CSE 515 — Winter 2004

Mobile Objects in Emerald

Lecture 8

What's the Theme?

Interface:

- Do it right!
- Clean, simple abstractions for the programmer

Implementation:

- Careful Engineering Compromises
- Be clear about your goals!
  - local performance is more important than remote

Example

Interface

– Every object is represented by a globally unique object reference

var x: point ← point.at[3, 4]

Implementation

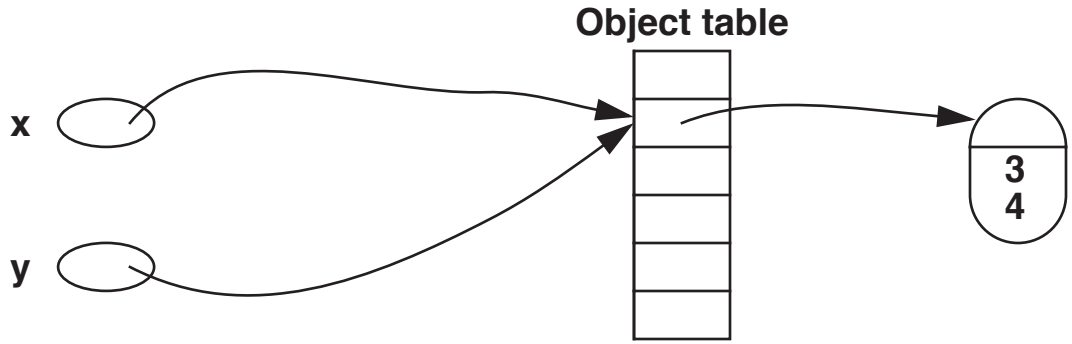
– There are actually different representations for *x*

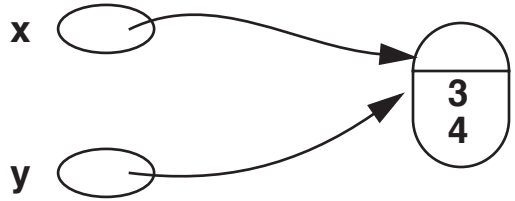
- on different machines, and
- for different categories of object

An Emerald Object

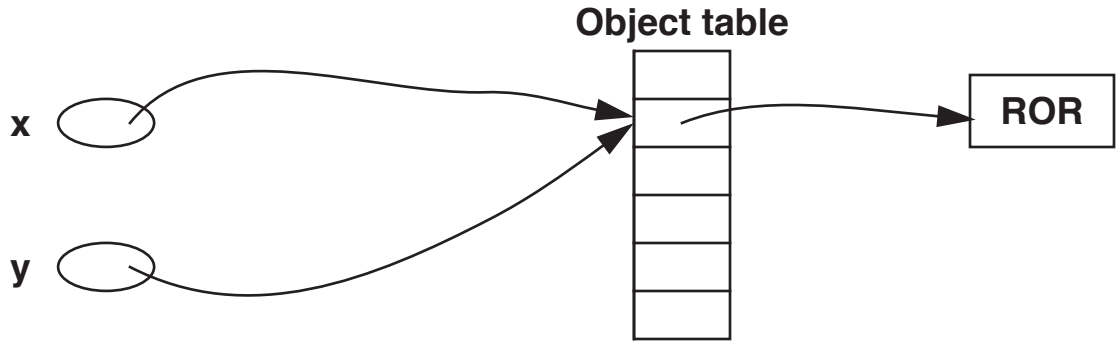
```
object aPoint
        var x: integer ← 3
        var y: integer ← 4
        var c: color ← color.blue[]
        op x [] → [r: integer];  r ← x  end x
        op y [] → [r: integer];  r ← y  end y
        op color [] →[r: color]; r ← c end color
        …
end aPoint
```

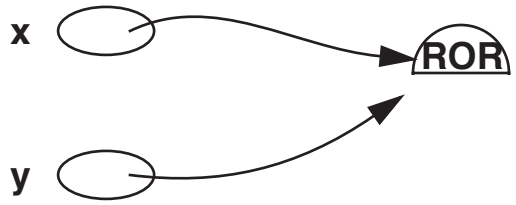
Object Table *vs.* Direct Pointer





Move point object to Remote machine



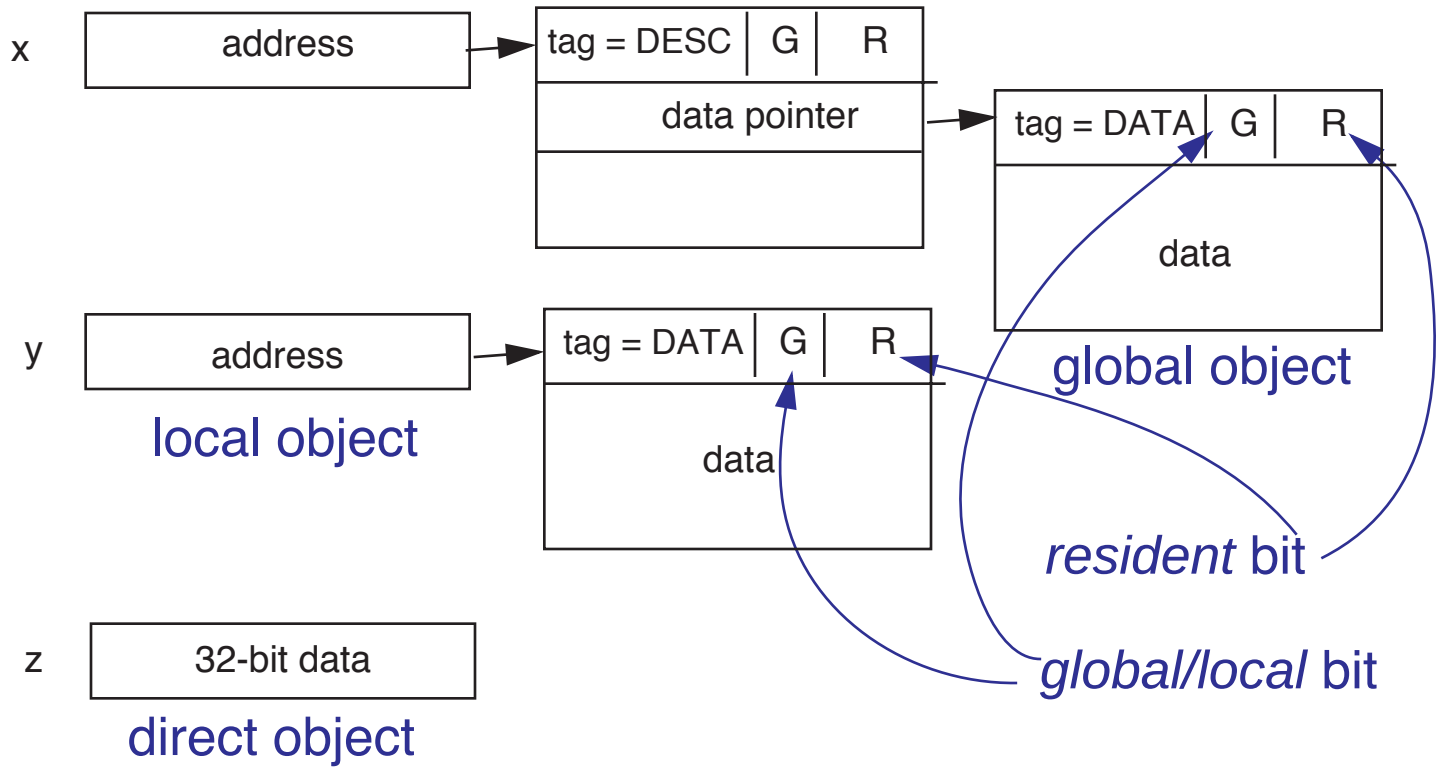


"Morph" point object into ROR

Representation of Objects

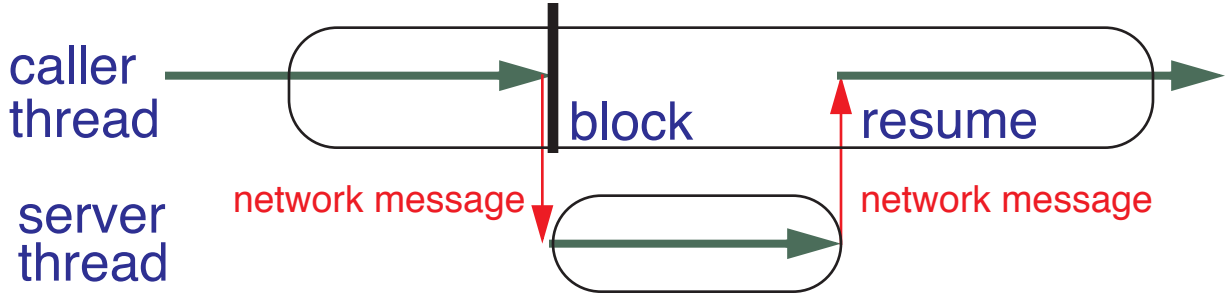
- global object
  - object can be moved
  - can be referenced from anywhere
  - location check required on message send
- local object
  - referenced only from inside its creator
  - heap allocated, message send by procedure call
- direct object
  - date represented “inline”

After [Jul88] Figure 3



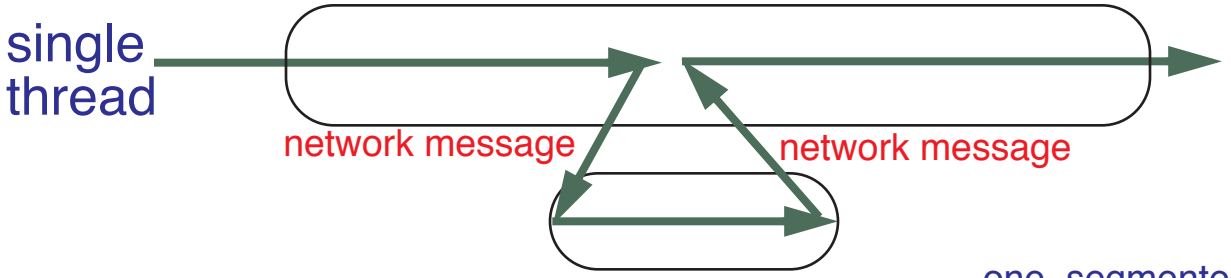
Thinking about RMS

Conventional



multiple stacks, one per machine

Emerald



one, segmented stack

Finding Objects

Two alternatives

- Keep track of where object are
- Find them when you need them

- How to choose?

Forwarding Pointers: when do they fail?

Pointer Translation

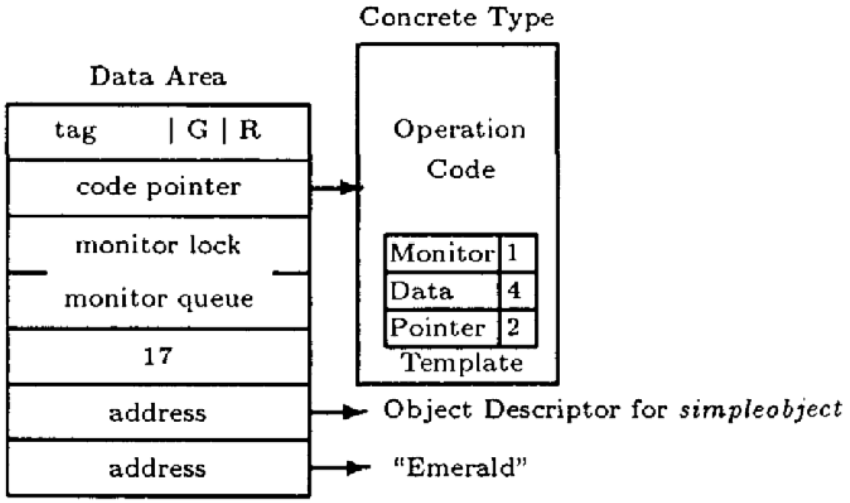
Multiple representations for Object Reference

- System must translate from one to the other
- Object Refs were local pointers
  - meaningless on remote side
  - append a “translation table” local pointer → GUID
- Templates are used to find pointers

[Jul88] Figures 4 & 5

```
const simpleobject ← object simpleobject
    monitor
        var myself : Any ← simpleobject
        var name : String ← "Emerald"
        var i : Integer ← 17
        operation GetMyName → [n : String]
            n ← name
        end GetMyName
        :
    end monitor
end simpleobject
```

Fig. 4. Simple Emerald object definition.



Micro benchmark Performance

Table II. Remote Operation Timing

Operation type	Time/ms
Local invocation	0.019
Kernel CPU time, remote invocation	3.4
Elapsed time, remote invocation	27.9
Remote invocation, local reference parameter	31.0
Remote invocation, call-by-move parameter	33.0
Remote invocation, call-by-visit parameter	37.4
Remote invocation, remote reference parameter	61.8
System & Network round trip time	24.5

Mail System Performance

Table IV. Mail System Traffic

	Without mobility	With mobility
Total elapsed time (in seconds)	71	55
Remote invocations	1,386	666
Network messages sent	2,772	1,312
Network packets sent	2,940	1,954
Total bytes transferred	568,716	528,696
Total bytes moved	0	382,848

- How important is mobility in a distributed object system?