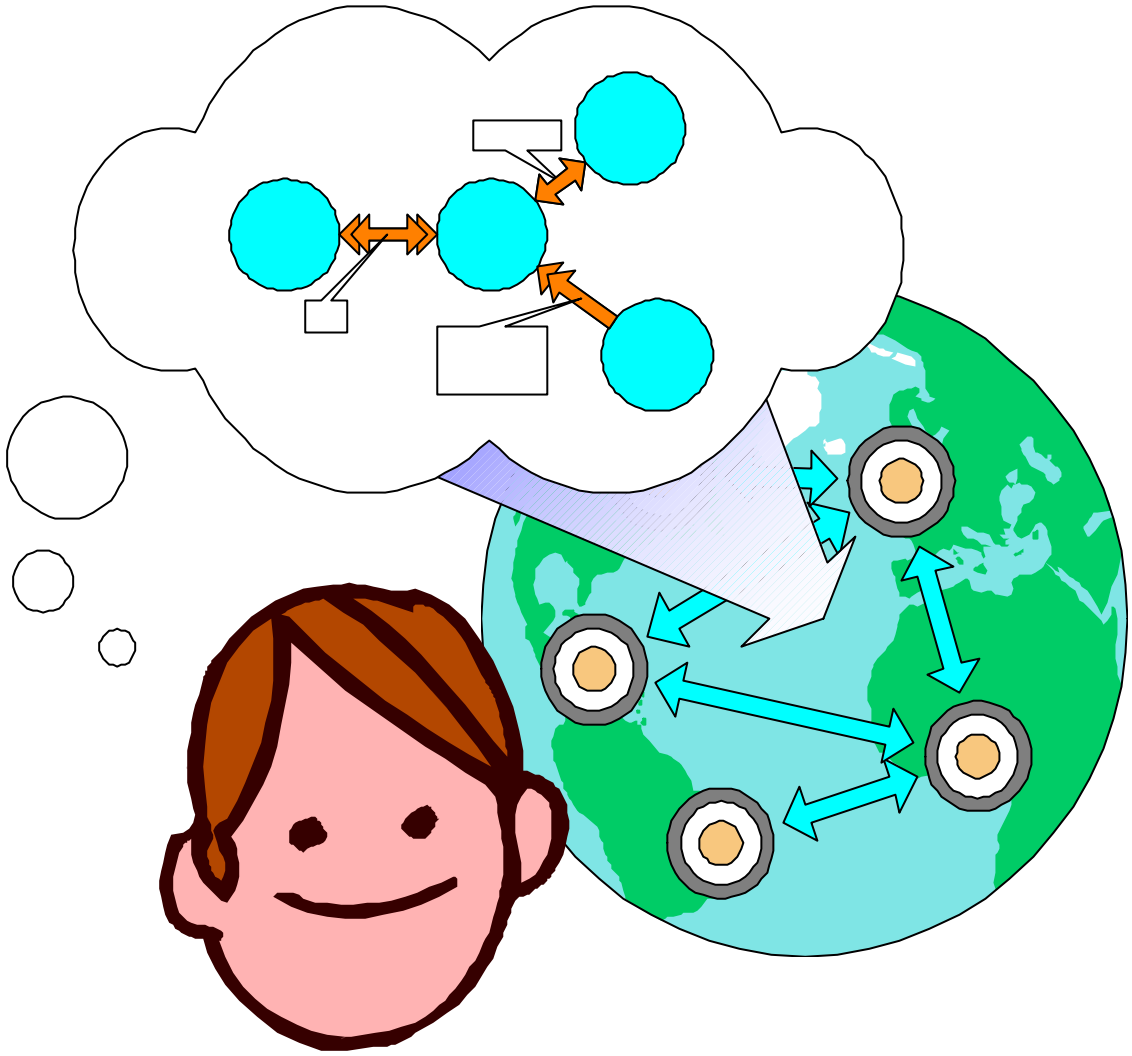
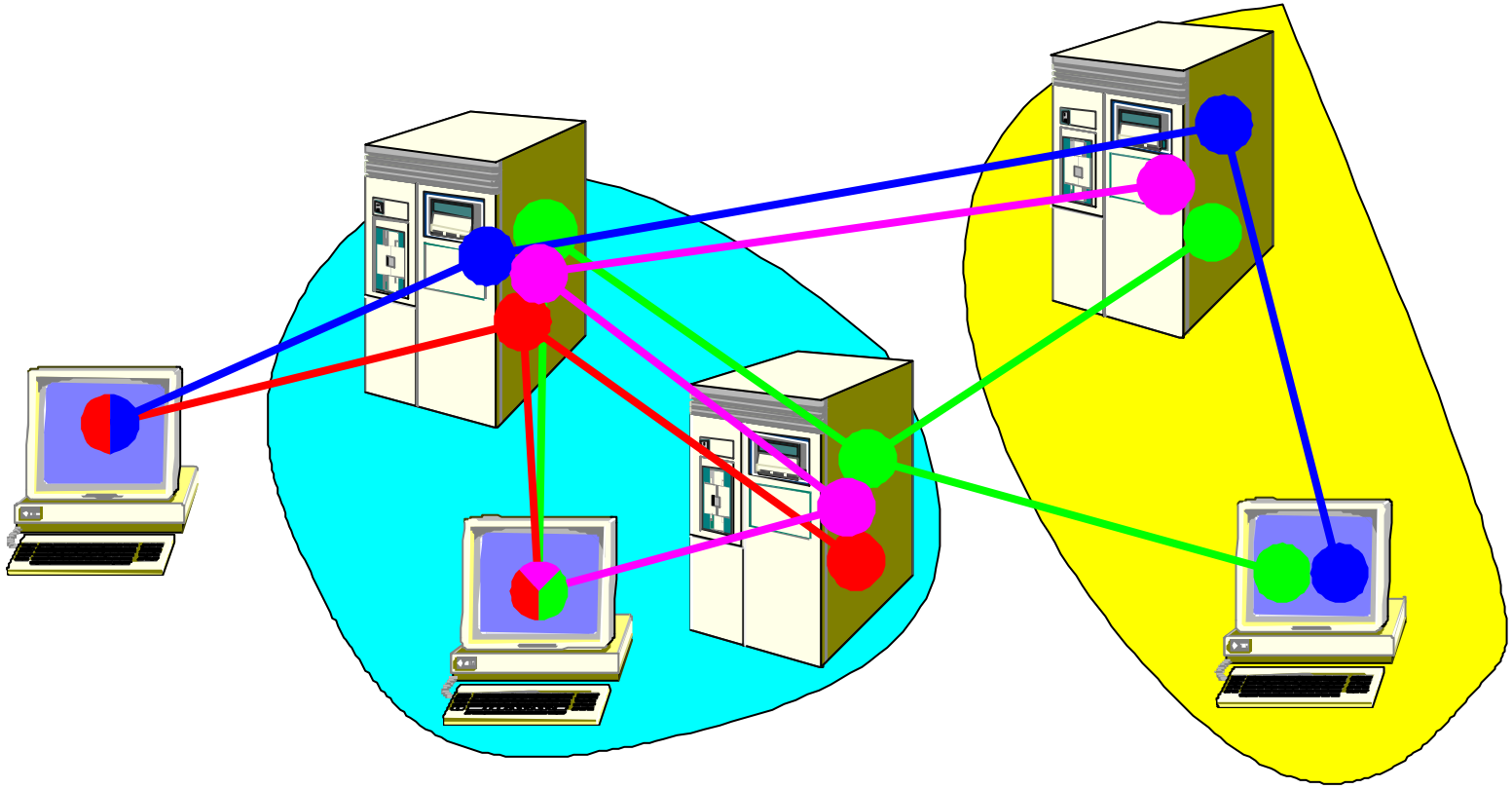


FlexiNet

Matthew Faupel & Richard Hayton

Utilising WWW increases Complexity

Global Organisations - Electronic Commerce - Devolved Management



- *Policy based deployment across enterprise boundaries is the key issue*

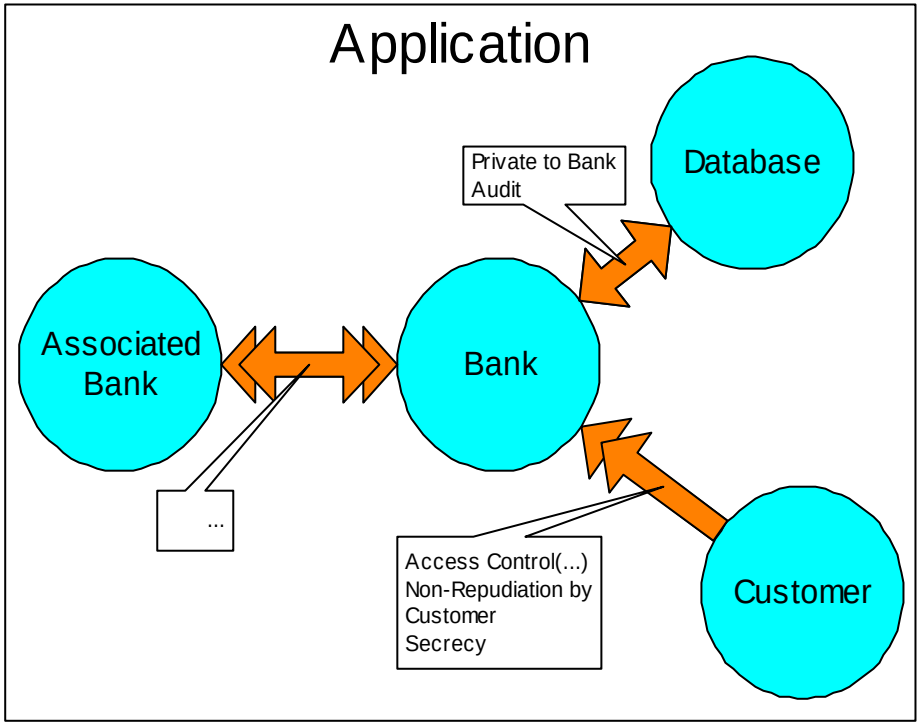
Applications are more than Components

Function

Environment

Policies

Costs



✓ *Application*

✓ *Deployment Policy*

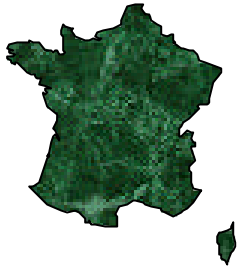
Deployable Application

Characteristics of FlexiNet Applications

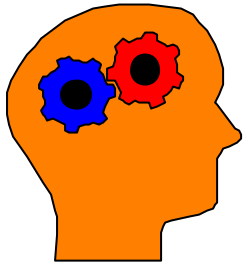
- Large and Long Lived
  - potentially complex and poorly understood
- Distributed
  - many interconnected processes (both clients and servers)
- Multi-Organisation
  - No single point of policy control
- Heterogeneous Environment
  - different organisations
  - different facilities
  - different security domains
  - different costing factors

Policy is not static

Contracts and Partnerships

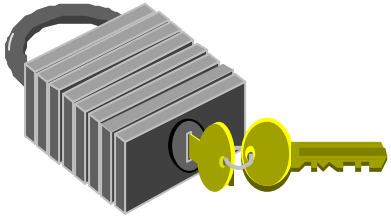


Legislation



Business Practices

New Opportunities

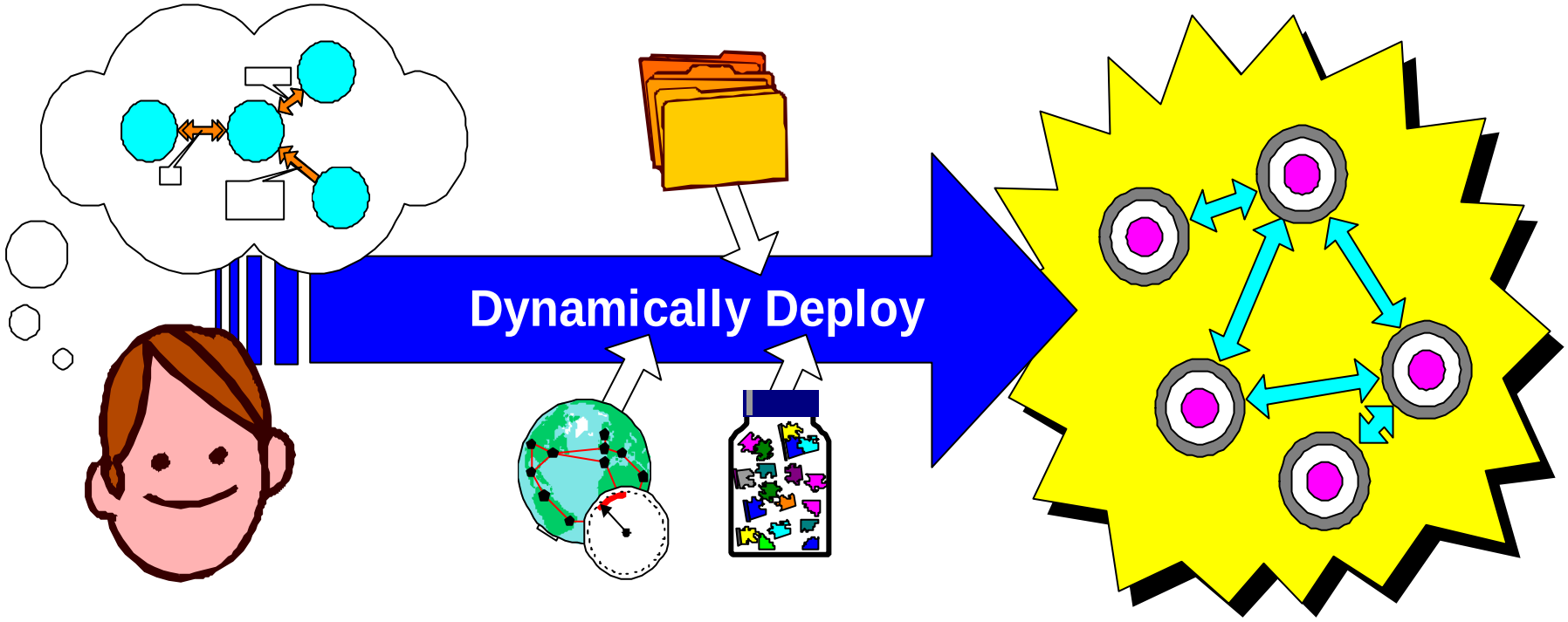


Security Mechanisms

Networks and Infrastructure

- Over time, requirements change
- The new requirements may be short lived
  - change in policy for short term contract
  - deal with network outage
- It may not be possible to anticipate the changes
  - Legislation
  - New developments and facilities
  - New opportunities

FlexiNet: Application deployment driven by policy



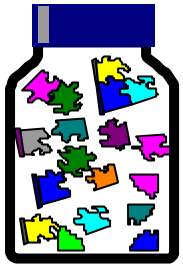
- *FlexiNet framework for wrapping applications*
- *Policy selects components to populate framework*

  ⇒*component orientated middleware*

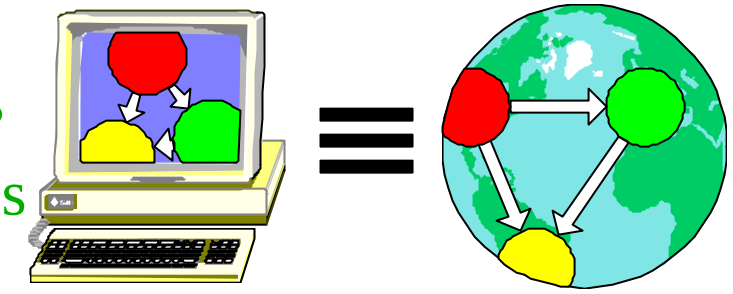
Aims

- Support Development
  - reduce domain of knowledge of developers - separation of concerns
  - make errors easier to spot - leverage strong typing
- Support Deployment
  - aid engineering decisions - declarative specification
  - enable reusable code/services - component based
- Support Evolution
  - of application - function / policy
  - of infrastructure - new mechanisms
  - of environment - changing costs

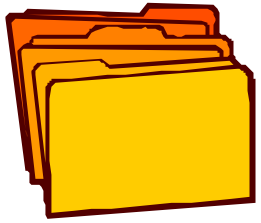
Approach



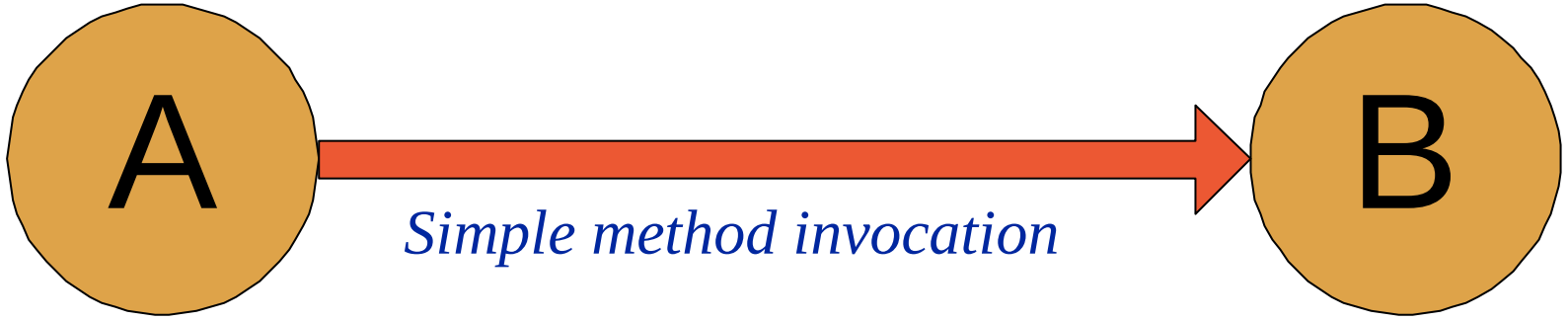
- Components for selective transparency
  - reusable units for management/mobility/checkpointing etc.
- Modular Engineering
  - simple API for high level abstractions
  - external control of reflective interfaces



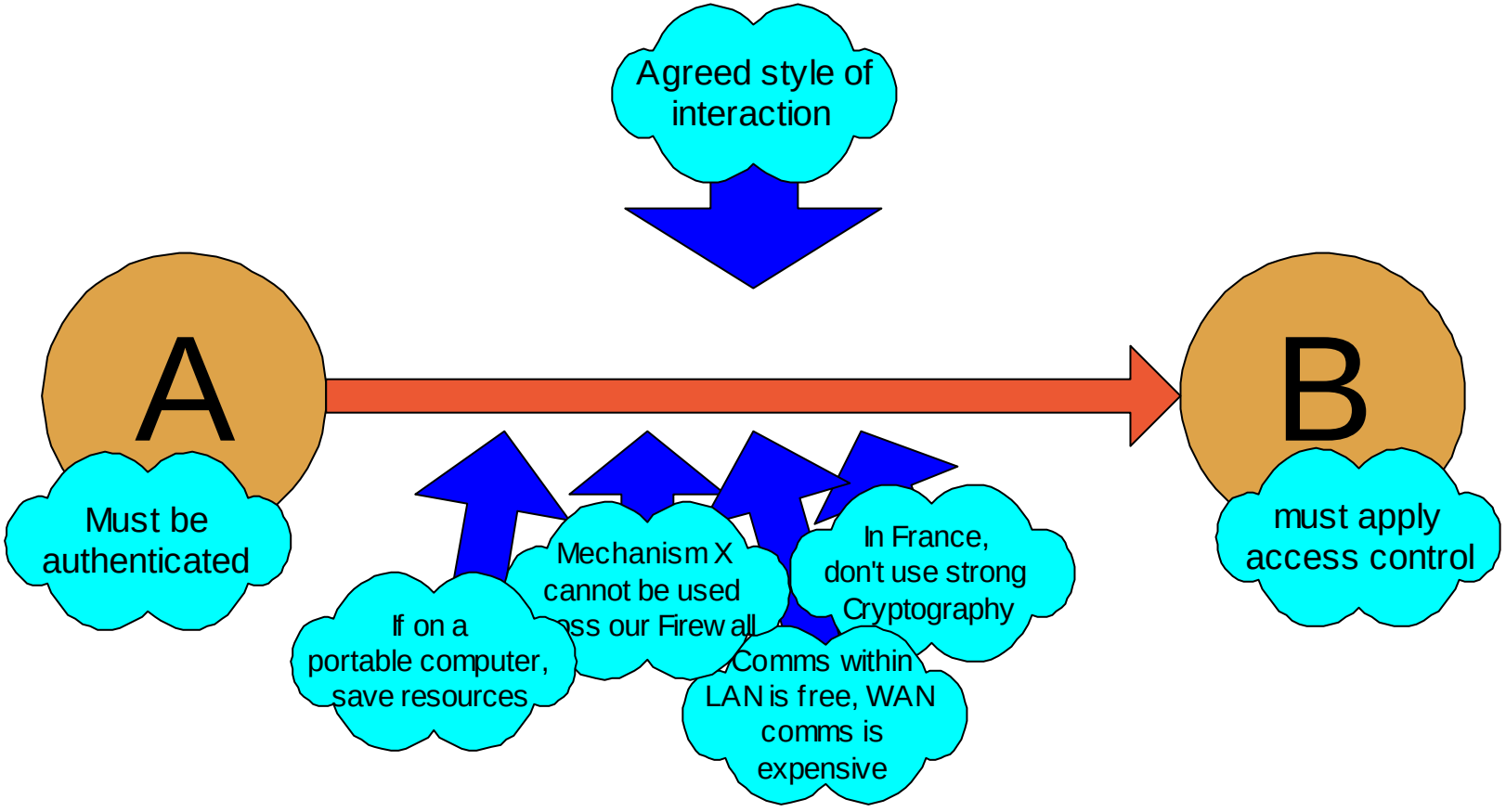
- Declarative Descriptions
  - describe requirements and resources
  - drives selection and configuration of transparency components



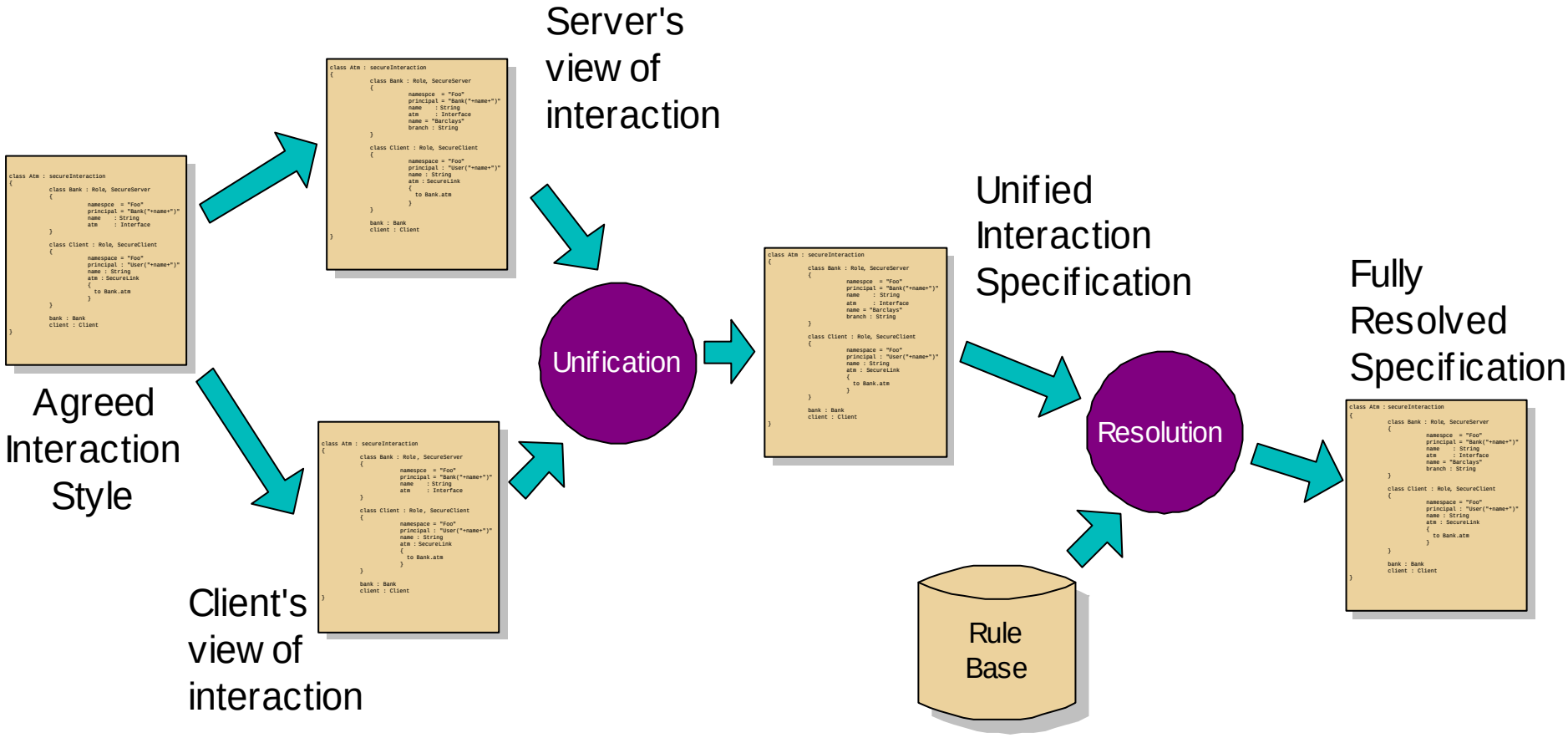
Application Programmer's view of an interaction



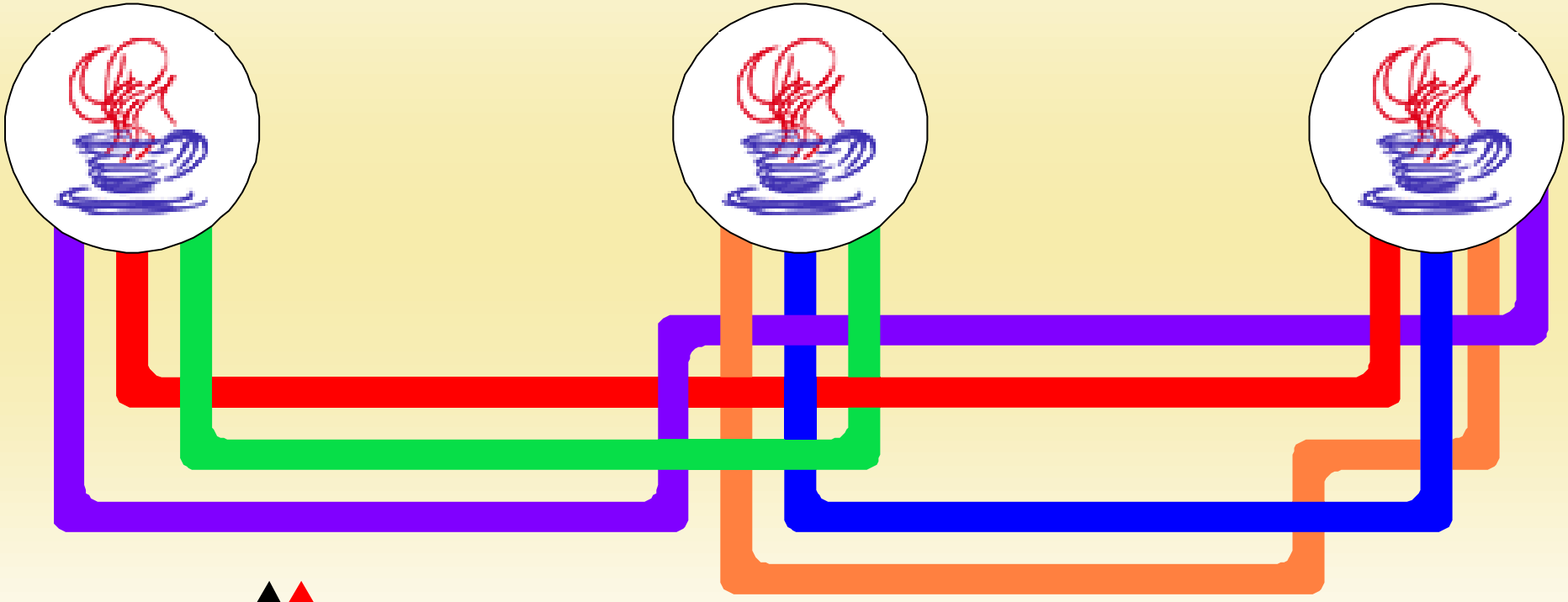
Engineering requirements of an interaction

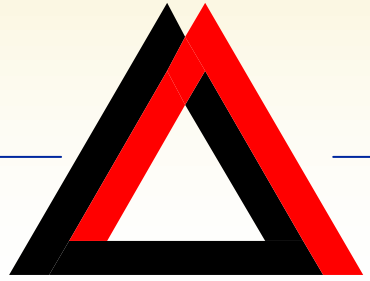


Choosing a binding mechanism via Resolution



FlexiNet Open ORB Framework





Engineering Framework Overview

- Core FlexiNet framework
  - future work will be built on this
- Provides transparent binding
  - For local or remote interconnection
- Open binding architecture
  - To allow general reflection or communication
- Minimalist API
  - External control via reflection

Topics

- Generic Communications and Reflection
- Naming and Binding
- Resource Management
- API
- Performance & Summary

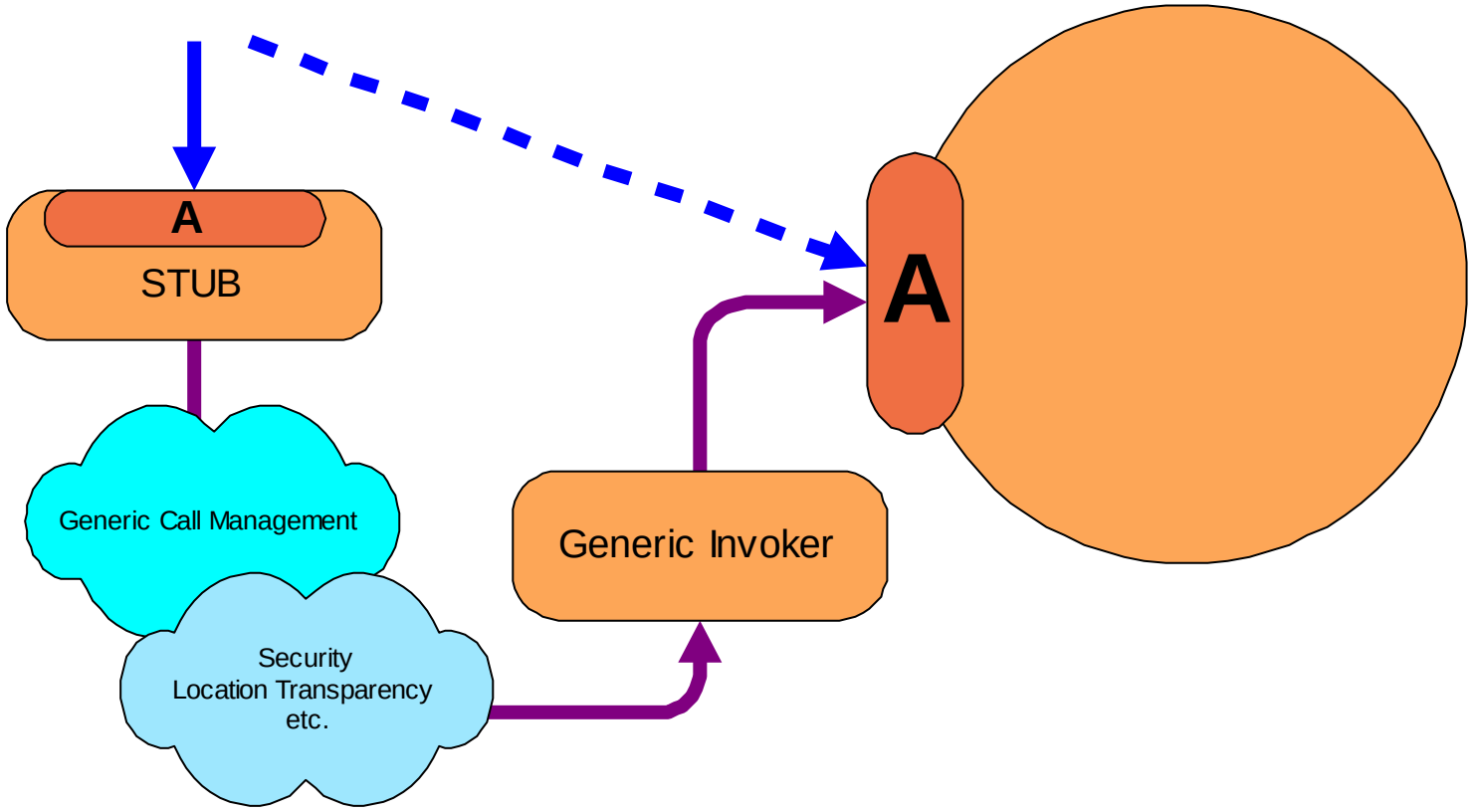
Topics

- **Generic Communications and Reflection**
- Naming and Binding
- Resource Management
- API
- Performance & Summary

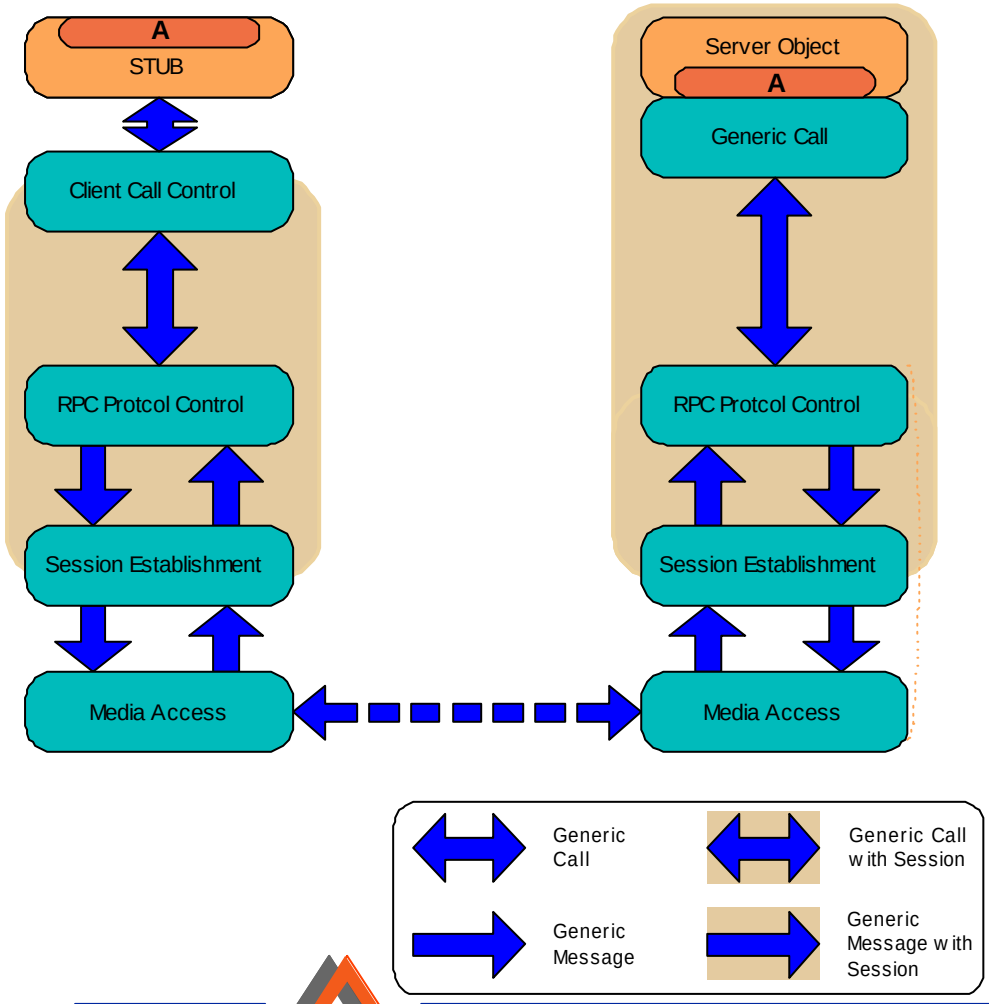
Generic Communications

- Transparent but Powerful
  - No special compiler - stub generated on the fly
- Utilise Java Core Reflection
  - Standard representation of a method call
  - Use Java generic method invocation
- Flexible Reflection
  - Can apply any transformations or restrictions on call
  - Stub is not dependant on interconnect mechanism
- Evolvable: engineering can change under the feet of API

Generic Communication

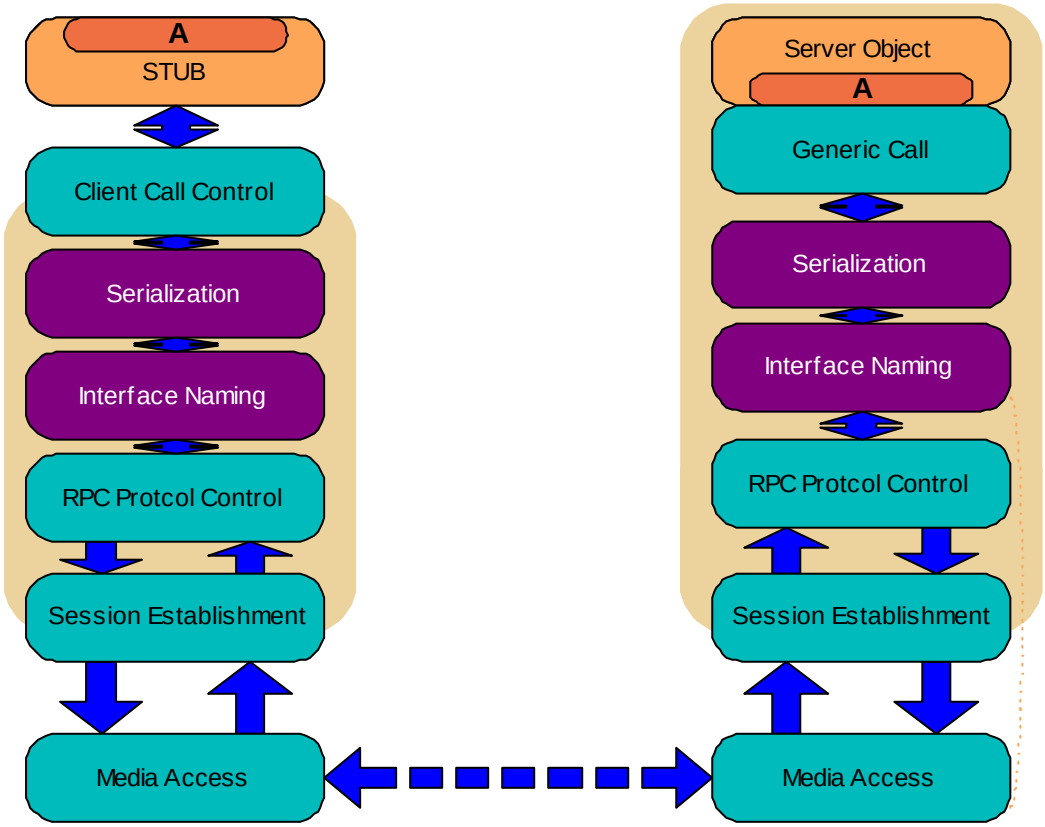


Generic Communications Stack



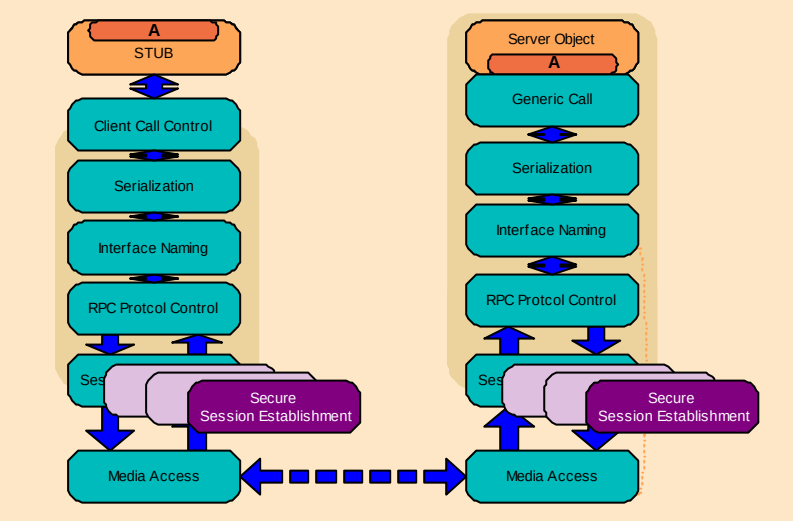
- We have designed a communications stack that identifies and separates different styles of communication.
- This allows simple addition or refinement of functionality
- Layers can be replaced, or new layers added between existing ones.

Simple Remote Invocation Stack

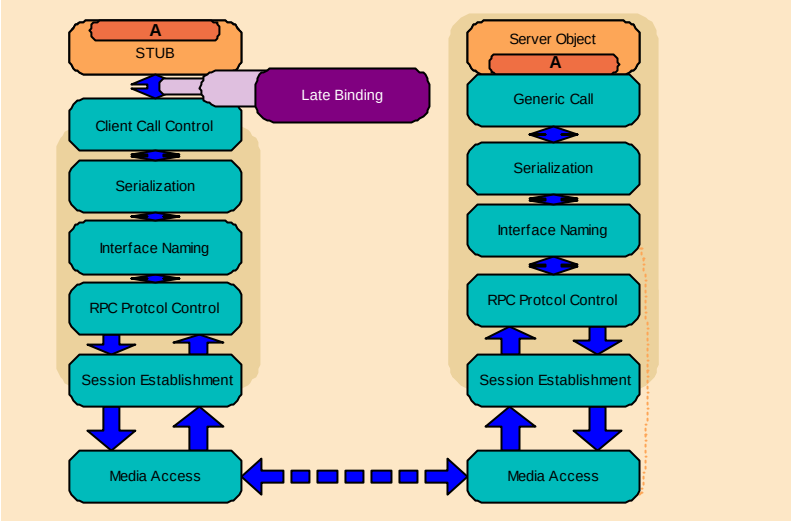


- For simple Remote Method Invocation, we must add two layers.
- This protocol stack was delivered to ANSA sponsors in October 97
- It is a simple example of how the generic protocol stack can be used.

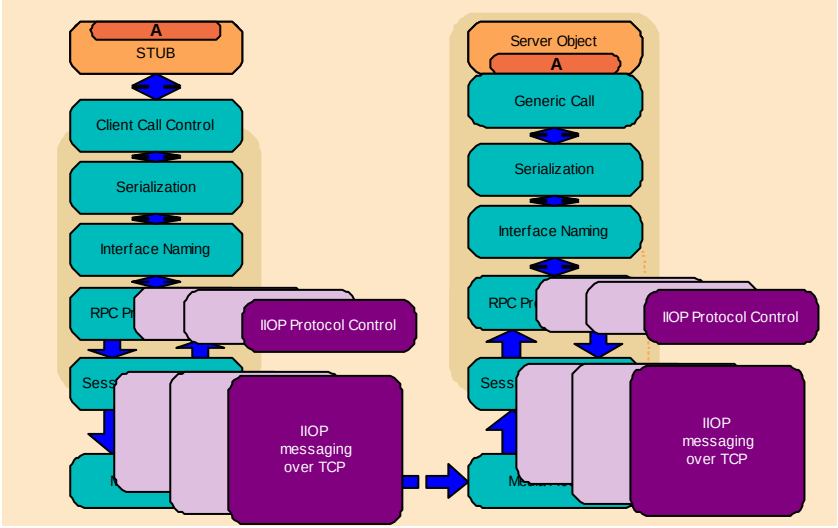
Other Remote Invocation Stacks



SECURE SESSIONS

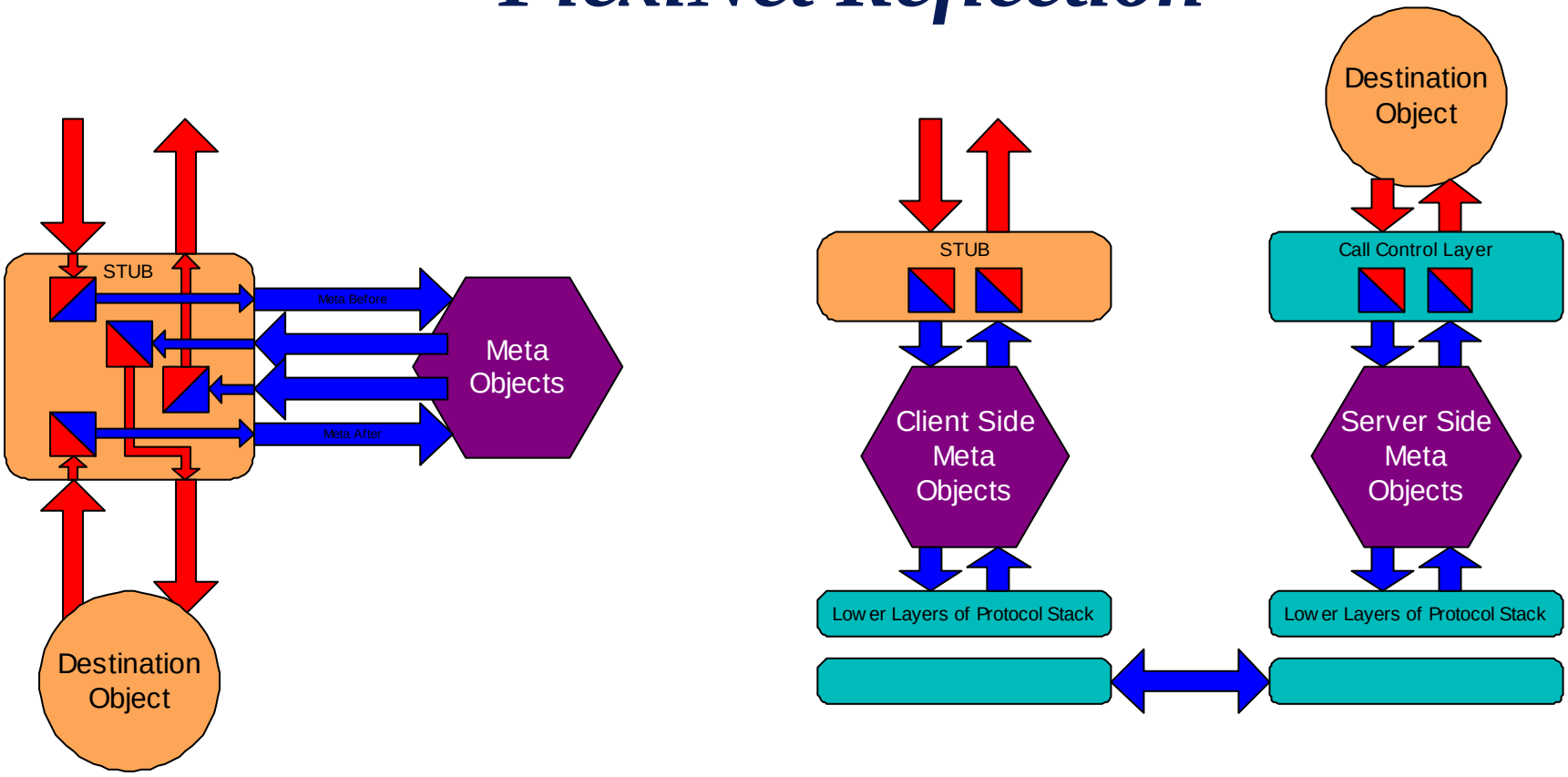


LATE BINDING



IIOP TRANSPORT

FlexiNet Reflection

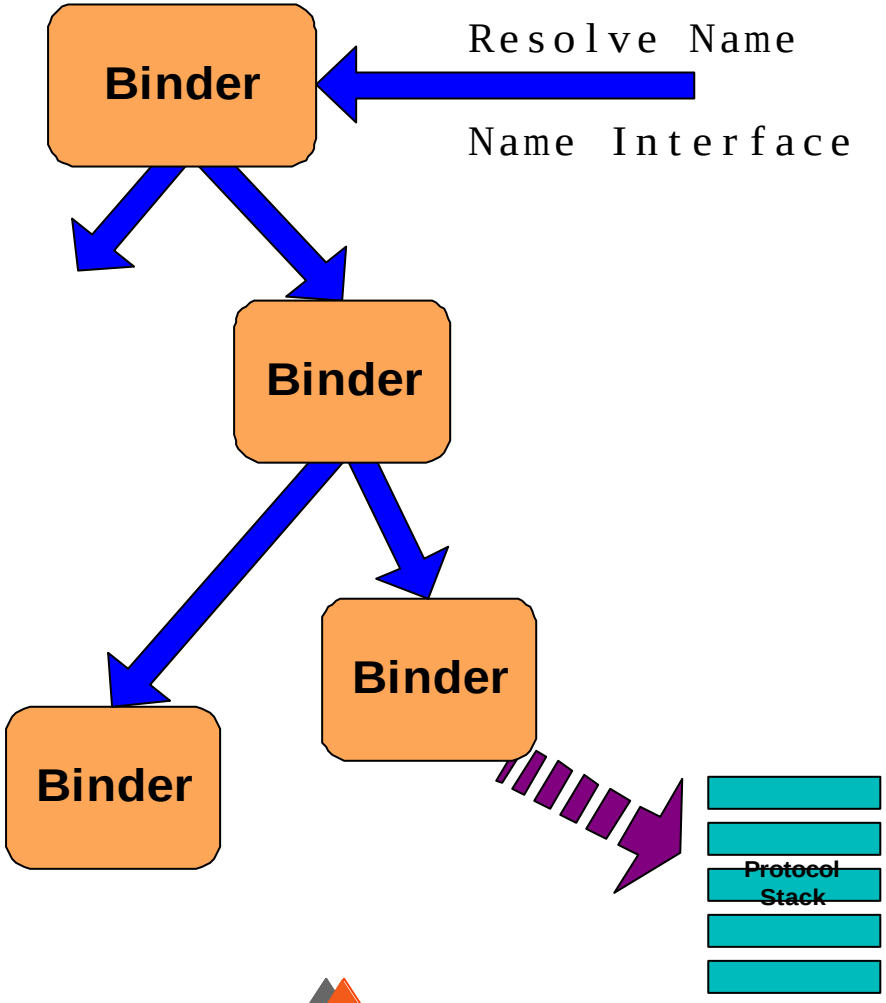


Generic communications provides general purpose reflection.
We integrate Reflection and RMI for performance

Topics

- Generic Communications and Reflection
- **Naming and Binding**
- Resource Management
- API
- Performance & Summary

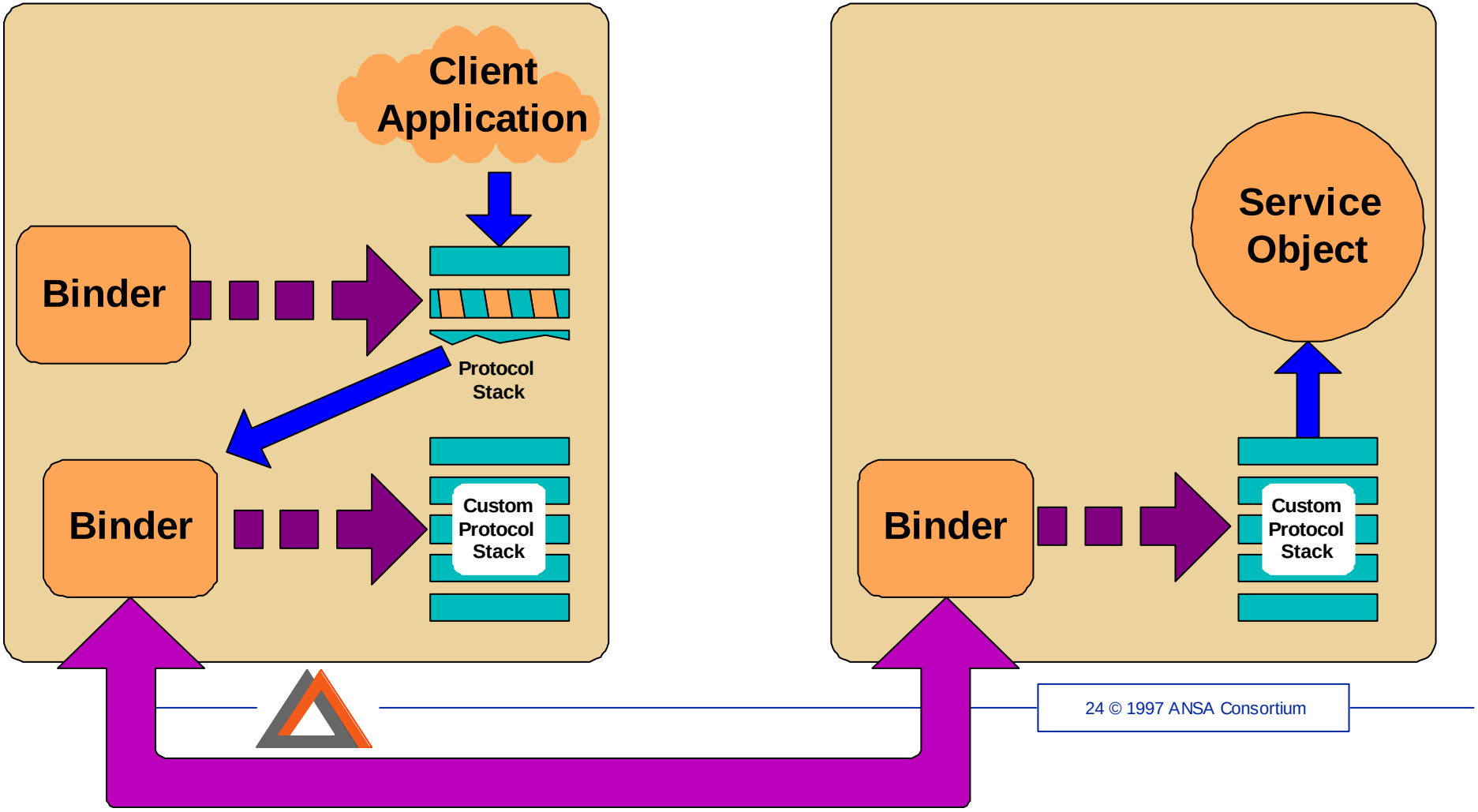
Naming and Binding



- There may be many binders
  - Binders may manage different protocols and name spaces.
- Binders may be composed.
  - One binder decides which protocol to use and forwards the request to an appropriate protocol specific binder.
  - A binder may simply audit bind requests, and then forward requests to another binder.

Late / Negotiated Binding

We can make an initial binding, and later negotiate for a custom protocol



Ongoing Naming/Binding Work

- Negotiation for choice of Binder
  - negotiation based on declarative specifications
- Parametric Binder
  - slot in transparencies
- Integration with transactions
  - layer to maintain transactional context
- Integration with mobility
  - naming layer for interfaces that move

Topics

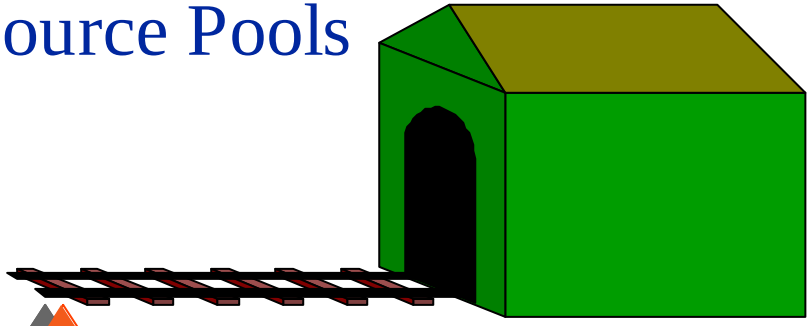
- Generic Communications and Reflection
- Naming and Binding
- **Resource Management**
- API
- Performance & Summary

Resource Management

Resources

Management Policies

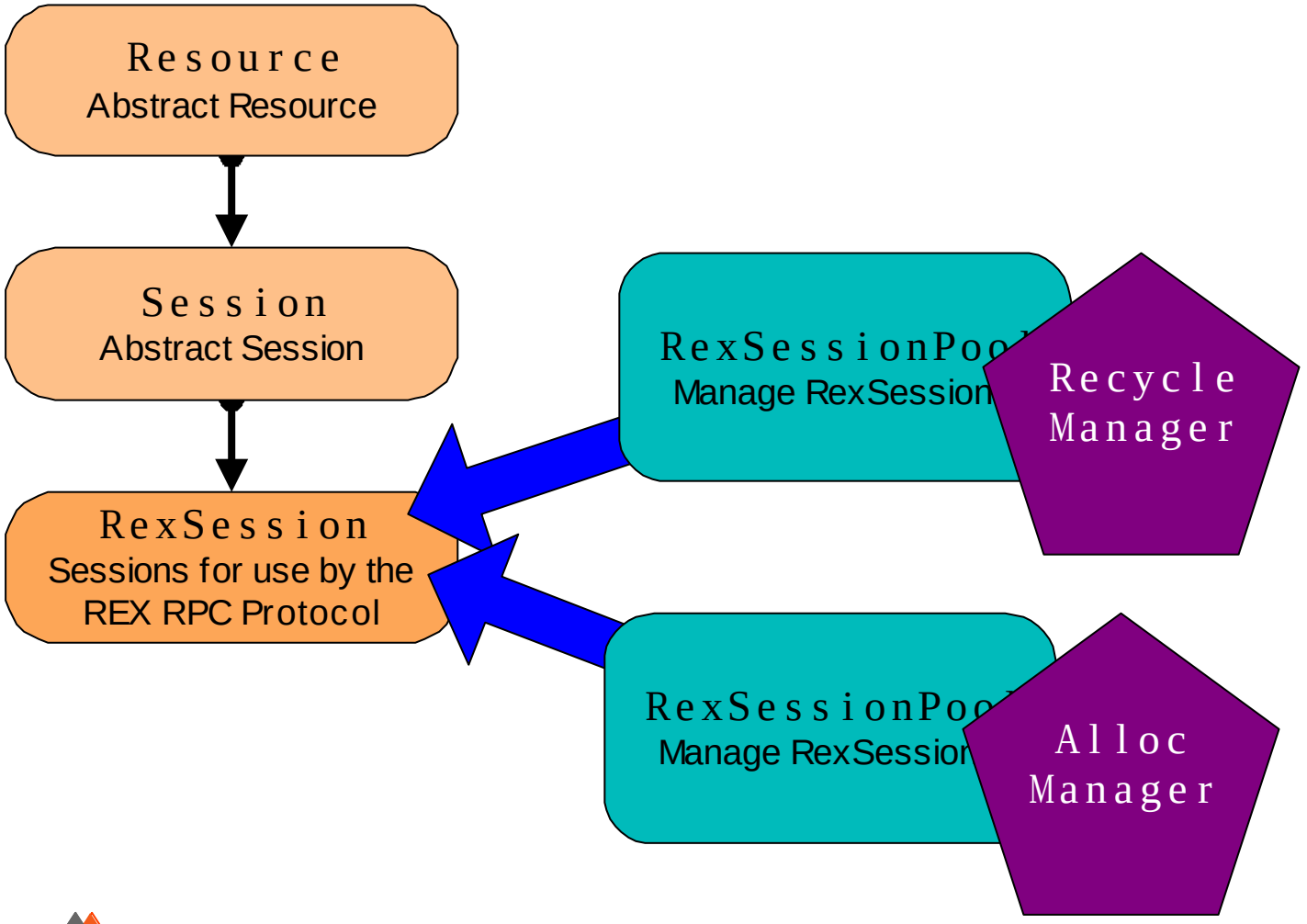
Resource Pools



Resource Management Issues

- Abstract resource management
  - one layer creates resource, another layer uses it
  - separation functions of related layers
- Decouple resource and management classes
  - don’t redesign management for every class
  - simplify sub-classing of resources
  - enable reuse
- Dynamic Configuration
  - allow run time choice of management strategies

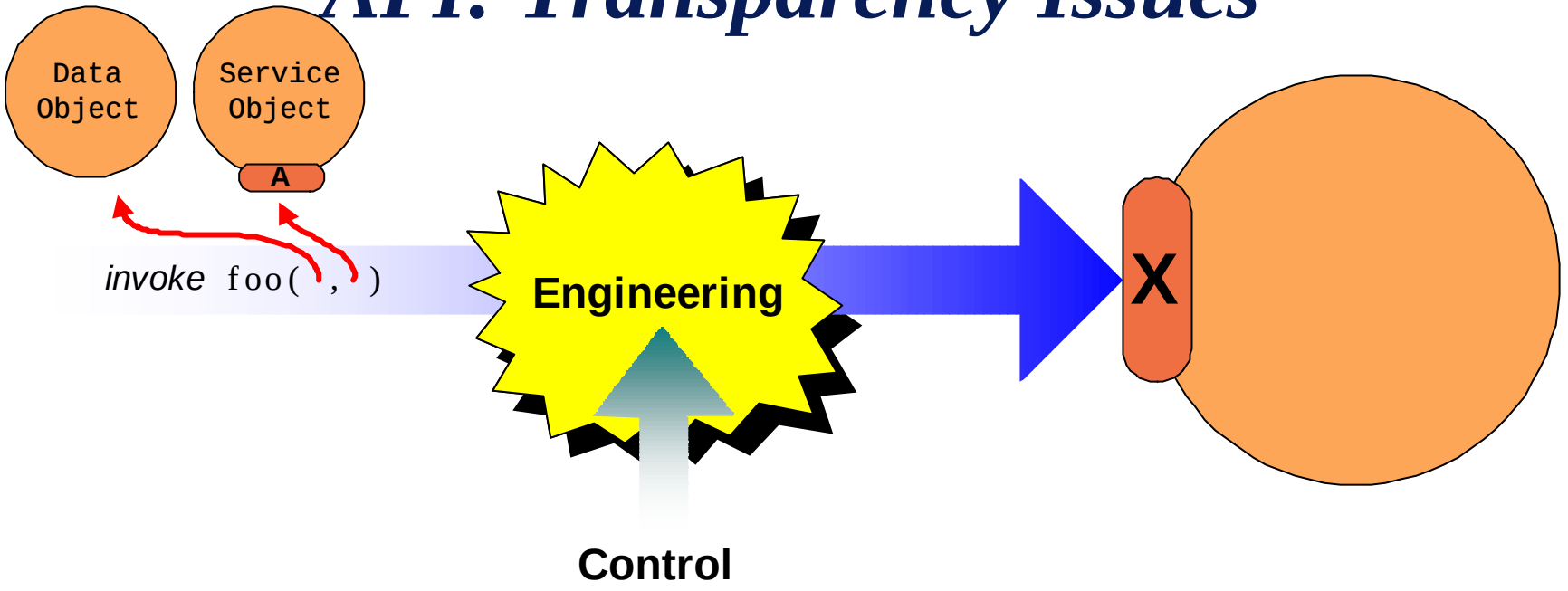
Resources, Pools, Pool Managers



Topics

- Generic Communications and Reflection
- Naming and Binding
- Resource Management
- **API**
- Performance & Summary

API: Transparency Issues

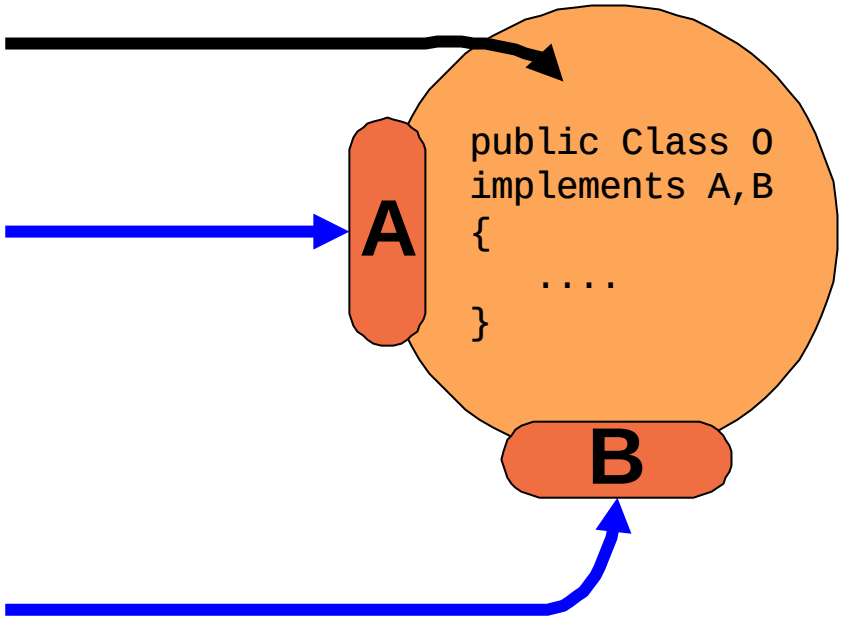


- Application programmer requires simplicity
  - needs to pass data and references to services
  - is not concerned with engineering details

When to pass by reference?

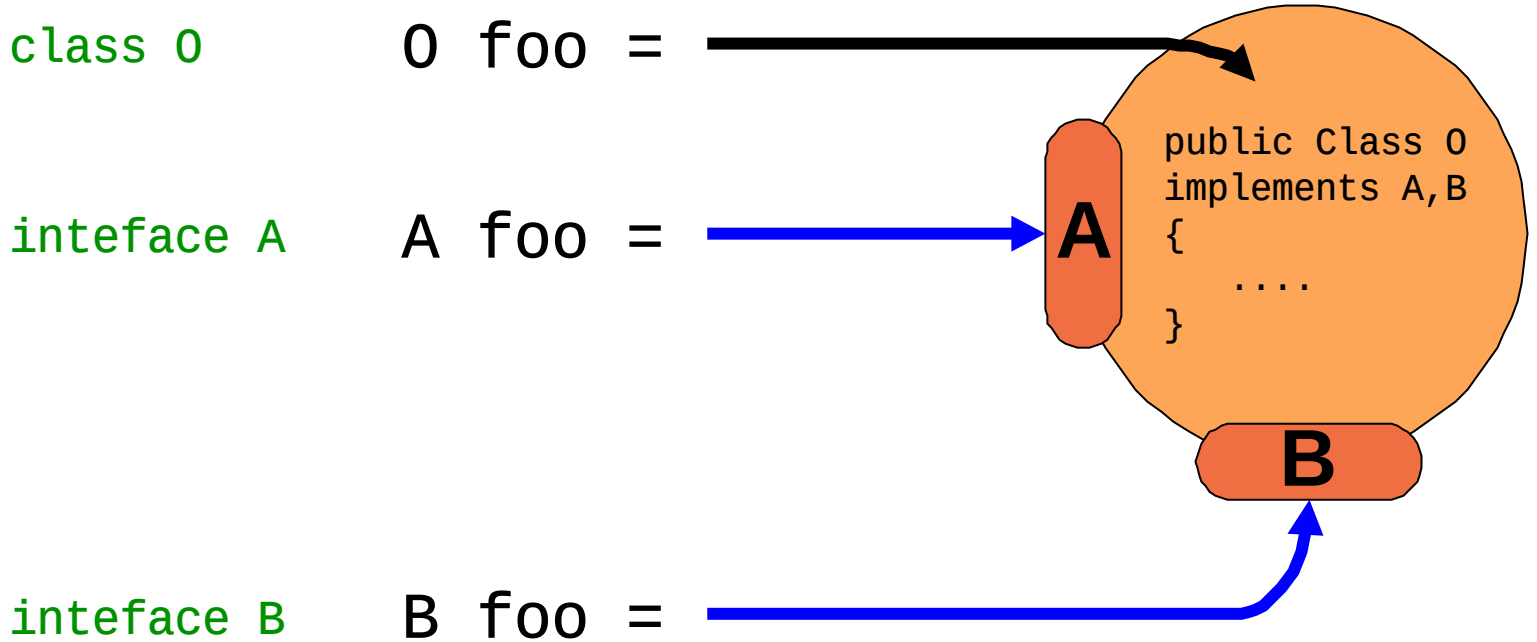
Pointer to an object, pass the state of the object.

Pointer to an interface, pass a reference to the interface.



Problem: In Java these three pointers have identical values

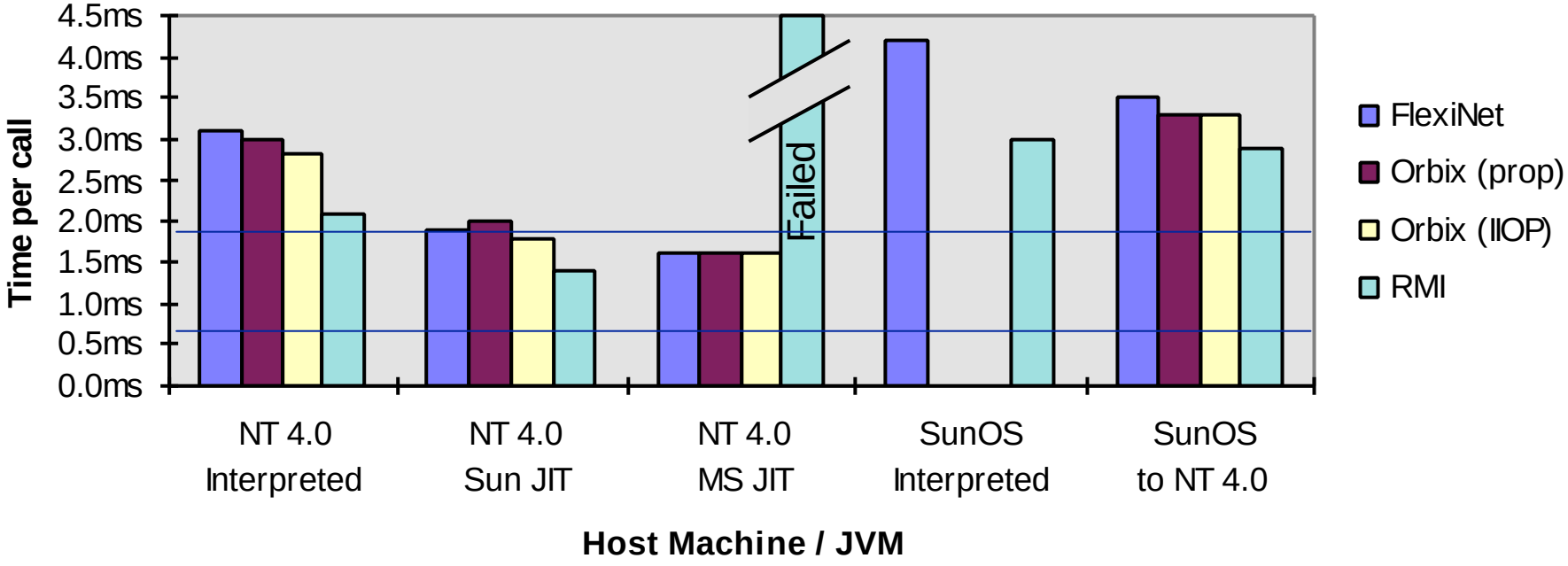
Solution: examine the class of the reference



Topics

- Generic Communications and Reflection
- Naming and Binding
- Resource Management
- API
- **Performance & Summary**

Comparative Performance for Null-Operation



Java Lessons Learnt

- Core reflection is really nice
  - But: too much left to JVM implementation
- Pure Java?
  - Suffer from ‘next release syndrome’
  - Different interpretations of the spec.
  - But: better than the C++ story!
- Speed
  - Can be made to go fast
  - Slow in unexpected places - e.g. object creation

Summary of Engineering Framework

- FlexiNet ORB Framework
  - built, working and tested
- Available for use
  - released to sponsors
  - works in applications and applets
- Pure Java
  - some clever (but legal) techniques
- Future research built on this testbench
  - transactions, mobility, security

Current Status

- Engineering Framework
  - version 1.0 delivered to sponsors
  - working on mobility issues:
    - naming for mobile interfaces
    - encapsulation of mobile objects
- Declarative Specification
  - prototype language & approach to resolution
  - design for negotiation framework
- Abstractions
  - initial design of transactional framework
  - secure communications in progress