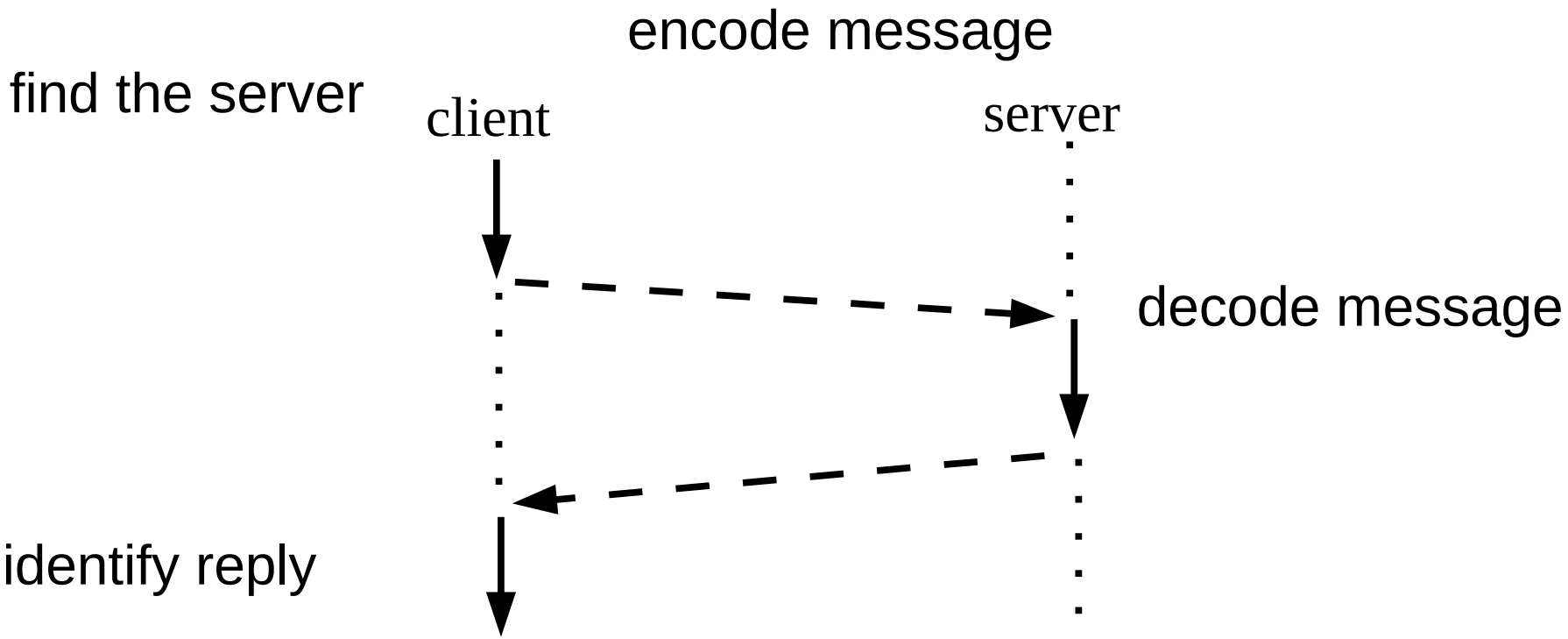
Distributed Systems

ID2201

Remote Invocation

Johan Montelius

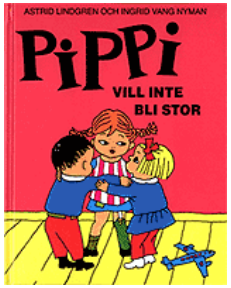
Request Reply



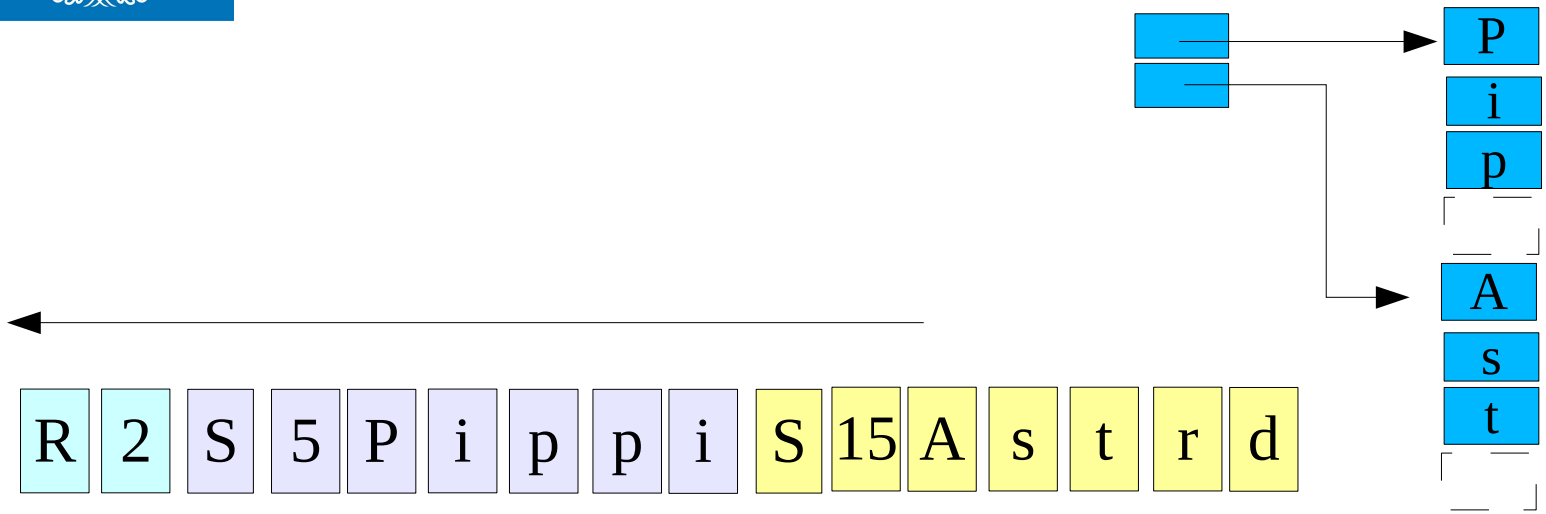
Find the process

- A process can listen to a port.
  - Use the IP address and port number to find the process.
- Problems
  - What if the process moves or should be taken over by another process.

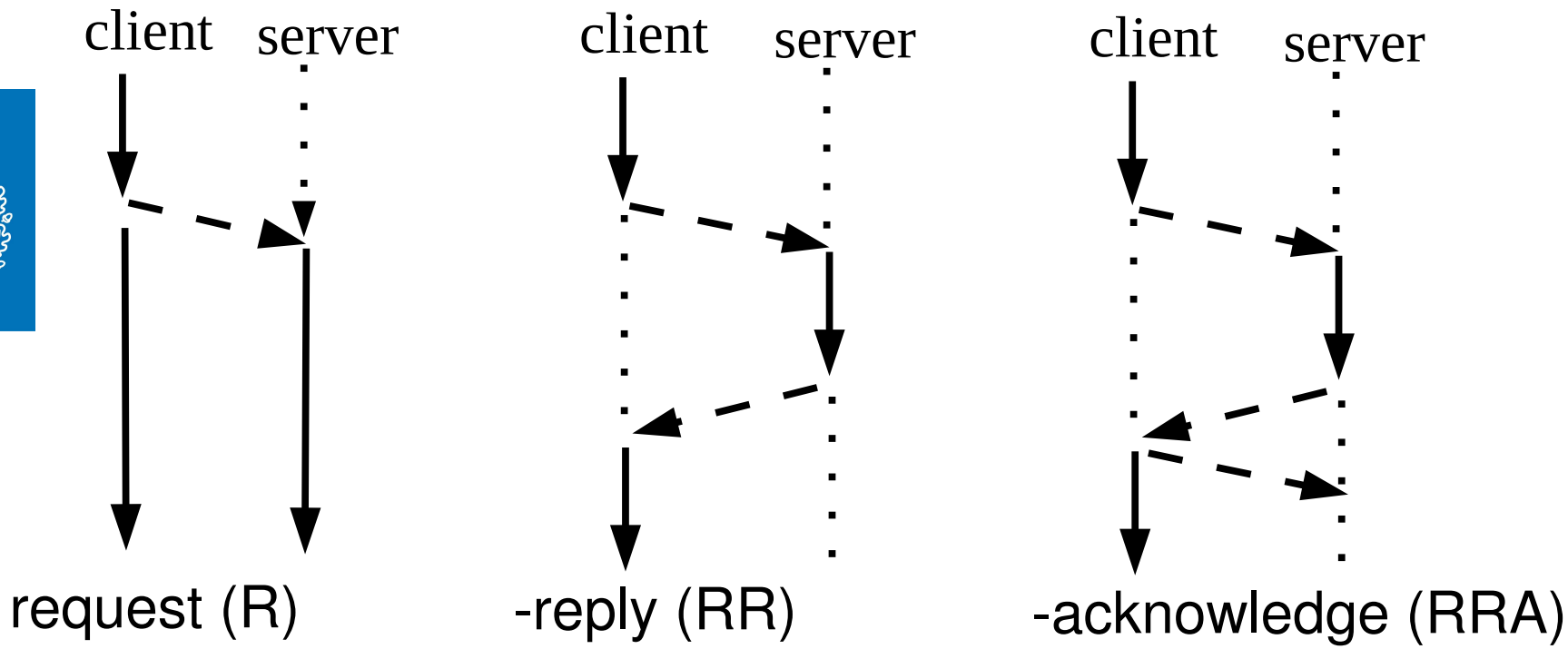
Marshaling



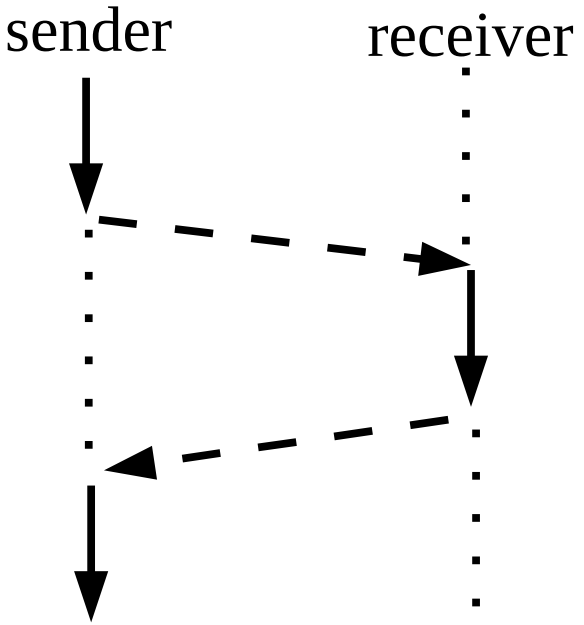
book = {“Pippi”, “Astrid Lindgren”}

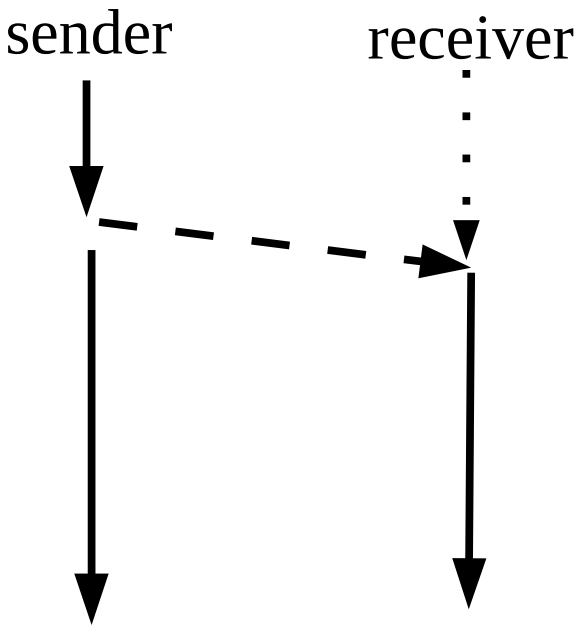


Reply

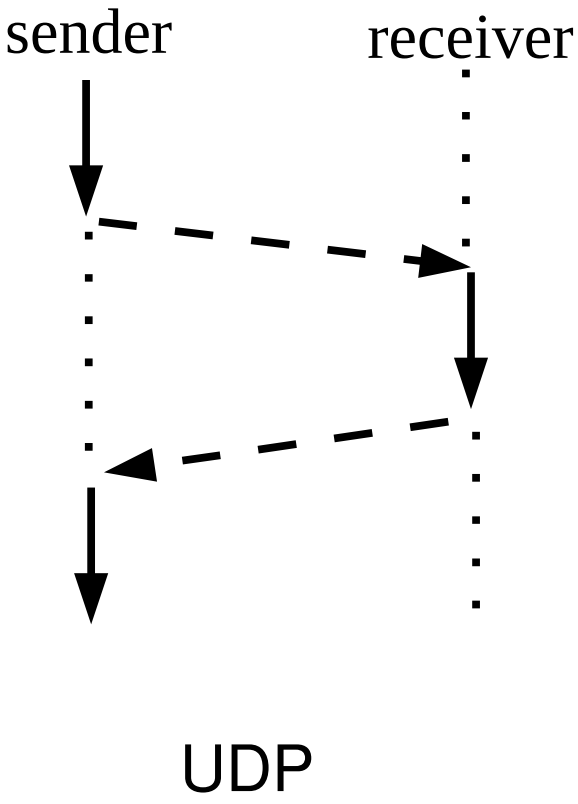

client
server
request (R)
client
server
-reply (RR)
client
server
-acknowledge (RRA)

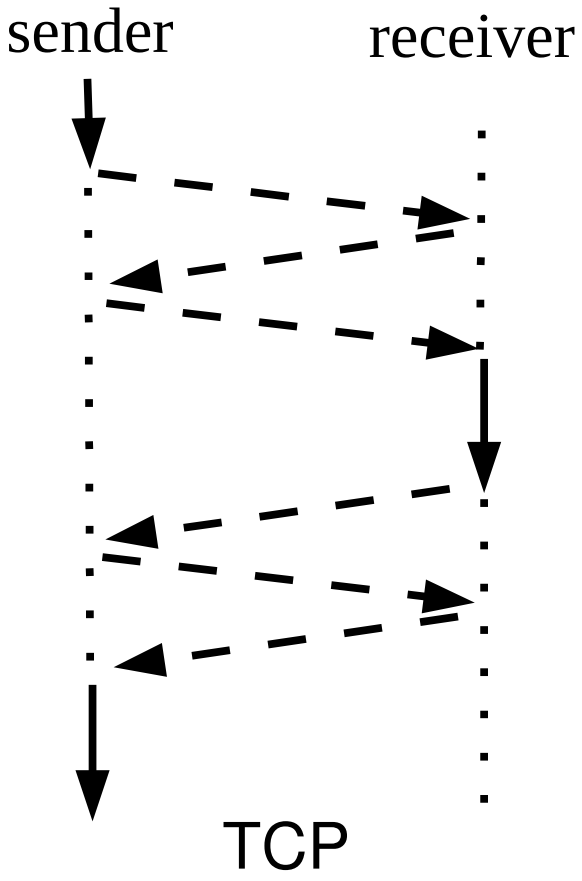
Synchronous / Asynchronous





TCP or UDP





Make it easy

- Hide the problems of distributed programming.
  - Use a regular construct in the language.
  - Why not a procedure call?

Procedure calls

- What is a procedure call:
  - find the procedure
  - give the procedure access to arguments
  - pass control to the procedure
  - collect the reply if any
  - continue execution
- What are the open issues?
- How do we turn this into a tool for distributed programming.

operational semantics


```
int x, n;
n = 5;
proc(n);
x = n;
```

```
int x, arr[3];
arr[0] = 5;
proc(arr);
x = arr[0];
```

operational semantics

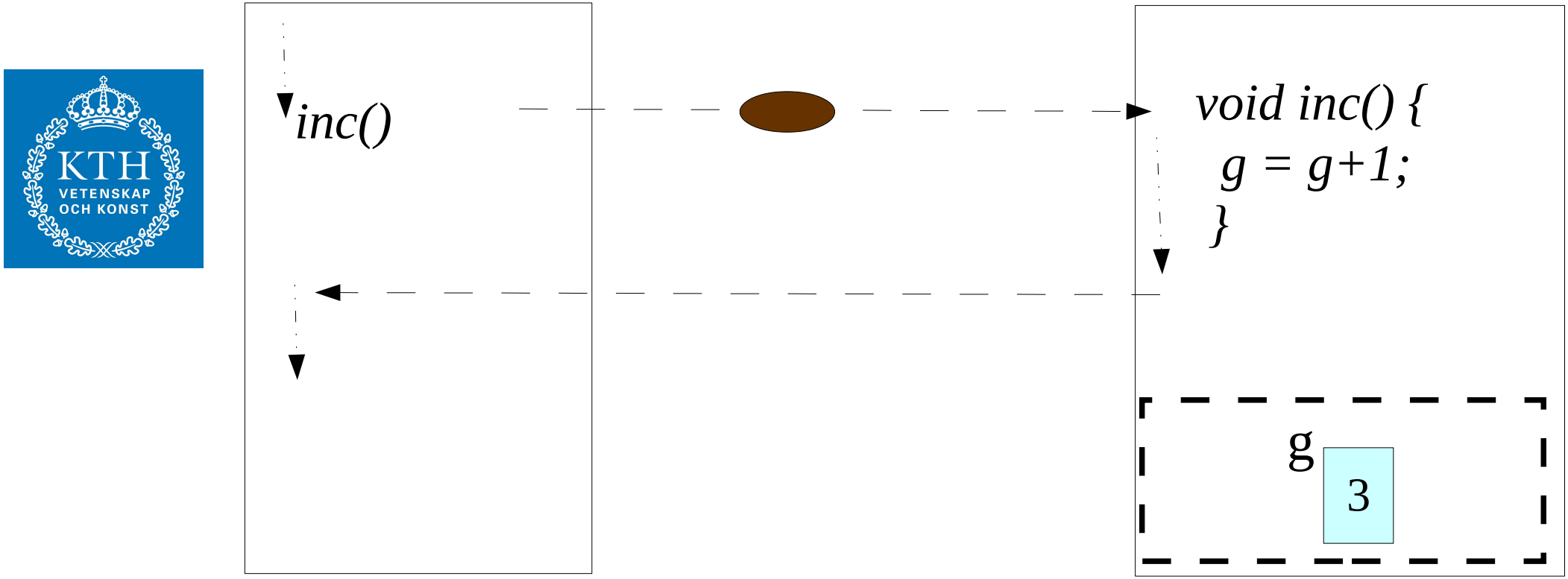
```
void proc(int x[], y[]) {
        x[0] = x[0]+1;
        y[0] = y[0] + 1;
}
```


```
int  n, arr[3];
arr[0] = 5;
proc(arr,arr);
n = arr[0];
```

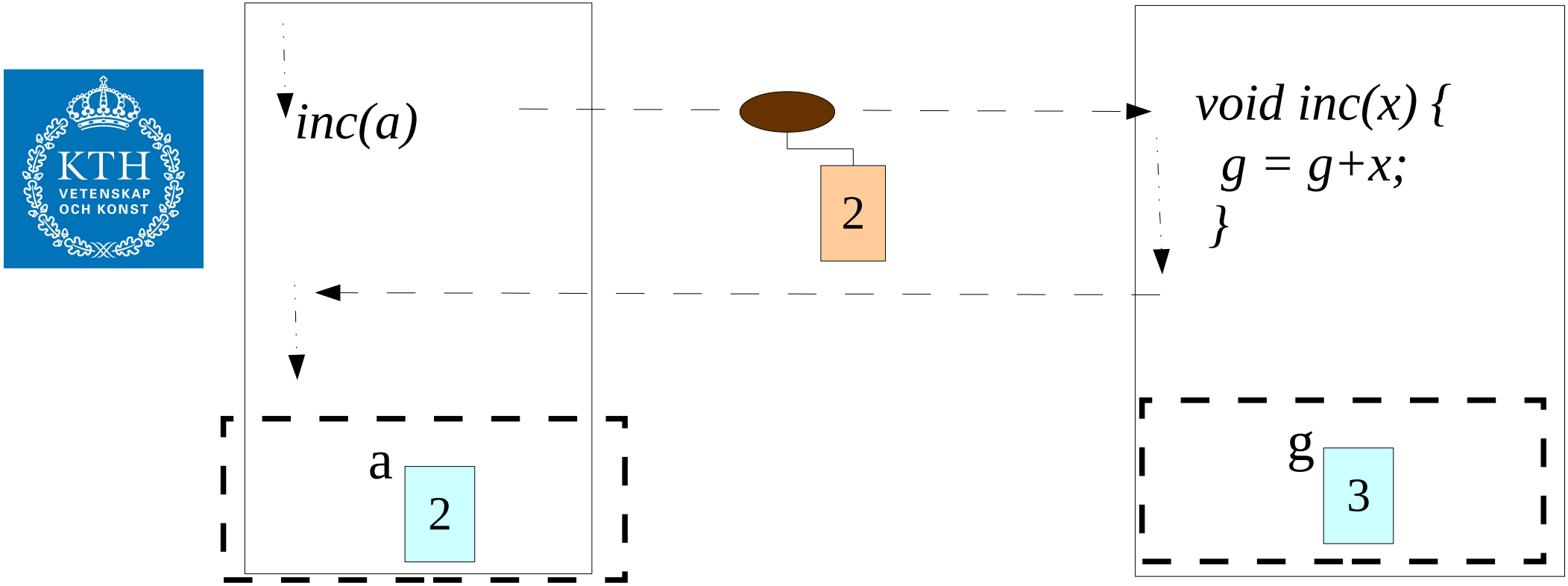
call by value / call by reference

- call by value
  - procedures are given *a copy of the datum*
- call by reference
  - procedures are given *a reference to the datum*
- confusion
  - what if the datum is a reference and we pass a copy of the datum
- why is this important?

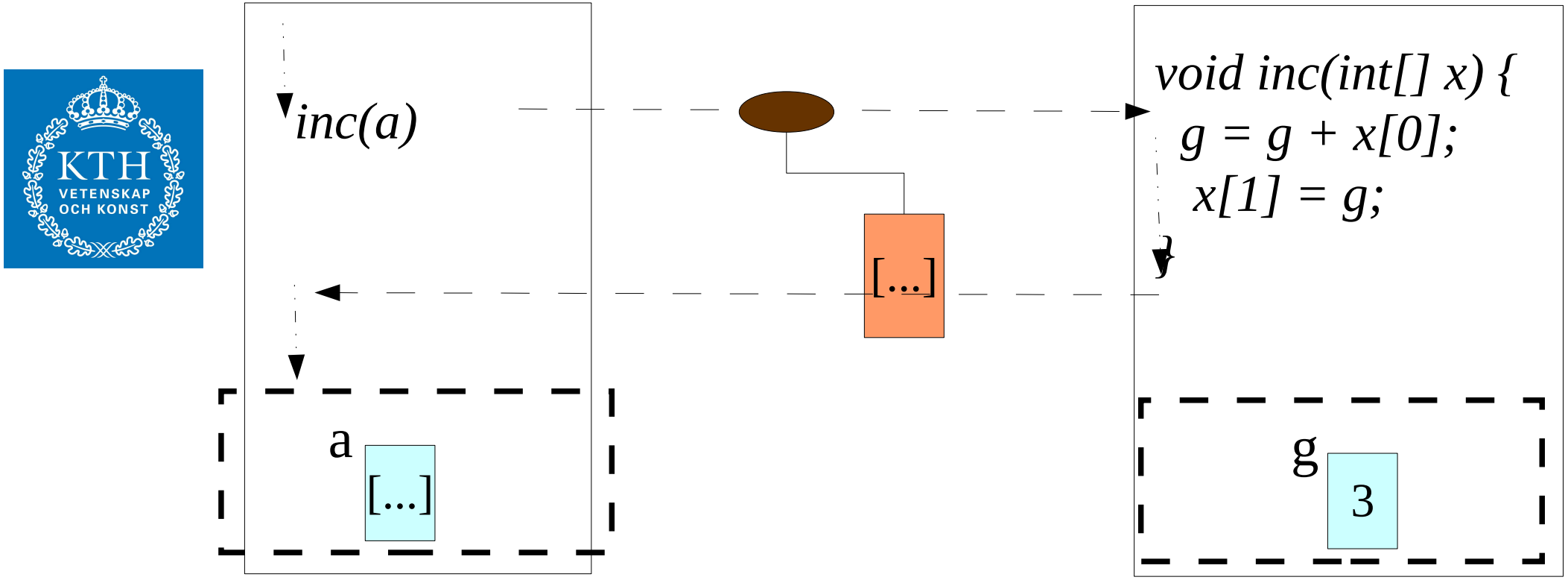
Remote procedure call



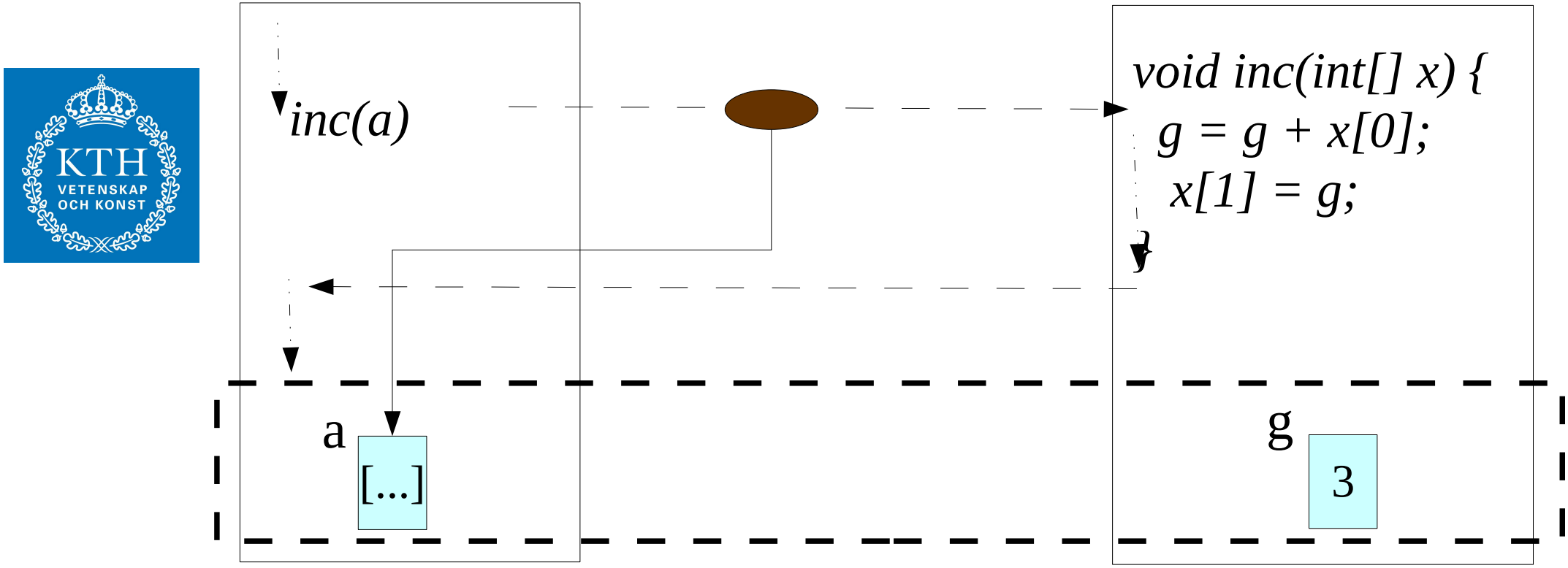
Remote procedure call



remote procedure call



distributed memory



remote procedure calls

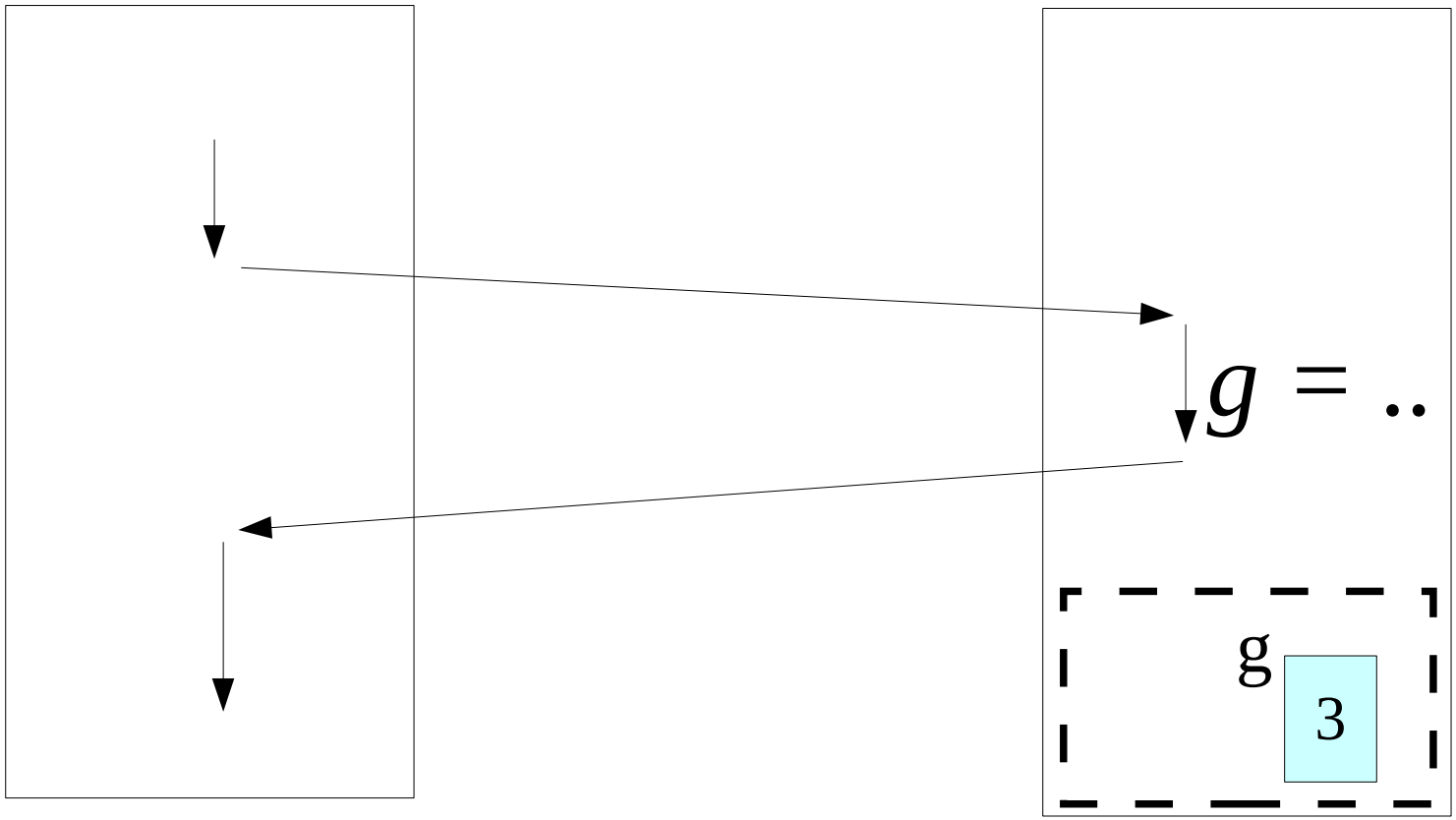
- Normally implemented using call by value since we want to avoid remote references.
- Local and remote procedure calls will have different operational semantics:
  - call by reference (local)
  - call by value (remote)
- Anything else?

operational semantics

inc(5);

...what is the value of g

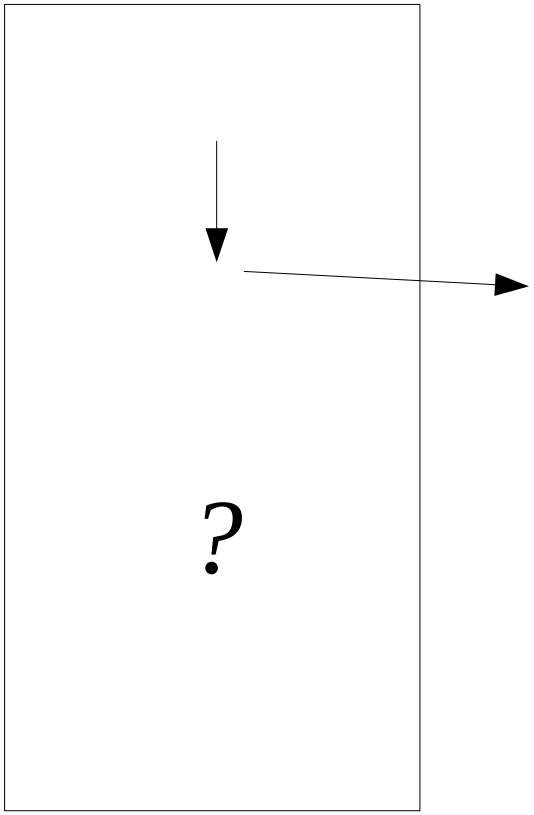
synchronous call



if everything works fine then we know that g is updated

what if

- we send a request and then hear nothing
- was the request received
- was it executed
- should we send it again?



RPC semantics

- hope-for-the-best (Erlang)
  - send the request
- at-most-once (Java RMI)
  - send the request and wait for reply
- at-least-once (Sun RPC)
  - send the request and wait for reply
  - if no reply re-send the request
- exactly-once
  - how would we do this?

Fill in the table

Result	ok	error
hope-for-the-best		
at-most-once		
at-least-once		
exactly-once		

what to do

- How can we live with ...
  - at-most-once semantics
  - at-least-once semantics
- Should the underlying middleware provide exactly-once semantics?
- “making a remote procedure call is just like making a local procedure call”
  - true or false?

what more

- How do we find the remote procedure?
- How can we describe the interface to the procedure?
- How can we represent data structures in a sequence of bytes.

finding the procedure

- How do we find a remote procedure.
- We would need to know what *node* and what *port* to contact.
- Solution: a binder
  - one known port
  - remote procedures register
  - clients can access procedures by name

describing the procedure

- Before accessing a procedure we need to know what the interface looks like.
- Interface Description Language
  - describes input and output
  - defines possible data structures
  - could be independent of programming language
  - could be used to produce stub code

marshaling

- How do we code programming language data structures?
- Network layer provides transport of sequences of bytes.
- How do we code:
  - integers,floats,boolean
  - array, structures
  - functions, procedures ??

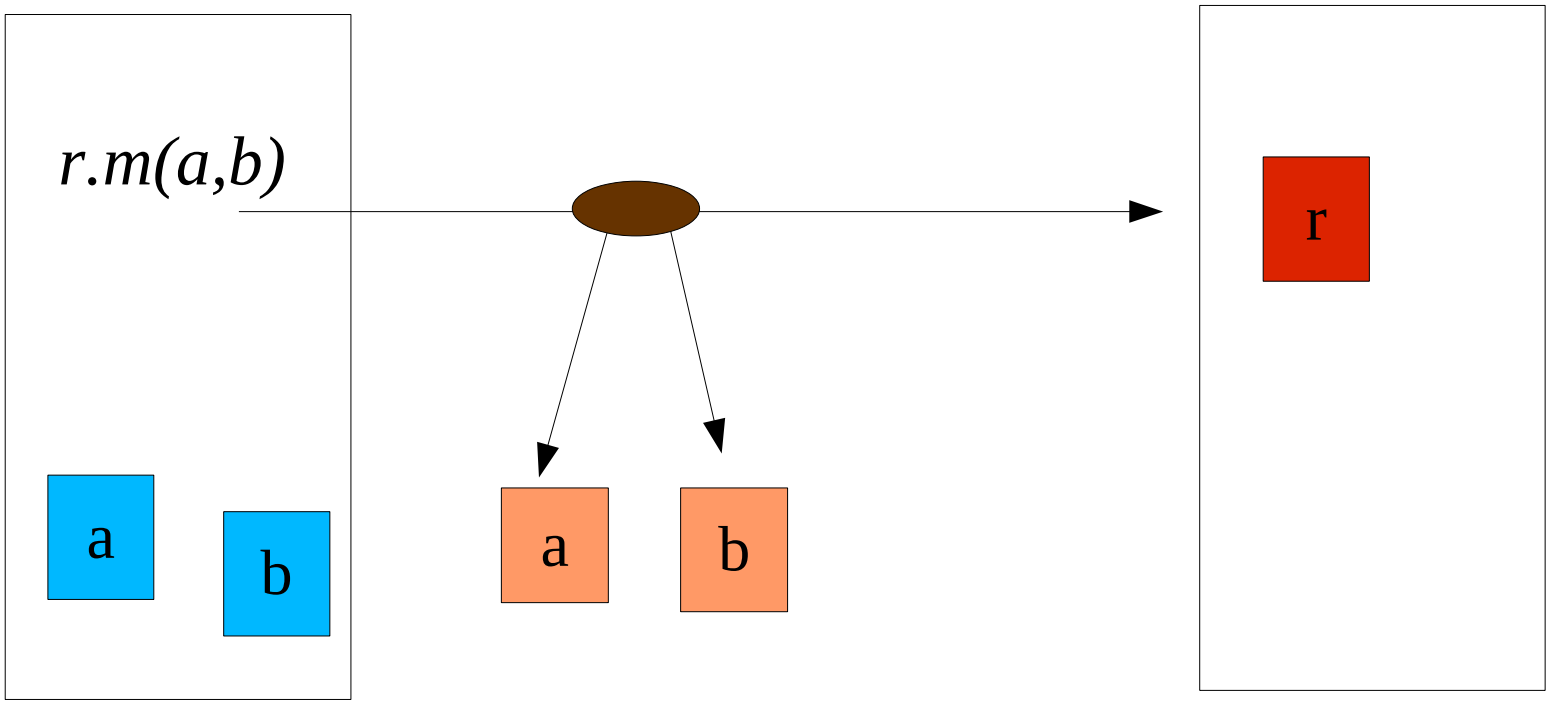
same, same but different

system:	Sun RPC	CORBA	Java RMI	WS	Erlang
binder	rpcbind	ORB	Registry	UDDI	epmd
description	XDR	IDL	*interface*	WSDL	-
marshaling	binary	binary/XML	binary	XML	binary
language	indep	indep	Java	indep	Erlang

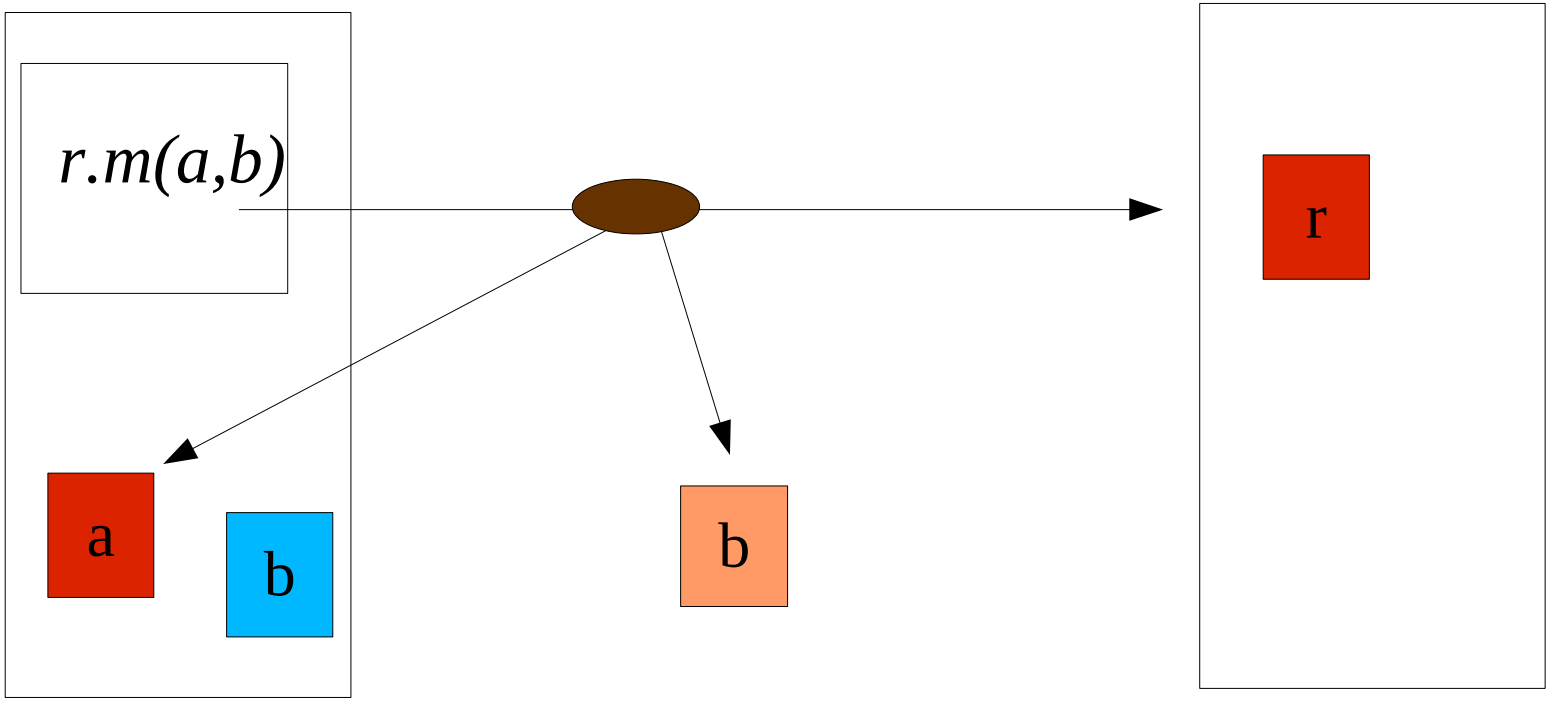
a closer look at Java RMI

- similar to RPC but
  - we now invoke methods of *remote object*
- Objects can of course be passed as arguments, how should this be done?
  - by value, a copy of the object
  - by reference, a remote reference to the object

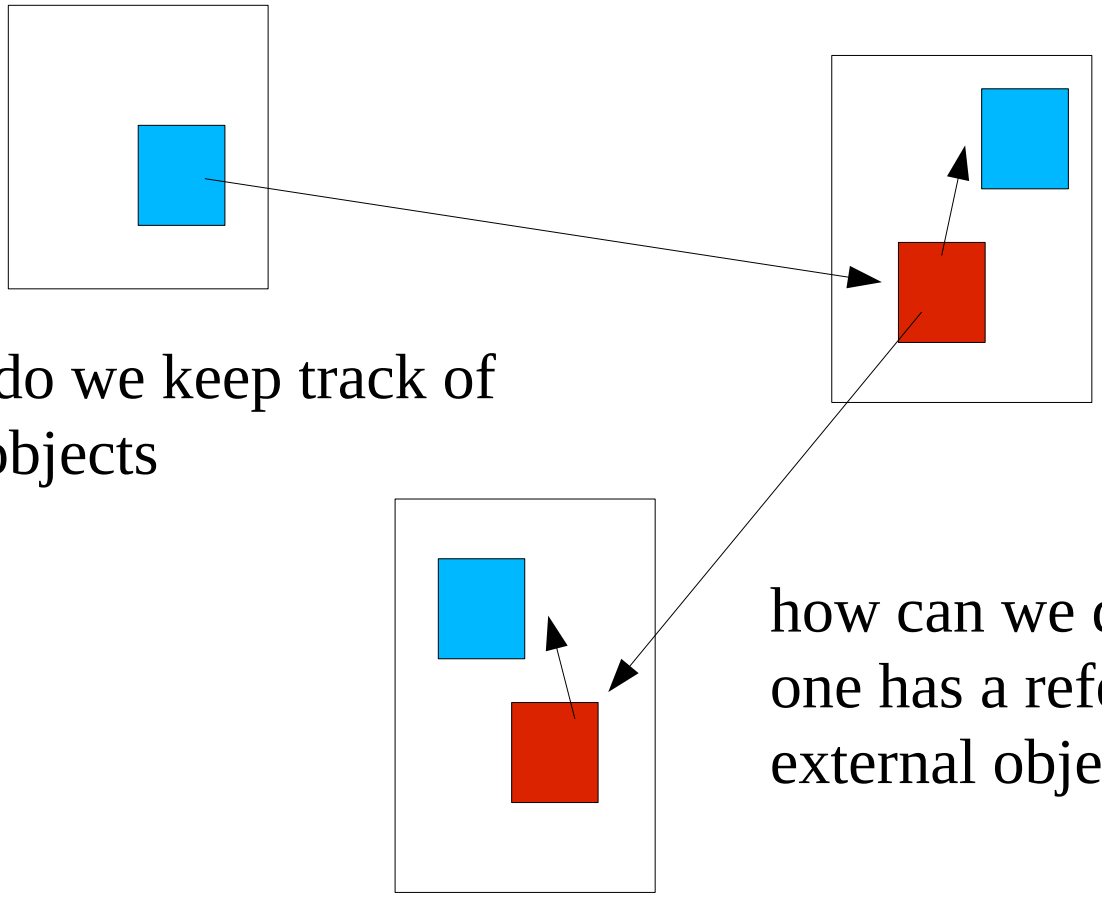
Remote method invocation



Remote method invocation



distributed garbage collection



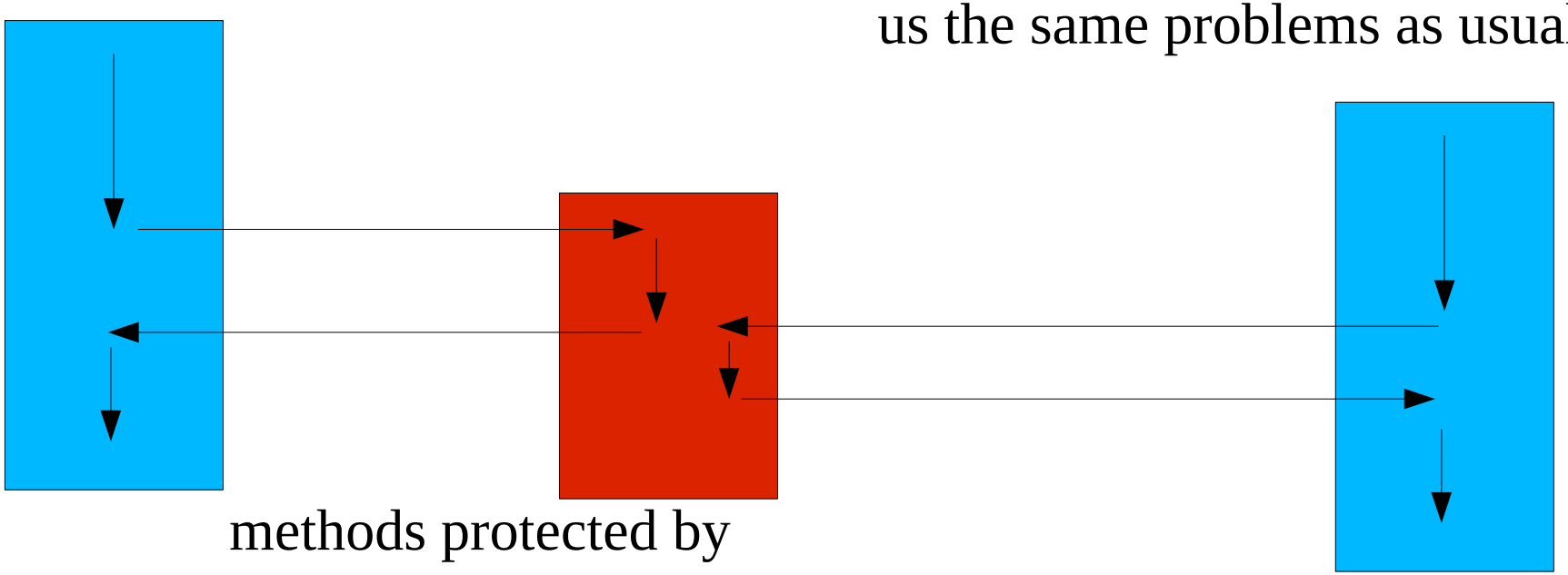
how do we keep track of live objects

how can we determine that no one has a reference to an external object

thread of control

a method invocation is within one thread of control

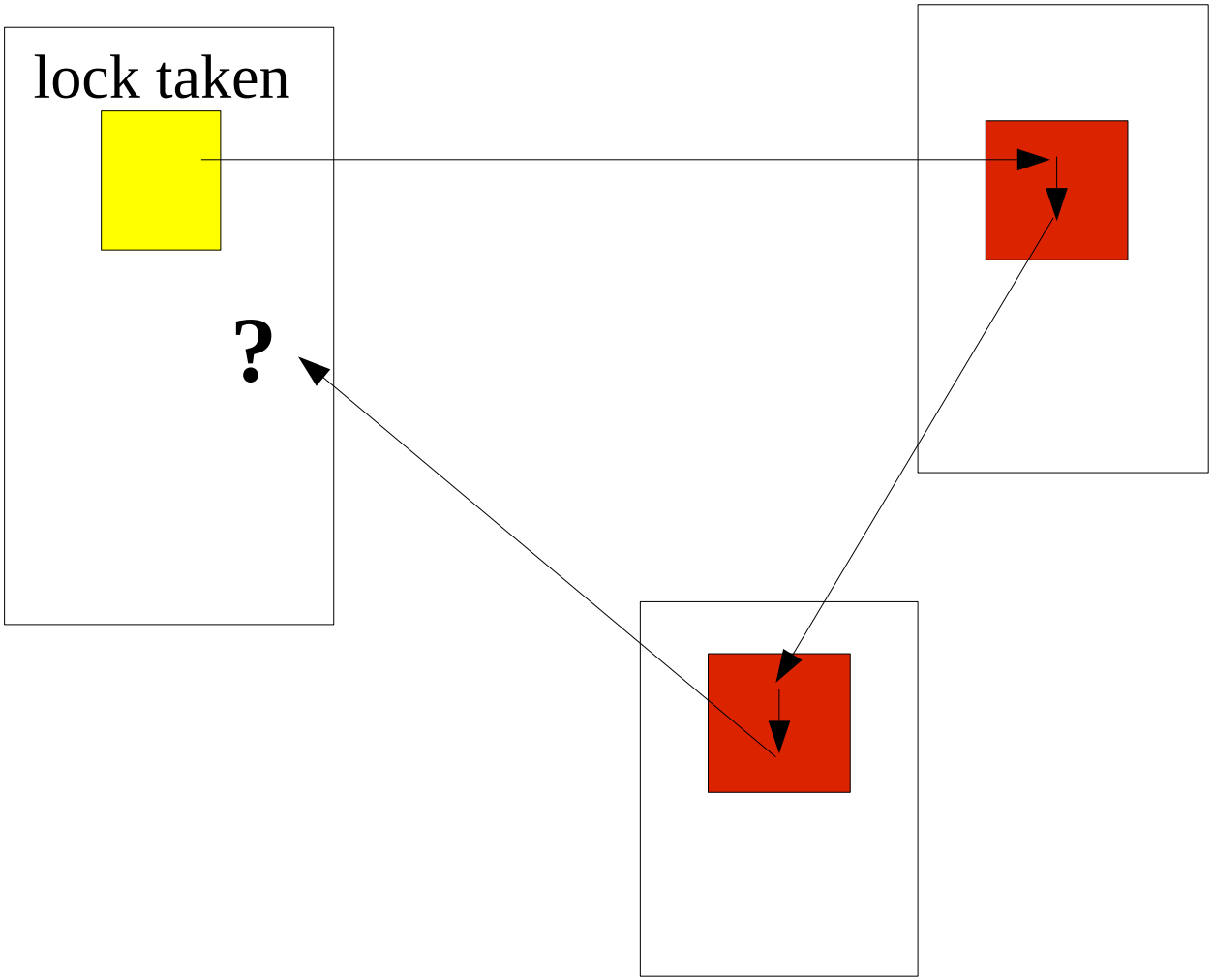
having multiple threads gives us the same problems as usual



methods protected by locks; “synchronized”

a thread is allowed to enter the same object if it holds the lock

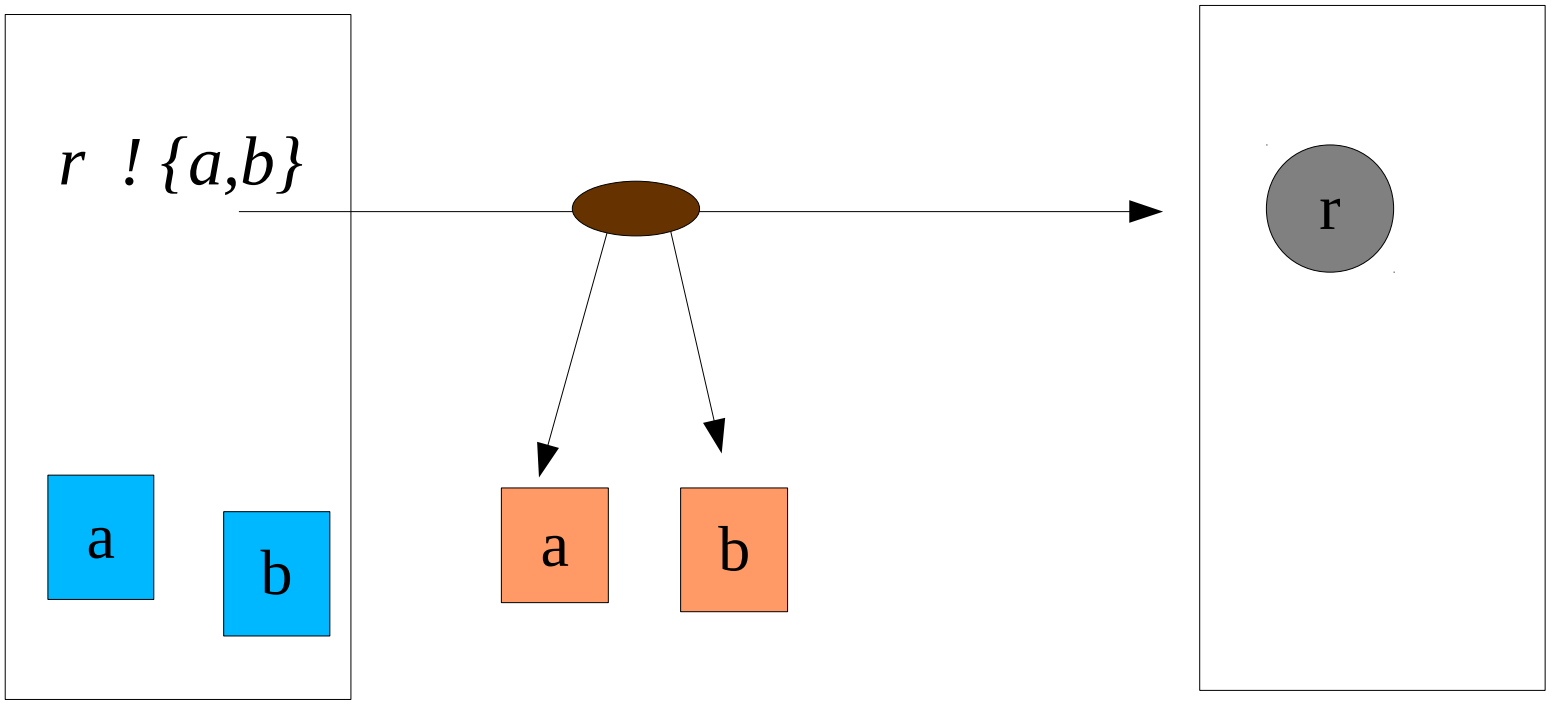
dead lock



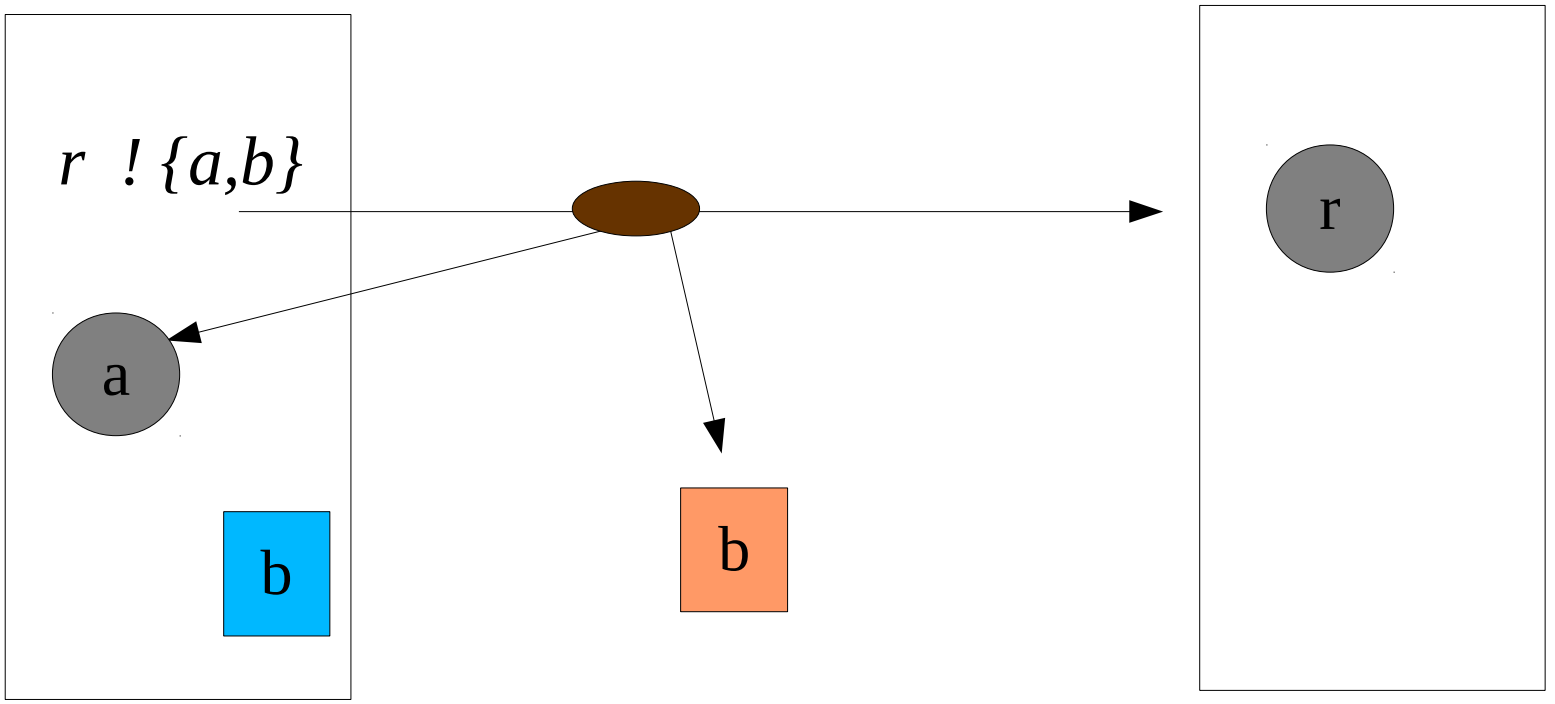
RPC and RMI

- RPC nor RMI is access transparent
  - semantics
  - error behavior
  - memory management
  - dead lock

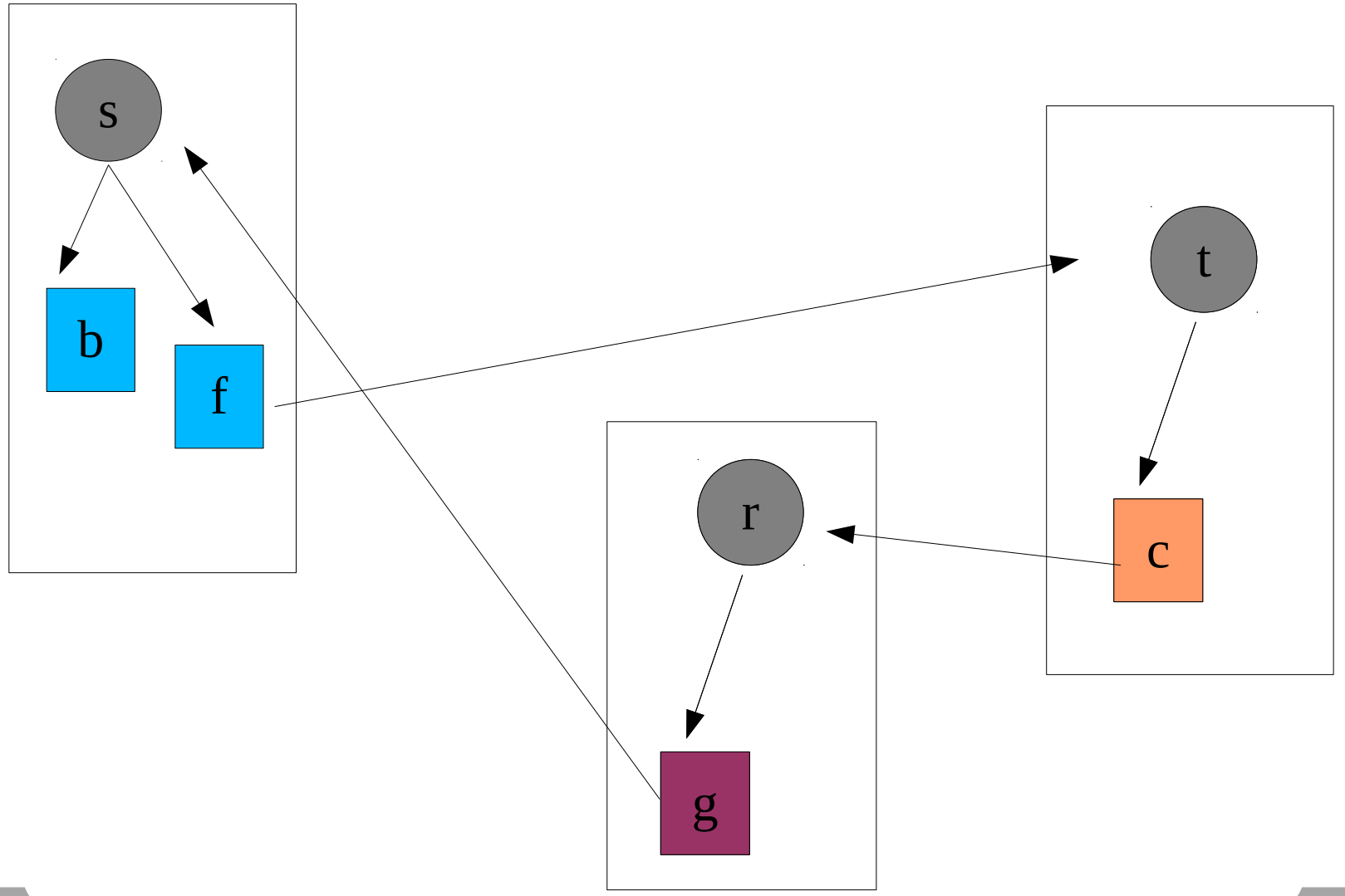
Erlang – copy data



Erlang – remote pid



Erlang – garbage collection



Erlang

- The distributed programming model is very close to the local programming model.
  - Data is always local.
  - Messages are always copied.
  - Processes live independent of external references.
- You have to understand asynchronous communication.