Ibis tutorial & hands-on session

Jason Maassen
jason@cs.vu.nl

Thursday 13 October 2005
Sophia Antipolis

Overview

- Philosophy / design / implementation
  - Why do we need Ibis ?
  - Ibis design
  - Performance
  - Cool features

Overview

- Programming models
  - IPL (bare bones Ibis)
  - RMI (remote invocation)
  - GMI (group communication)
  - Satin (divide and conquer)
  - MPJ (MPI to Java binding)
- Hands-on session
  - How to roll your own Ibis applications

We are interested in…

- Parallel applications on "the Grid"
  - Single site runs
    - Grid == big collection of clusters
      - Only communicate within cluster
      - Use fast local network (Myrinet/Infiniband/...)
  - Multi site runs
    - Grid == big processor pool
      - Communicate between clusters
      - Use regular network & internet

So why Ibis?

- Ideally, grid computing should be “fire and forget”
  - Develop application locally
  - Submit to some grid scheduler which
    - Finds some suitable site(s)
    - Transfers your application and data to the sites, and runs it.
    - Returns the result

Problems

- Lots of problems
  - Resource selection
  - Data transfer
  - Security and authentication
  - ...
  - Heterogeneity
- Globus, Gridlab (GAT), etc.

Problems

- Grids are heterogeneous:
  - Intel / PowerPC / Mips / Arm / ...
  - Windows / Linux / Unix / OSX / ...
  - Different OS/library/tool versions
- Compiled (C/MPI) apps. huge pain:
  - Need executable for every combination of CPU/network/OS/libraries etc.
  - Makes 'fire & forget' runs really hard...

ibis
efficient Java-based grid computing

vrije Universiteit

Solution (partly)

- So, we use Java instead C or Fortran
  - No recompilation required
  - Runs (almost) anywhere
    - Doesn't work on supercomputers (Hitachi SR8000, IBM BlueGene, etc.)
    - Most sites have clusters anyway
  - Acceptable performance
- But … only part of the solution!

How about 'portable' communication?

- Class libraries are portable, but ...
  - Sockets are too low-level
  - RMI model/performance is limited

- Most parallel libraries not portable ...
  - MpiJava requires native code
    - recompilation
    - no malleable runs

Ibis

- Solution: Ibis !
  - A “run-anywhere” communication library
  - Just send it along with your application!
- Flexible communication models
  - More than just unicast communication
    - More about this later
  - Malleability & Fault-Tolerance
    - Change number of machines during the run

Ibis

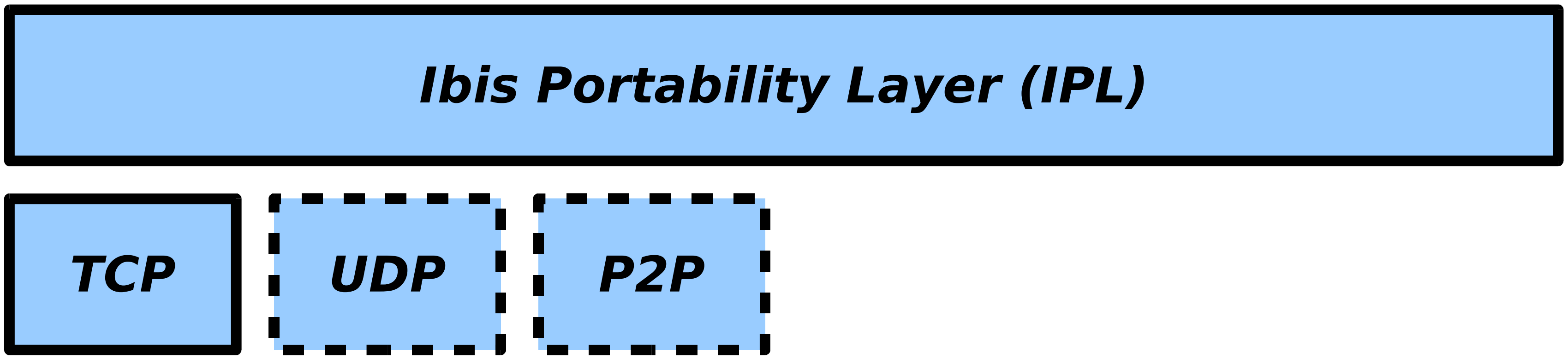
- Portability vs. performance
  - On a single site run you often want to use the fast local network
  - Ibis allows specialized implementation
    - Designed for Myrinet, Infiniband, etc.
    - Usually use native code
    - Installed in advance
      - not portable
      - cannot be shipped with application

Ibis

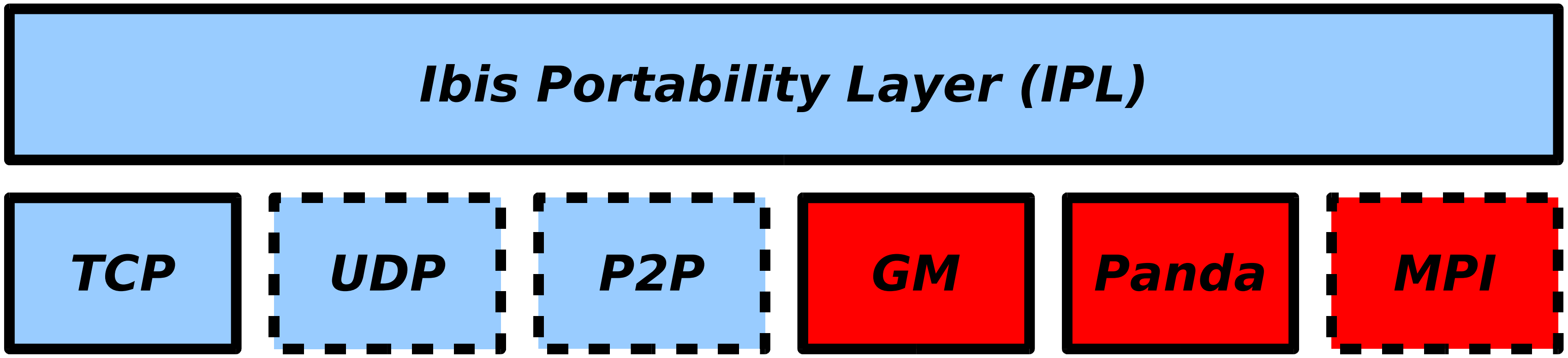
- As a result, there may be multiple Ibis' available on a site
  - Automatically choose 'best' at startup
  - Based on requirements specified by
    - Application & user (using properties)
- Not every impl. needs all features
  - Pick one at startup that suits your needs....

Ibis Design

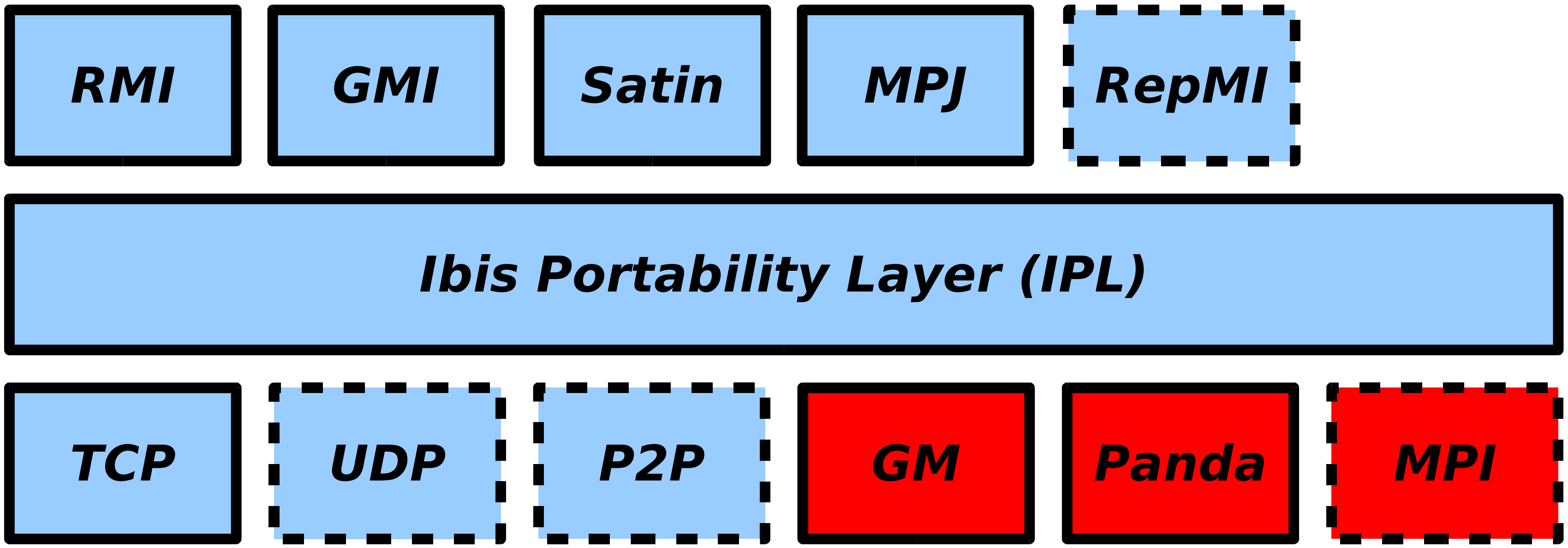
Ibis Design



Ibis Design



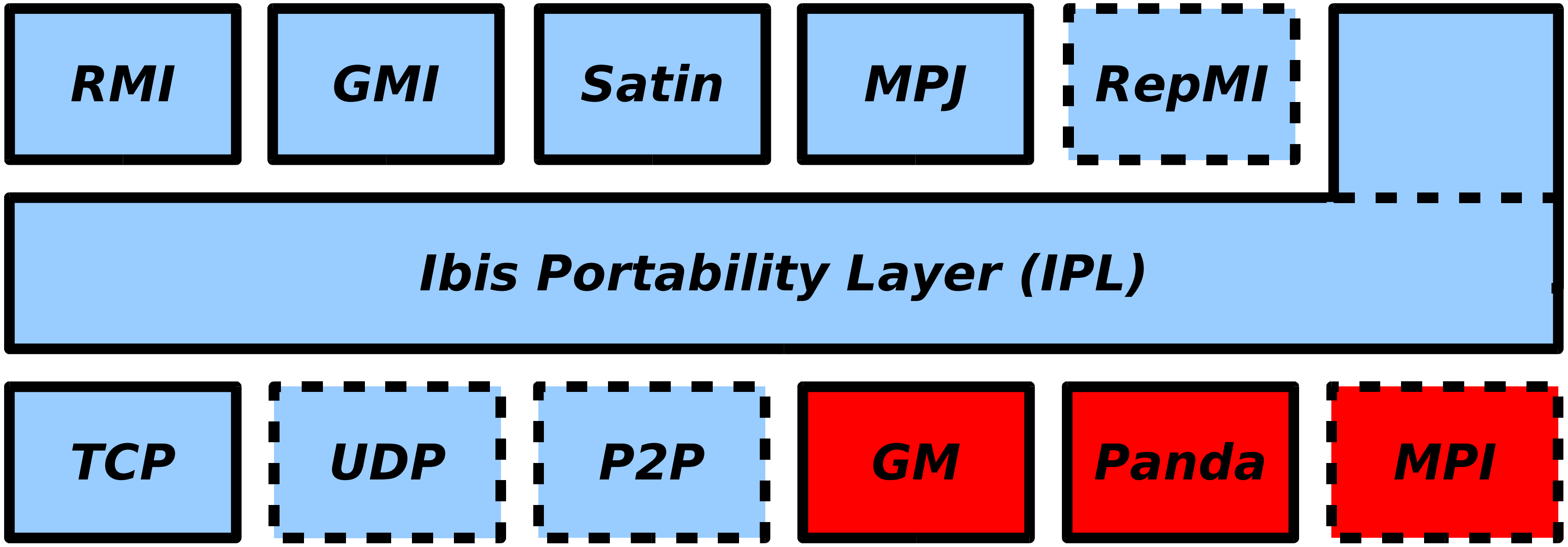
Ibis Design



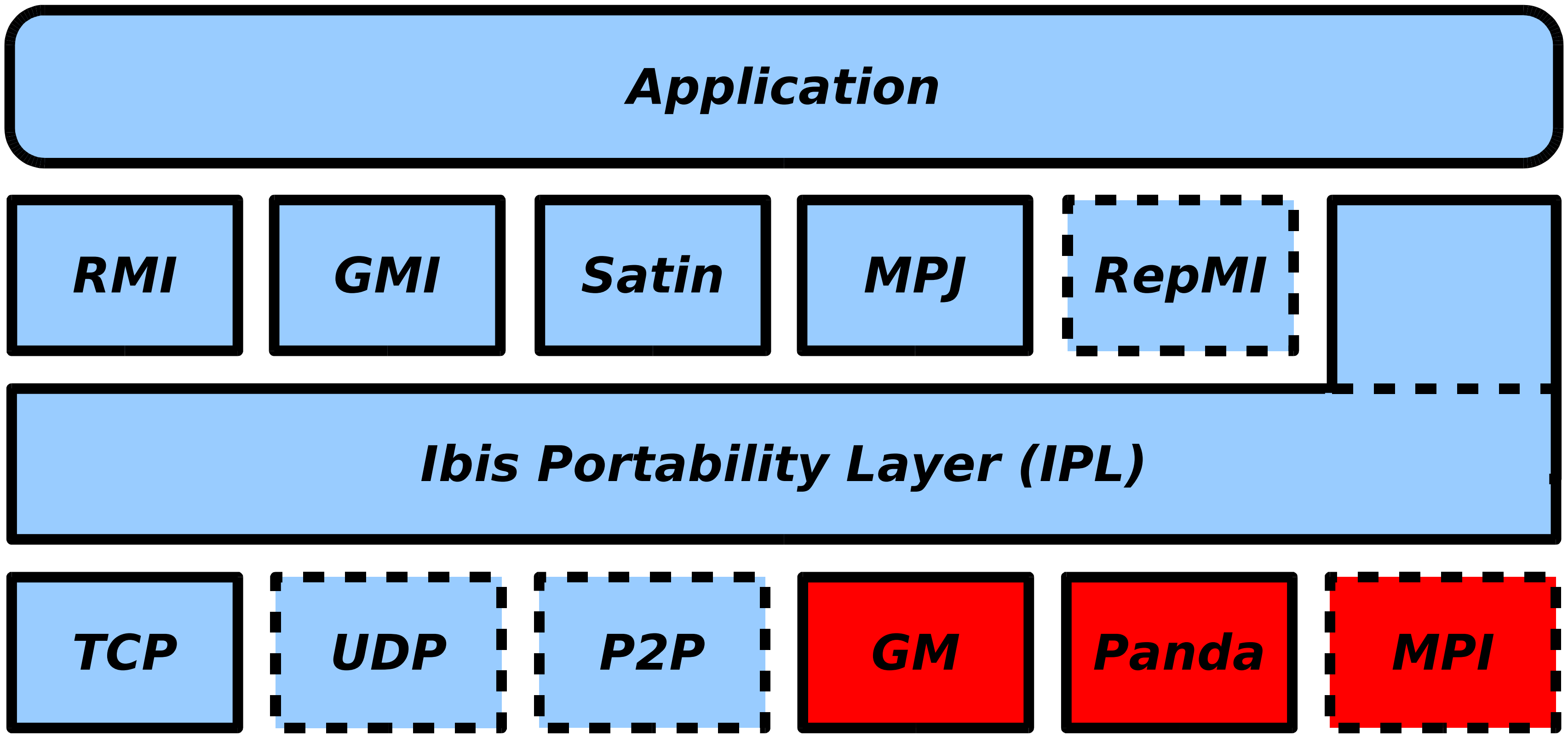
ibis
efficient Java-based
grid computing

vrije Universiteit

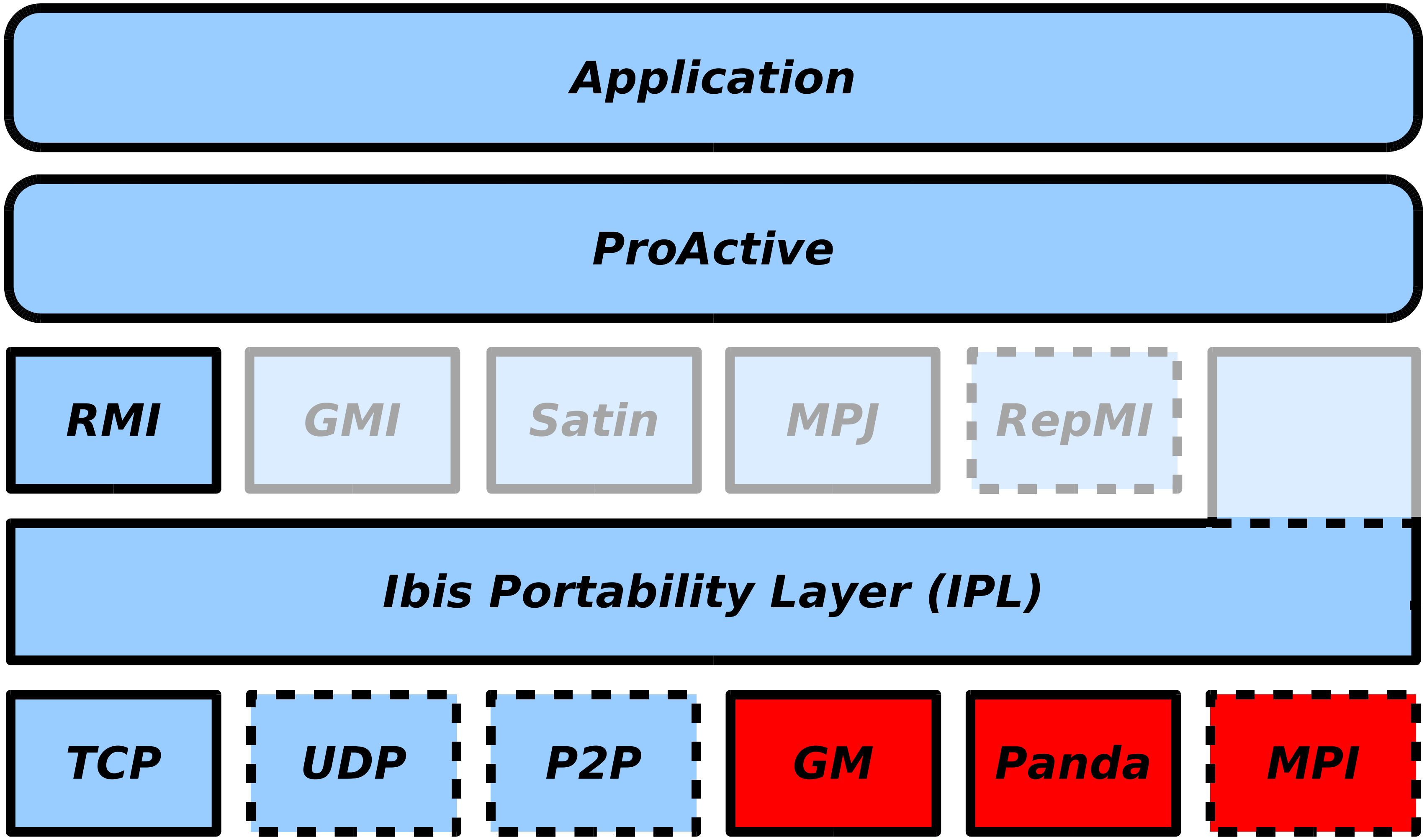
Ibis Design


RMI
GMI
Satin
MPJ
RepMI
Ibis Portability Layer (IPL)
TCP
UDP
P2P
GM
Panda
MPI

Ibis Design


Application
RMI
GMI
Satin
MPJ
RepMI
Ibis Portability Layer (IPL)
TCP
UDP
P2P
GM
Panda
MPI

Ibis Design



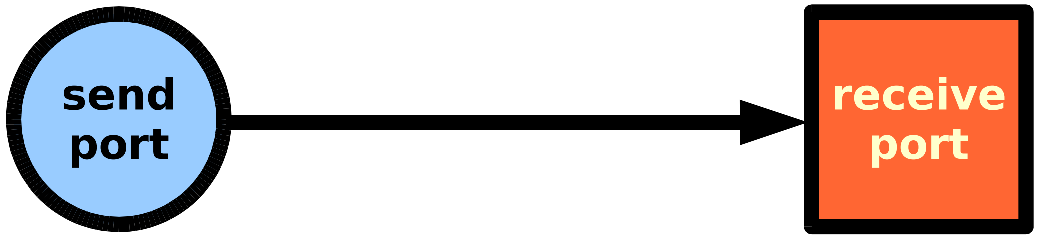
Application
ProActive
RMI
GMI
Satin
MPJ
RepMI
Ibis Portability Layer (IPL)
TCP
UDP
P2P
GM
Panda
MPI

Ibis Portability Layer

- Basic Ibis interface
  - Reasonably simple (5 classes & 12 interf.)
- Contains methods for
  - Loading an Ibis
  - Malleability (adding & removing machines)
  - Connection handling
  - Communication primitives (low-level)

IPL Communication

- 'Low-level' communication model
- Unidirectional pipes
- Two end points
- Connection oriented



ibis efficient Java-based grid computing

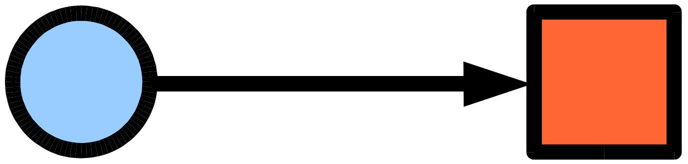
vrije Universiteit

Send & receive ports

- Can be connected in arbitrary ways

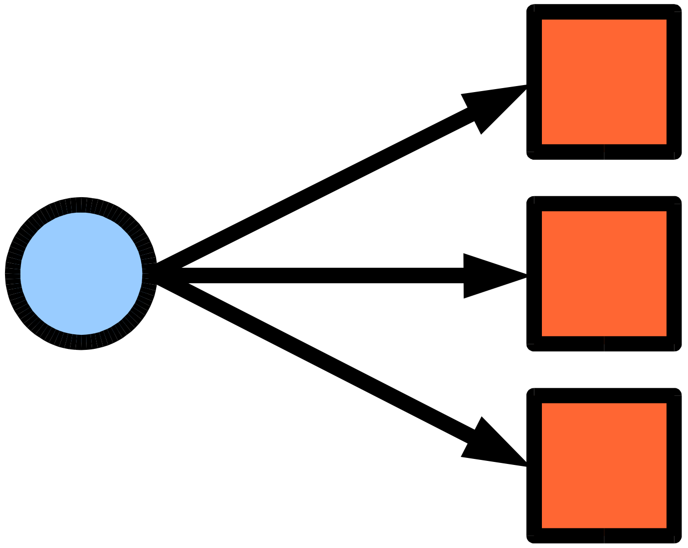
Send & receive ports

- Can be connected in arbitrary ways
- One to one (unicast) ...



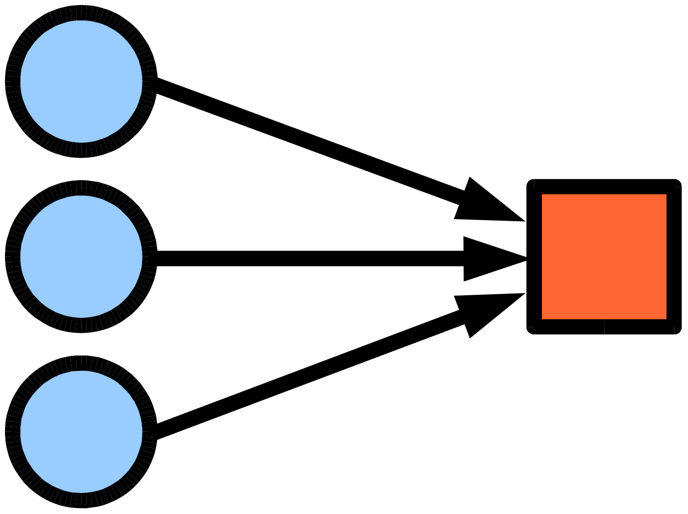
Send & receive ports

- Can be connected in arbitrary ways
- ... one to many (multicast) ...



Send & receive ports

- Can be connected in arbitrary ways
- ... many to one ...

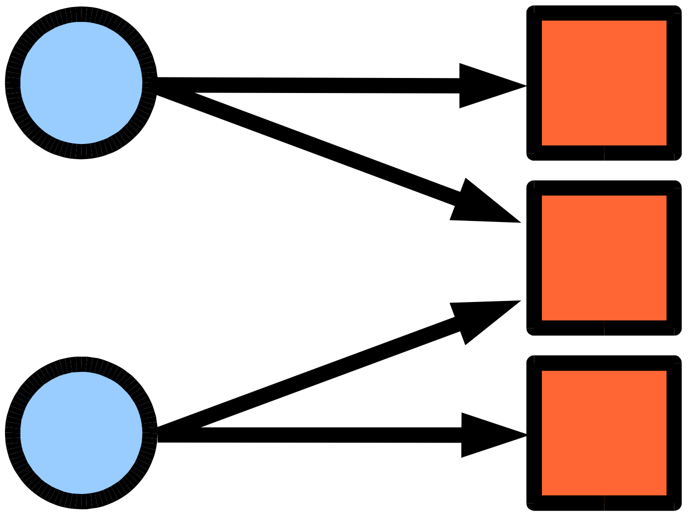


efficient Java-based grid computing

vrije Universiteit

Send & receive ports

- Can be connected in arbitrary ways
- ... or some combination!

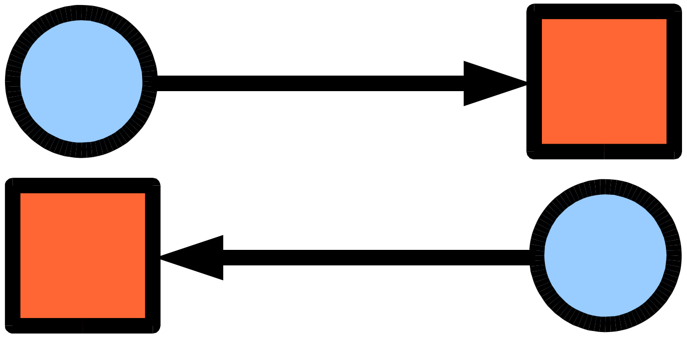


Send & receive ports

- Advantages:
  - Very simple & abstract model
  - Easy to implement using TCP/UDP/MPI/etc.
  - Allows multicast, many-to-one, etc.
    - Useful for parallel programs
  - <u>Allows efficient implementation</u>
    - Can be implemented using efficient low-level primitives (i.e., mpi-broadcast)
    - Other models do prevent this (e.g., RMI)

Send & receive ports

- Disadvantage:
  - Simplicity may cause some overhead...
  - Example: need two pairs for RPC / RMI

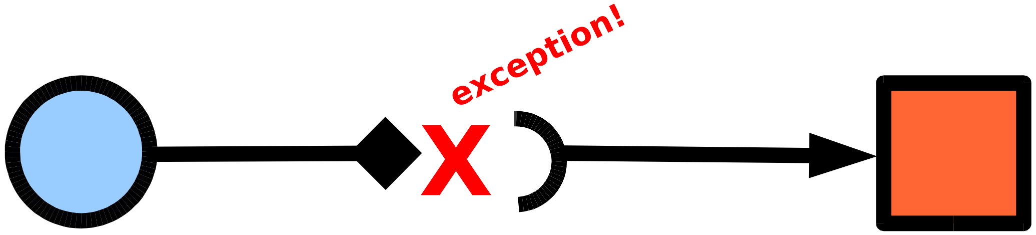


Port Types

- All ports have a type consisting of:
  - Unique name
  - Set of properties, e.g.:
    - Supports unicast and multicast
    - Is Reliable
    - Is fifo ordered
    - Supports object serialization
    - ...

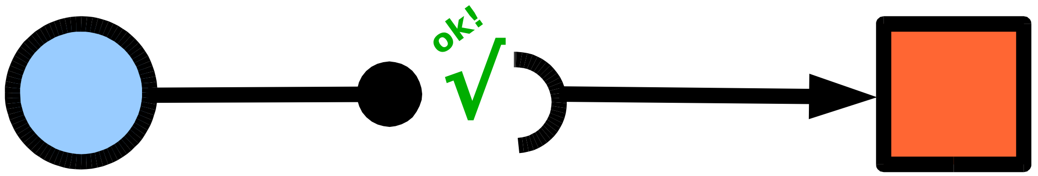
Port Types

- Defined at runtime
  - Specify name and set of properties
- Types must match when connecting!



Port Types

- Defined at runtime
  - Specify name and set of properties
- Types must match when connecting!



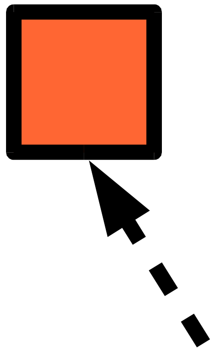
Port Types

- Forces programmer to specify how each communication channel is used
  - Prevents bugs
    - Exception when contract is breached
  - Allows efficient impl. to be selected
    - Unicast only ?
    - Bytes only ?
    - Saves a lot of overhead!

Connection setup

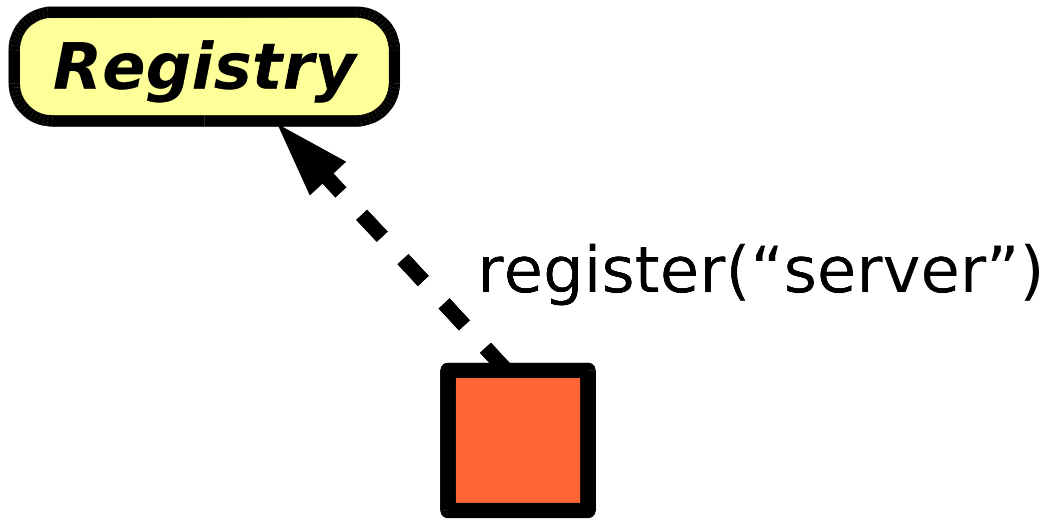
- Need the ReceivePortIdentifier
  - Uniquely identifies a receiveport
  - Created when ReceivePort is created
  - May also have unique name (String)
    - Human-readable (usually)
    - Use for registry lookup

Connection setup

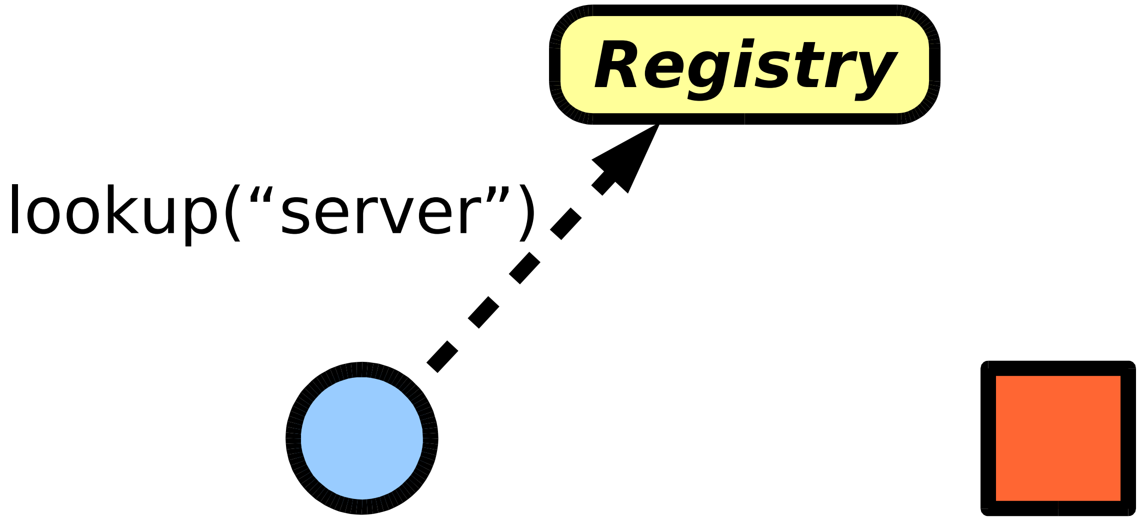


Create ReceivePort
“server”

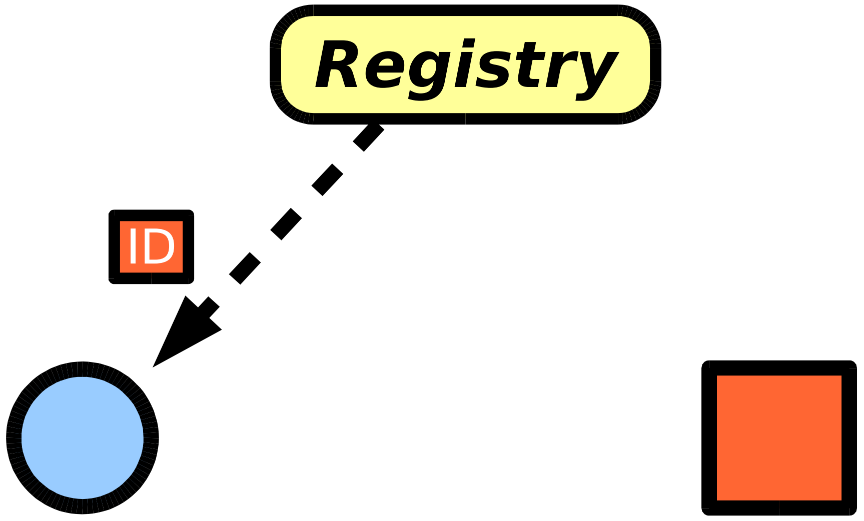
Connection setup



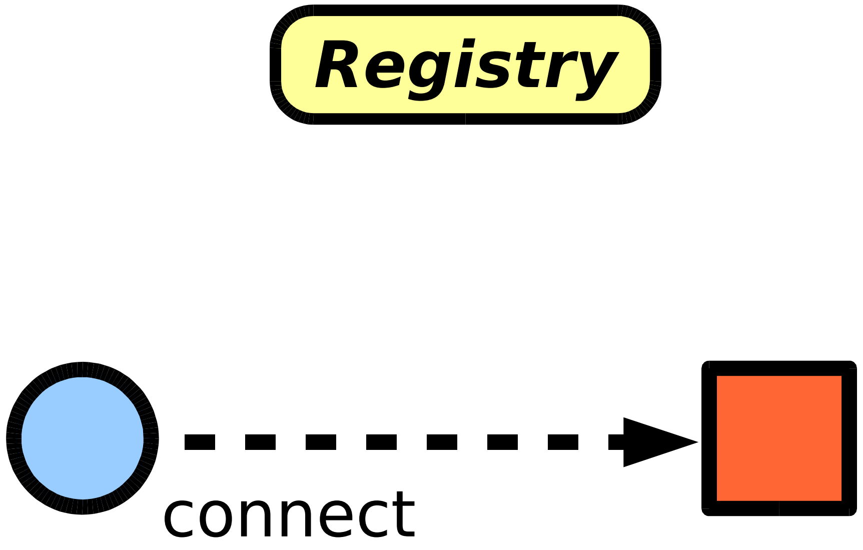
Connection setup



Connection setup



Connection setup



Connection setup

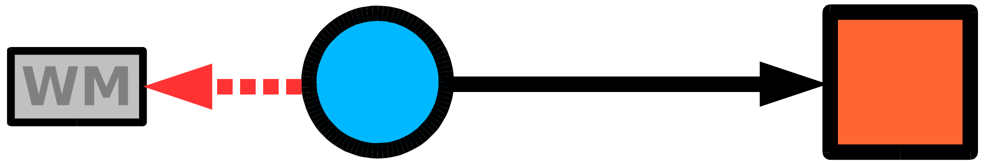
- Advantage of ReceivePortIdentifiers
  - Hides implementation details
  - Independent of
    - IP-addresses
    - Host names
    - Port numbers
    - MPI-ranks
    - etc...
  - Abstract way of addressing

Messages

- Ports communicate using 'messages'
- Contain read or write methods for
  - Primitive types (byte, int, ...)
  - Object
  - Arrays slices (partial write / read in place)
- Unlimited message size

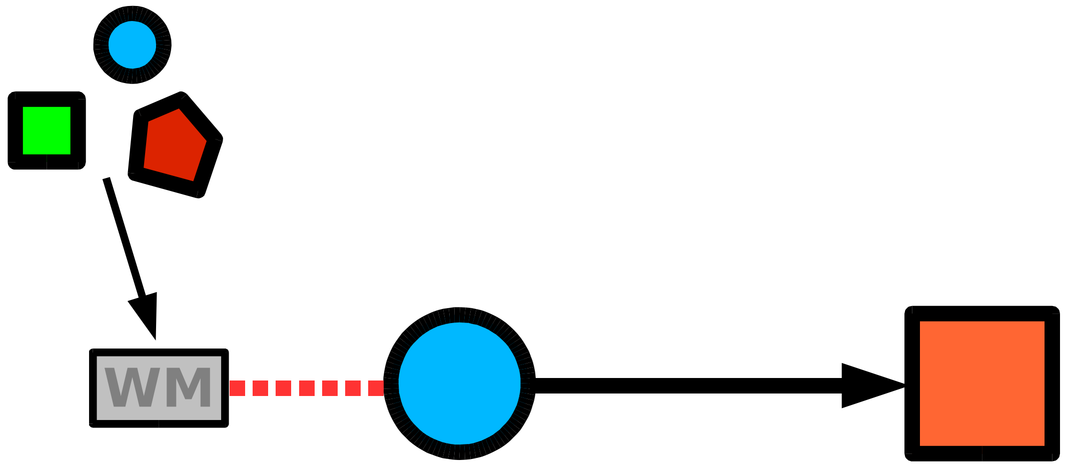
Messages

- Get WriteMessage from SendPort



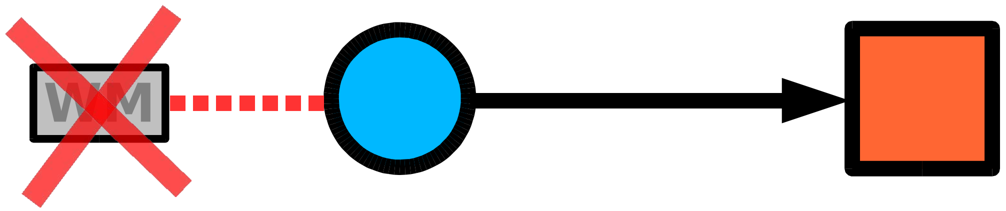
Messages

- Write data into WriteMessage



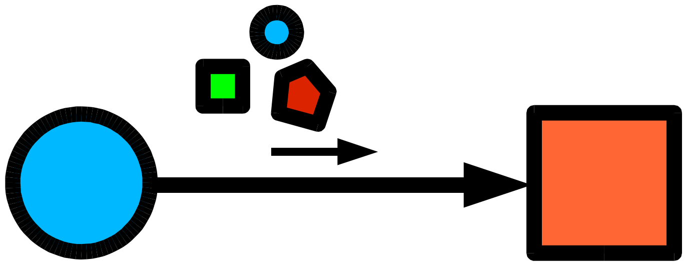
Messages

- Finish the WriteMessage



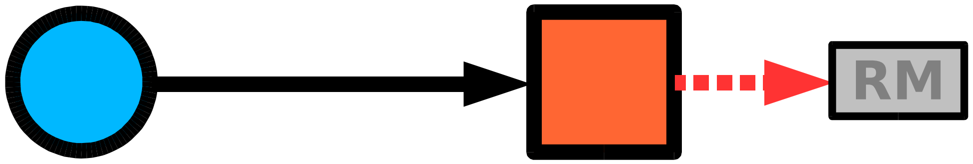
Messages

- Data is send to ReceivePort



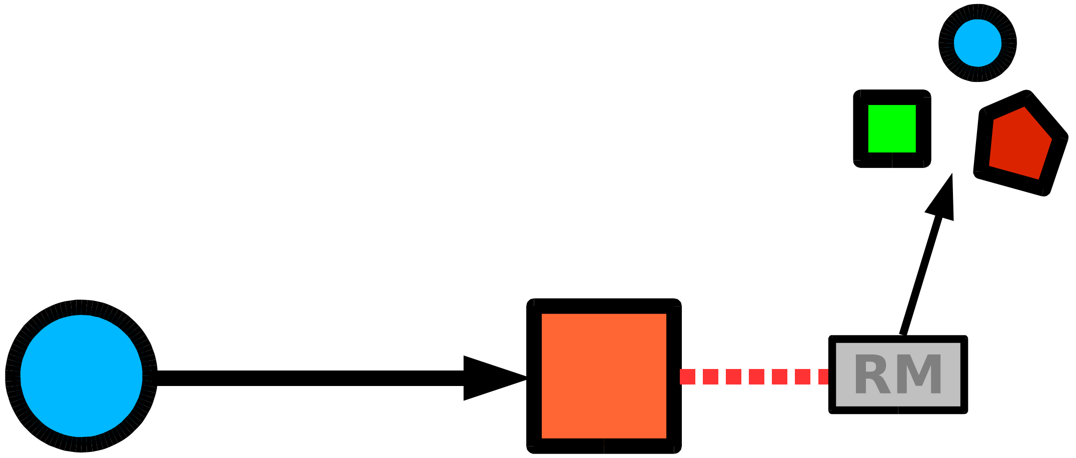
Messages

- ReceivePort produces ReadMessage
  - Explicit receive or callback (upcall)



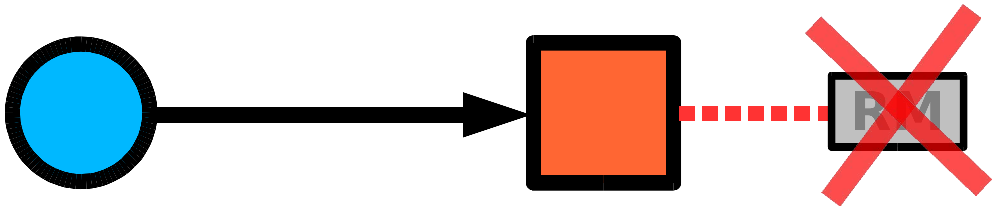
Messages

- Read data from ReadMessage



Messages

- Finish the ReadMessage

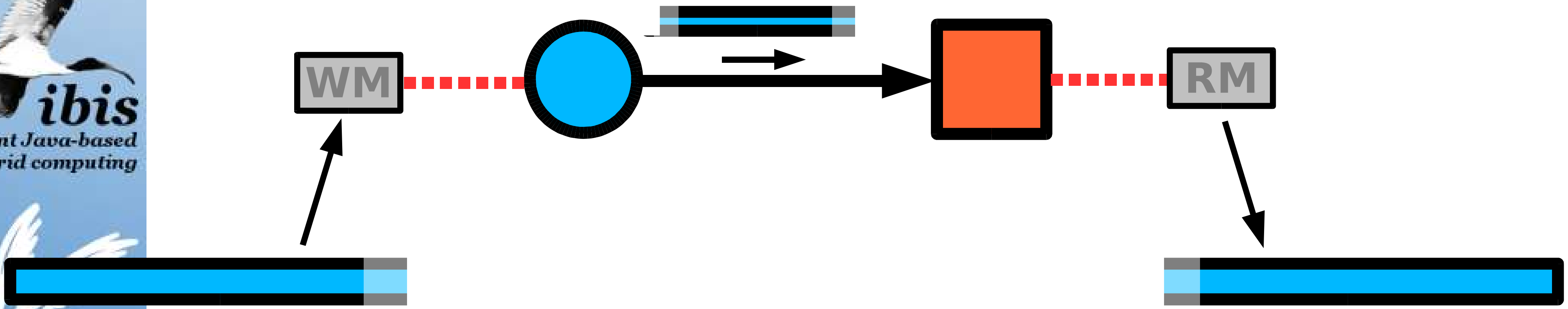


Messages

- Done!

Messages or streams ?

- Message size is unlimited
  - Data may be forwarded at any time
  - Both S. & R. messages alive at same time
  - There's streaming!



Restrictions

- Must write and read data in same order
- A port can have only one message 'alive' at a time
- Not thread safe (but ports are)
- If there is no receiver, the sender may block (there may be flow-control)

Serialization

- Ibis supports 4 types of serialization
  - Bytes (i.e., no serialization at all)
  - Data (only primitive types/arrays)
  - Sun (standard Sun serialization)
  - Ibis (efficient Ibis serialization)
- To select one of the last two use 'Object'

Ibis Serialization

- Based on bytecode-rewriting
  - Adds serialization and deserialization code to serializable types
  - Prevents reflection overhead during (de-)serialization
  - Has fallback mechanism for non-rewritten classes
- Future work: runtime rewriting

Short Recap

- First create PortType
- PortType creates Send & ReceivePort
  - Type is checked when connecting
- Use ReceivePortID's to connect
  - Abstact addressing
- Use Messages to communicate
  - Allows streaming
  - 4 types of serialization

Creating an Ibis

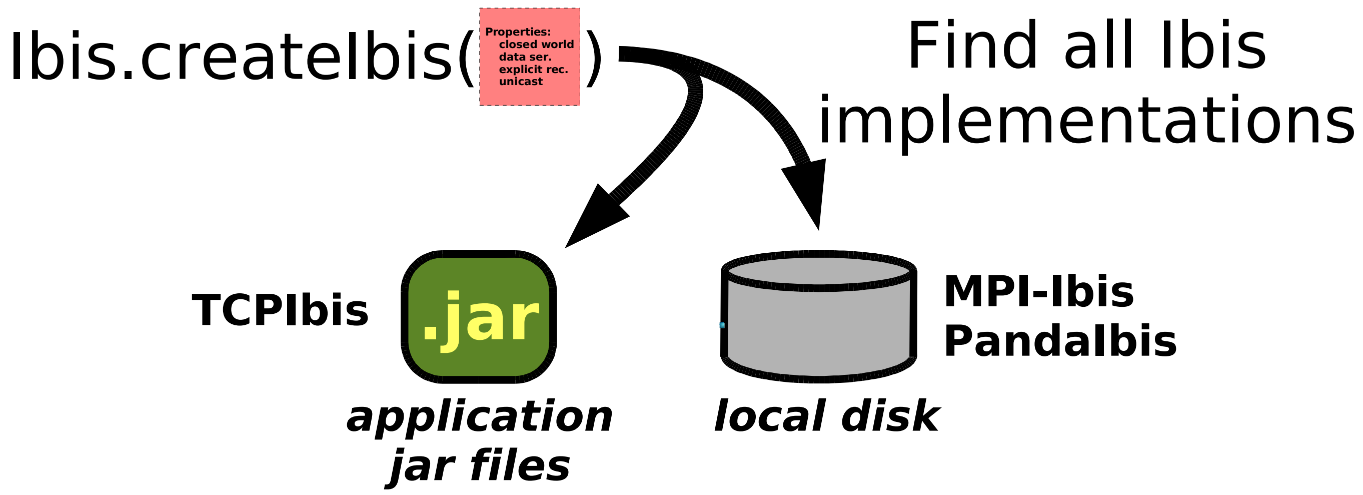
- First step in application
  - IPL is only abstract classes & interfaces
- Ibis selects implementation for you
  - Multiple may be available
  - Selected on the basis of properties
    - Specify the needs of the application

Selecting an Ibis

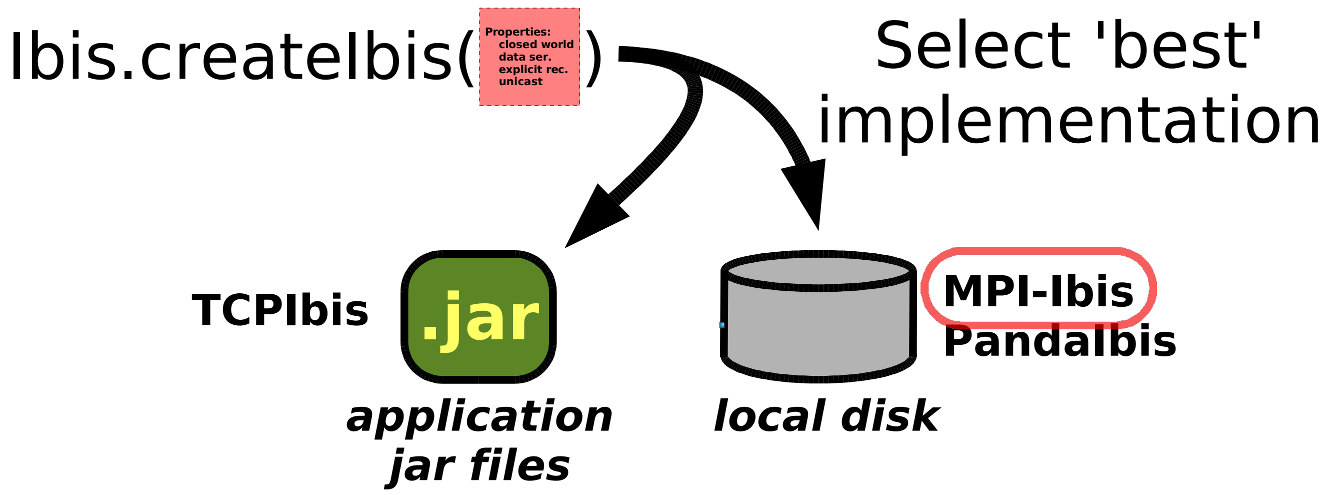
Properties:
closed world
data ser.
explicit rec.
unicast

Ibis.createIbis(...)

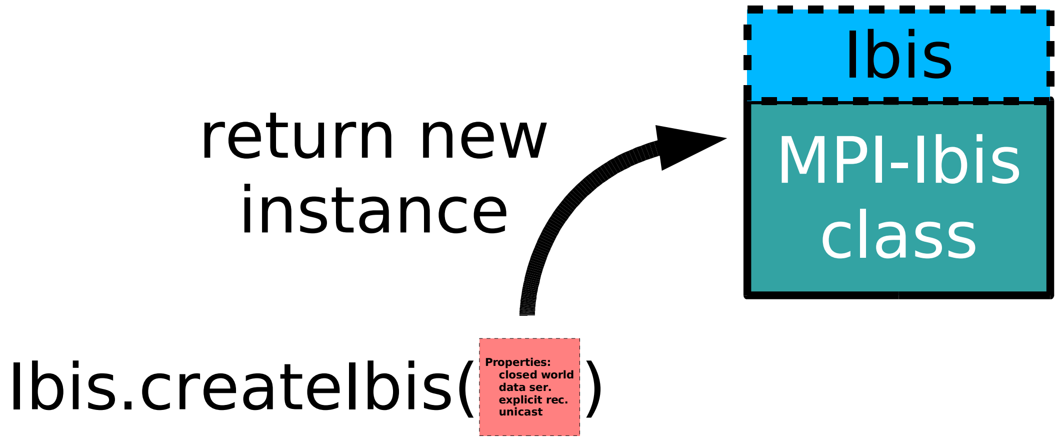
Selecting an Ibis



Selecting an Ibis



Selecting an Ibis



Properties

- Usual Java properties
  - Set of key-value pairs
    - “serialization”, “object”
    - “communication”, “OneToOne”
    - “worldmodel”, “open”
- Very flexible
  - good <u>and</u> bad

Properties

- Good
  - Introduce features without IPL changes
    - Just add more properties
  - Allows impl. specific properties
- Bad
  - No compile time checks (only runtime)
    - Just strings
    - Sensitive to typos

Example

Property	***Values***	***Description***
Worldmodel	Open	Support malleability
	Closed	Fixed set of machines
Serialization	Byte	
	Data	
	Object	
Communication	OneToOne	
	OneToMany	
	ManyToOne	
	ExplicitReceive	Support explicit rec.
	AutoMessageUpcalls	Automatic callback
	PollingMessageUpcalls	Callback triggered by polling
	ConnectionUpcalls	Callback when machin join or leave

Example

Property	*TCP*	**Ibis** *Panda*	*MPI*
Worldmodel	Open	Closed	Closed
Serialization	B/D/O	B/D/O	B/D/O
Communication	OO/OM/MO	OO/OM/MO	OO/OM/MO
Explicit receive	Yes	Yes	Yes
AutoMessageUpcalls	Yes	Yes	No
PollingMessageUpcalls	Yes	Yes	No
ConnectionUpcalls	Yes	Yes	Yes
ConnectionDowncalls	Yes	No	No

Malleability

- Ibis can notify the application if a machine joins or leaves
  - Callbacks/upcalls to a 'ResizeHandler'
  - Calls are delivered in the same order on all machines
- Each 'Ibis' has a unique IbisIdentifier
  - Abstract identification of the machine
  - Impl. using IP-addresses / MPI-ranks / etc.

Elections

- Ibis offers an 'election' mechanism
  - Allows a group of machines to determine who's in charge
- Each election
  - Has a name (String)
  - Produces an IbisIdentifier identifying the winner
  - Is not democratic

Other cool features

- TCP Ibis supports
  - Parallel streams
    - For high-latency & high-bandwidth links
  - NAT / firewall traversal
    - TCP splicing
    - Routing messages through external point

Code example

```
import ibis.ipl.*;

public class Example {

    public static void main(String args[]) throws Exception {

              // Step 1: create ibis
              StaticProperties p1 = new StaticProperties();
              p1.add("communication", "OneToOne, Reliable, ExplicitReceipt");
              p1.add("serialization", "object");

              Ibis ibis = Ibis.createIbis(p1, null);

              // Step 2: create porttype
              PortType type = ibis.createPortType("Test Type", p1);

              // Step 3: elect server
              Registry reg = ibis.registry();
              IbisIdentifier server = reg.elect("Server");
              boolean amServer = server.equals(ibis.identifier());

              if (amServer) {
                        // Step 4, create port
                        ReceivePort rp = type.createReceivePort("server");
                        rp.enableConnections();

                        // Step 5, receive message and read data
                        ReadMessage rm = rp.receive();
                        String tmp = (String) rm.readObject();
                        rm.finish();

                        System.out.println("Client says: " + tmp);

                        // Step 6, close port
                        rp.close();
              } else {

                        // Step 4, create port, find receivePort and connect
                        SendPort sp = type.createSendPort();
                        sp.connect(reg.lookupReceivePort("server"));

                        // Step 5, get message and write data
                        WriteMessage wm = sp.newMessage();
                        wm.writeObject("Hello World");
                        wm.finish();

                        // Step 6, close port
                        sp.close();
              }

              ibis.end();
    }
}
```

Code example

```
import ibis.ipl.*;

public class Example {

    public static void main(String args[]) throws Exception {

              // Step 1: create ibis
              StaticProperties p1 = new StaticProperties();
              p1.add("communication", "OneToOne, Reliable, ExplicitReceipt");
              p1.add("serialization", "object");

              Ibis ibis = Ibis.createIbis(p1, null);

              // Step 2: create porttype
              PortType type = ibis.createPortType("Test Type", p1);

              // Step 3: elect server
              Registry reg = ibis.registry();
              IbisIdentifier server = reg.elect("Server");
              boolean amServer = server.equals(ibis.identifier());

              if (amServer) {
```

```
// Step 1: create ibis
StaticProperties p1 = new StaticProperties();
p1.add("communication", "OneToOne, Reliable, ExplicitReceipt");
p1.add("serialization", "object");

Ibis ibis = Ibis.createIbis(p1, null);
```

```
                        sp.connect(reg.lookupReceivePort("server"));

                        // Step 5, get message and write data
                        WriteMessage wm = sp.newMessage();
                        wm.writeObject("Hello World");
                        wm.finish();

                        // Step 6, close port
                        sp.close();
              }

              ibis.end();
    }
}
```

Code example

```
import ibis.ipl.*;

public class Example {

    public static void main(String args[]) throws Exception {

              // Step 1: create ibis
              StaticProperties p1 = new StaticProperties();
              p1.add("communication", "OneToOne, Reliable, ExplicitReceipt");
              p1.add("serialization", "object");

              Ibis ibis = Ibis.createIbis(p1, null);

              // Step 2: create porttype
              PortType type = ibis.createPortType("Test Type", p1);

              // Step 3: elect server
              Registry reg = ibis.registry();
              IbisIdentifier server = reg.elect("Server");
              boolean amServer = server.equals(ibis.identifier());

              if (amServer) {
                        // Step 4, create port
                        ReceivePort rp = type.createReceivePort("server");
```

```
// Step 2: create porttype
PortType type = ibis.createPortType("Test Type", p1);
```

```
                        System.out.println("Client says: " + tmp);

                        // Step 6, close port
                        rp.close();
              } else {

                        // Step 4, create port, find receivePort and connect
                        SendPort sp = type.createSendPort();
                        sp.connect(reg.lookupReceivePort("server"));

                        // Step 5, get message and write data
                        WriteMessage wm = sp.newMessage();
                        wm.writeObject("Hello World");
                        wm.finish();

                        // Step 6, close port
                        sp.close();
              }

              ibis.end();
    }
}
```

Code example

```
import ibis.ipl.*;

public class Example {

    public static void main(String args[]) throws Exception {

              // Step 1: create ibis
              StaticProperties p1 = new StaticProperties();
              p1.add("communication", "OneToOne, Reliable, ExplicitReceipt");
              p1.add("serialization", "object");

              Ibis ibis = Ibis.createIbis(p1, null);

              // Step 2: create porttype
              PortType type = ibis.createPortType("Test Type", p1);

              // Step 3: elect server
              Registry reg = ibis.registry();
              IbisIdentifier server = reg.elect("Server");
              boolean amServer = server.equals(ibis.identifier());

              if (amServer) {
                        // Step 4, create port
                        ReceivePort rp = type.createReceivePort("server");
                        rp.enableConnections();

                        // Step 5, receive message and read data
                        ReadMessage rm = rp.receive();
```

```
// Step 3: elect server
Registry reg = ibis.registry();
IbisIdentifier server = reg.elect("Server");
boolean amServer = server.equals(ibis.identifier());
```

```
                        // Step 5, get message and write data
                        WriteMessage wm = sp.newMessage();
                        wm.writeObject("Hello World");
                        wm.finish();

                        // Step 6, close port
                        sp.close();
              }

              ibis.end();
    }
}
```

Code example

```
import ibis.ipl.*;

public class Example {

    public static void main(String args[]) throws Exception {

              // Step 1: create ibis
              StaticProperties p1 = new StaticProperties();
              p1.add("communication", "OneToOne, Reliable, ExplicitReceipt");
              p1.add("serialization", "object");

              Ibis ibis = Ibis.createIbis(p1, null);

              // Step 2: create porttype
              PortType type = ibis.createPortType("Test Type", p1);

              // Step 3: elect server
              Registry reg = ibis.registry();
              IbisIdentifier server = reg.elect("Server");
              boolean amServer = server.equals(ibis.identifier());

              if (amServer) {
                        // Step 4, create port
                        ReceivePort rp = type.createReceivePort("server");
                        rp.enableConnections();

                        // Step 5, receive message and read data
                        ReadMessage rm = rp.receive();
                        String tmp = (String) rm.readObject();
                        rm.finish();

                        System.out.println("Client says: " + tmp);
```

```
if (amServer) {
        // Step 4, create port
        ReceivePort rp = type.createReceivePort("server");
        rp.enableConnections();
```

```
                        // Step 6, close port
                        sp.close();
              }

              ibis.end();
    }
}
```

Code example

```
import ibis.ipl.*;

public class Example {

    public static void main(String args[]) throws Exception {

              // Step 1: create ibis
              StaticProperties p1 = new StaticProperties();
              p1.add("communication", "OneToOne, Reliable, ExplicitReceipt");
              p1.add("serialization", "object");

              Ibis ibis = Ibis.createIbis(p1, null);

              // Step 2: create porttype
              PortType type = ibis.createPortType("Test Type", p1);

              // Step 3: elect server
              Registry reg = ibis.registry();
              IbisIdentifier server = reg.elect("Server");
              boolean amServer = server.equals(ibis.identifier());

              if (amServer) {
```

```
// Step 4, create port, find receivePort and connect
SendPort sp = type.createSendPort();
sp.connect(reg.lookupReceivePort("server"));
```

```
                        // Step 6, close port
                        rp.close();
              } else {

                        // Step 4, create port, find receivePort and connect
                        SendPort sp = type.createSendPort();
                        sp.connect(reg.lookupReceivePort("server"));

                        // Step 5, get message and write data
                        WriteMessage wm = sp.newMessage();
                        wm.writeObject("Hello World");
                        wm.finish();

                        // Step 6, close port
                        sp.close();
              }

              ibis.end();
    }
}
```

Code example

```
import ibis.ipl.*;

public class Example {

    public static void main(String args[]) throws Exception {

              // Step 1: create ibis
              StaticProperties p1 = new StaticProperties();
              p1.add("communication", "OneToOne, Reliable, ExplicitReceipt");
              p1.add("serialization", "object");

              Ibis ibis = Ibis.createIbis(p1, null);

              // Step 2: create porttype
              PortType type = ibis.createPortType("Test Type", p1);

              // Step 3: elect server
              Registry reg = ibis.registry();
              IbisIdentifier server = reg.elect("Server");
              boolean amServer = server.equals(ibis.identifier());

              if (amServer) {
```

```
// Step 5, get message and write data
WriteMessage wm = sp.newMessage();
wm.writeObject("Hello World");
wm.finish();
```

```
                        rp.close();
              } else {

                        // Step 4, create port, find receivePort and connect
                        SendPort sp = type.createSendPort();
                        sp.connect(reg.lookupReceivePort("server"));

                        // Step 5, get message and write data
                        WriteMessage wm = sp.newMessage();
                        wm.writeObject("Hello World");
                        wm.finish();

                        // Step 6, close port
                        sp.close();
              }

              ibis.end();
    }
}
```

ibis
efficient Java-based
grid computing

vrije Universiteit

Code example

```
import ibis.ipl.*;

public class Example {

    public static void main(String args[]) throws Exception {

              // Step 1: create ibis
              StaticProperties p1 = new StaticProperties();
              p1.add("communication", "OneToOne, Reliable, ExplicitReceipt");
              p1.add("serialization", "object");

              Ibis ibis = Ibis.createIbis(p1, null);

              // Step 2: create porttype
              PortType type = ibis.createPortType("Test Type", p1);

              // Step 3: elect server
              Registry reg = ibis.registry();
              IbisIdentifier server = reg.elect("Server");
              boolean amServer = server.equals(ibis.identifier());

              if (amServer) {
                        // Step 4, create port
                        ReceivePort rp = type.createReceivePort("server");
                        rp.enableConnections();

                        // Step 5, receive message and read data
                        ReadMessage rm = rp.receive();
                        String tmp = (String) rm.readObject();
                        rm.finish();

                        System.out.println("Client says: " + tmp);

                        // Step 6, close port
                        rp.close();
              } else {
```

```
// Step 5, receive message and read data
ReadMessage rm = rp.receive();
String tmp = (String) rm.readObject();
rm.finish();

System.out.println("Client says: " + tmp);
```

Code example

```
import ibis.ipl.*;

public class Example {

    public static void main(String args[]) throws Exception {

              // Step 1: create ibis
              StaticProperties p1 = new StaticProperties();
              p1.add("communication", "OneToOne, Reliable, ExplicitReceipt");
              p1.add("serialization", "object");

              Ibis ibis = Ibis.createIbis(p1, null);

              // Step 2: create porttype
              PortType type = ibis.createPortType("Test Type", p1);

              // Step 3: elect server
              Registry reg = ibis.registry();
              IbisIdentifier server = reg.elect("Server");
              boolean amServer = server.equals(ibis.identifier());

              if (amServer) {
                        // Step 4, create port
                        ReceivePort rp = type.cre
                        rp.enableConnections();

                        // Step 5, receive messag
                        ReadMessage rm = rp.recei
                        String tmp = (String) rm.readObject();
                        rm.finish();

                        System.out.println("Client says: " + tmp);

                        // Step 6, close port
                        rp.close();
              } else {

                        // Step 4, create port, find receivePort and connect
                        SendPort sp = type.createSendPort();
                        sp.connect(reg.lookupReceivePort("server"));

                        // Step 5, get message and w
                        WriteMessage wm = sp.newMess
                        wm.writeObject("Hello World"
                        wm.finish();

                        // Step 6, close port
                        sp.close();
              }

              ibis.end();
    }
}
```

```
// Step 6, close port
rp.close();
```

```
// Step 6, close port
sp.close();
```

Code example

```
import ibis.ipl.*;

public class Example {

    public static void main(String args[]) throws Exception {

              // Step 1: create ibis
              StaticProperties p1 = new StaticProperties();
              p1.add("communication", "OneToOne, Reliable, ExplicitReceipt");
              p1.add("serialization", "object");

              Ibis ibis = Ibis.createIbis(p1, null);

              // Step 2: create porttype
              PortType type = ibis.createPortType("Test Type", p1);

              // Step 3: elect server
              Registry reg = ibis.registry();
              IbisIdentifier server = reg.elect("Server");
              boolean amServer = server.equals(ibis.identifier());

              if (amServer) {
                        // Step 4, create port
                        ReceivePort rp = type.createReceivePort("server");
                        rp.enableConnections();

                        // Step 5, receive message and read data
                        ReadMessage rm = rp.receive();
                        String tmp = (String) rm.readObject();
                        rm.finish();

                        System.out.println("Client says: " + tmp);

                        // Step 6, close port
                        rp.close();
              } else {

                        // Step 4, create port, find receivePort and connect
                        SendPort sp = type.createSendPort();
                        sp.connect(reg.lookupReceivePort("server"));

                        // Step 5, get message and write data
                        WriteMessage wm = sp.newMessage();
                        wm.writeObject("Hello World");
                        wm.finish();

                        // Step 6, close port
                        sp.close();
              }

              ibis.end();
    }
}
```

```
ibis.end();
```

Code Example

- Live demo

Nameserver

- Used by Ibises to find each other
  - Needs to be in a well known place
  - Used to implements joins & leaves
  - Used to implement Registry
  - Supports multiple namespaces
    - So 'conflicting' apps can use the same server

Higher level programming models

- Remote Method Invocation (RMI)
- Group Method Invocation (GMI)
- Satin (Divide & Conquer)
- MPJ (MPI Java 'standard')
- Others are being developed
  - Balutek (data parallel)
  - Replicated Method Invocation (RepMI)

Satin

- Parallel Divide-and-conquer
  - Divide work into independent parts
  - Spawn sub-jobs
  - Combine sub-results
  - Repeat recursively
- Master-Worker is a subset of this
  - Only one level of recursion
- Targeted at the grid (and clusters)

Sequential Fibonacci

```
public long fib(int n) {
        if (n < 2) return n;

        long x = fib(n - 1);
        long y = fib(n – 2);

        return x + y;
}
```

Parallel Fibonacci

```
interface FibInterface extends ibis.satin.Spawnable {
    public long fib(int n);
}
public long fib(int n) {
        if (n < 2) return n;

        long x = fib(n - 1);
        long y = fib(n – 2);
        sync();
        return x + y;
}
```

Parallel Fibonacci

```
interface FibInterface extends ibis.satin.Spawnable {
    public long fib(int n);
}
public long fib(int n) {
        if (n < 2) return n;

        long x = fib(n - 1);
        long y = fib(n – 2);
        sync();
        return x + y;
}
```

Mark methods as Spawnable.

They are allowed to run in parallel.

Parallel Fibonacci

```
interface FibInterface extends ibis.satin.Spawnable {
    public long fib(int n);
}
public long fib(int n) {
        if (n < 2) return n;

        long x = fib(n - 1);
        long y = fib(n – 2);
        sync();
        return x + y;
}
```

Mark methods as Spawnable.

They are allowed to run in parallel.

Wait until spawned methods are done.

efficient Java-based grid computing

vrije Universiteit

Satin features

- Satin distributes jobs across machines
- Load-balancing is done automatically
  - Algorithm has been proven to be optimal on homogeneous systems
  - Additional highly-efficient grid-aware algorithms

Satin features

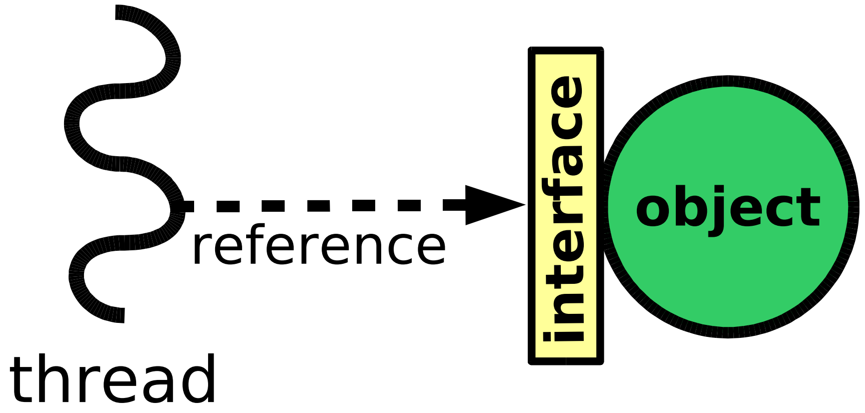
- Malleability
  - Add/remove machines on the fly
- Fault-tolerance
  - When a machine leave suddenly (crashes) the others continue the computation and automatically recompute the lost work
- Shared Objects (added recently)
  - Allows machines to share 'global data'

Satin Applications

- More interesting applications
  - Numerical functions
  - N-body simulations
  - Game-tree search
  - Raytracer
  - Satisfiability solver
  - Grammar-based text analysis
  - Bioinformatics applications
  - …

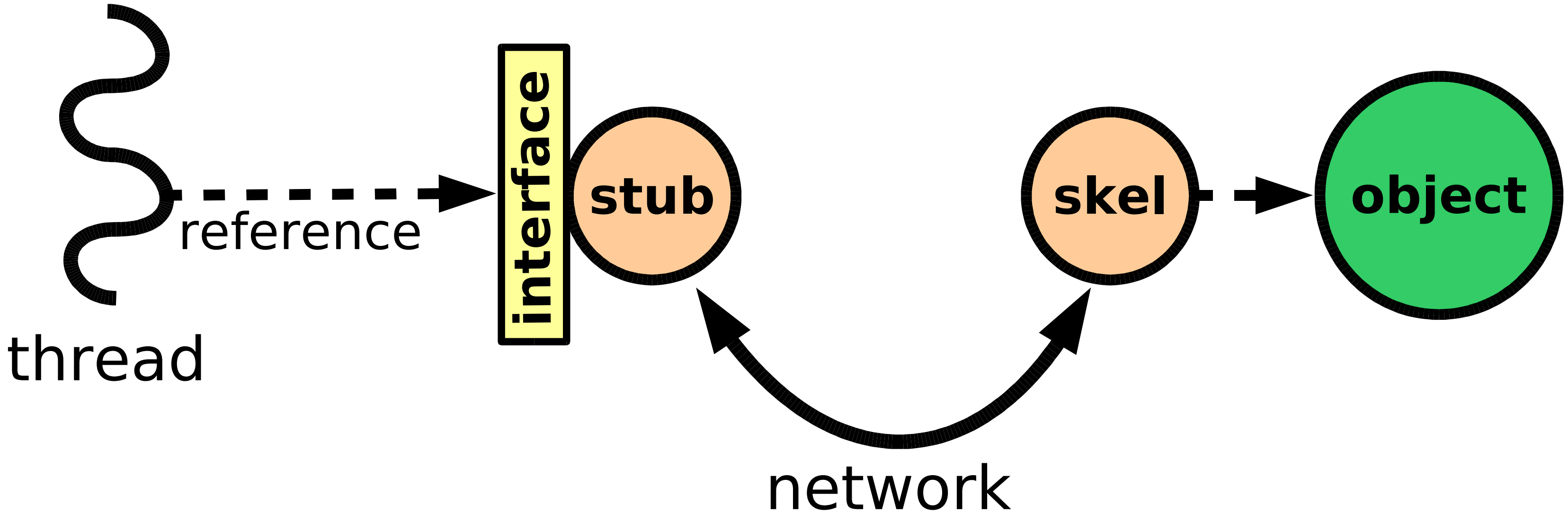
Ibis RMI

- Replacement for Sun RMI
  - Has the same interface
  - Used different stub compiler (rmic)
    - Generates Ibis specific stubs/skeletons



Ibis RMI

- Replacement for Sun RMI
  - Has the same interface
  - Used different stub compiler (rmic)
    - Generates Ibis specific stubs/skeletons



Ibis RMI

- Not interoperable with Sun RMI
  - uses a different protocol
- No socket factories
  - Ibis doesn't have to use sockets!
- No activatable objects

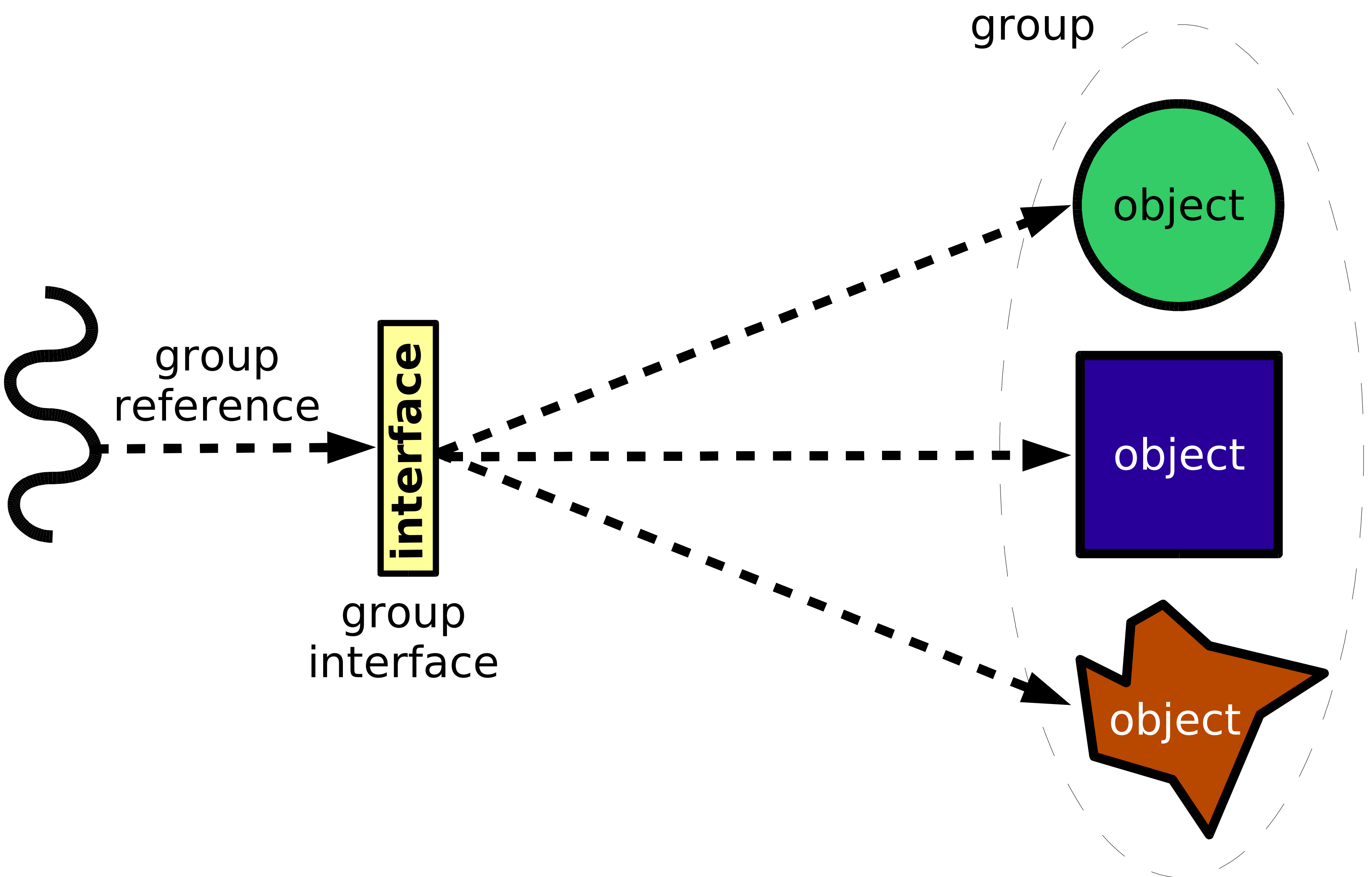
GMI

- Generalized RMI model
  - Allows communication with groups
    - A single stub refers to an entire group
  - Allows more 'advanced' communication
    - By offering different ways of forwarding a method invocation and handling the reply

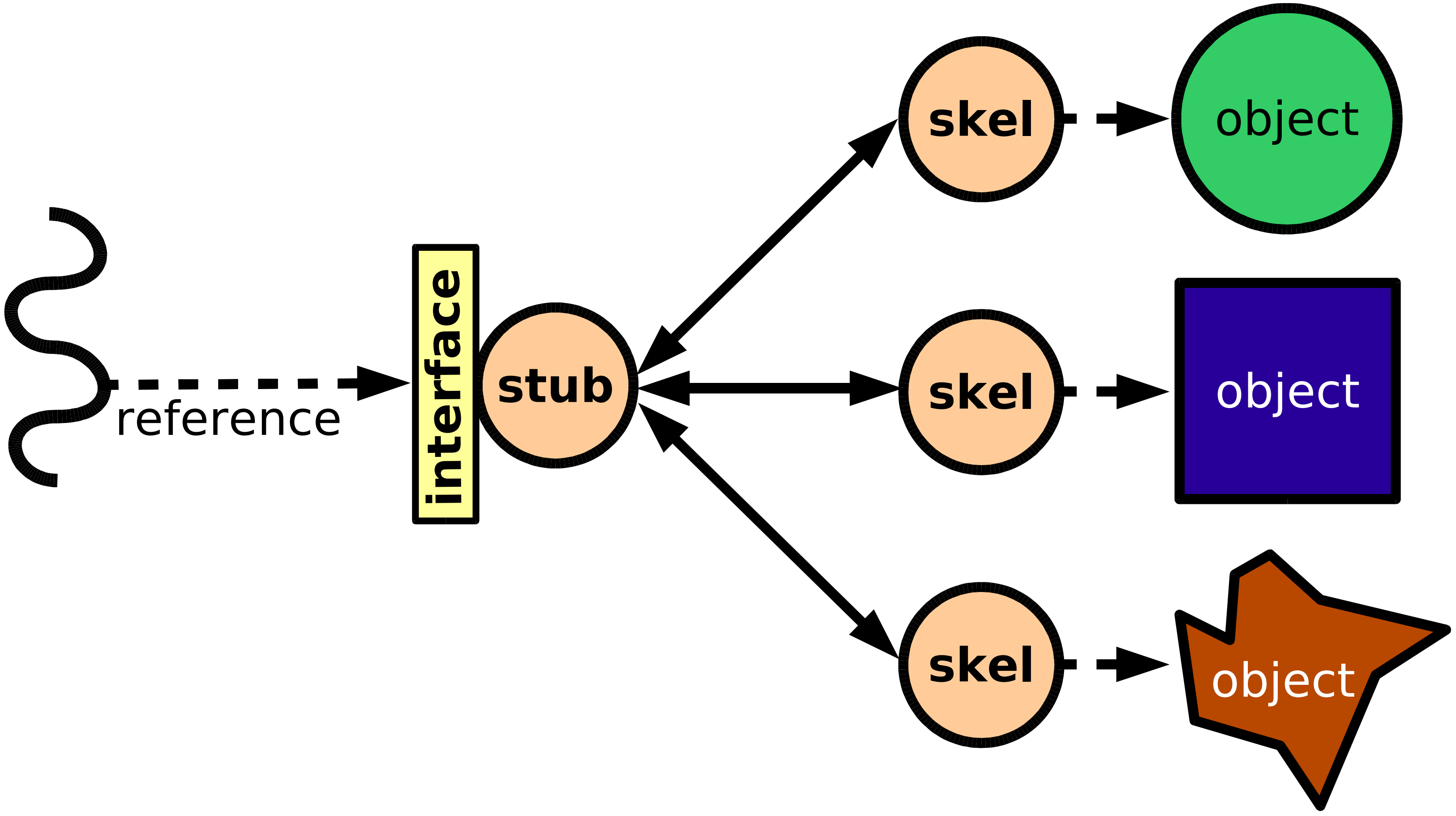
GMI

- Group
  - Contains 1 or more objects
    - Fixed size (set when it is created)
  - All objects must implement the same group interface
    - But objects may have different type!
  - Unique name
    - Used in lookup (produces group reference)
  - Group members have rank
    - Ranks are 'per-group'

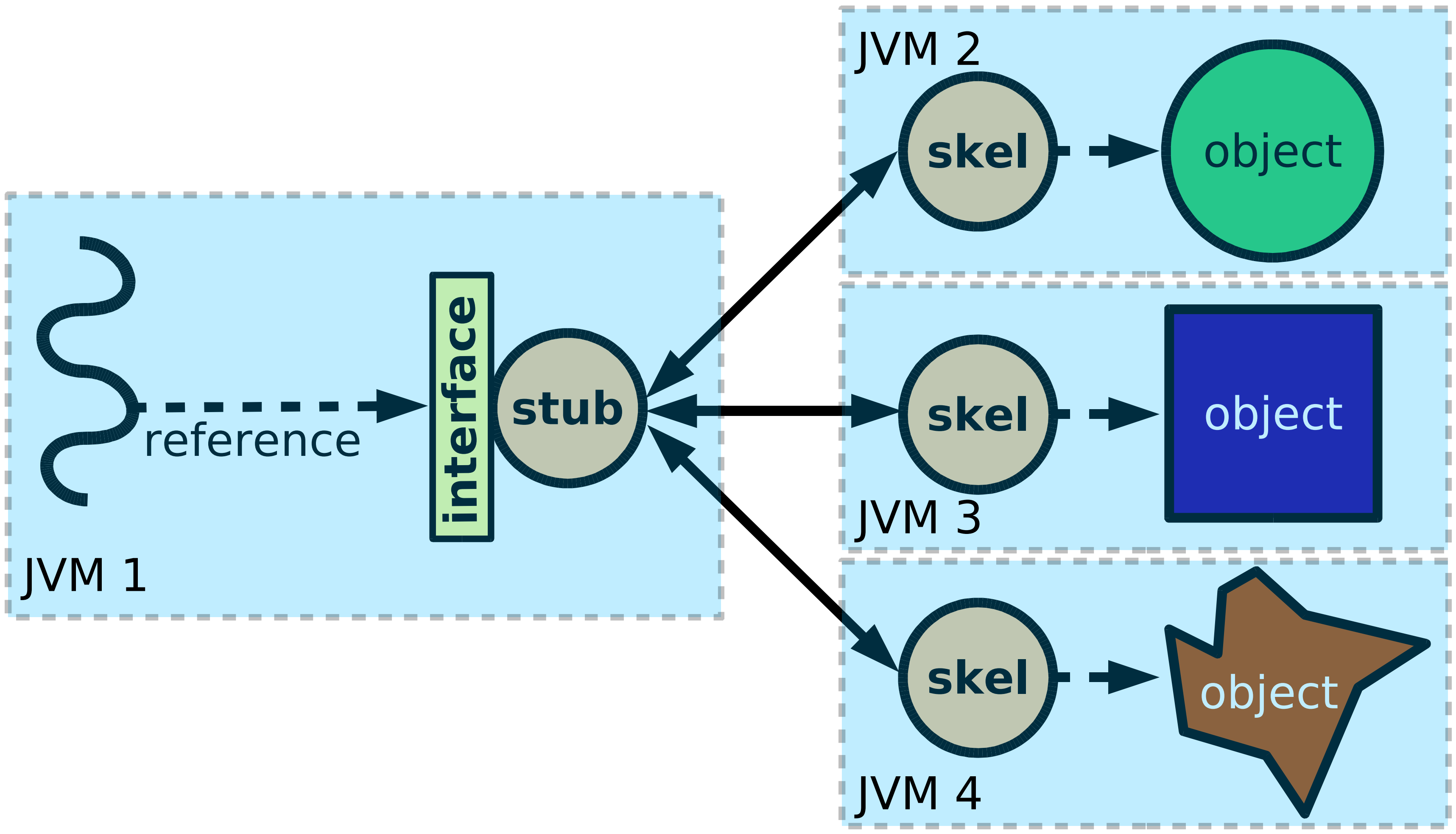
GMI Example


group
group reference
interface
group interface
object
object
object

GMI Implementation



GMI Implementation



Group operations

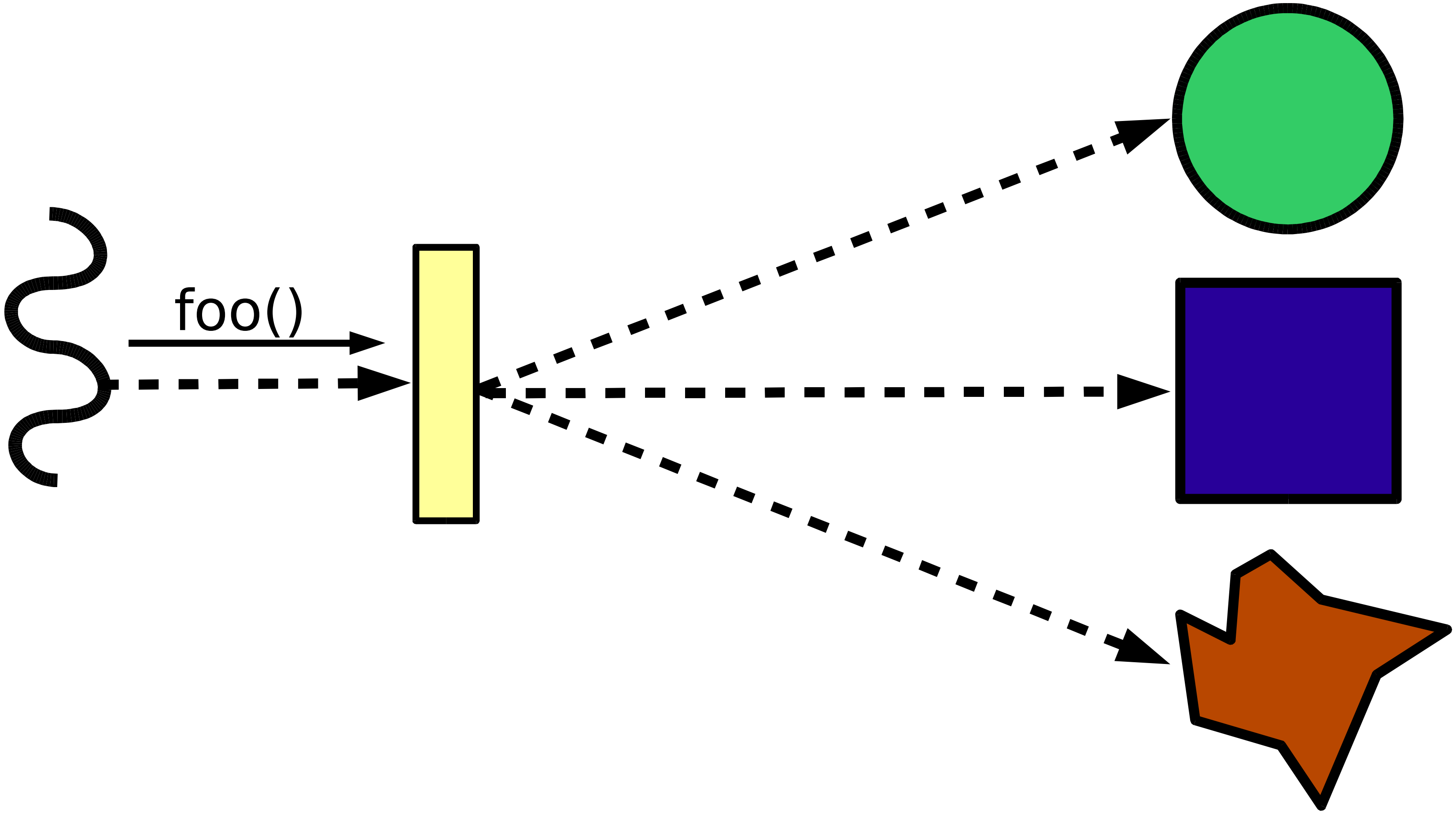
- The group reference can be configured
  - How is a method invocation handled
  - How is the method result handled
  - Configuration per method

- Implemented by selecting different communication code in the generated stubs and skeletons

Invocation Schemes

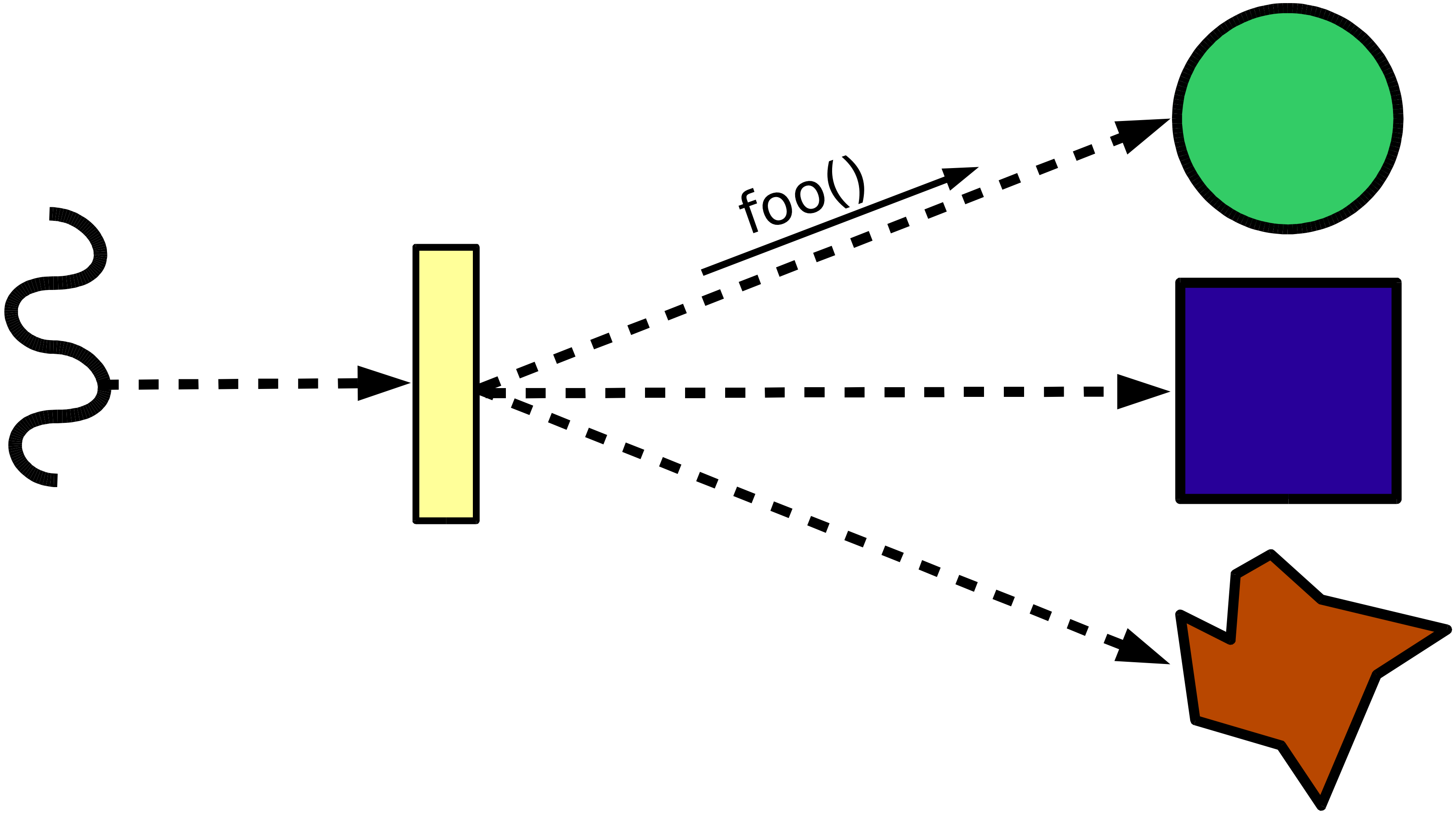
- **Single**
  - Forward to 1 object in group
- **Group**
  - Forward to all objects in group
- **Personalized**
  - Forward to all objects, but personalize parameters for each target
- **Combined**
  - Combine several invocation into one, then foward to the group using one of the above

Single



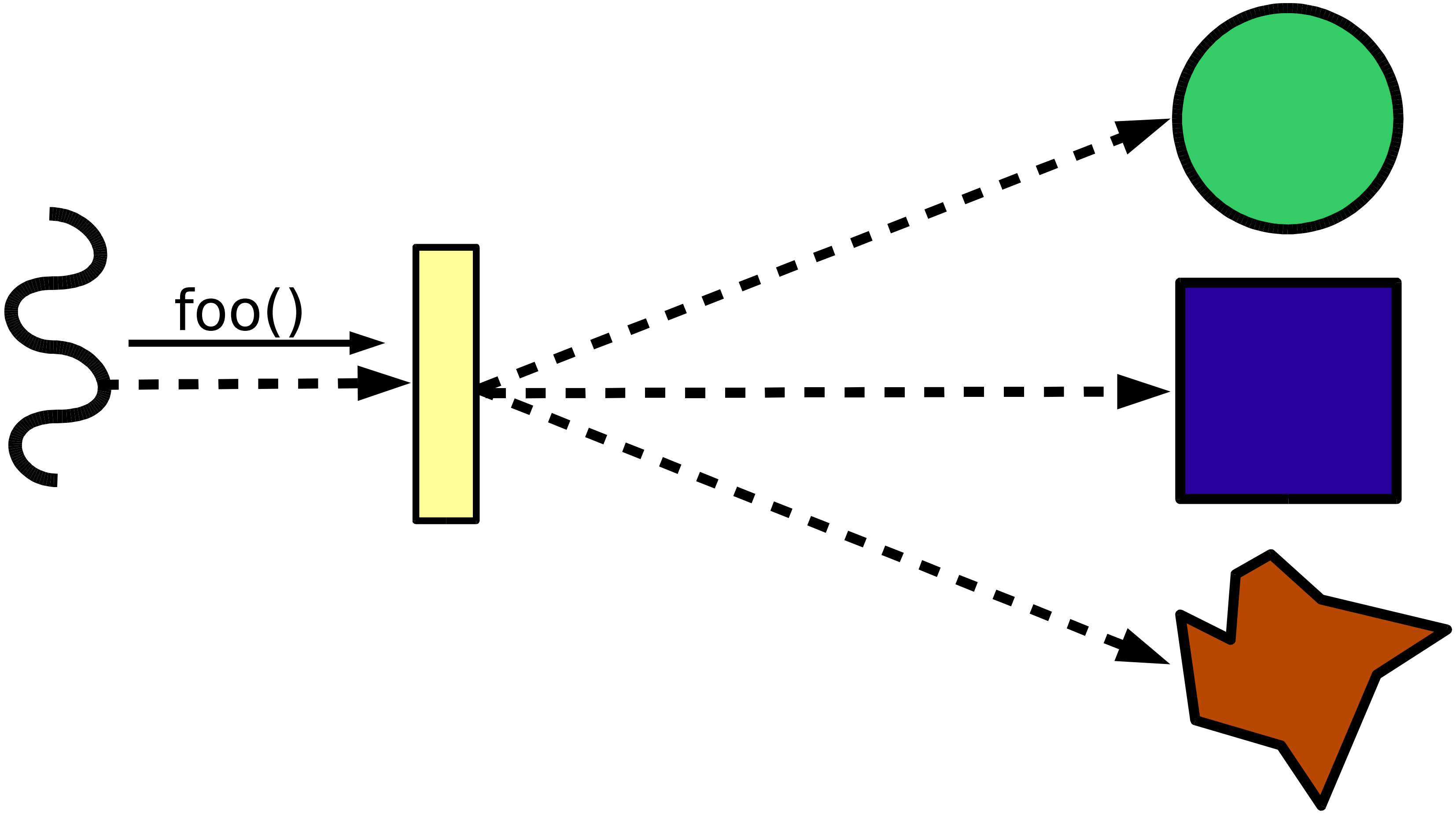
ibis
efficient Java-based grid computing
vrije Universiteit

Single

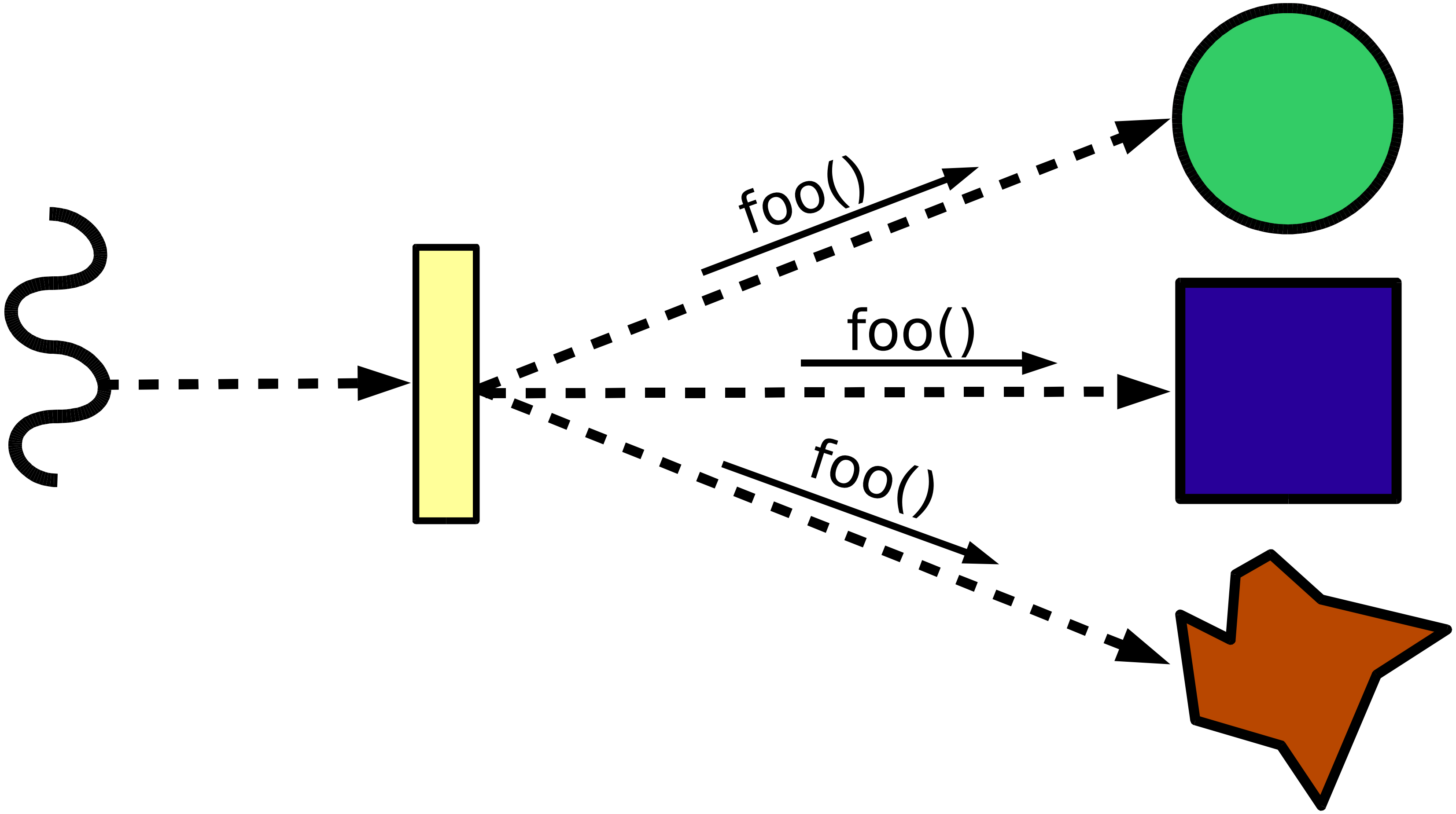


vrije Universiteit

Group



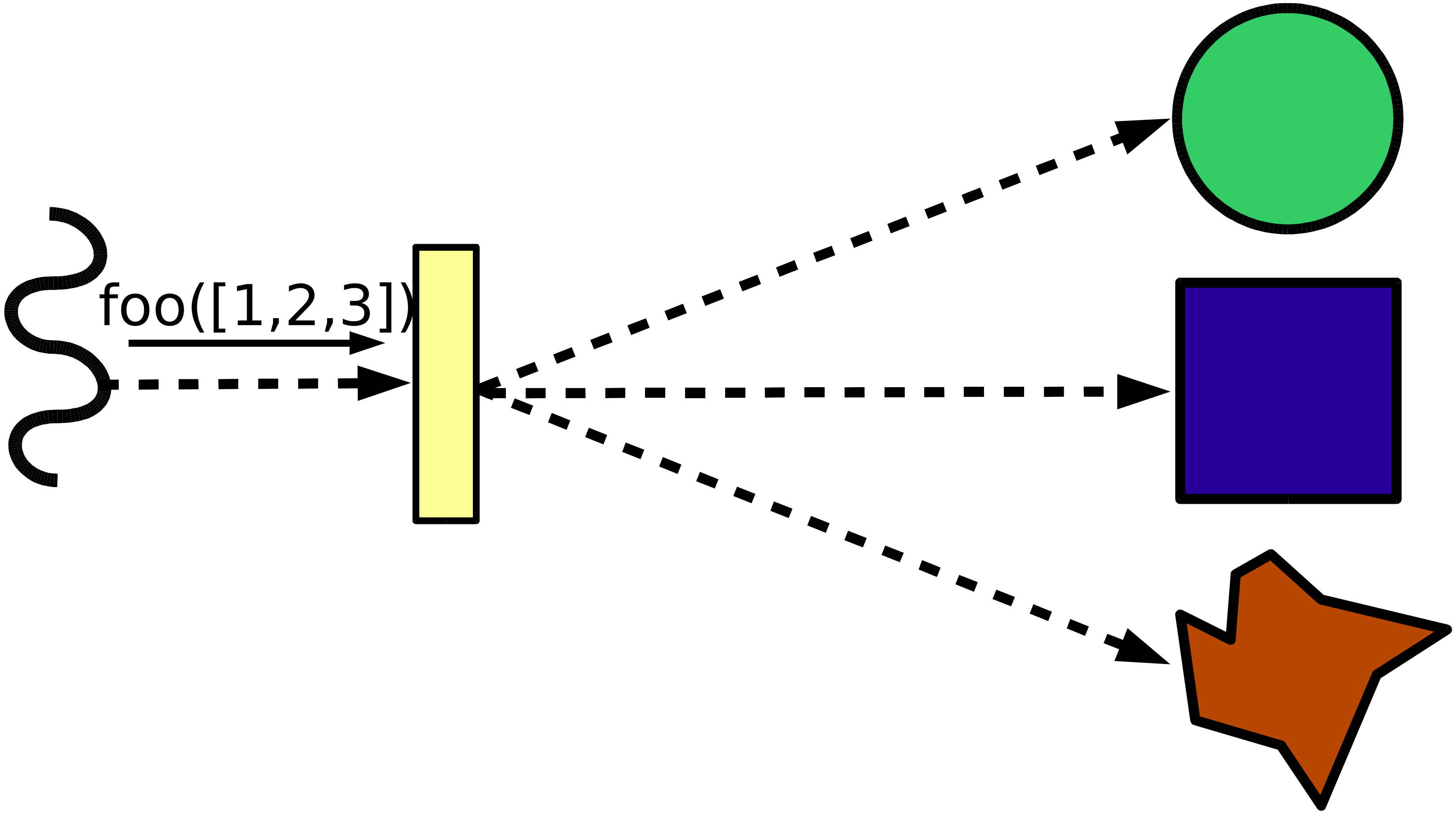
Group



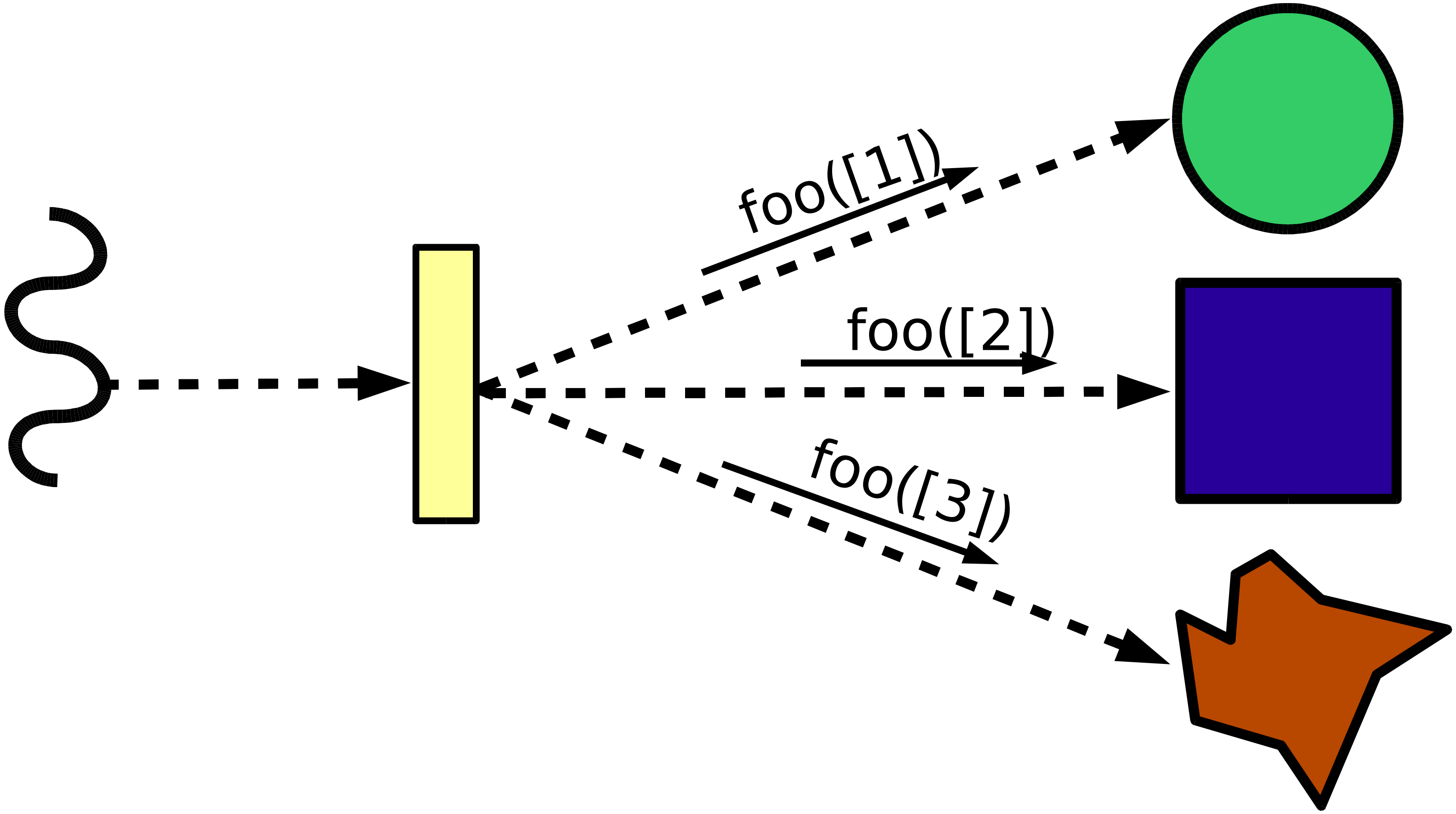
efficient Java-based grid computing

vrije Universiteit

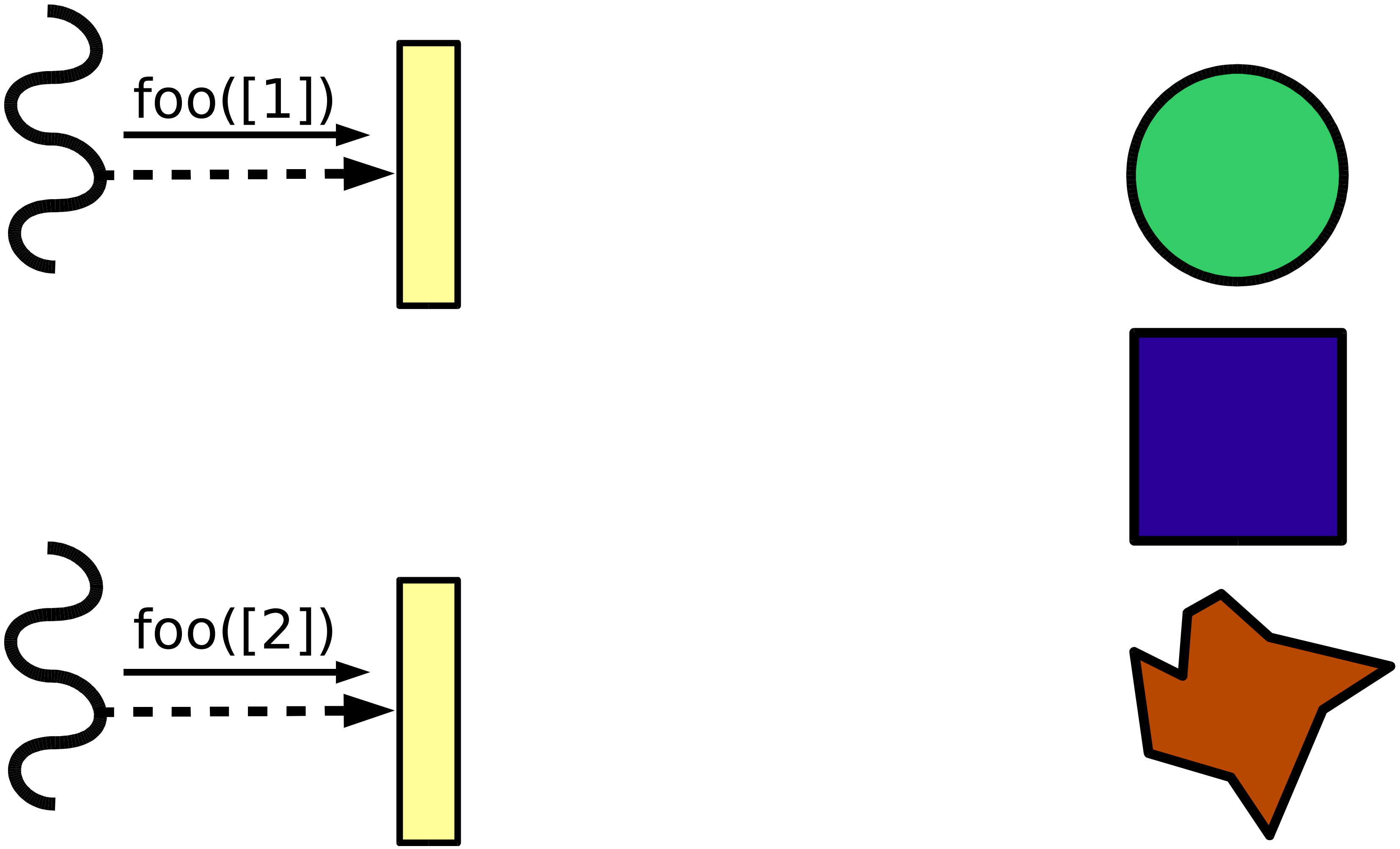
Personalized



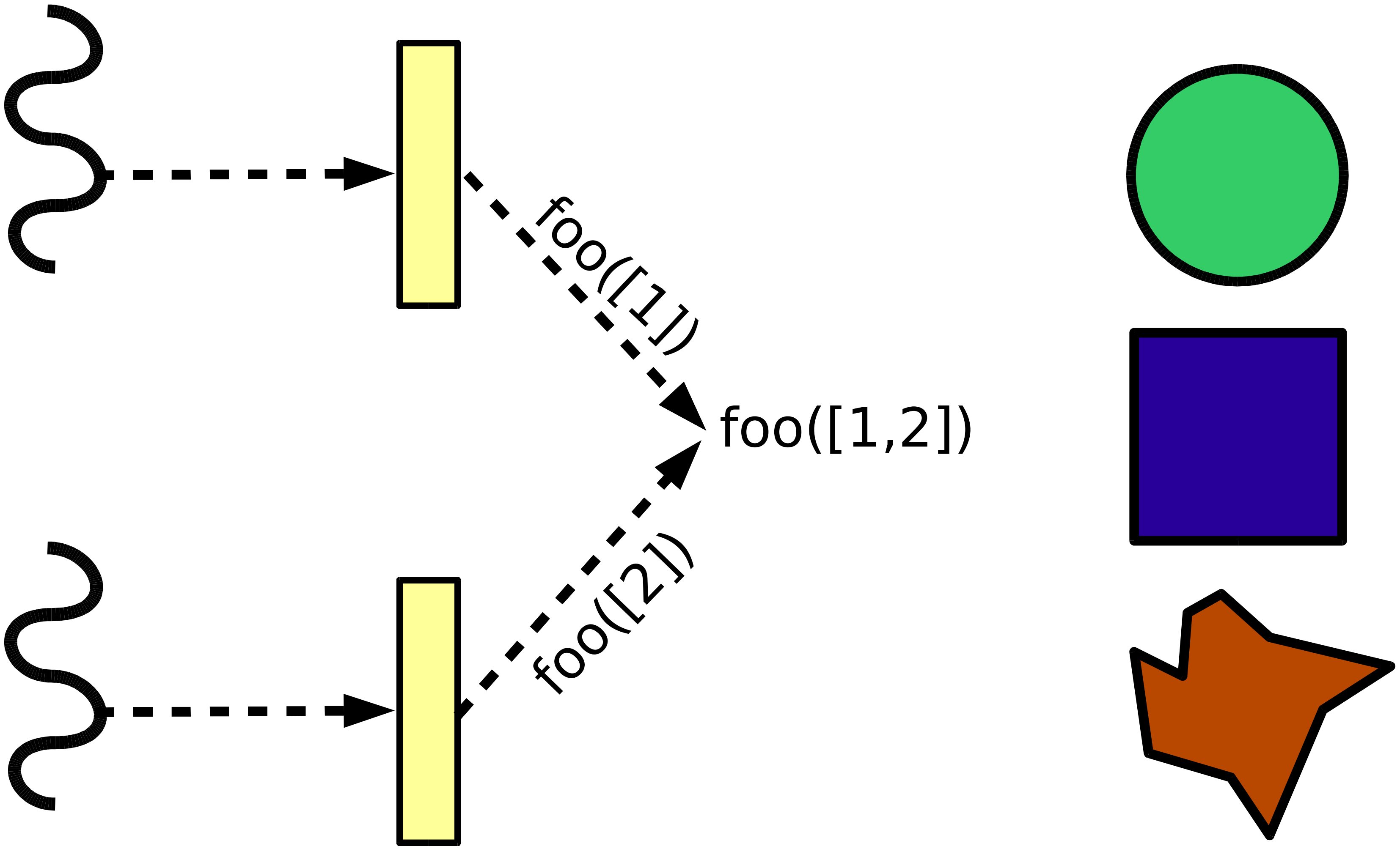
Personalized



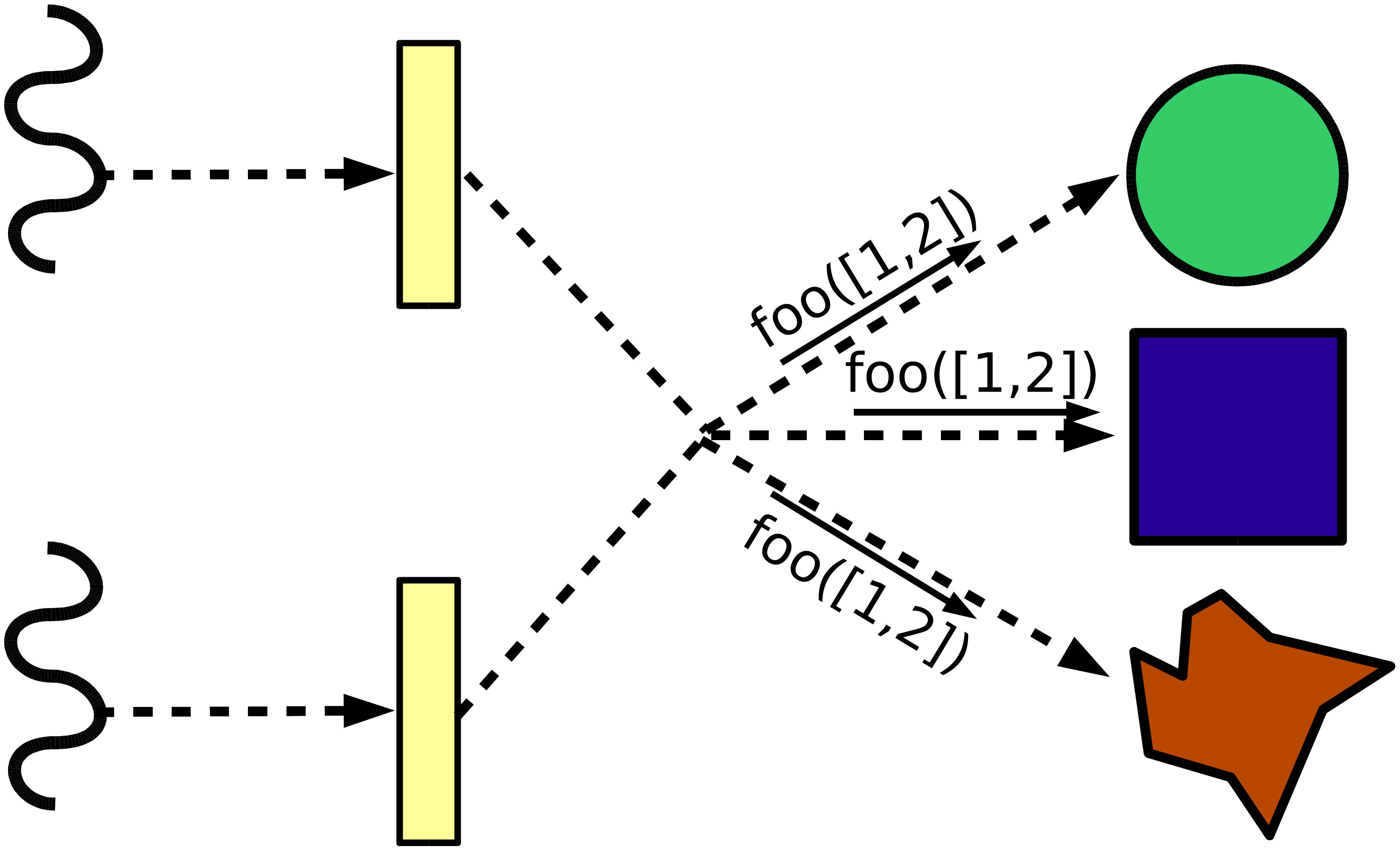
Combined



Combined



Combined



Reply handling schemes

- **Discard**
- **Return**
- **Forward**
  - Reply is forwarded to a seperate object
- **Combine**
  - Multiple replies are combined into one
- **Personalize**
  - A personalized result is returned to each participant of a combined invocation

GMI Communication

Operation	Invocation	Reply
RMI	Single	Return
Async. RMI	Single	Discard
Future	Single	Forward
Broadcast	Group	Discard
Scatter	Personalized	Discard
Reduce results	Group	Combine (binomial)
Gather results	Group	Combine (flat)
Reduce inv.	Combine + Single	Discard
Gather inv.	Combine + Single	Discard

Code example

```
public interface i_SimpleGroup extends GroupInterface {
    void ping();
}

public class SimpleGroup extends GroupMember implements i_SimpleGroup {

    public SimpleGroup() {
        super();
    }

    public void ping() {
        System.out.println("ping");
    }
}

public class MulticastNoReply {
    public static void main(String[] args) throws Exception {

        int rank = Group.rank();
        int size = Group.size();

        // Create the group
        if (rank == 0) {
            Group.create("GroupNoReply", i_SimpleGroup.class, size);
        }

        // Everyone adds an object
        SimpleGroup s = new SimpleGroup();
        Group.join("GroupNoReply", s);

        if (rank == 0) {
            // Perform lookup to get group reference
            i_SimpleGroup g = (i_SimpleGroup) Group.lookup("GroupNoReply");

            // Configure reference to perform group invocation
            GroupMethod m = Group.findMethod(g,"void ping()");
            m.configure(new GroupInvocation(), new DiscardReply());

            // Perform the invocation
            g.ping();
        }

        // Done
        Group.exit();
    }
}
```

Code example

```
public interface i_SimpleGroup extends GroupInterface {
    void ping();
}

public class SimpleGroup extends GroupMember implements i_SimpleGroup {

    public SimpleGroup() {
        super();
```

```
public interface i_SimpleGroup extends GroupInterface {
    void ping();
}
```

```
public class MulticastNoReply {
    public static void main(String[] args) throws Exception {

        int rank = Group.rank();
        int size = Group.size();

        // Create the group
        if (rank == 0) {
            Group.create("GroupNoReply", i_SimpleGroup.class, size);
        }

        // Everyone adds an object
        SimpleGroup s = new SimpleGroup();
        Group.join("GroupNoReply", s);

        if (rank == 0) {
            // Perform lookup to get group reference
            i_SimpleGroup g = (i_SimpleGroup) Group.lookup("GroupNoReply");

            // Configure reference to perform group invocation
            GroupMethod m = Group.findMethod(g,"void ping()");
            m.configure(new GroupInvocation(), new DiscardReply());

            // Perform the invocation
            g.ping();
        }

        // Done
        Group.exit();
    }
}
```

Code example

```
public interface i_SimpleGroup extends GroupInterface {
    void ping();
}

public class SimpleGroup extends GroupMember implements i_SimpleGroup {

    public SimpleGroup() {
        super();
    }

    public void ping() {
        System.out.println("ping");
    }
}

public class MulticastNoReply {
    public static void main(String[] args) throws Exception {

        int rank = Group.rank();
        int size = Group.size();
```

```
public class SimpleGroup extends GroupMember implements i_SimpleGroup {

    public SimpleGroup() {
        super();
    }

    public void ping() {
        System.out.println("ping");
    }
}
```

```
        }

        // Done
        Group.exit();
    }
}
```

ibis
efficient Java-based grid computing

vrije Universiteit

Code example

```
public interface i_SimpleGroup extends GroupInterface {
    void ping();
}

public class SimpleGroup extends GroupMember implements i_SimpleGroup {

    public SimpleGroup() {
        super();
    }

    public void ping() {
        System.out.println("ping");
    }
}

public class MulticastNoReply {
    public static void main(String[] args) throws Exception {

        int rank = Group.rank();
        int size = Group.size();

        // Create the group
        if (rank == 0) {
            Group.create("GroupNoReply", i_SimpleGroup.class, size);
```

```
public class MulticastNoReply {
    public static void main(String[] args) throws Exception {

        int rank = Group.rank();
        int size = Group.size();
```

```
            GroupMethod m = Group.findMethod(g,"void ping()");
            m.configure(new GroupInvocation(), new DiscardReply());

            // Perform the invocation
            g.ping();
        }

        // Done
        Group.exit();
    }
}
```

ibis
efficient Java-based grid computing

vrije Universiteit

Code example

```java
public interface i_SimpleGroup extends GroupInterface {
    void ping();
}

public class SimpleGroup extends GroupMember implements i_SimpleGroup {

    public SimpleGroup() {
        super();
    }

    public void ping() {
        System.out.println("ping");
    }
}

public class MulticastNoReply {
    public static void main(String[] args) throws Exception {

        int rank = Group.rank();
        int size = Group.size();

        // Create the group
        if (rank == 0) {
            Group.create("GroupNoReply", i_SimpleGroup.class, size);
        }

        // Everyone adds an object
        SimpleGroup s = new SimpleGroup();
        Group.join("GroupNoReply", s);
```

```java
// Create the group
if (rank == 0) {
    Group.create("GroupNoReply", i_SimpleGroup.class, size);
}
```

```java
        }

        // Done
        Group.exit();
    }
}
```

ibis
efficient Java-based
grid computing

vrije Universiteit

Code example

```
public interface i_SimpleGroup extends GroupInterface {
    void ping();
}

public class SimpleGroup extends GroupMember implements i_SimpleGroup {

    public SimpleGroup() {
        super();
    }

    public void ping() {
        System.out.println("ping");
    }
}

public class MulticastNoReply {
    public static void main(String[] args) throws Exception {

        int rank = Group.rank();
        int size = Group.size();

        // Create the group
        if (rank == 0) {
            Group.create("GroupNoReply", i_SimpleGroup.class, size);
        }

        // Everyone adds an object
        SimpleGroup s = new SimpleGroup();
        Group.join("GroupNoReply", s);

        if (rank == 0) {
            // Perform lookup to get group reference
            i

            /
            G
            m

            /
            g.ping();
        }

        // Done
        Group.exit();
    }
}
```

```
// Everyone adds an object
SimpleGroup s = new SimpleGroup();
Group.join("GroupNoReply", s);
```

Code example

```
public interface i_SimpleGroup extends GroupInterface {
    void ping();
}

public class SimpleGroup extends GroupMember implements i_SimpleGroup {
```

```
if (rank == 0) {
    // Perform lookup to get group reference
    i_SimpleGroup g = (i_SimpleGroup) Group.lookup("GroupNoReply");

    // Configure reference to perform group invocation
    GroupMethod m = Group.findMethod(g,"void ping()");
    m.configure(new GroupInvocation(), new DiscardReply());

    // Perform the invocation
    g.ping();
}
```

```
        Group.join("GroupNoReply", s);
```

```
        if (rank == 0) {
            // Perform lookup to get group reference
            i_SimpleGroup g = (i_SimpleGroup) Group.lookup("GroupNoReply");

            // Configure reference to perform group invocation
            GroupMethod m = Group.findMethod(g,"void ping()");
            m.configure(new GroupInvocation(), new DiscardReply());

            // Perform the invocation
            g.ping();
        }
```

```
        // Done
        Group.exit();
    }
}
```

ibis
efficient Java-based
grid computing

vrije Universiteit

Code example

```
public interface i_SimpleGroup extends GroupInterface {
    void ping();
}

public class SimpleGroup extends GroupMember implements i_SimpleGroup {

    public SimpleGroup() {
        super();
    }

    public void ping() {
        System.out.println("ping");
    }
}

public class MulticastNoReply {
    public static void main(String[] args) throws Exception {

        int rank = Group.rank();
        int size = Group.size();

        // Create the group
        if (rank == 0) {
            Group.create("GroupNoReply", i_SimpleGroup.class, size);
        }

        // Everyone adds an object
        SimpleGroup s = new SimpleGroup();
        Group.join("GroupNoReply", s);

        if (rank == 0) {
            // Perform lookup to get group reference
            i_SimpleGroup g = (i_SimpleGroup) Group.lookup("GroupNoReply");

            // Configure reference to perform group invocation
            GroupMethod m = Group.findMethod(g,"void ping()");
            m.configure(new GroupInvocation(), new DiscardReply());

            // Perform th
            g.ping();
        }

        // Done
        Group.exit();
    }
}
```

```
// Done
Group.exit();
```

Code Example

- Live demo

Function Objects

- Some operations need user defined functions
  - Personalizing a method invocation
  - Combining a result or invocation
  - Forwarding of results
- GMI uses function objects
  - Extend a class from the GMI package

Result Combiners

- Use 'combiner' to merge the results of an invocation
- FlatCombiner
  - Combines all results in one go
  - Similar to 'gather' operation of MPI
- BinomialCombiner
  - Pairwise combines results
  - Similar to 'reduce' operation of MPI

FlatCombiner

```
public class FlatCombiner {

    public boolean combine(boolean[] results, Exception[] ex)
    public byte combine(byte[] results, Exception[] ex)
    public char combine(char[] results, Exception[] ex)
    public short combine(short[] results, Exception[] ex)
    public int combine(int[] results, Exception[] ex)
    public long combine(long[] results, Exception[] ex)
    public float combine(float[] results, Exception[] ex)
    public double combine(double[] results, Exception[] ex)

    public Object combine(Object[] results, Exception[] ex)

    public void combine(Exception[] exceptions)
}
```

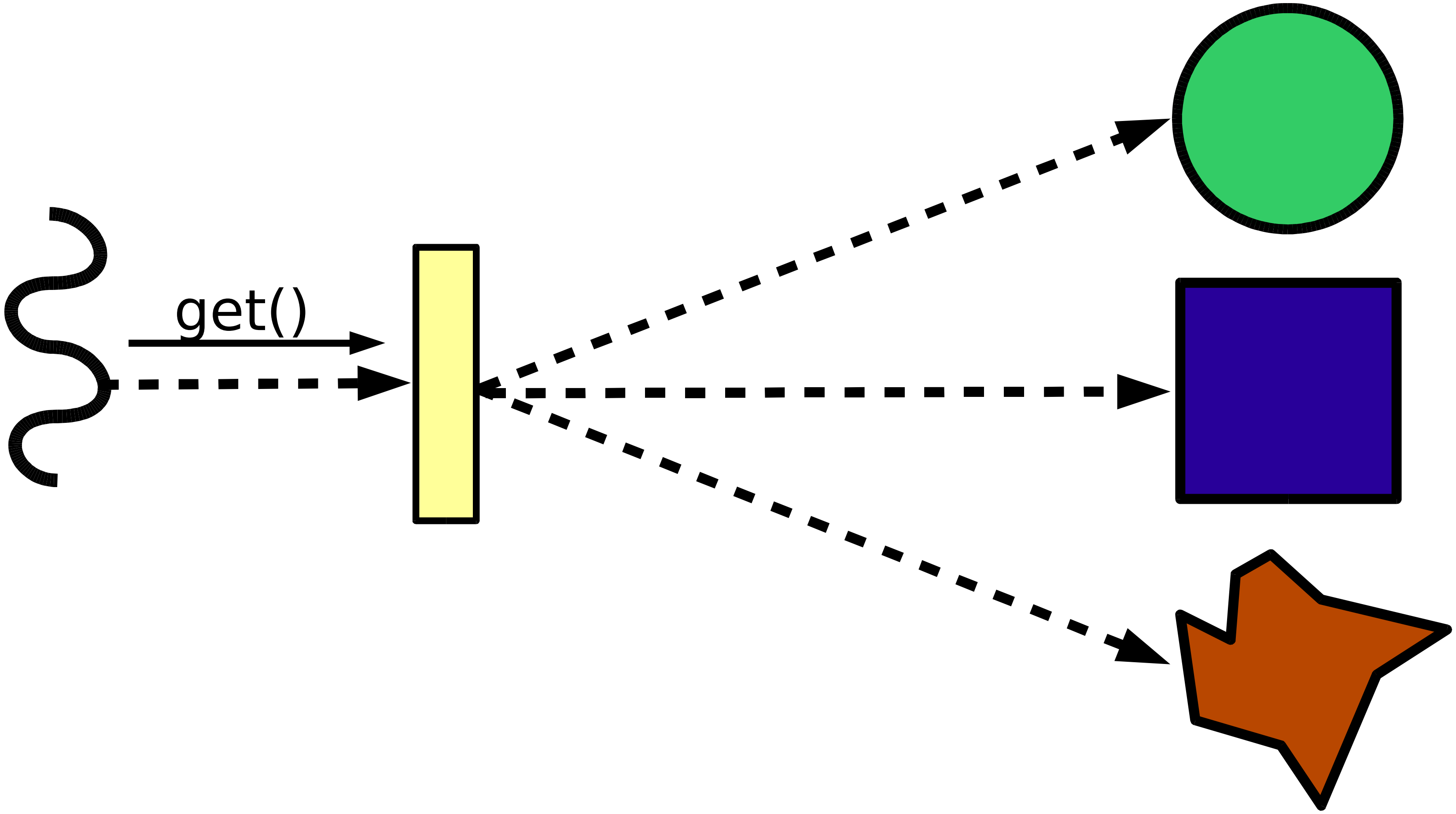
FlatCombiner

```
public class MyCombiner extends FlatCombiner {

    public int combine(int[] results, Exception[] ex) {

        int sum = 0;

        for (int i=0;i<results.length;i++) {
            sum += results[i];
        }

        return sum;
    }
}
```

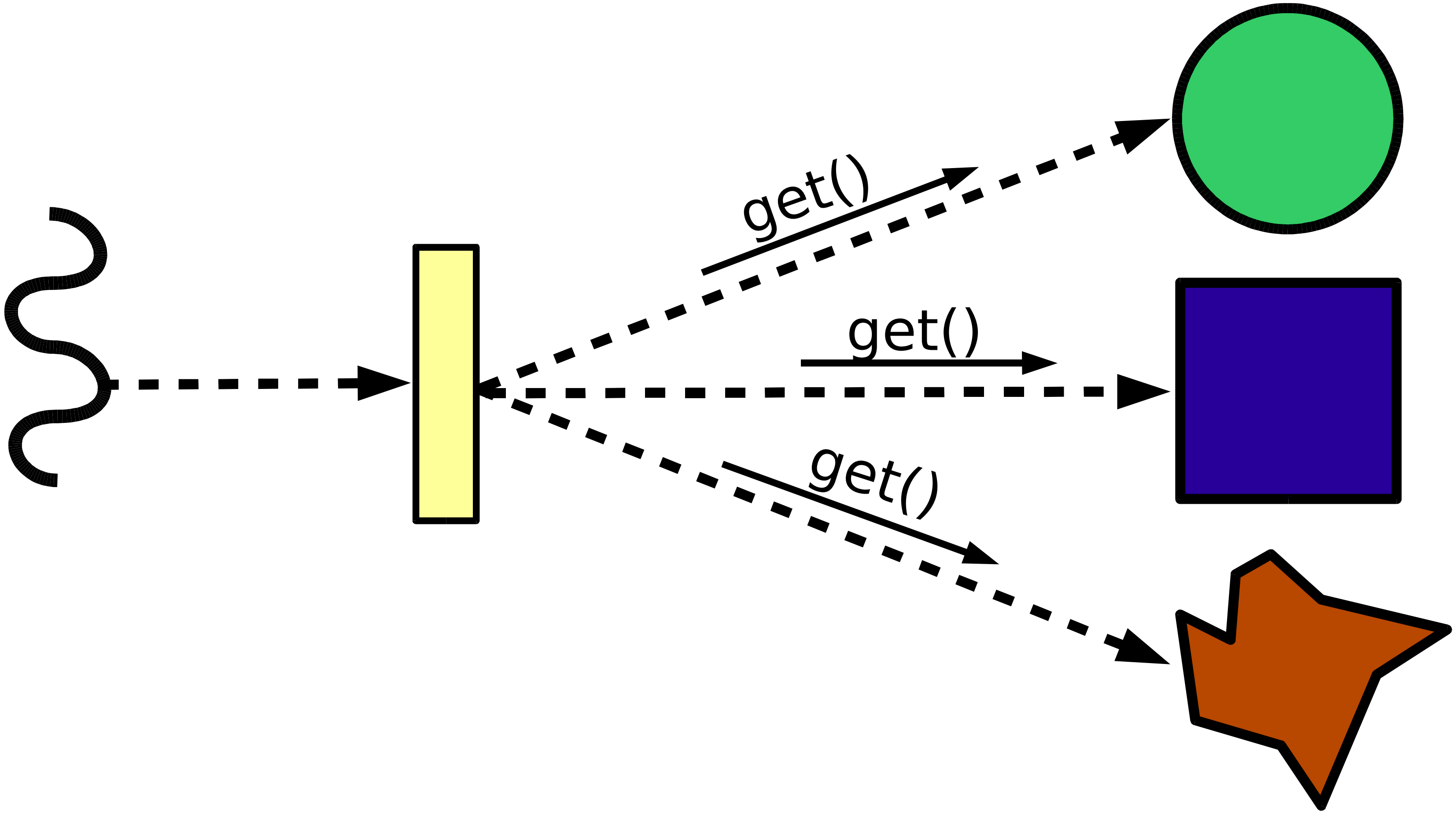
FlatCombiner

```
// Get a group reference
X g = (X) Group.lookup("your group");


// Configure reference to perform group invocation,
// and combine the replies using 'MyCombiner'
GroupMethod m = Group.findMethod(g, "int get()");
m.configure(new GroupInvocation(),
            new CombineReply(new MyCombiner()));


// Perform the invocation
int result = g.get();
```

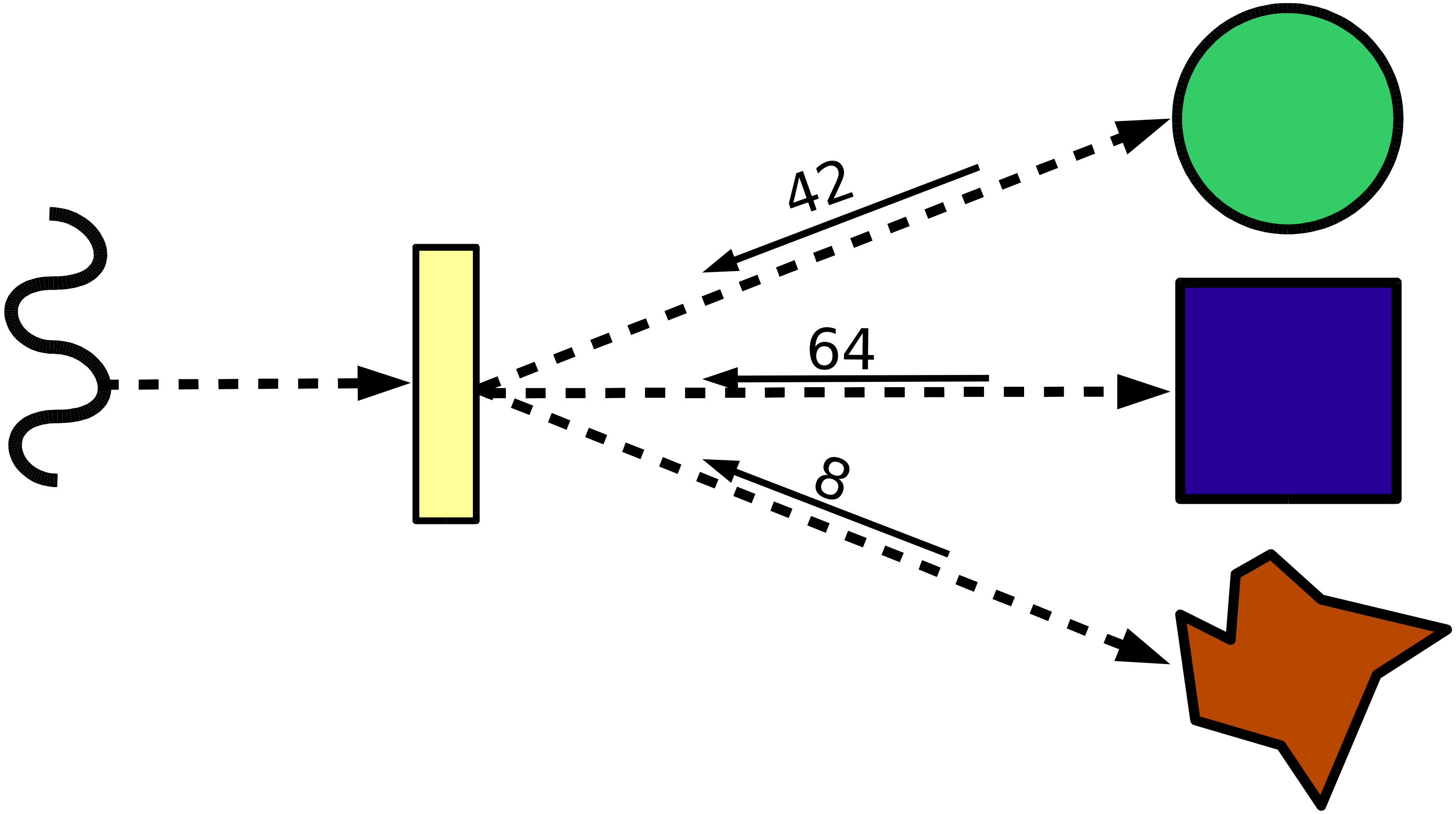
FlatCombiner



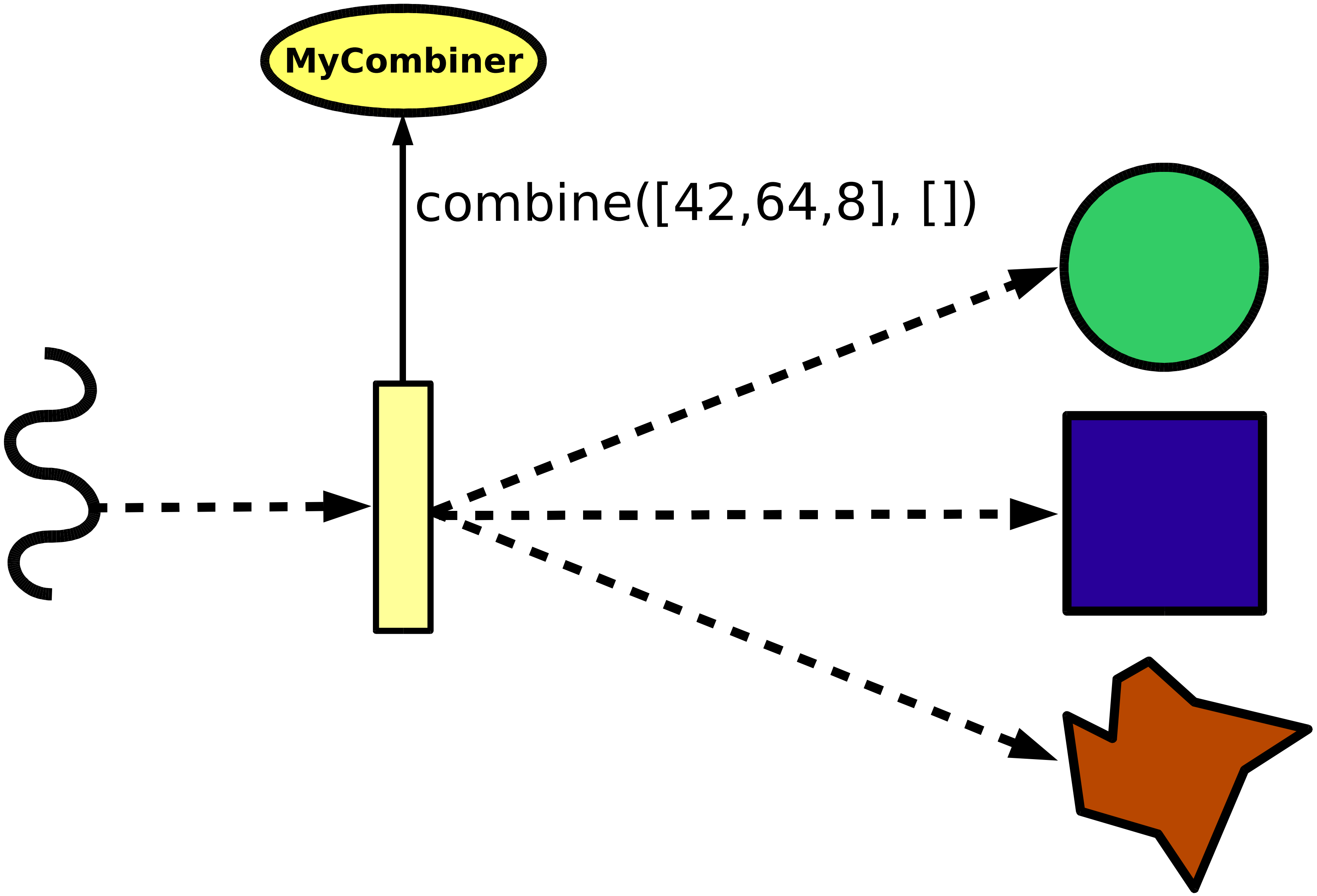
FlatCombiner



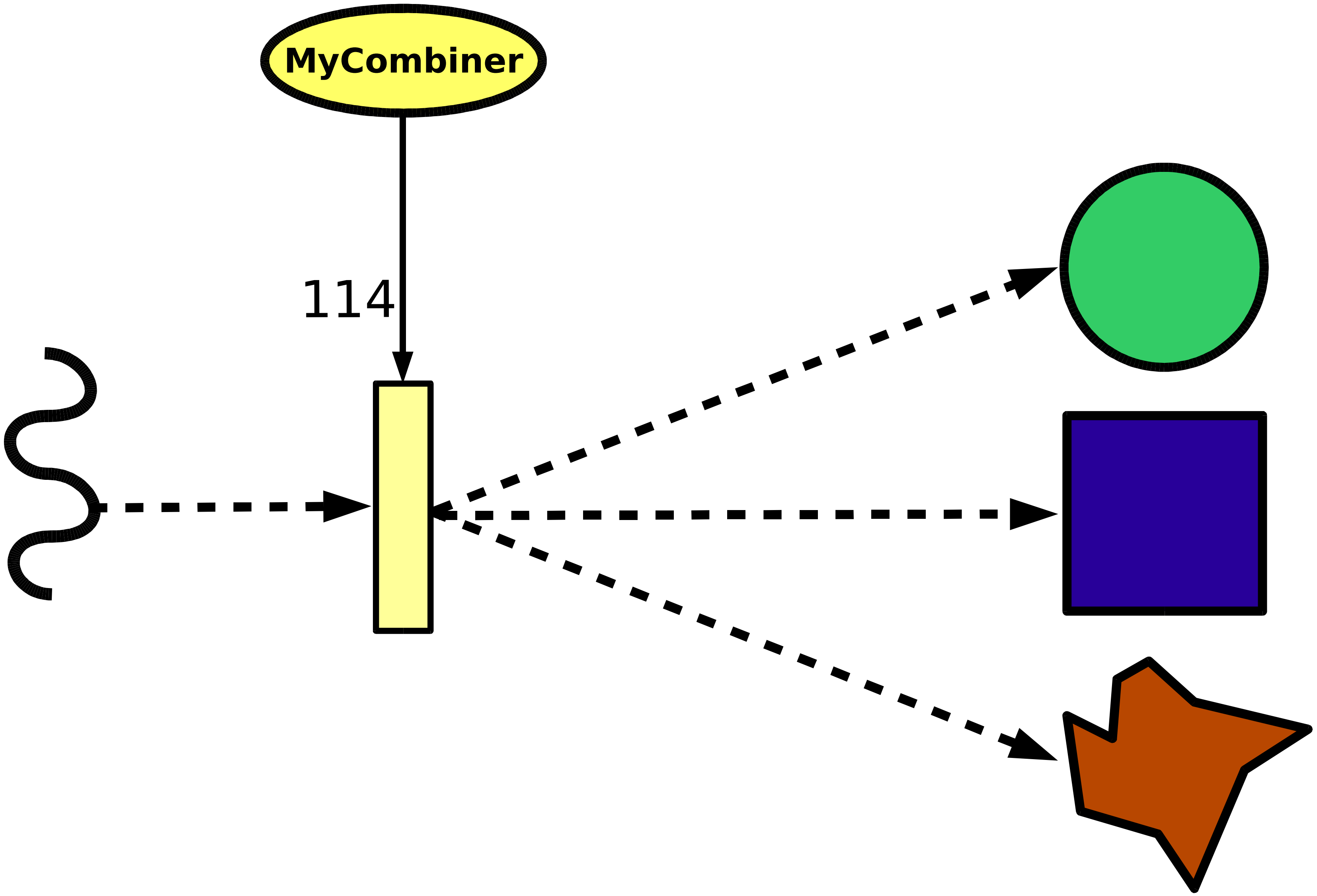
FlatCombiner



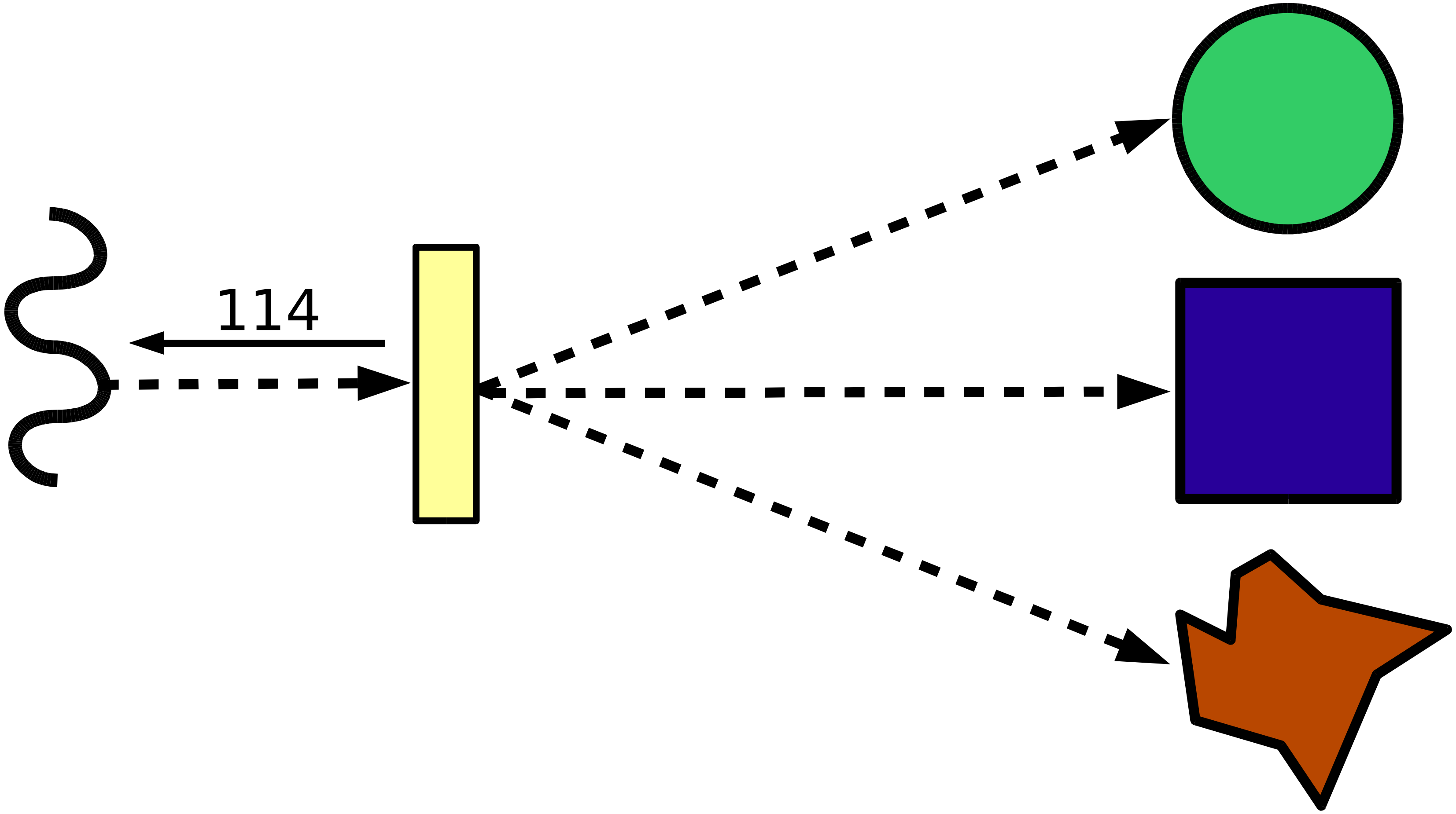
FlatCombiner



FlatCombiner



FlatCombiner



FlatCombiner Demo

- Live demo

After the Break

- Hands-on session
  - Installing ibis
  - Running applications
    - Scripts and command line
  - Writing your own applications