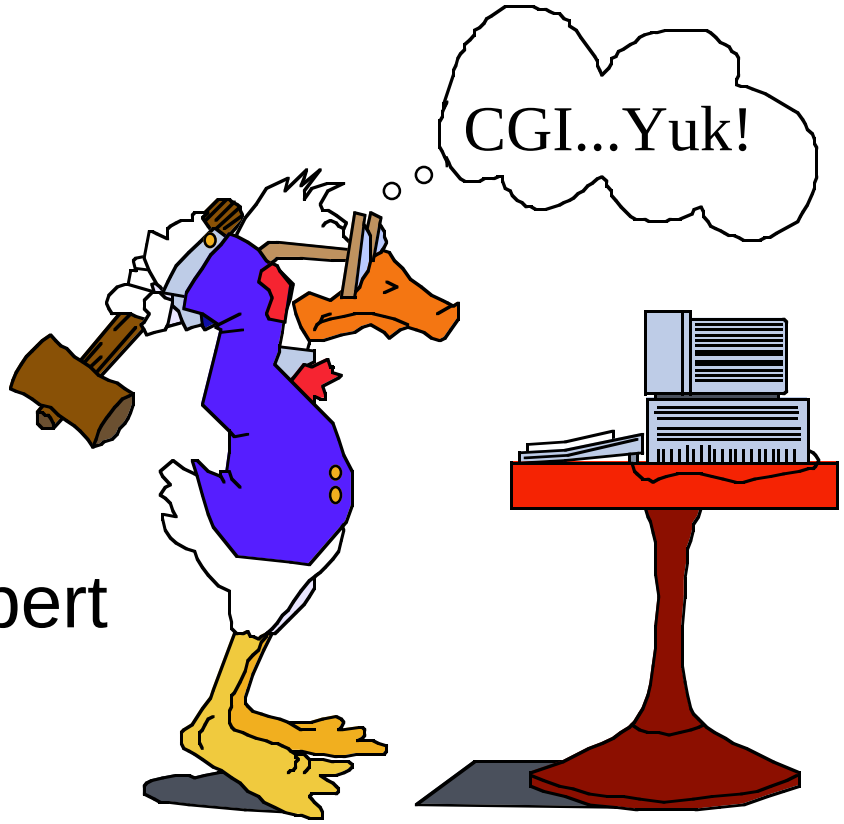
Let CORBA do the talking.....

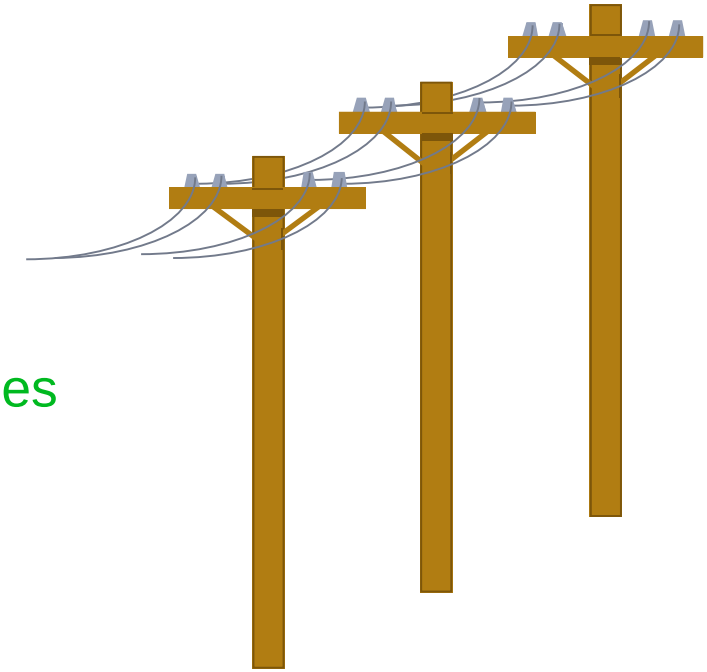


Andrew Herbert

APM Ltd

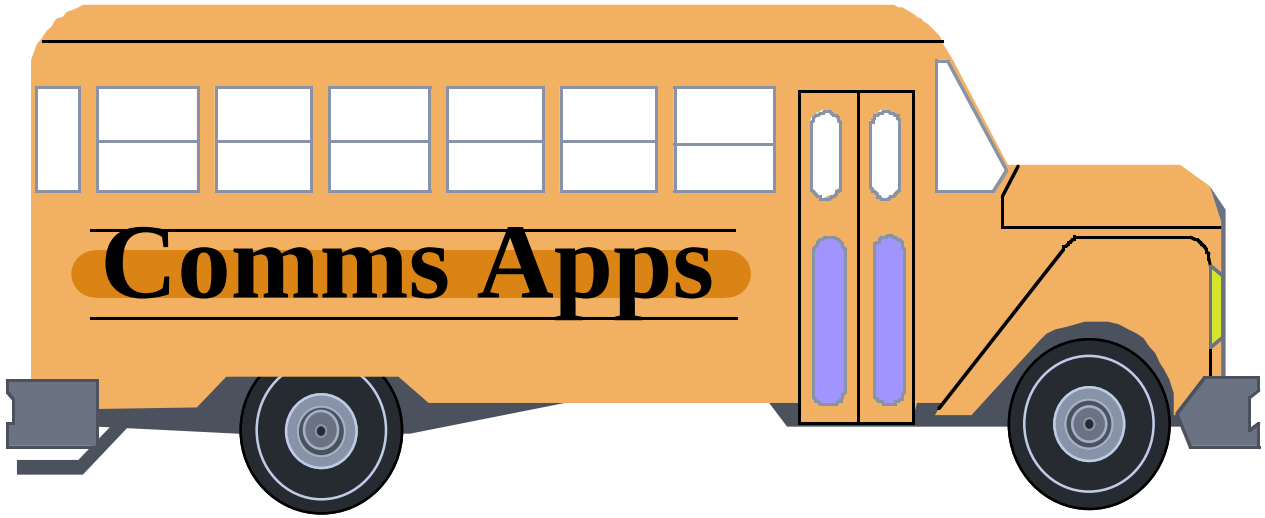
Comms Application Market

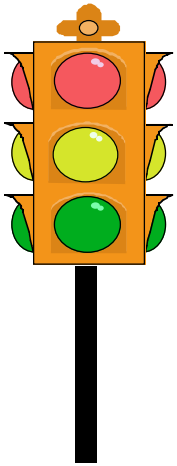
- Is huge
  - Message queueing
    - EDI, EFT, ...
  - Cooperative working
    - publish / subscribe
    - process groups
  - Interactive multimedia services
    - WWW, DAVIC, TINA, ...
- And lacks architecture
  - Low level of abstraction
    - e.g., queue managers, sockets, ASCII
  - Needs CORBA benefits, now



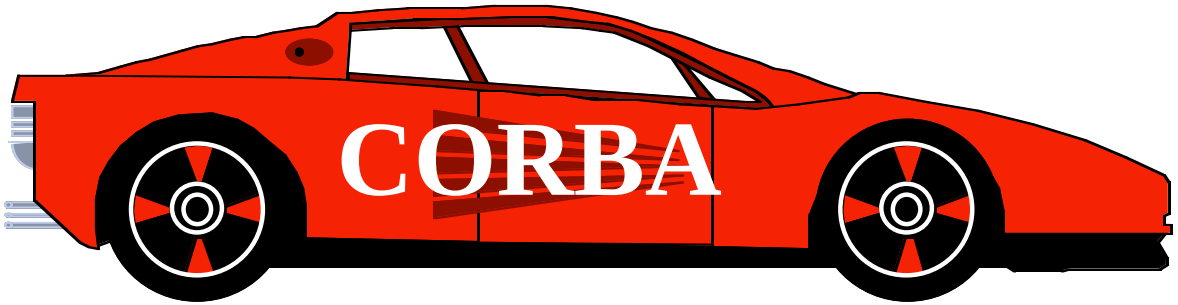
Interactive Multimedia

- Live video and audio applications
  - Entertainment, education, telepresence
- Build comms as application software
  - MIPs available
  - cheaper than hardware or kernel development
- Many formats and protocols
  - soft engineering for flexibility
  - plug and play converters

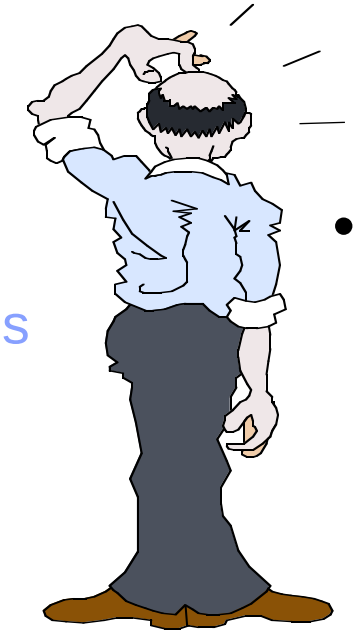




Who gets the market?



What to add to CORBA?



- Nothing
  - build as services
- Connection management
  - plugs and sockets

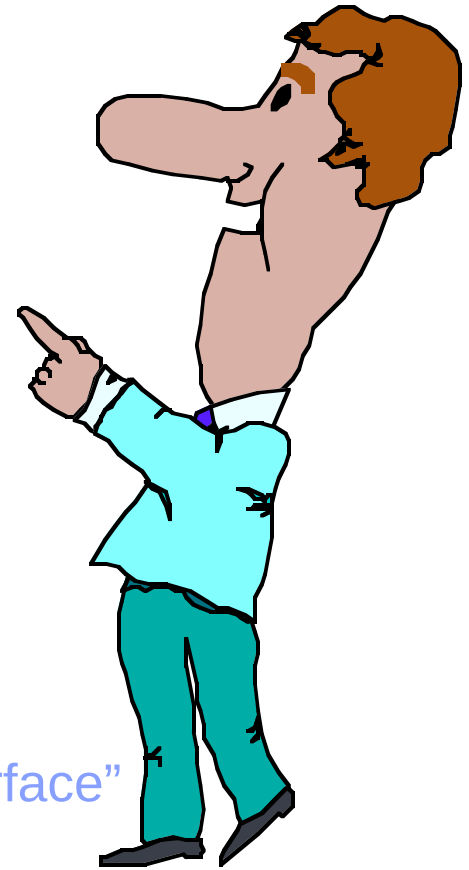
- Connections and Stream interaction
  - sources ands sinks

Necessary Extensions

- Streams
  - interact through flows of data “frames”
- Explicit connection management
  - multi-party coordination, QoS control
- Resource pools
  - user driven scheduling and control
- Synchronous programming [optional]
  - predictable software, QoS guarantees

OMG Standards - Implications

- Streams
  - IDL extensions
- Connections
  - Generalise object adapters
  - Standard object adapter templates
- Resource pools
  - ORB extensions
- *Synchronous programming*
  - new “Synchronous Invocation Interface”

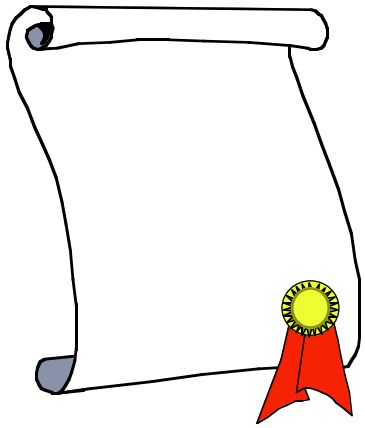


Benefits

- Convergence of communications and distributed processing technology
  - CORBA = universal systems integration “glue”
- Comms services enabled CORBA
  - new services
    - store and forward (esp. for mobile, low bandwidth)
    - publish and subscribe
    - multimedia, multiparty
    - quality of service guarantees
  - faster service development

Foundations

- ISO/ITU RM-ODP
- TINA-C
- ANSA prototypes
  - CNET / Chorus demos
  - ANSAware/RT
  - Quartz (CORBA/MBone interface)
  - DCAN ("Lightweight ATM" ORB)
- EEC *RETINA* project ("Telecoms ORB")



New concepts

- Architecture
  - Connections (ODP “bindings”)
  - Streams and Stream [Object] Adaptors
- Data types
  - Flows, frames (ODP “flows”,“signals”)
- Scheduling
  - Resource pools
- Synchronous programming

CORBA connection model

- Implicit
  - not under programmer control
  - engineered for efficiency and scaling
    - resources bound as late as possible
    - resources released when connection is dormant
    - multiplex wherever possible
- Asymmetric
  - server has no knowledge of potential clients
- Best effort, no QoS bounds

When do we want more?

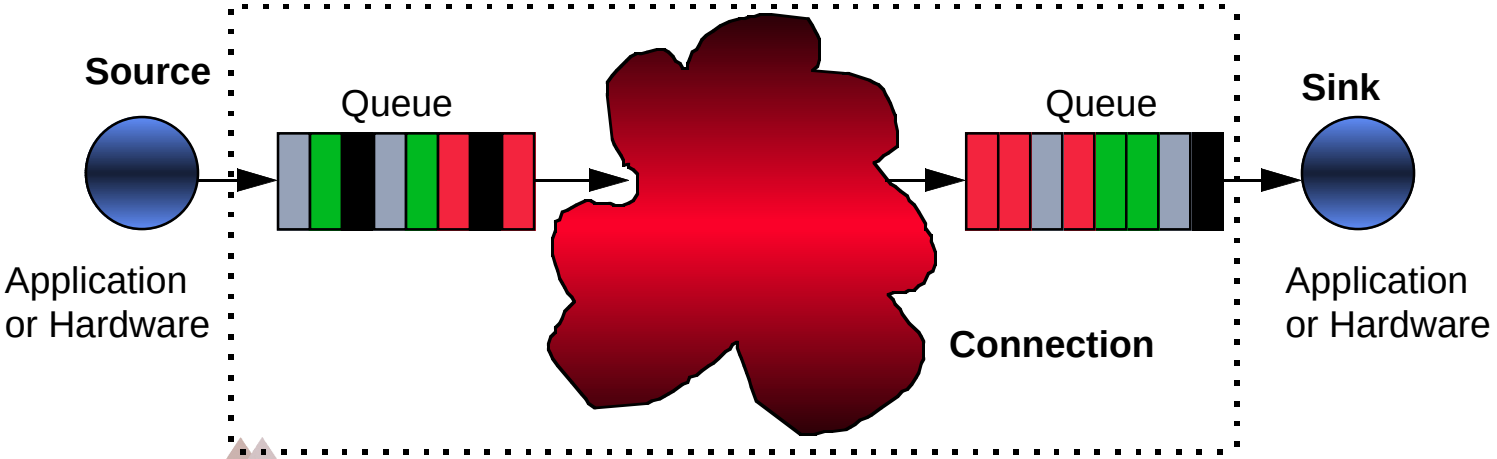
- When exercising control
  - to provide predictable communications
  - to prioritise communications
  - to synchronize communications
  - to give QoS guarantees
    - ordering, security, reliability, performance
  - to control connect/disconnect time
  - to batch connnections
  - to monitor delivery of guarantees
  - to change connections after initial connect

“Connection” is an application

- QoS negotiation and connection setup
  - is too complex to build into the network
  - is to open ended to engineer a generic solution
  - a small set of standard solutions won’t do
  - would benefit from distributed objects
- Connection “by the ORB” defeats encapsulation and breaches security
  - if an object cannot control connections to itself, how can it secure itself?

Flows

- Unidirectional sequence of typed frames from a source to a sink
- Explicitly connect sources and sinks through logical “connection”



Application Defined Framing

- Some flows have natural formats, others don't
- Engineering decision to decide best payload for each processing step
  - hardware framing format might differ
  - alternative application formats might exist for same hardware format
  - marshalled by flow "stubs"
    - abstraction, efficency, portability

Flows

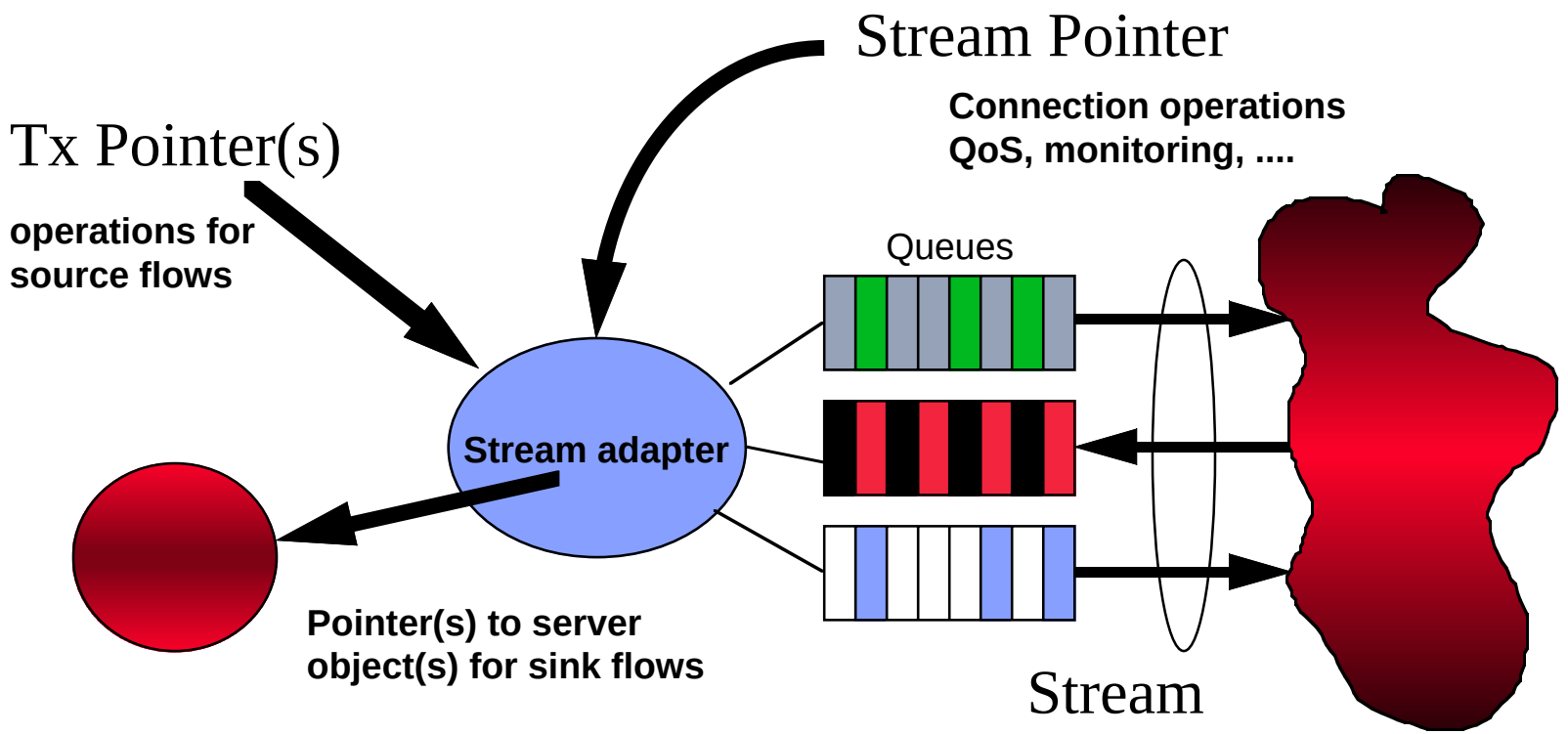
- Multiple frame types in flow enable:
  - base + delta coding
  - changes in compression algorithm
  - application mux / demux
  - in-band control
- Model frame interactions as *oneway* operation with *in* arguments
  - n.b. interaction is with the connection “object”, not the ultimate sink(s)

Streams

- A bundle of related flows to / from an application
  - named by a *stream pointer* (c.f. ObjPtr)
  - connected as a single entity
  - connection enables communication
- Can be bi-directional
  - exploit duplex comms technology
  - abstract over connection models

Stream Adaptor

- Analogous to server object adaptor



Stream IDL

- Connection operations
  - regular CORBA IDL
- Frame interaction operations
  - re-interpret *oneways* as frames
    - source and sink for every frame, one IN flow, one OUT flow
  - pragmas for flows, IN vs OUT
  - native syntax
    - FRAMES clause, IN and OUT flow tags, frame formats
    - FRAMES { IN flowA {frameX (arg1, ...argn); ...}; ...}

Stream adapter template

- Create new adaptor, disconnected
  - <Adaptor>.New (RxObj, &StrRef, &TxRef1,..., QoS)
- Create new adaptor, connected
  - <Adaptor>.New (RxObj, StrRef, &TxRef1,..., QoS)
- Dynamic connection control
  - YourBindingRef.Bind(MyBindingRef, ..., QoS)
  - MyBindingRef.ChangeQoS(....)
- Interaction
  - TxRef.Message(Contents)

Template library

- Compiler has library of templates
  - defining portability API for CORBA comms apps
- Programmer instantiates template for application type
  - gives type safe connections
- Engineering substitutes generic adapters
  - link through ‘Any (objPtr)’.

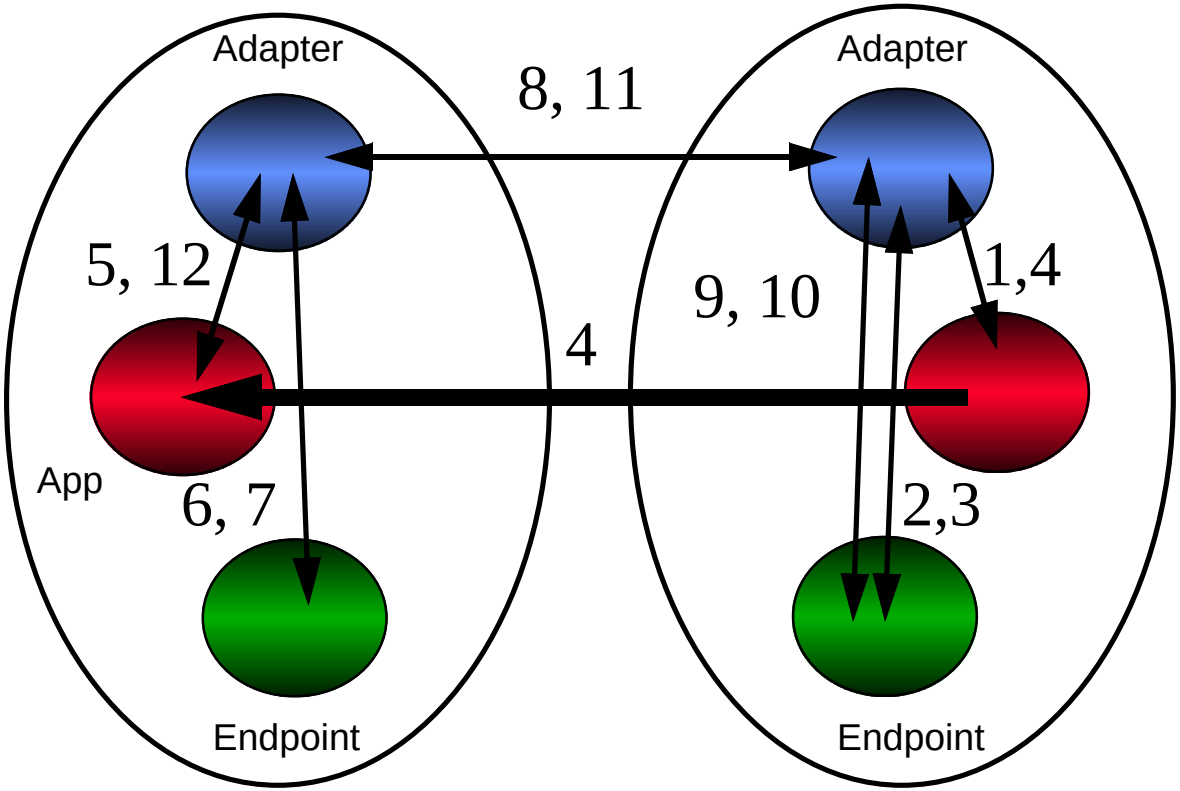
Adaptable Adaptors

- Lots of possibilities
  - keep connection model, resource model and formats separate
  - allow for composition (inheritance)
  - base set for major industry standards
  - reap all the benefits of distributed objects
  - allow for “internal” connections
    - transparent format conversion, synchronisation etc
  - allow for network connection managers
- N.B. Remoting TxRef sends StreamRef

Architectural Footnote

- All endpoints (clients, servers, producers, consumers have object adapters)
  - but adaptors can be transparent
    - and client ones currently are; server adapters could be!!
- An adaptor manages binding between application, local ORB and connection to remote ORB
- Binding clients to servers can be implicit OR explicit

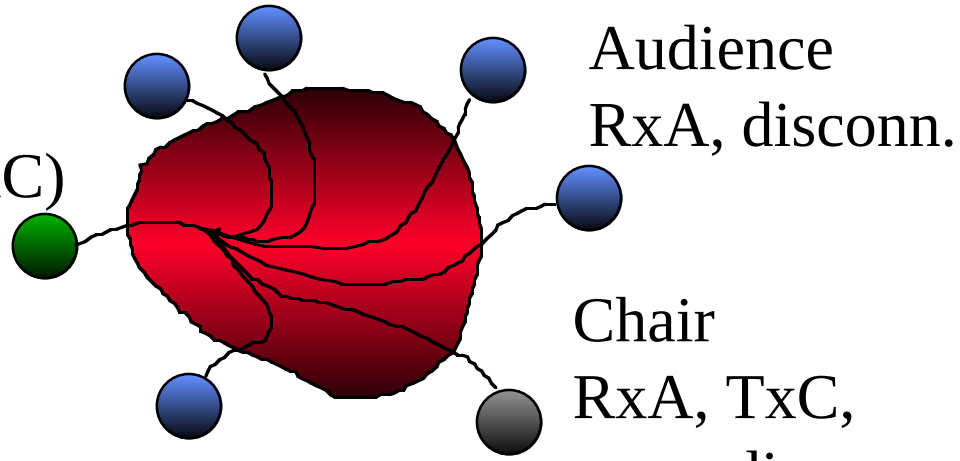
Making Connections



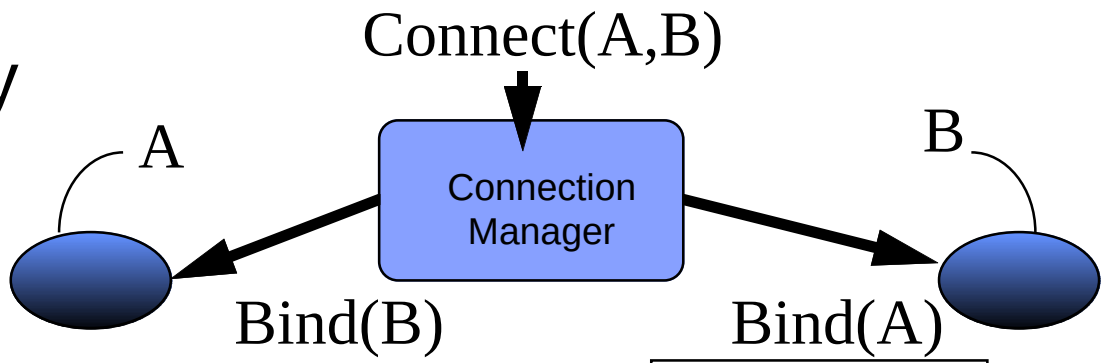
1: Create Adapter
2: Create Endpoint
3: Return address
4: Pass BindingRef
5: Create and Connect Adapter
6: Create and Connect Endpoint
7: Return address
8: Negotiate
9: Connect Endpoint

Other Forms of Connections

- Multi-party

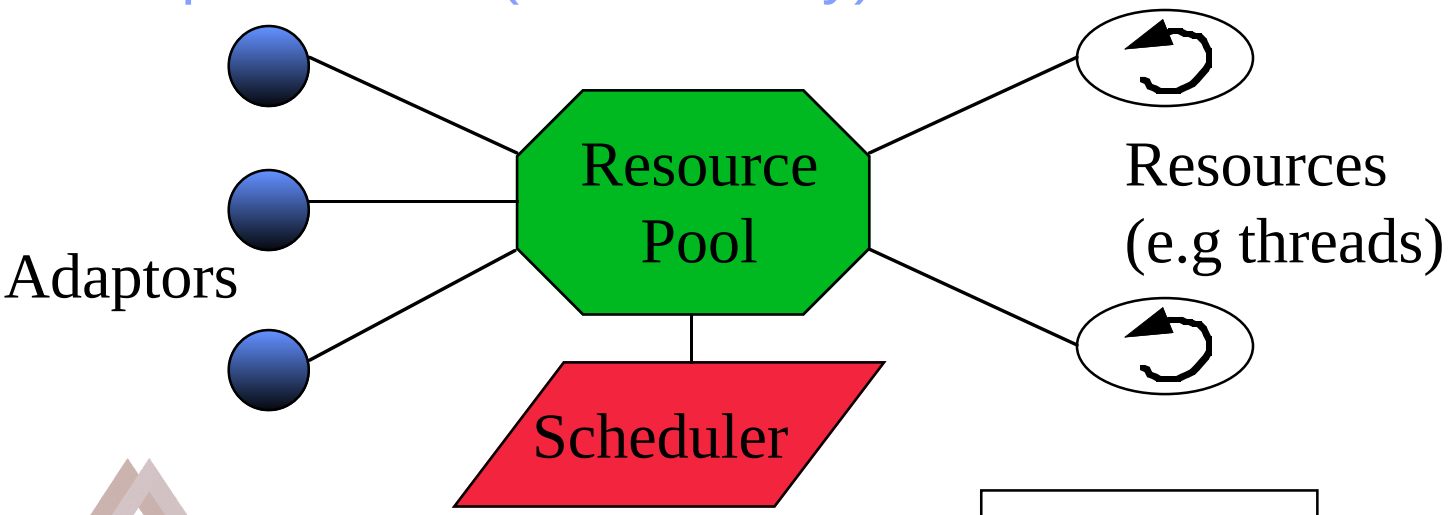


- Third Party



Resource Pools

- Control muxing of threads, buffers, memory, objects
  - Pool.New (Resources, Scheduler)
  - Pool.Change (Resources)
  - Adaptor.Share (Pool, Policy)

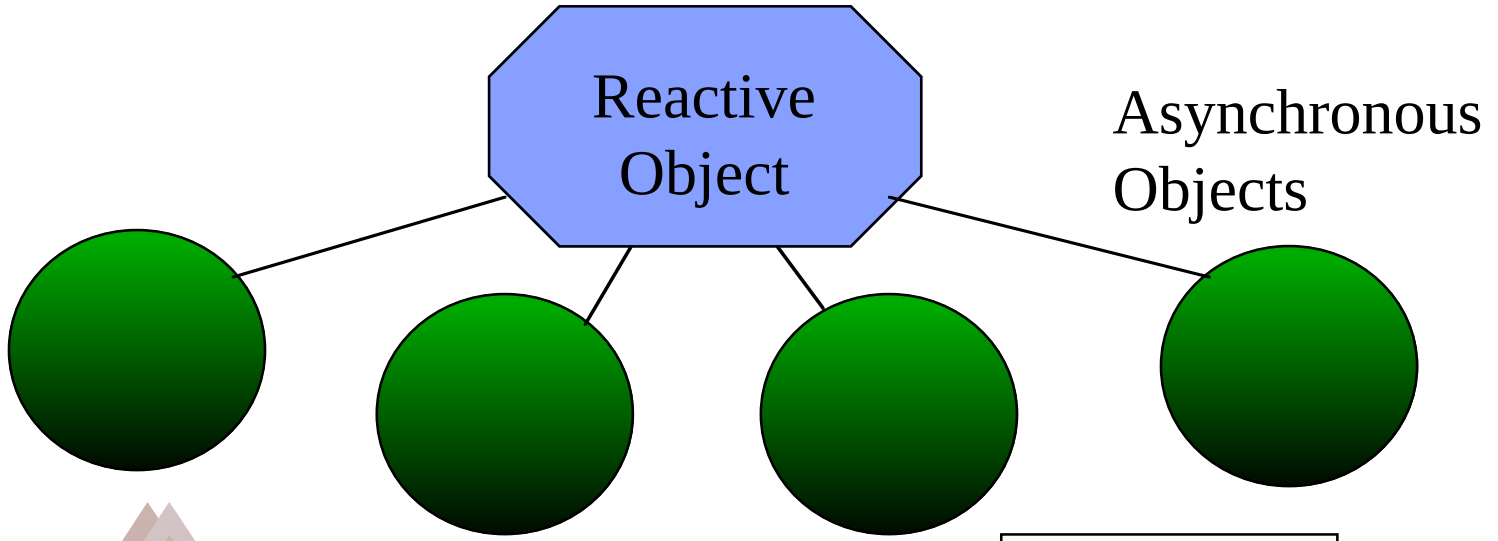


Quality of Service

- QoS is end-to-end
  - negotiate parameter values as part of connection
  - trade-off resources against formats
    - e.g. MBone ABR strategies vs. Telco CBR
  - need guarantees from each element in path
    - RSVP in Internet, Periodic threads in OS Scheduler
    - guarantees come from predictability
- Synchronous hardware is predictable
- How to make software predictable?

Synchronous Programming

- Model controller as a *reactive* object
  - at each instant take inputs, produce results
  - resources and real-time synchronisation guaranteed by the operating system



The Theory

- Synchronous hypothesis
  - execution is a sequence of discrete events
  - reactions are instantaneous
    - no concurrency between instants
    - concurrency within instant is serialised
  - deterministic (worst case) behaviour
    - bounded execution paths, calculable in advance
    - with known resource requirement
    - gives predictable timing and reproducible behaviour
  - e.g. ESTEREL, Reactive-C

Implementation

- Treat frame tx and rx as an event (signal)
  - signal has name and arguments
  - signal has direction (in or out)
  - signals are grouped into interfaces
    - c.f. ESTREL etc global broadcast
      - gives same local and remote semantics
      - enables event “broadcast” to be scoped
      - enables multiple instances of same interface
  - real-time synchronization via in signals from scheduler

SII Semantics (1)

- `tx = expr “!” signalName arguments`
  - fire and forget
- `rx = expr “?” signalName`
  - wait for signal and return results
- `clock ? hour ; bell ! ring`
  - chime on the hour
- `[clock ? second]; [clock ? second] || [button ? press ; bell ! ring]`
  - debounce the button

SII Semnatics (2)

- `presence = “present” condition`
- `await = “await” condition`
- `cond’n = “(“ { expr “?” signal } “)”`
- `watchdog = “during” block “watch” condition block`

  – signals are held over until the instant in which the dynamically last signal fires

Wrap Up

- Important need to fulfil
  - reconcile CORBA and comms-oriented apps
  - strong pressure from TelSIG
- Some new concepts
  - streams, connections, pools, QoS, reactive objects
- CORBA extensions
  - stream IDL, stream [object] adapters, SII
- Firm foundations
  - ODP framework, working examples