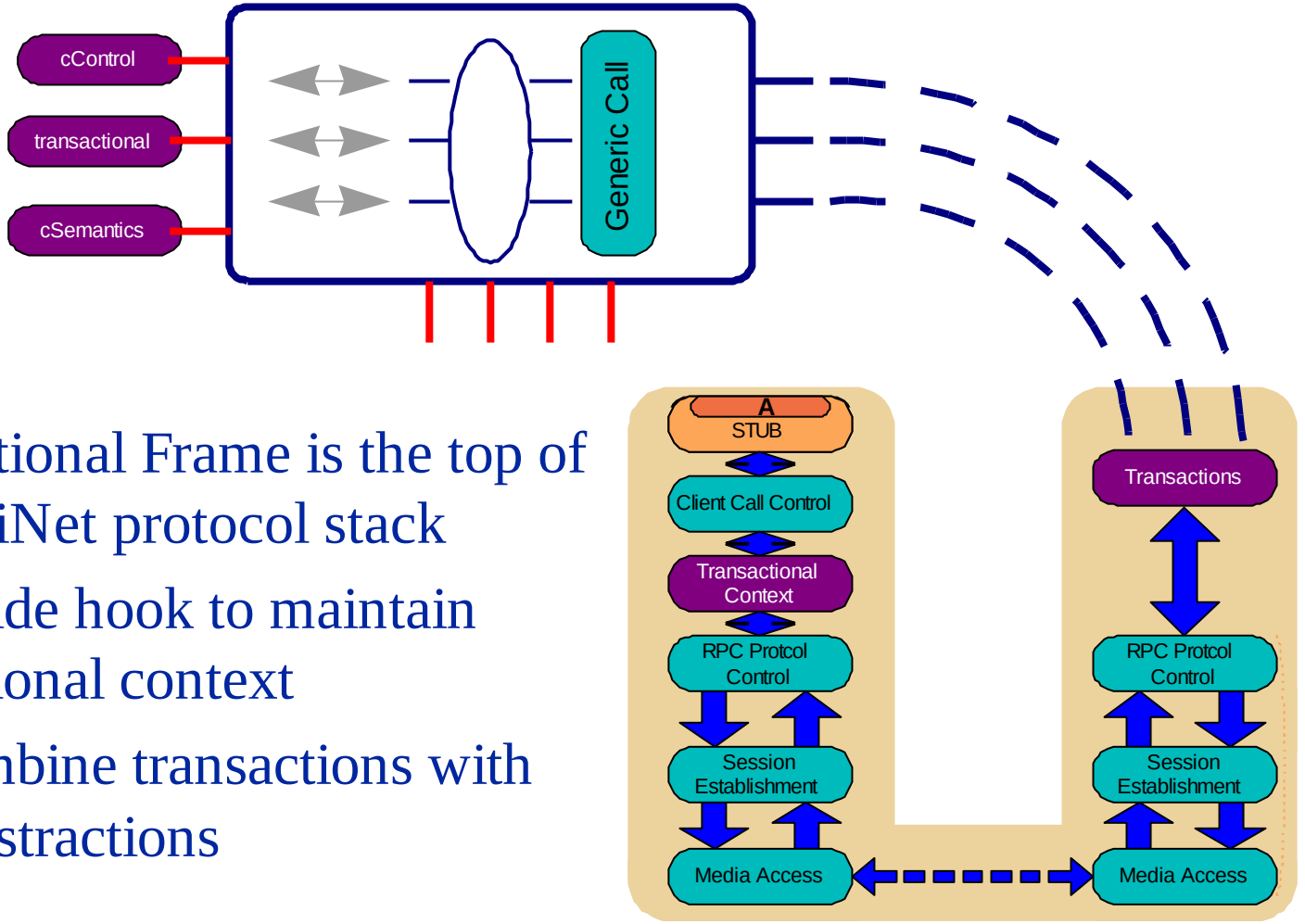


Demonstration

- So Far…
  - Placing an application object in a transaction frame
  - Using reflection to generate a locking meta object for the application object, to act as a transaction manager
  - Using a configuration interface to customise the concurrency control semantics
- Not Yet…
  - CCBeans for concurrency control and recovery
  - Transaction scripts for describing policies
  - Transaction Processing Manager Beans, e.g. 2-phase commit

Link to FlexiNet

Transactional Bean



- Transactional Frame is the top of the FlexiNet protocol stack
- Client-side hook to maintain transactional context
- Can combine transactions with other abstractions

Summary

- Support for developing the ‘middle tier’
  - benefits of component approach
  - separating application logic from system issues
  - ease of configuration