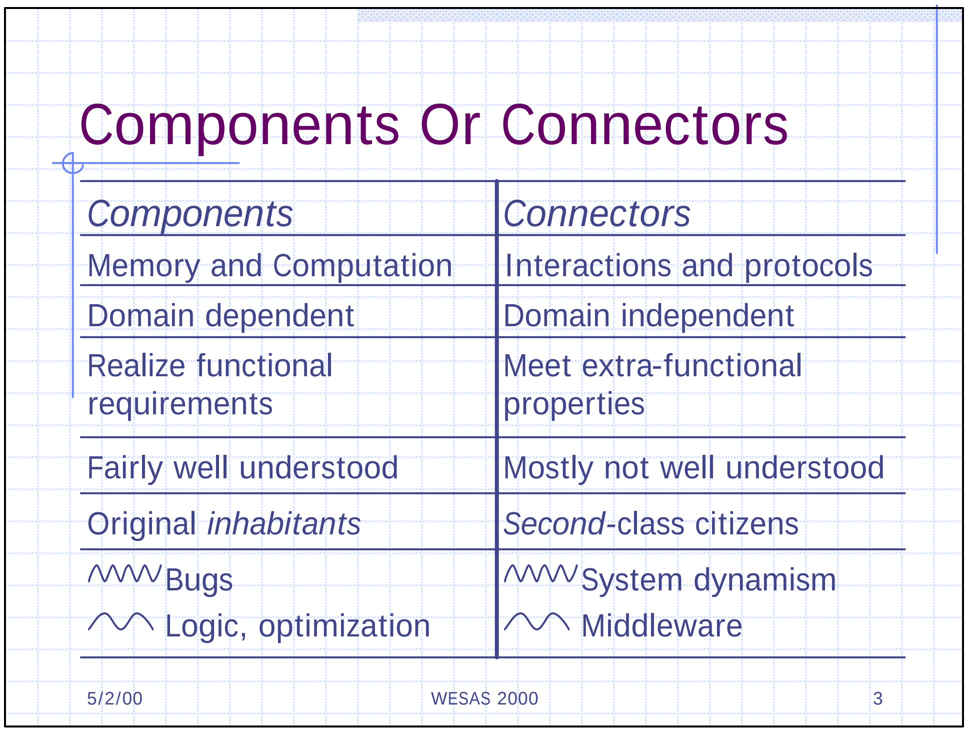
Software Connectors – A Taxonomy Approach

Nikunj Mehta
USC-CSE

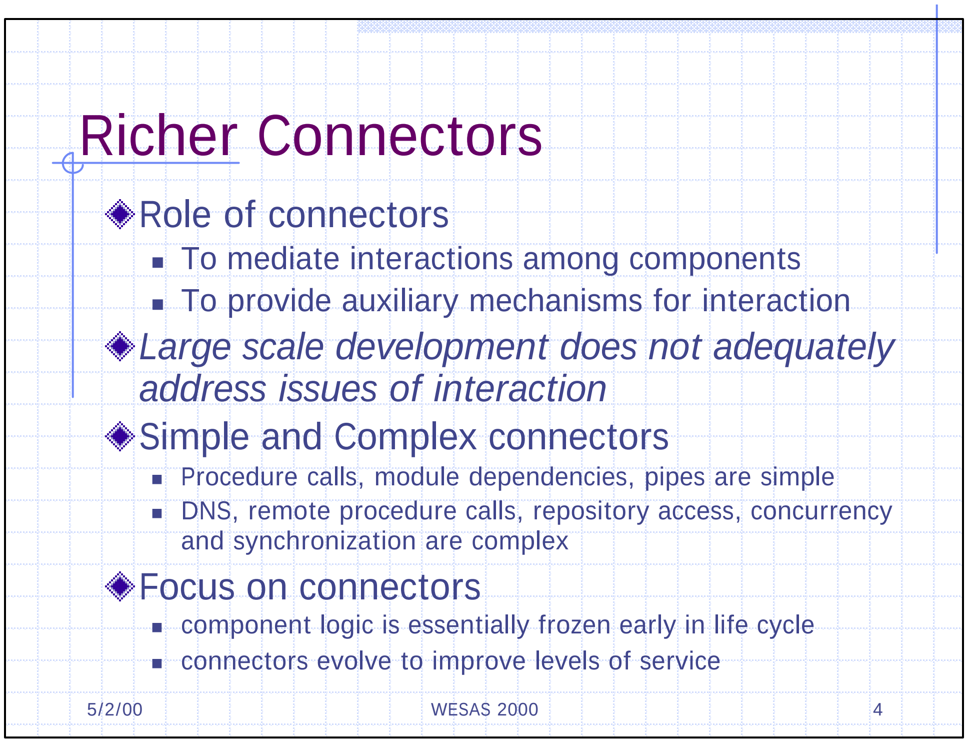
Software Architectures

- ◈Model run time structure of applications
- ◈Describe the elements, form and rationale
- ◈Used to analyze system properties
- ◈Formal representation of elements
- ◈Domain specific styles
- ⌘Limited modeling primitives
- ⌘Implementation decisions considered outside the scope



Components Or Connectors

Components	*Connectors*
Memory and Computation	Interactions and protocols
Domain dependent	Domain independent
Realize functional requirements	Meet extra-functional properties
Fairly well understood	Mostly not well understood
Original *inhabitants*	*Second*-class citizens
∿∿∿∿ Bugs ∿∿ Logic, optimization	∿∿∿∿ System dynamism ∿∿ Middleware

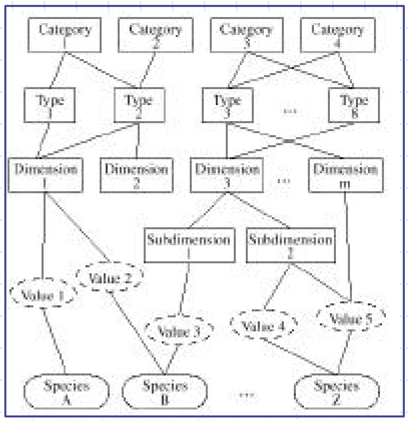


Richer Connectors

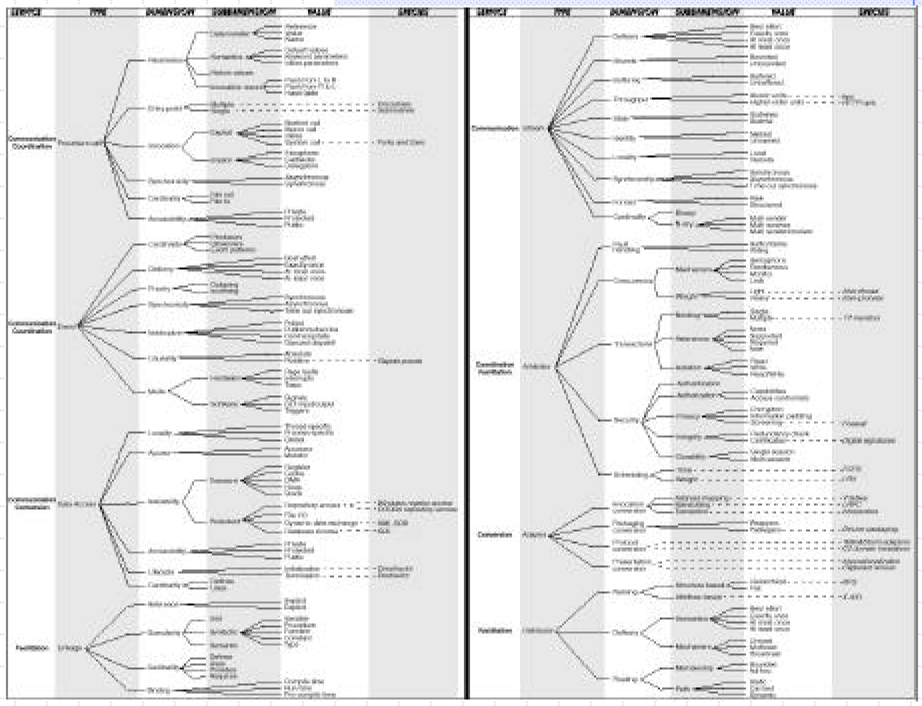
- Role of connectors
  - To mediate interactions among components
  - To provide auxiliary mechanisms for interaction
- *Large scale development does not adequately address issues of interaction*
- Simple and Complex connectors
  - Procedure calls, module dependencies, pipes are simple
  - DNS, remote procedure calls, repository access, concurrency and synchronization are complex
- Focus on connectors
  - component logic is essentially frozen early in life cycle
  - connectors evolve to improve levels of service

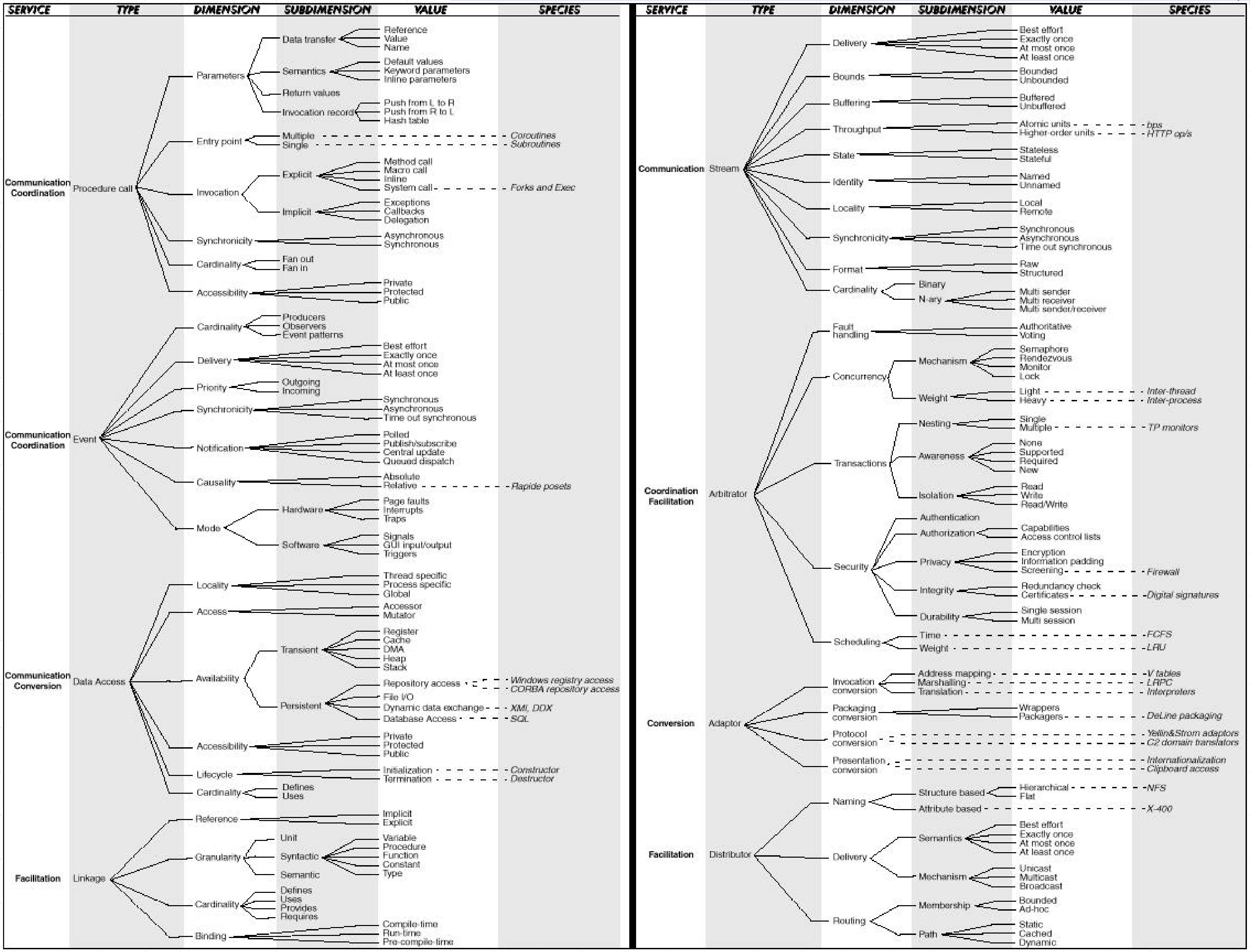
Classification Framework

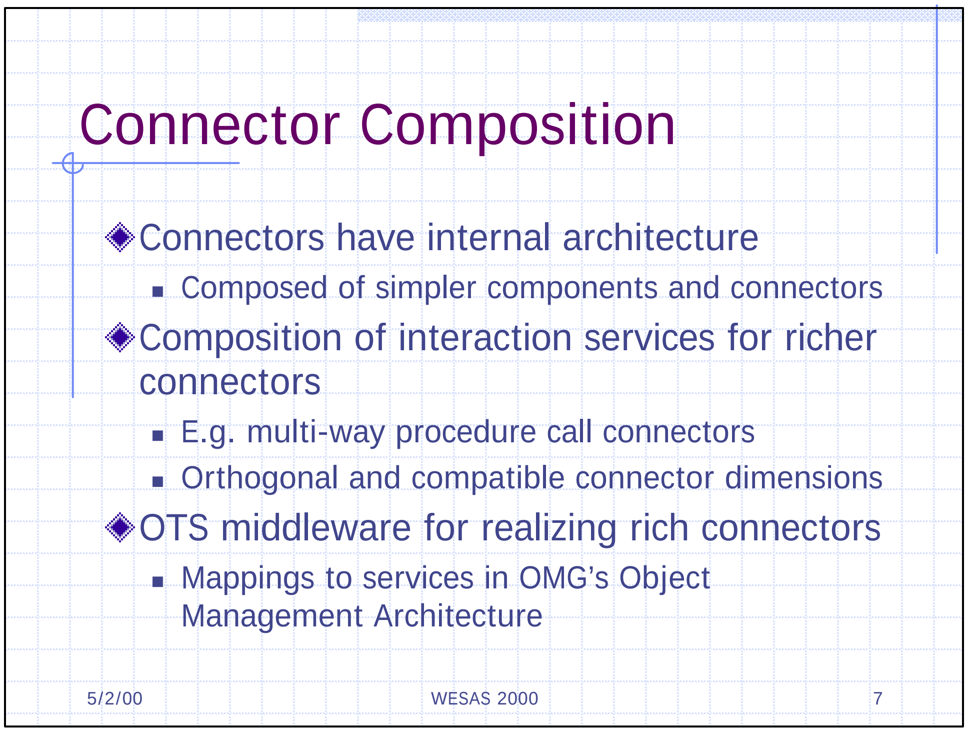
- Atomic elements
  - Ducts, data transfer and control transfer
- Service categories
  - Communication
  - Coordination
  - Conversion
  - Facilitation
- Connector types, dimensions
  - Finitely many values



5/2/00 WESAS 2000 5



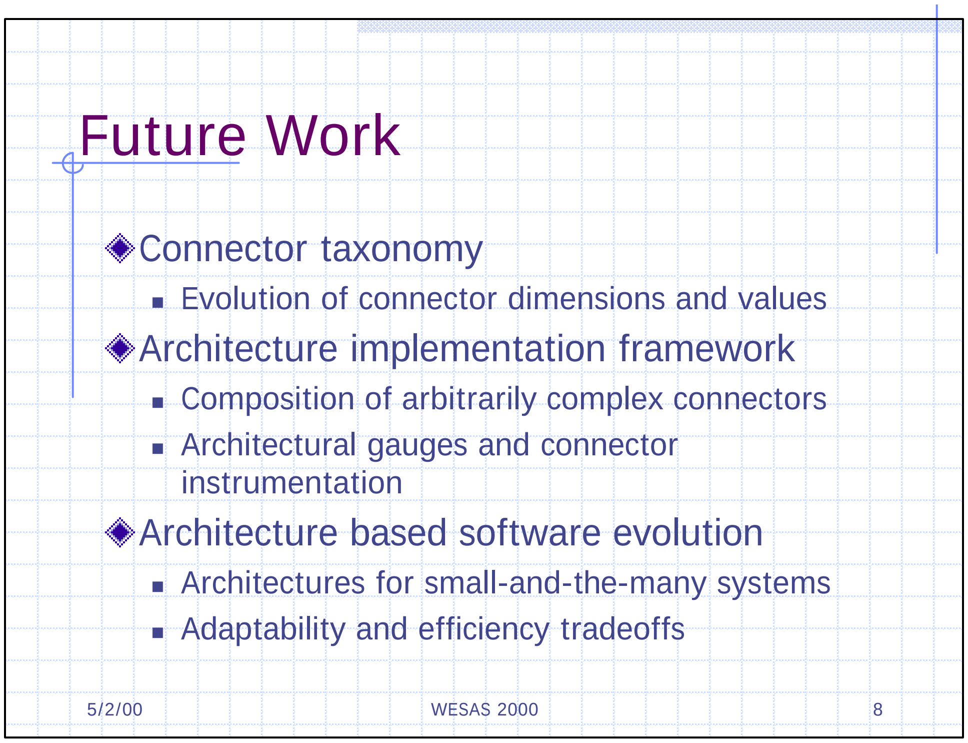

SERVICE | TYPE | DIMENSION | SUBDIMENSION | VALUE | SPECIES
Communication Coordination – Procedure call
  Parameters
    Data transfer: Reference, Value, Name
    Semantics: Default values, Keyword parameters, Inline parameters
    Return values
    Invocation record: Push from L to R, Push from R to L, Hash table
  Entry point
    Multiple – Coroutines
    Single – Subroutines
  Invocation
    Explicit: Method call, Macro call, Inline, System call – Forks and Exec
    Implicit: Exceptions, Callbacks, Delegation
  Synchronicity: Asynchronous, Synchronous
  Cardinality: Fan out, Fan in
  Accessibility: Private, Protected, Public
Communication Coordination – Event
  Cardinality: Producers, Observers, Event patterns
  Delivery: Best effort, Exactly once, At most once, At least once
  Priority: Outgoing, Incoming
  Synchronicity: Synchronous, Asynchronous, Time out synchronous
  Notification: Polled, Publish/subscribe, Central update, Queued dispatch
  Causality: Absolute, Relative – Rapide posets
  Mode
    Hardware: Page faults, Interrupts, Traps
    Software: Signals, GUI input/output, Triggers
Communication Conversion – Data Access
  Locality: Thread specific, Process specific, Global
  Access: Accessor, Mutator
  Availability
    Transient: Register, Cache, DMA, Heap, Stack
    Persistent: Repository access – Windows registry access, CORBA repository access; File I/O; Dynamic data exchange – XML, DDX; Database Access – SQL
  Accessibility: Private, Protected, Public
  Lifecycle: Initialization – Constructor; Termination – Destructor
  Cardinality: Defines, Uses
Facilitation – Linkage
  Reference: Implicit, Explicit
  Granularity
    Unit
    Syntactic: Variable, Procedure, Function, Constant, Type
    Semantic
  Cardinality: Defines, Uses, Provides, Requires
  Binding: Compile-time, Run-time, Pre-compile-time
Communication – Stream
  Delivery: Best effort, Exactly once, At most once, At least once
  Bounds: Bounded, Unbounded
  Buffering: Buffered, Unbuffered
  Throughput: Atomic units – bps; Higher-order units – HTTP op/s
  State: Stateless, Stateful
  Identity: Named, Unnamed
  Locality: Local, Remote
  Synchronicity: Synchronous, Asynchronous, Time out synchronous
  Format: Raw, Structured
  Cardinality
    Binary
    N-ary: Multi sender, Multi receiver, Multi sender/receiver
Coordination Facilitation – Arbitrator
  Fault handling: Authoritative, Voting
  Concurrency
    Mechanism: Semaphore, Rendezvous, Monitor, Lock
    Weight: Light – Inter-thread; Heavy – Inter-process
  Transactions
    Nesting: Single, Multiple – TP monitors
    Awareness: None, Supported, Required, New
    Isolation: Read, Write, Read/Write
  Security
    Authentication
    Authorization: Capabilities, Access control lists
    Privacy: Encryption, Information padding, Screening – Firewall
    Integrity: Redundancy check, Certificates – Digital signatures
    Durability: Single session, Multi session
  Scheduling
    Time – FCFS
    Weight – LRU
Conversion – Adaptor
  Invocation conversion
    Address mapping – V tables
    Marshalling – LRPC
    Translation – Interpreters
  Packaging conversion: Wrappers; Packagers – DeLine packaging
  Protocol conversion – Yellin&Strom adaptors; C2 domain translators
  Presentation conversion – Internationalization; Clipboard access
Facilitation – Distributor
  Naming
    Structure based: Hierarchical – NFS; Flat
    Attribute based – X-400
  Delivery
    Semantics: Best effort, Exactly once, At most once, At least once
    Mechanism: Unicast, Multicast, Broadcast
  Routing
    Membership: Bounded, Ad-hoc
    Path: Static, Cached, Dynamic



Connector Composition

- Connectors have internal architecture
  - Composed of simpler components and connectors
- Composition of interaction services for richer connectors
  - E.g. multi-way procedure call connectors
  - Orthogonal and compatible connector dimensions
- OTS middleware for realizing rich connectors
  - Mappings to services in OMG's Object Management Architecture

5/2/00 WESAS 2000 7



Future Work

- Connector taxonomy
  - Evolution of connector dimensions and values
- Architecture implementation framework
  - Composition of arbitrarily complex connectors
  - Architectural gauges and connector instrumentation
- Architecture based software evolution
  - Architectures for small-and-the-many systems
  - Adaptability and efficiency tradeoffs

5/2/00 WESAS 2000 8