Remote method invocation

- RMI, and it's predecessor RPC (remote procedure call) are abstractions concerned with a programming model for distributed systems.
- RPC was first: basic procedure calls.
- RMI is the object oriented equivalent. Methods are procedures associated with object instances.

- Most modern systems are RMI.

- RMI is a middleware layer, built on top of request/reply protocols, XDRs, and point to point communications.

Transparency

- In week one, we discussed the notion of transparencies.
- RMI aims to provide what is called *location transparency*.
  - By this, we intend that the caller not be aware of whether the procedure runs in the local process or within a remote one.
- RMI sometimes also attempts to provide transparency with respect to the languages on either end of the call.

Interfaces and language heterogeneity

- In RMI, we are concerned only with the interface of the method.
- The interface is what a method provides to a caller.
- For example:
  - int sum(Array<int>, i);

- How do we read this?
  - Sum is a function that takes an array of integers as input, and returns an integer containing their sum.
- This is an abstract interface representation. The RMI middleware is then responsible for binding it to the appropriate languages.

Interfaces and binding

- int sum(Array<int>, i);
- C:
  - `int sum(int *i, int length);`
- Java:
  - `int sum(int i[]);`
- Fortran:
  - `function sum`
    - `INTEGER :: sum`
    - `INTEGER, DIMENSION(:) :: I`

Interfaces and binding

- How does this help?
- On the caller side, the RMI system provides a thin layer that exposes the correct interface signature for that language.
- Behind this layer, the arguments are packaged into an XDR format and sent to the server.
- The server has a similar layer, that unpacks the arguments and invokes the actual method in the appropriate way for it’s implementation language.

Interface Definition Languages

- Interface Definition Languages (IDLs) provide this language-neutral method for describing interfaces to be provided via RMI.
  - CORBA IDL
  - SunRPC IDL
  - Babel SIDL (Scientific IDL)

- Java does not require an IDL for RMI. Java RMI only talks to Java, so no need. Reflection provides enough information about the class method signatures to make the calls on both sides.

IDLs

- IDLs have limits. For example, sometimes they prohibit passing pointers and references back and forth because you cannot directly address memory in a remote process without going through a well defined interface.
- IDLs also limit types to a subset that is reasonable to represent in all languages.
  - E.g.: Some languages have a native COMPLEX type. Most don't.