RMI and CORBA

Why both are valuable tools

by
Brian Gilstrap

with thanks to Mark Volkmann

Introduction

- This talk is about RMI and CORBA, not "RMI vs CORBA"
- It's about using the right tool for the job
- RMI and CORBA have different "design centers"; as a result, they are good for solving different (but overlapping) sets of problems
- This talk centers around Java

RMI & CORBA Characteristics

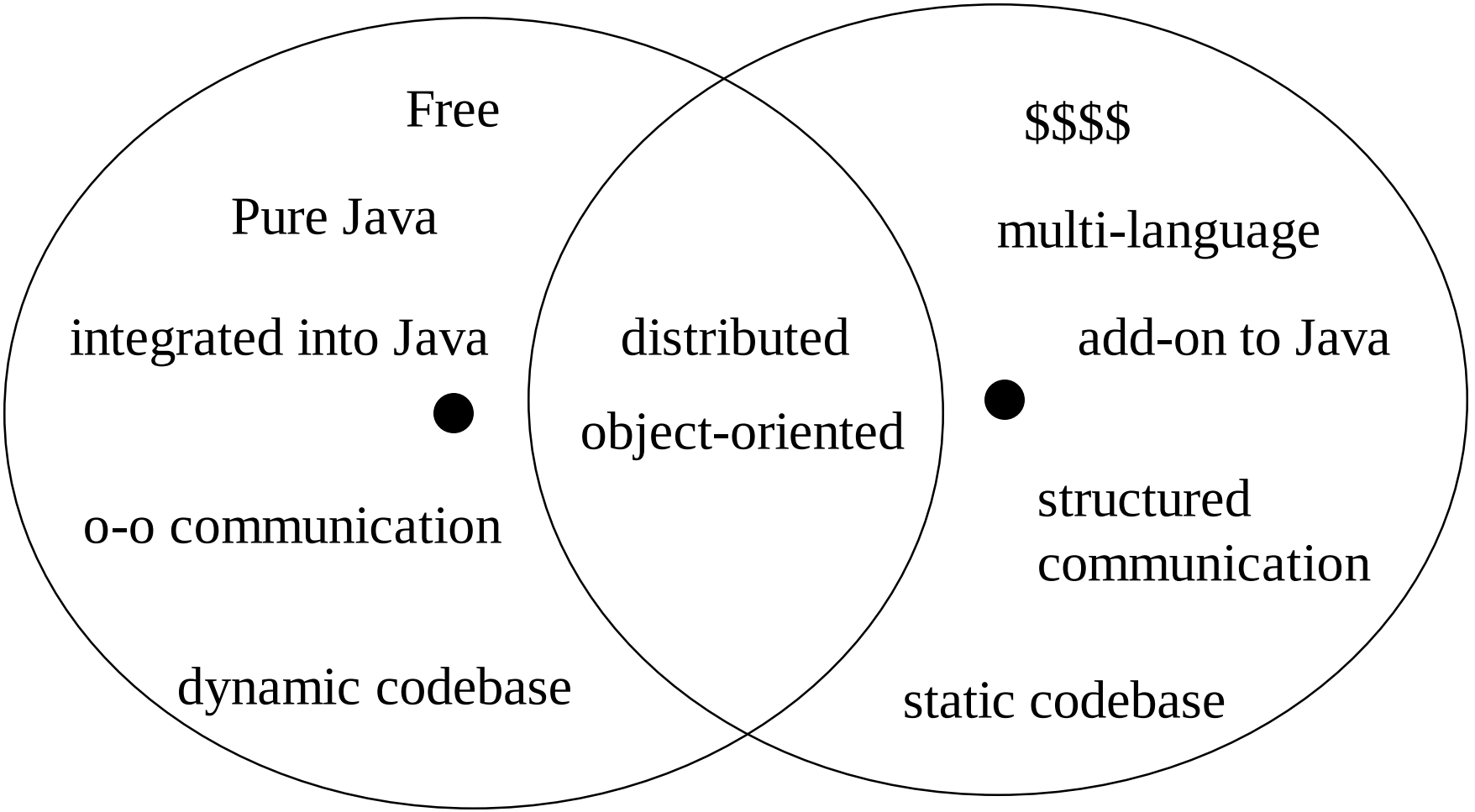
RMI

- Pure Java
- Free
- Simple to use
- Integrated into platform
- Object-oriented communication
- Dynamic codebase
- Performance is not as good (depending on application)
- Less mature (but learned from CORBA issues/mistakes)
- Maturing quickly

CORBA

- Multi-language
- $$$
- Harder to use
- Add-on
- Structured communication
- Static codebase
- Performance is better (depending on application)
- More mature (but 1st of its kind in the industry)
- Maturing slowly

These characteristics result in Design Centers



RMI features/limitations

- Pure Java
  - RMI is pure Java (no other languages), but it supports distributed OO computing in Java very well
  - Can still mix in socket communication and native code (JNI) to access non-Java (e.g. legacy) systems
    - this may be easier or harder than using CORBA, depending on the problem

RMI features/limitations (cont)

- Free
  - comes as part of the Java Development Kit (JDK) and is part of the Java Runtime Environment (JRE)
  - no runtime costs, developer costs, etc.

RMI Features/Limitations (Cont)

- Simple to use - providing remote access
  - extend “Remote” interface & handle exceptions
  - use the “rmic” utility to generate stubs & skeletons (produces ‘.class’ files)
  - registry provides bootstrap for finding remote objects
    - must know host

RMI Features/Limitations (Cont)

- Simple to use - sending objects
  - implement “Serializable” interface
  - optionally specify codebase
  - automatic version mismatch detection

RMI Features/Limitations (Cont)

- Integrated into platform
  - standard way of using RMI
  - source code is the same for any platform
  - bytecode is cross-platform just like other Java
  - remote method calls are nearly identical to normal method calls
  - distributed garbage collection preserves automatic memory management

RMI Features/Limitations (Cont)

- Object-oriented communication
  - can easily send references to remotely accessible objects to other VMs
    - using the “Remote” interface
  - can easily send copies of objects to other VMs
    - using “Serializable” interface
  - can get more sophisticated
    - overriding “read/writeObject”
    - using “Externalizable”

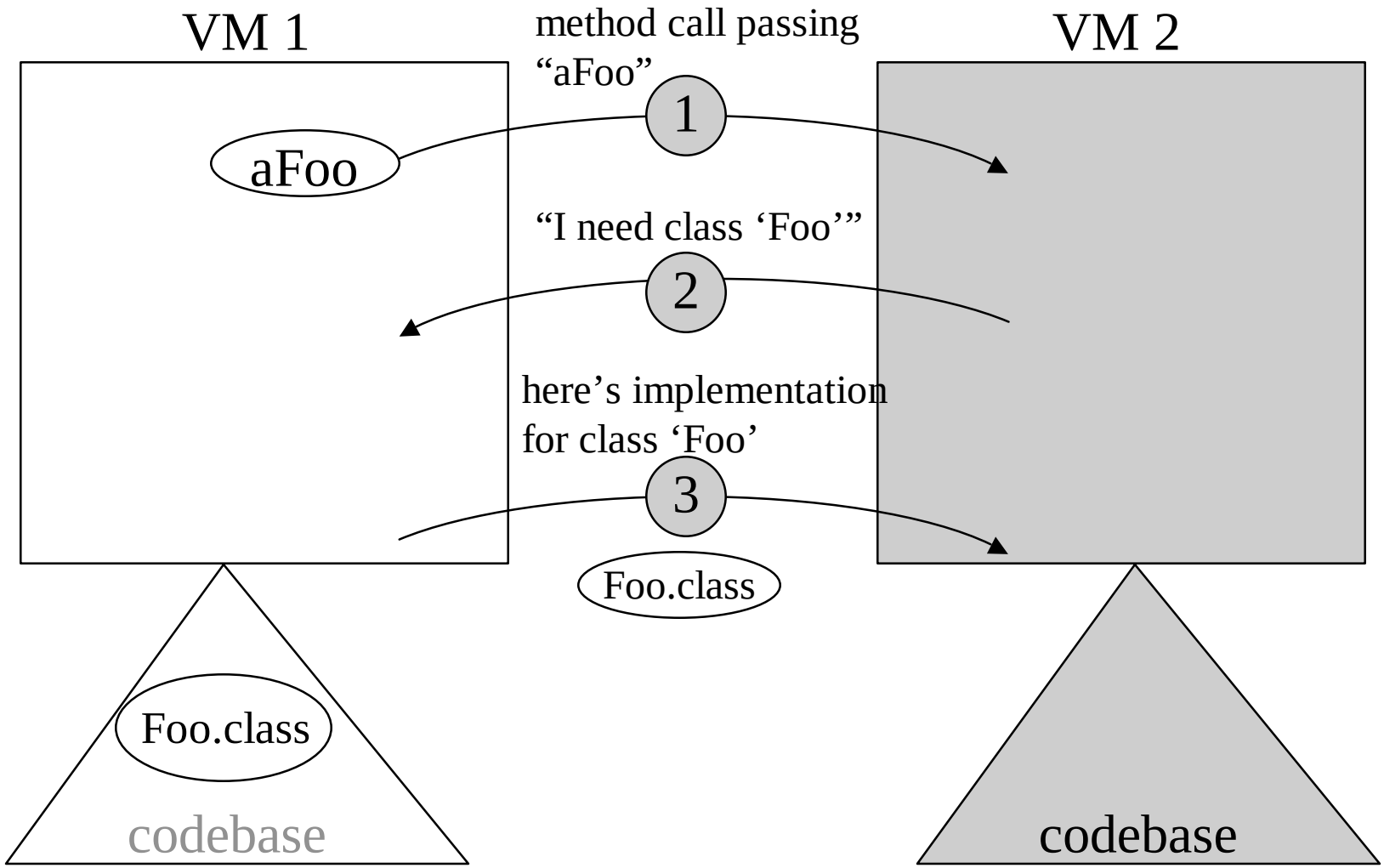
RMI Features/Limitations (Cont)

- Object-oriented communication (continued)
  - distributed garbage collection
    - allows server VMs to garbage collect unused objects/resources when those objects/resources are no longer needed

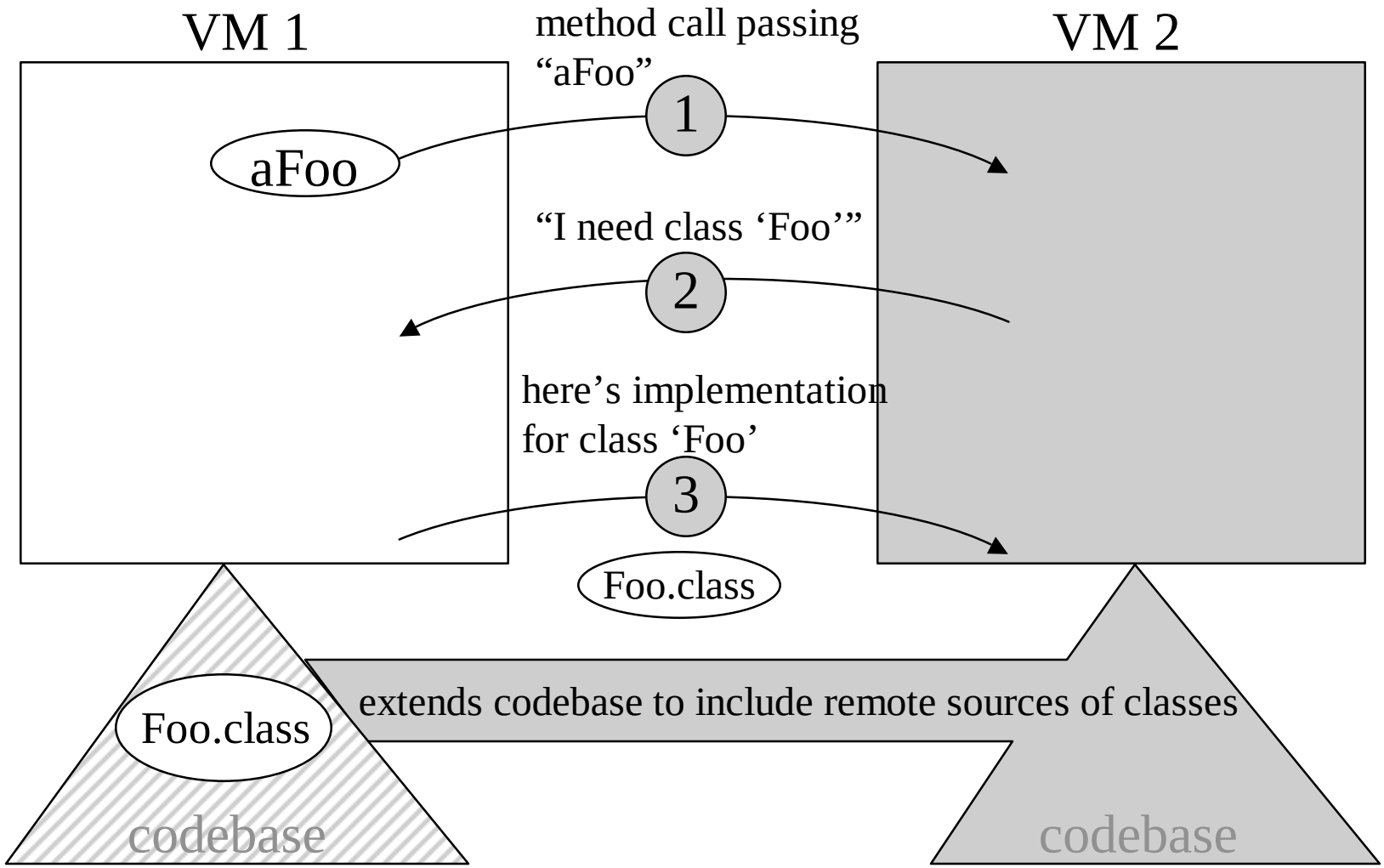
RMI Features/Limitations (Cont)

- Dynamic codebase
  - can send not only the state information (field values) but also the implementation (bytecode) of objects across the network
  - very powerful
    - send code to be executed by the server as part of performing work
      - example: a search object for searching a datastore
    - send an agent to a remote site to do work
    - send an object to a server to be stored persistently
    - etc.

RMI Features: Dynamic Codebase



RMI Features: Dynamic Codebase (cont)



RMI Features/Limitations (Cont)

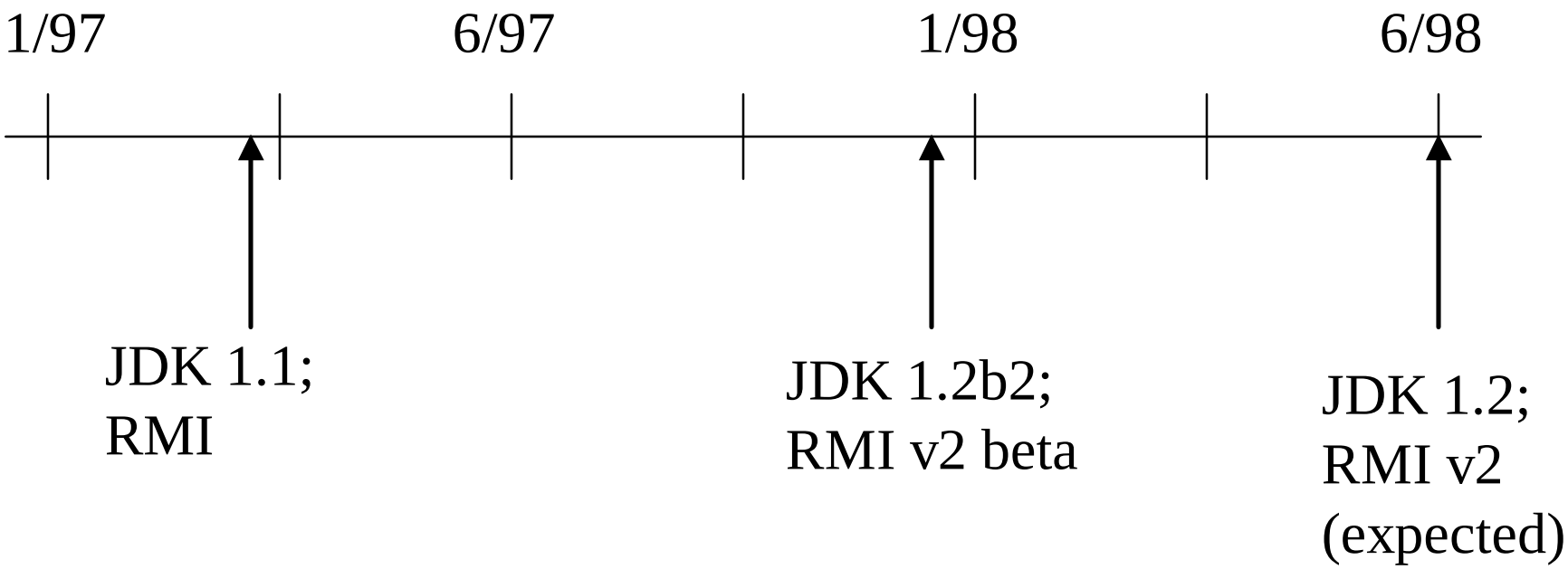
- Performance
  - for designs which pass data only (no objects), and for which version mismatch detection is not needed, RMI performs more slowly that CORBA
    - exact numbers depend upon many things
      - platform
      - Java JIT/no-JIT
      - ORB vendor
      - problem domain
      - etc.
  - for designs which pass objects, RMI is currently the only ticket in town (CORBA can’t pass objects)

RMI Features/Limitations (Cont)

- Performance (continued)
  - performance can be improved by custom serialization code
    - allows default implementation with targeted tuning for the 80/20 rule
  - BUT
    - using RMI features to solve problems produces a different design and can reduce or eliminate performance differences
    - passing objects, version mismatch, and distributed garbage collection are important issues when building distributed systems

RMI Features/Limitations (Cont)

- RMI is Maturing quickly



CORBA Features/Limitations

- Multi-language
  - C (requires layering object-oriented concepts on top)
  - C++
  - Smalltalk
  - Java - not yet standardized
  - COBOL
  - Ada

CORBA Features/Limitations (cont)

- Not free
  - costs can be substantial
    - developer licenses (-$800+ per developer)
    - runtime licenses (cost varies with vendor)
  - standardized extensions (“Object services” in OMG-speak) add more costs

CORBA Features/Limitations (cont)

- Harder to use
  - must
    - write IDL to provide interfaces and define data structures for passing information
    - transform IDL to Java stub & skeleton source code
    - compile stub/skeleton code to bytecode
  - different vendor's stub/skeleton bytecodes ('.class' files) are not compatible
  - details of using CORBA are different for each vendor: changing ORB vendors requires changing source code (eek!)

CORBA Features/Limitations (cont)

- Harder to use (continued)
  - more source code required to perform same tasks (vs. RMI)
  - no standard way to bootstrap communication
    - some vendor-specific options exist, but nothing standard across different vendors
  - no support for detecting software version mismatches
    - wing it or hand build it
    - most often, this isn't handled and problems arise at runtime, sometimes resulting in silent errors

CORBA features/limitations (cont)

- Add-on to the language
  - does not dovetail with standard Java features
    - breaks the object-oriented model when doing distributed communication
    - no built-in support for flattening objects onto/off of the wire
      - must use pseudo-objects or do the flattening/inflating by hand
    - no distributed garbage collection
      - e.g. don’t know when server objects & resources are no longer needed

CORBA features/limitations (cont)

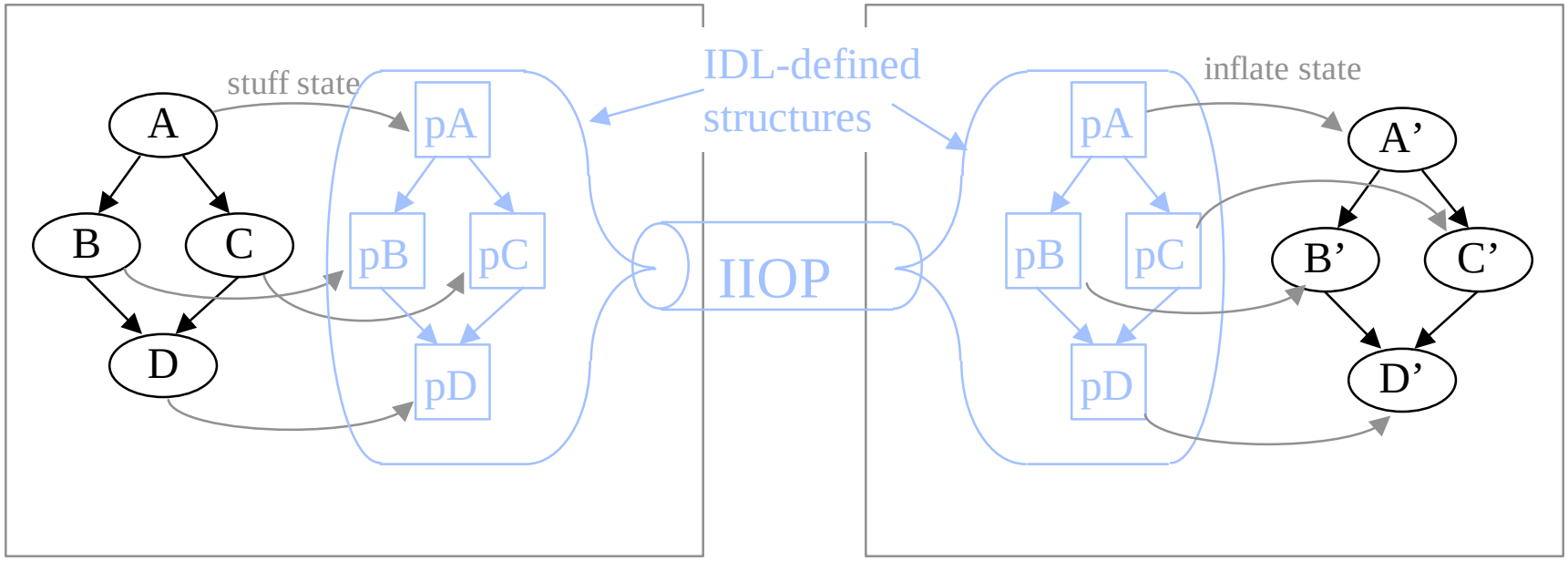
- Add-on to the language (continued)
  - BUT: Allows implementing in multiple languages
    - good for accessing legacy systems
      - especially if Java is not supported on the legacy platform
    - good for extending existing CORBA-based systems with Java
    - good for leveraging expertise in other languages
      - as long as there is a CORBA binding for that language

CORBA features/limitations (cont)

- Structured communication
  - must define pseudo-objects using IDL and translated to data structures
  - can’t send objects

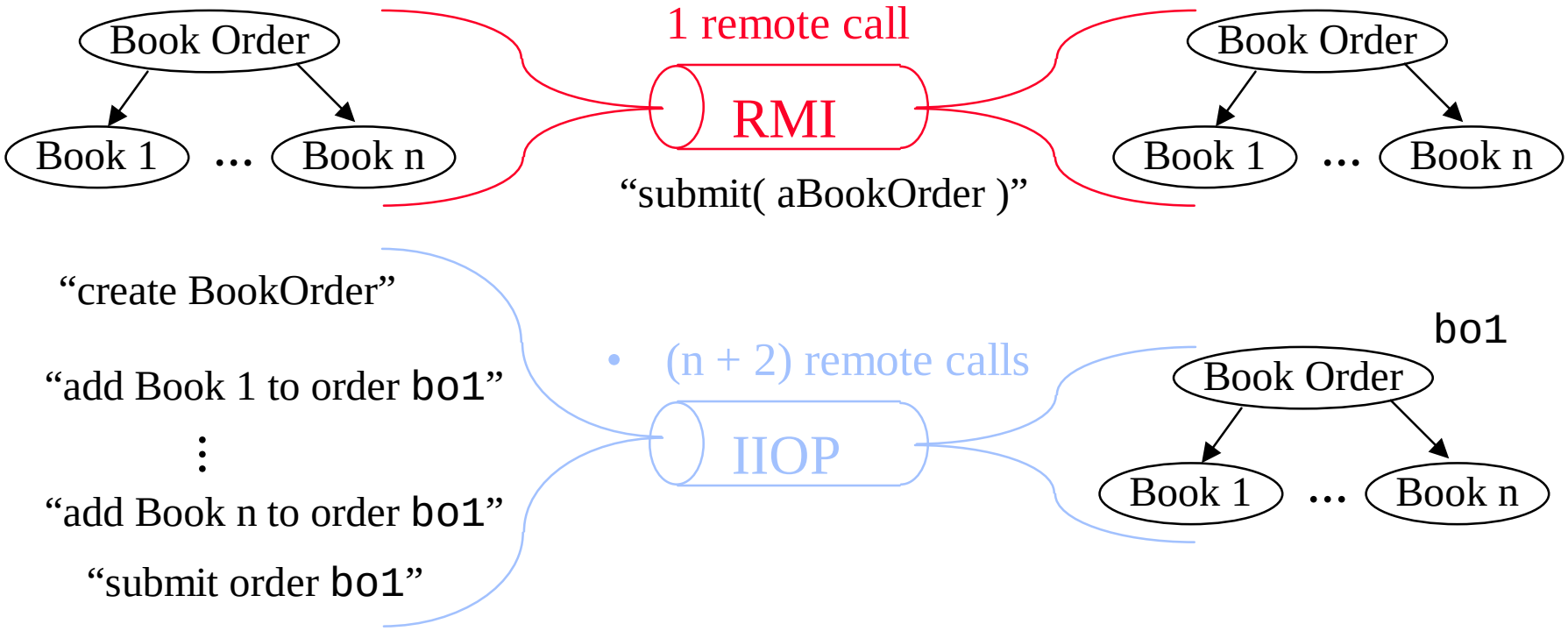
CORBA features/limitations (cont)

- Structured communication (continued)
  - there is substantial hand-coding to flatten/inflate a group of objects when sending their state information to a remote location



CORBA features/limitations (cont)

- Structured communication (continued)
  - lack of support for sending objects hampers design options and results in more remote calls across the wire



CORBA features/limitations (cont)

- Structured communication (continued)
  - assume an order for 3 books
  - we include marshalling/unmarshalling time
  - if 1 RMI call takes an average of 35 milliseconds (ms)
  - if 1 CORBA call takes
    - 21 ms => CORBA is • 105 ms & RMI is 35 ms
    - 10 ms => CORBA is • 50 ms & RMI is 35 ms
    - 6 ms => CORBA is • 35 ms & RMI is 35 ms

(numbers are illustrative only)

CORBA features/limitations (cont)

- Static codebase
  - can’t send implementations across the wire, therefore
    - can’t send previously unknown types of objects where they are needed to make computation more efficient
    - can’t dynamically extend the behavior of running VM
      - using DII only allows discovering new remote types and making remote method calls upon them

CORBA features/limitations (cont)

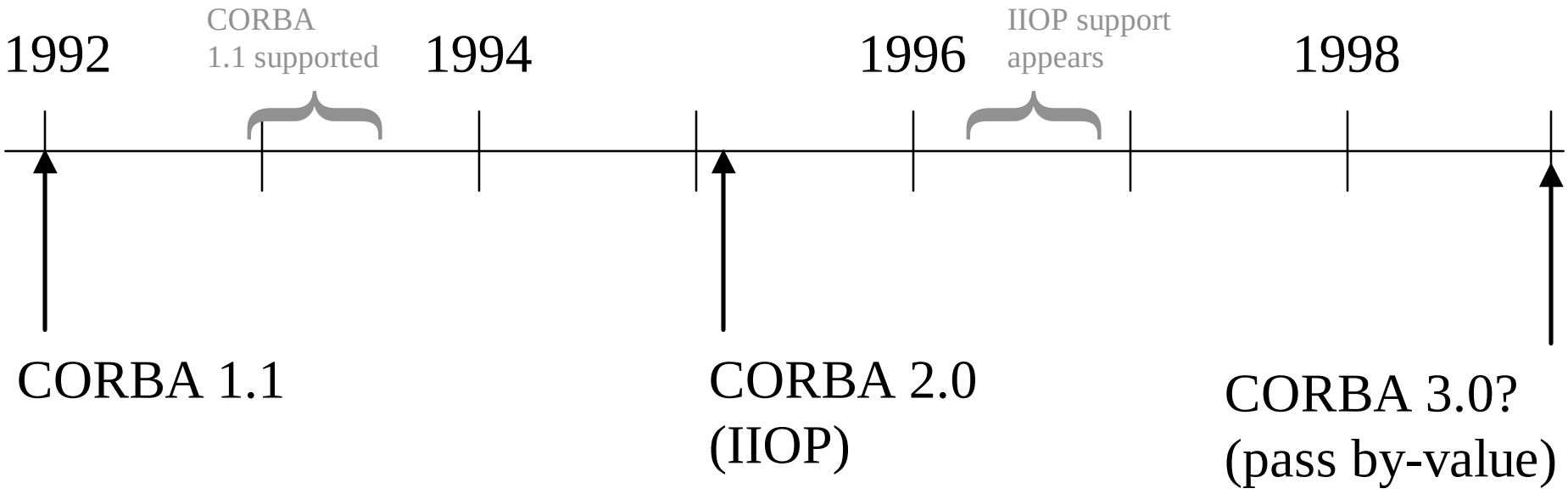
- Performance
  - performance of CORBA IIOP is not as good as custom protocols provided by ORB vendors
    - this talk only discusses IIOP because inter-vendor support is important
  - traditional client/server speed is better in ORB implementations than in RMI/JDK 1.1.x.
    - how much better varies, depending on how the application is structured, what OS you are running, etc.

CORBA features/limitations (cont)

- Performance (continued)
  - Compared to RMI, CORBA limits possible designs and removes at least some of the performance gain when including time for
    - marshalling/unmarshalling of objects into/out of IDL structures
    - more calls across the wire, which are substantially more costly than local calls
    - code to perform version mismatch detection
    - distributed garbage collection issues

CORBA features/limitations (cont)

- CORBA is Maturing slowly



The Future of RMI (JDK 1.2)

- Support for mixing different reference types
  - secure, multi-cast, etc.
- Performance improvements
  - RMI-specific improvements
  - JDK improvements: HotSpot + VM/library tuning
- Better control over (de)serialization of objects and ability to unexport a “Remote” object in your VM
- Activation of server objects
  - launching a VM to run a server object

The Future of RMI (2.0?)

- RMI over IIOP
  - there is an initiative by the OMG and Javasoft to extend CORBA's IIOP to support features needed to run RMI over IIOP
  - no known availability date
  - requires OMG's "pass by-value" (see next slide)

The Future of CORBA

- (3.0?) OMG is looking into supporting pass by-value
  - first response to first RFP was completed end of January
  - no known availability date
- (???) Other proposed features which have varying degrees of vendor support
  - see http://www.omg.org

The Future of CORBA (Cont)

- There are hard problems to be solved in order to support pass by-value in a heterogeneous environment
  - different languages at each end (e.g. C++ vs. COBOL)
    - Might only allow passing state, not implementation
      - introduces new failure modes
      - appears to be current OMG direction (except for Java)
    - If only passing state, might fail if appropriate implementation not available at receiving end
      - introduces new failure modes
  - what about security?
    - no security “sandbox” except in Java

The Future of CORBA (Cont)

- Supporting pass by-value (continued)
  - almost all languages don't support dynamic code inclusion in a running program (e.g. C, COBOL, C++)
    - introduces new failure modes
  - version mismatch problem must be solved for a multi-language environment. If not implemented
    - systems development becomes much more error prone
    - can result in "silent" errors (bad computation)
    - would introduce new failure modes

The Future - CORBA continued

- Therefore: don’t expect pass by-value to become standardized and supported on top of IIOP very soon

  (my guess: not before 1Q’99, more likely 3Q’99 or later)

Useful URL's

- RMI pages at Javasoft
  - www.javasoft.com/products/jdk/rmi
- Object Management Group (OMG)
  - www.omg.org
- CORBA FAQ
  - www.cerfnet.com/~mpcline/corba-faq

Useful Newsgroups/email lists

- rmi-users@java.sun.com
- comp.lang.java.*
- comp.object
- comp.object.corba
- general CORBA mailing list?

Questions/Comments

- Now
- Later: gilstrap@inlink.com