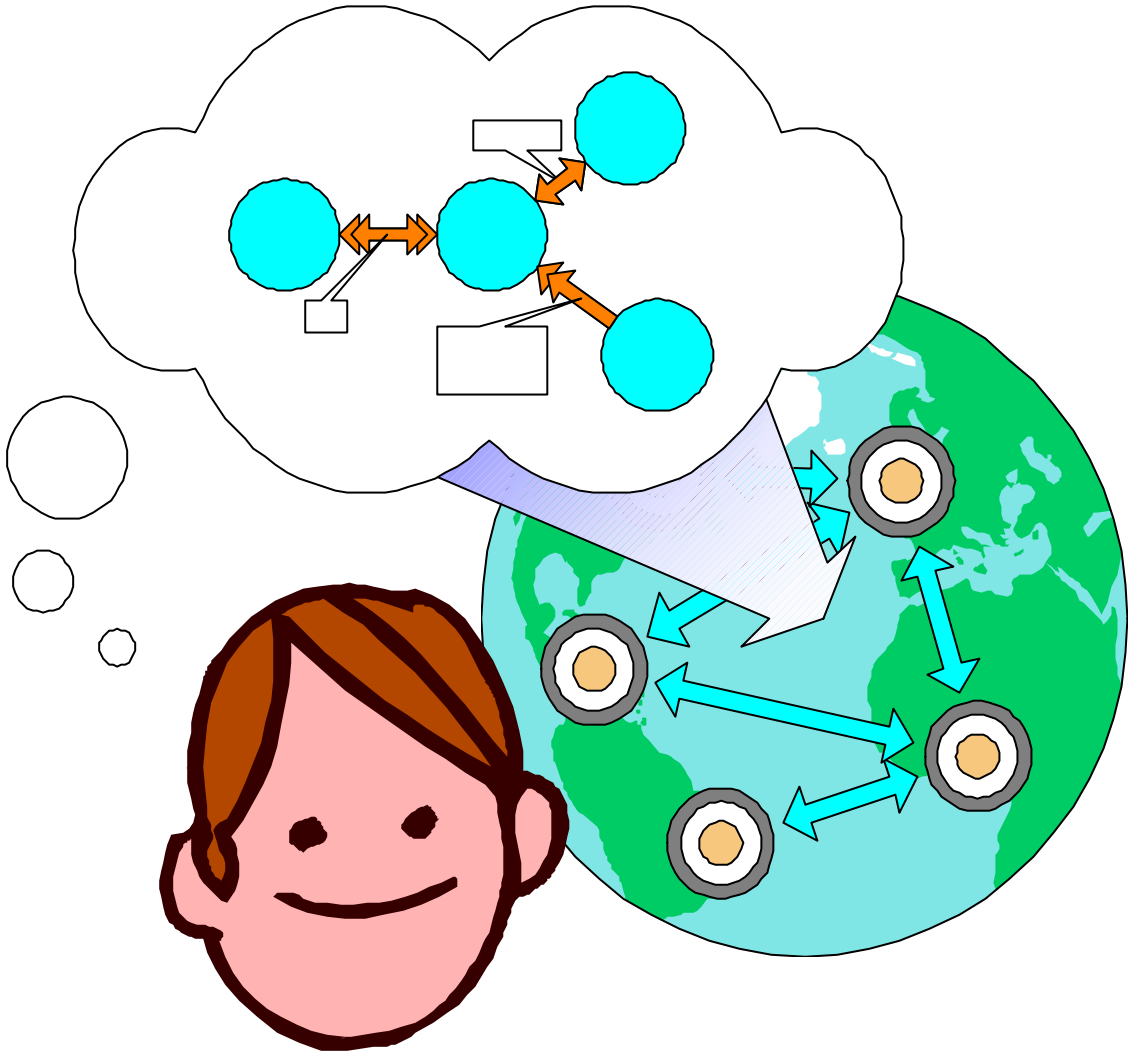


FlexiNet

Eighth
ACM
SIGOPS
European
Workshop-
Support for
Composing
Distributed
Applications

Richard Hayton, Andrew Herbert, Douglas Donaldson

Ideal World

- Application Development
  - Computational solutions use problem domain objects
- Application Deployment
  - Specify policy for rollout
    - object locations, availability, concurrency, transactional properties, authentication, privacy…
  - Engineering transparently configured from policy
  - Across enterprise boundaries

RM-ODP Distribution Transparencies

- Access
- Location

Access objects regardless of object or client location

- Migration
- Relocation

Allow the object and client to move

- Persistence
- Failure

Long lived, failure tolerant objects

- Transaction

Consistent, concurrent access

- Replication

For scale, performance, availability

- Security

Control and audit access

Current Situation

- Application Development uses Components but...
  - Influenced by deployment scenario
    - firewalls etc.
  - Requires knowledge of low level details
    - protocols, security etc.
- Application Deployment
  - Can't plug and play with engineering solutions
  - No standard approaches - 'black art'

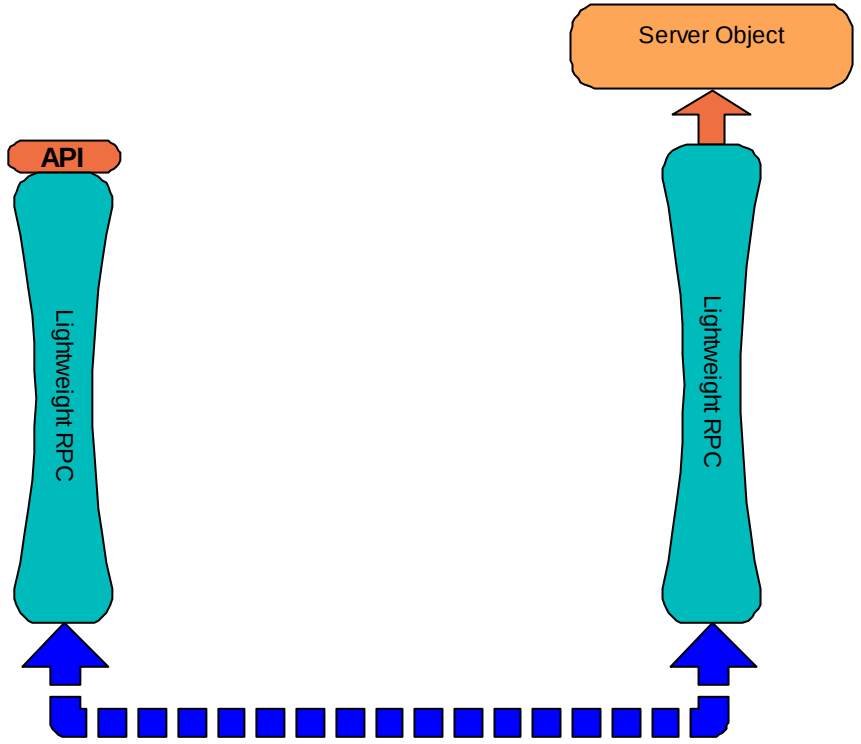
Middleware Development

- Lightweight
  - Lack of extensibility

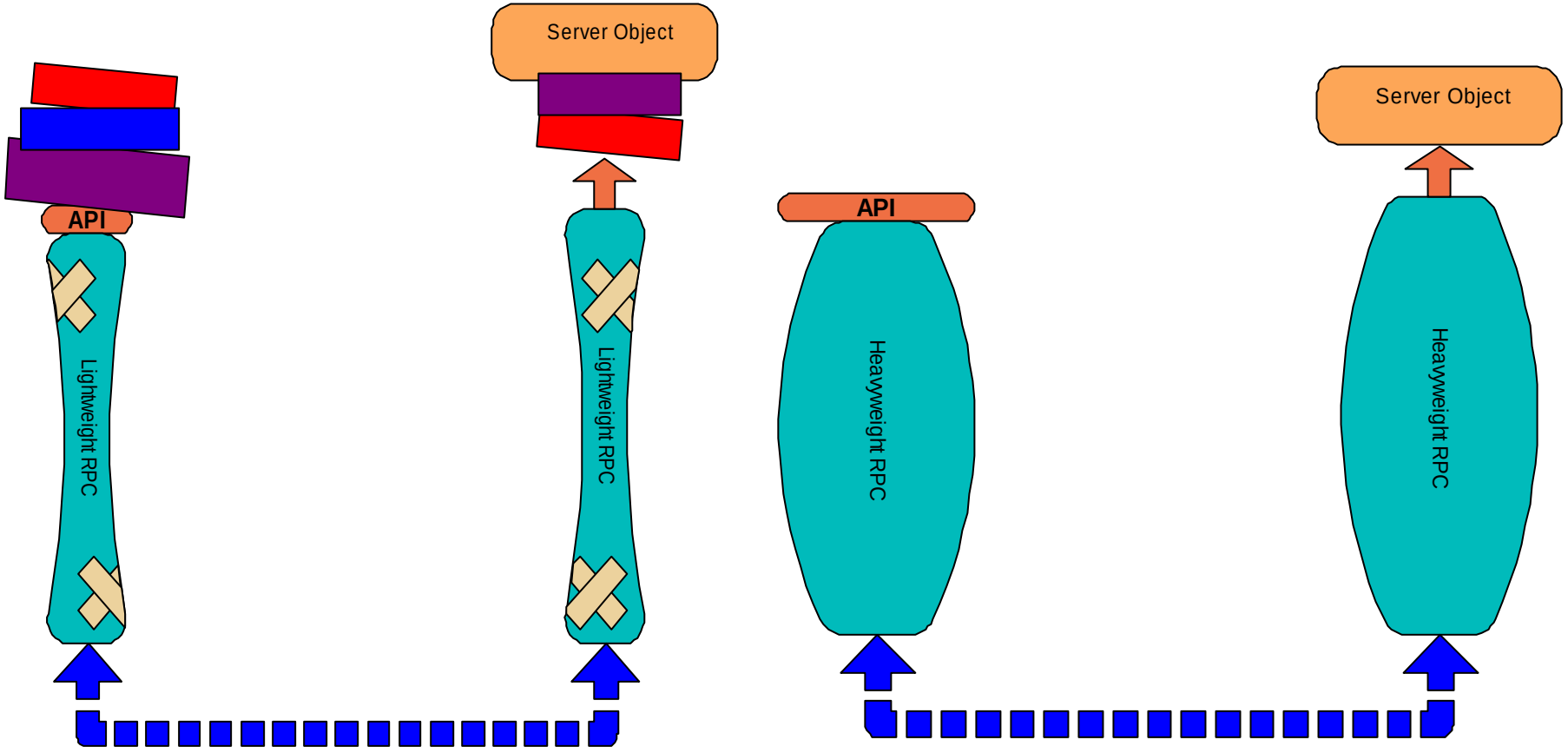
or

- Heavyweight
  - lack of flexibility.

- Requirement for engineering transparency



Middleware Problems



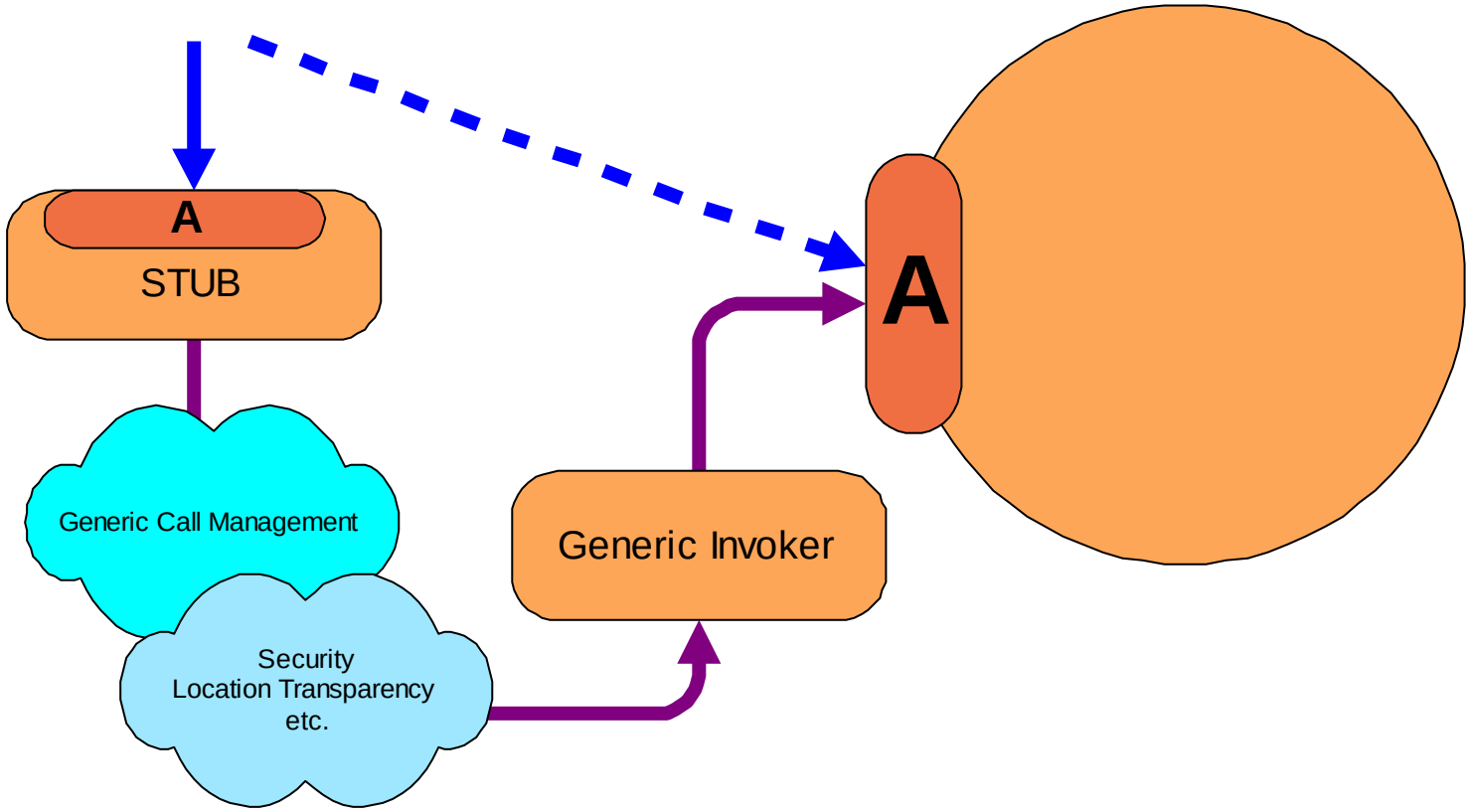
FlexiNet Aims

- 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

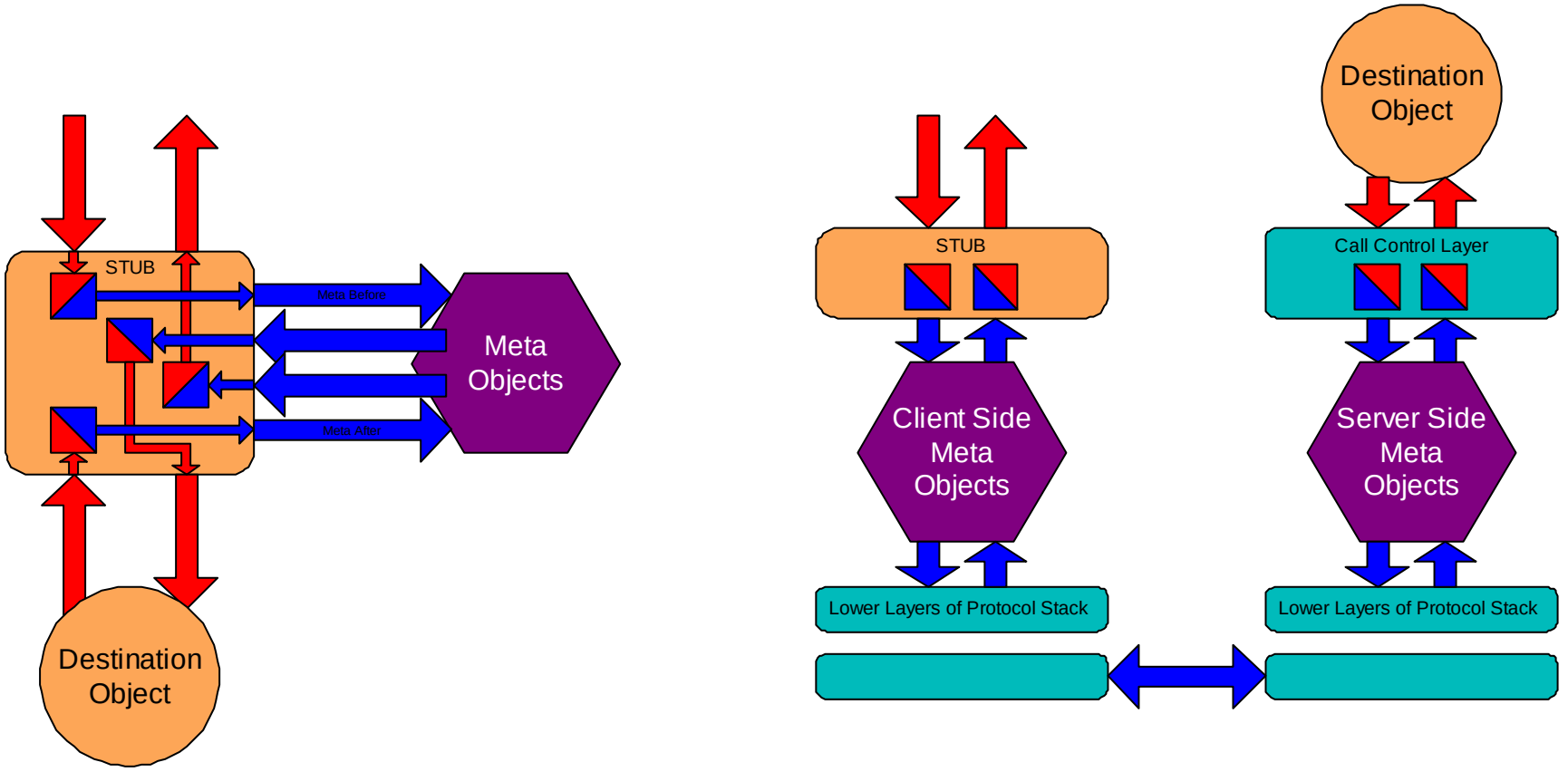
Approach

- Engineering Framework
  - 'Slot in' abstractions and enhancements
  - Selective transparency
  - Allow developer to make trade-offs.
- Straightforward functional APIs
  - Make life easy for application developers
  - Ease modelling to coding transition
- Reflection and introspection
  - to keep components modular
  - to allow configuration
- High level tools
  - Declarative specification driving reflective choice

Generic Communication



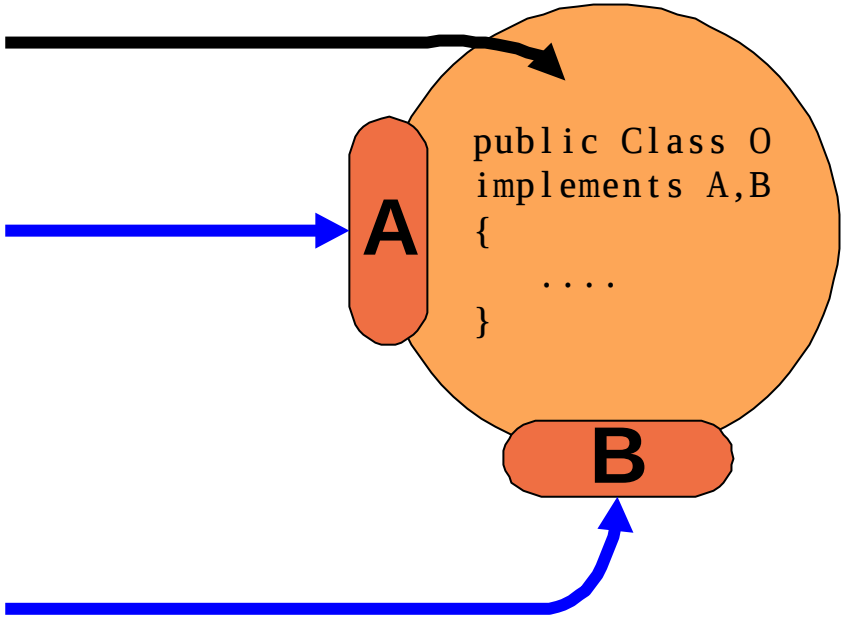
Reflective Communication



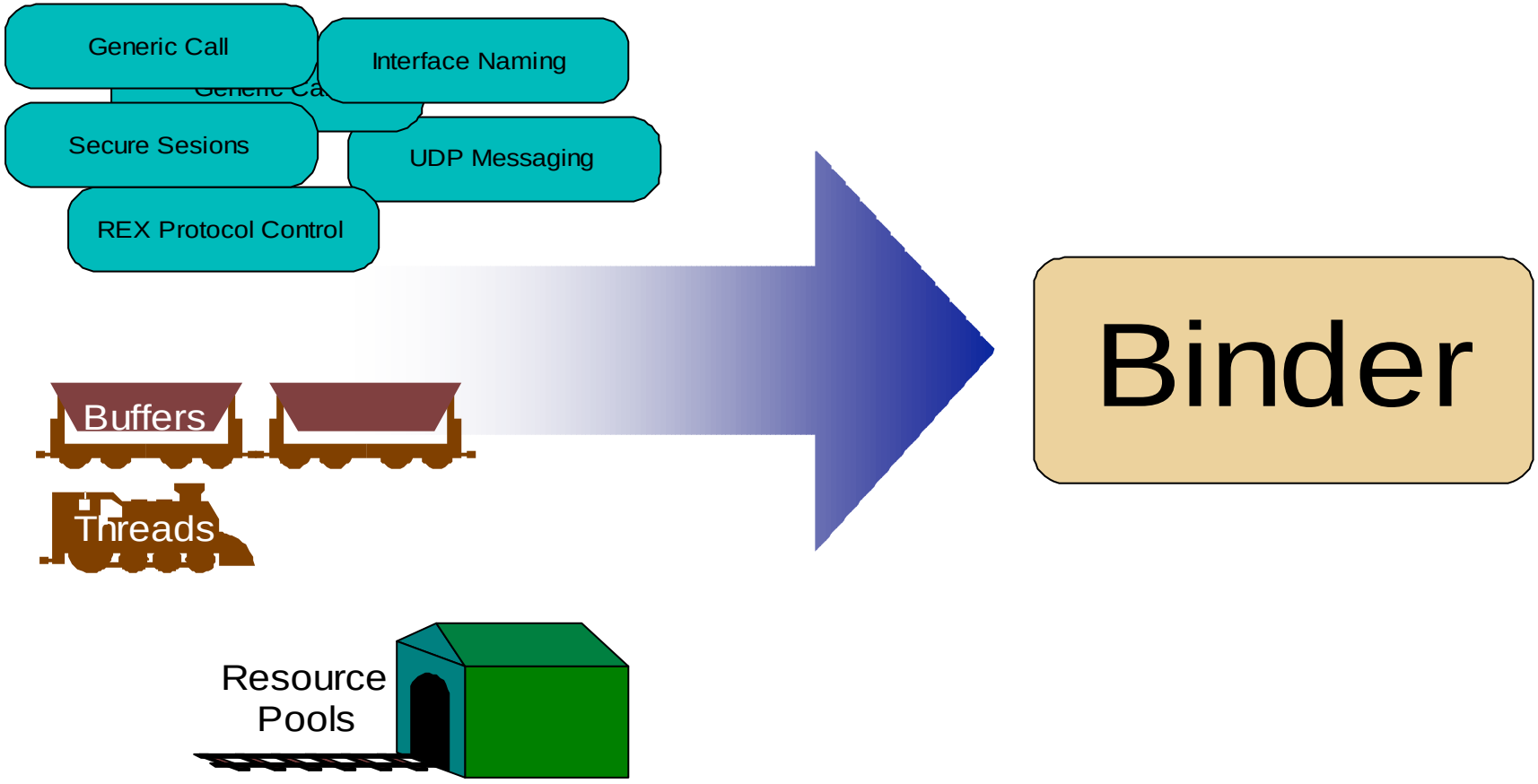
Meaning of an Invocation

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

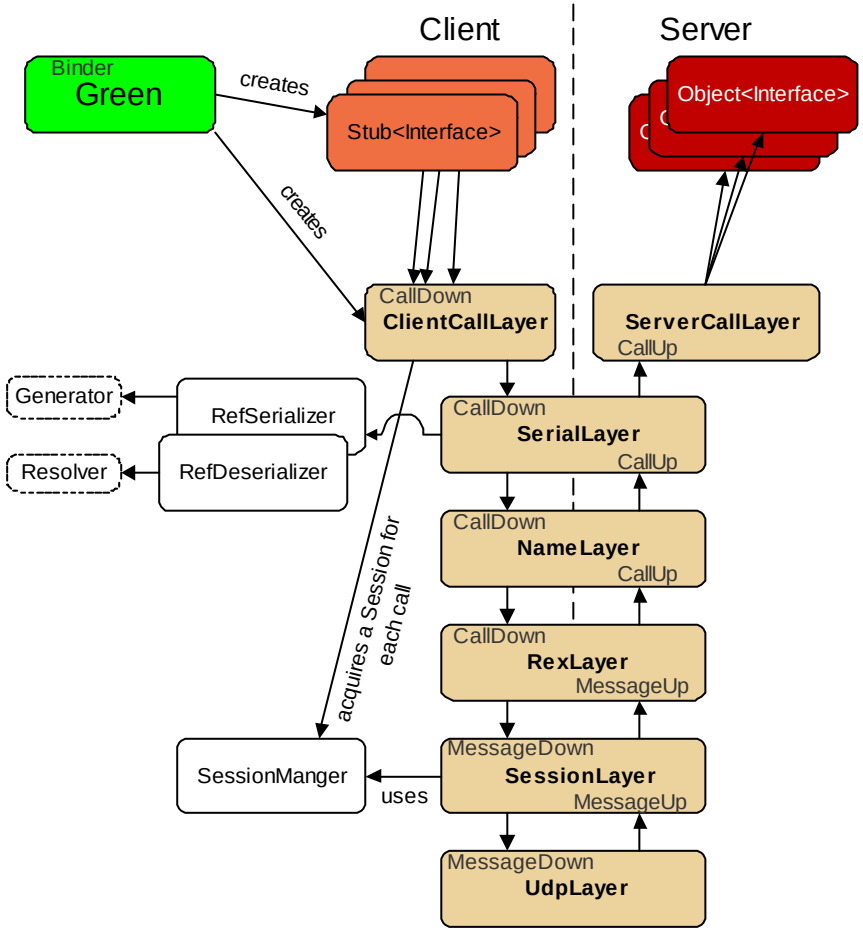
Pointer to an interface, pass a reference to the interface.



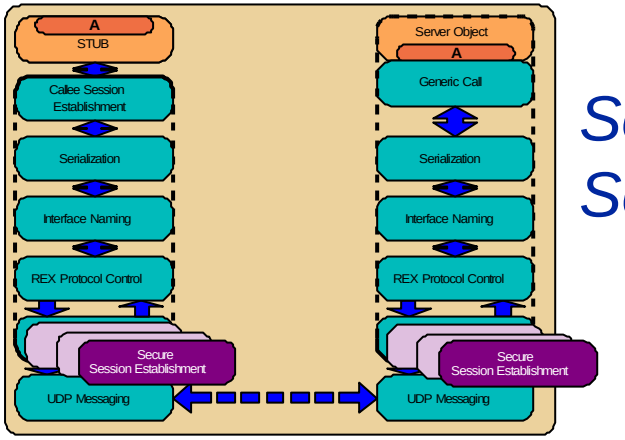
Bits & Pieces


Generic Call
Interface Naming
Secure Sesions
UDP Messaging
REX Protocol Control
Buffers
Threads
Resource
Pools
Binder

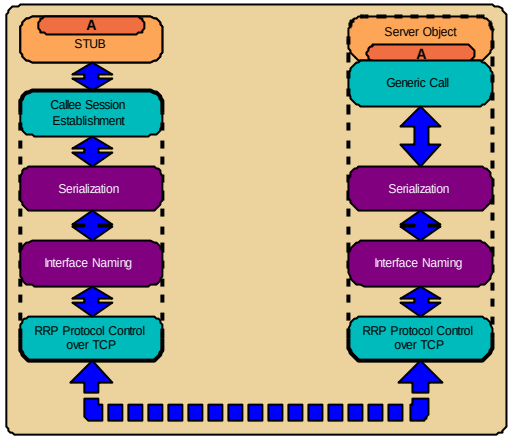
Green


Client
Server
Binder
Green
creates
Stub<Interface>
Object<Interface>
creates
CallDown
ClientCallLayer
ServerCallLayer
CallUp
Generator
RefSerializer
Resolver
RefDeserializer
CallDown
SerialLayer
CallUp
CallDown
NameLayer
CallUp
CallDown
RexLayer
MessageUp
acquires a Session for each call
SessionManger
uses
MessageDown
SessionLayer
MessageUp
MessageDown
UdpLayer

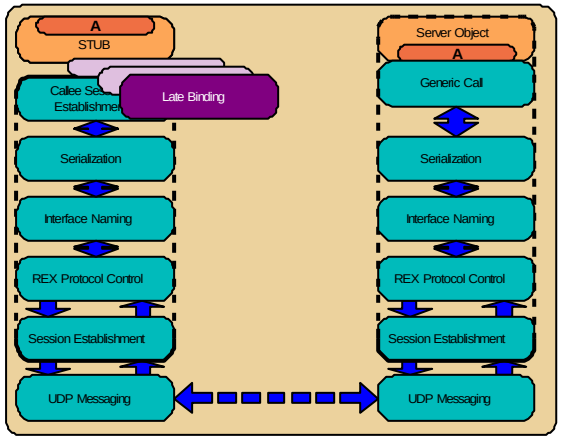
Other Remote Invocation Stacks



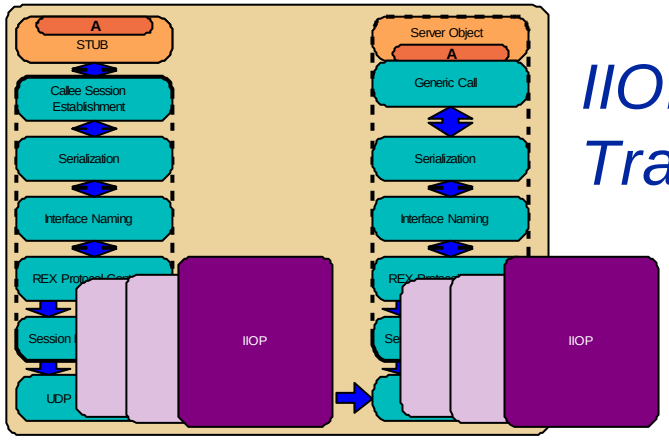
Secure Sessions



RRP Transport (Optimized TCP)

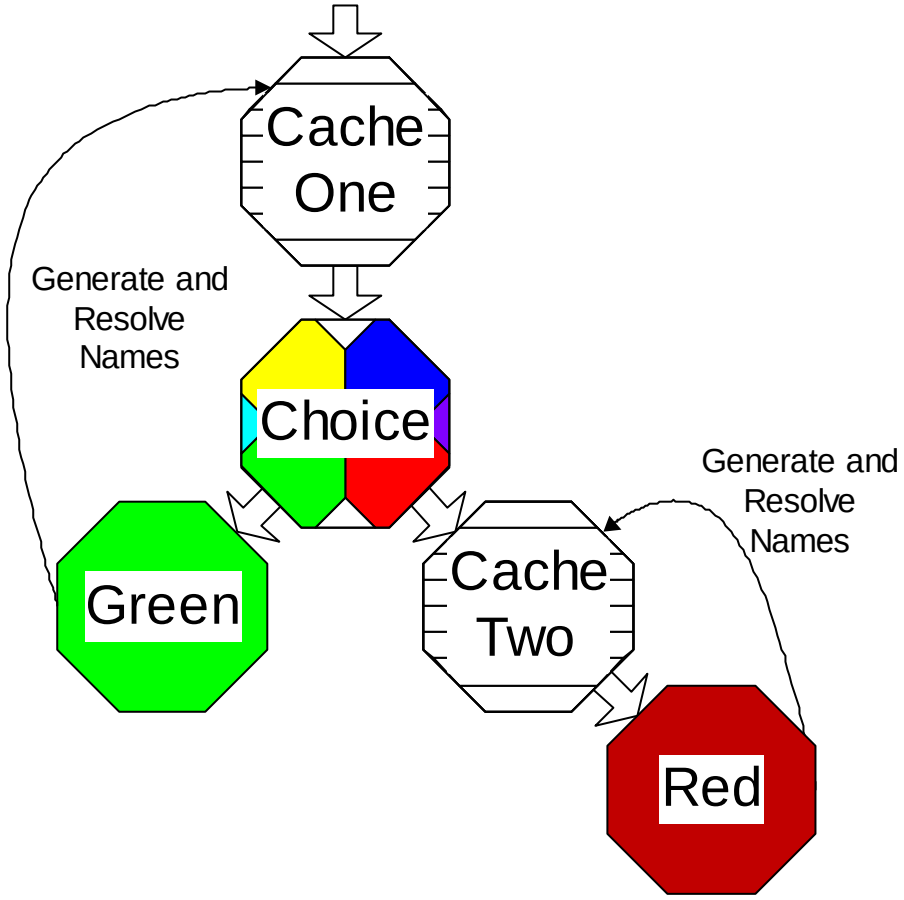


Late Binding



IIOP Transport

Multi Binders



What has been achieved?

- A Framework for composing distribution transparencies
  - Separation of concerns
  - Reuse of code
  - Policy driven configuration
- Other papers describe specific transparencies
  - Migration
  - Relocation
  - Persistence
  - Failure
  - Transaction
  - Security