Foundations of Distributed Systems

José Orlando Pereira

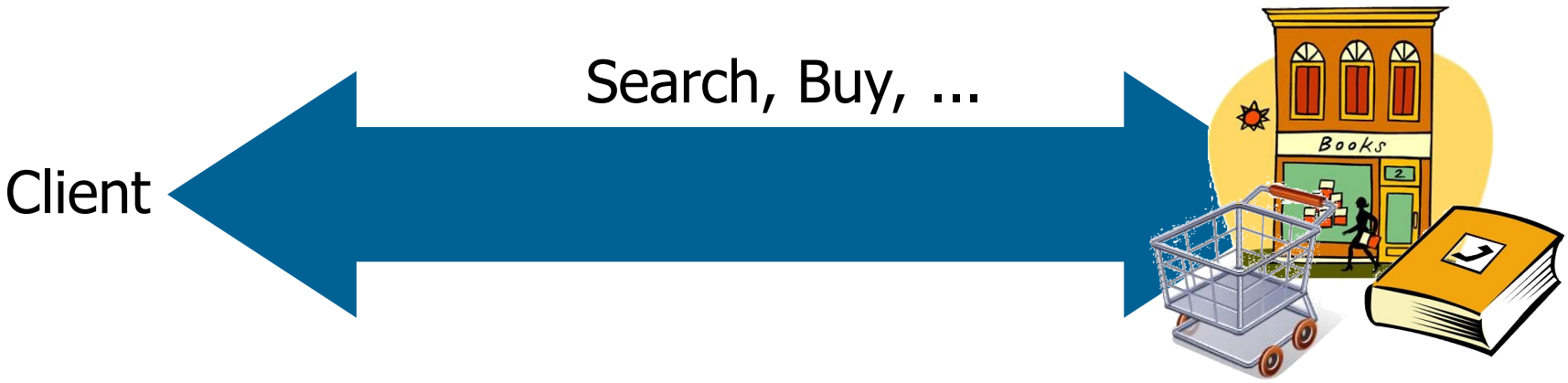
HASLab / Departamento de Informática
Universidade do Minho

2019/2020

Motivation

- A distributed program must deal with:
  - Business logic issues
  - Distributed systems issues
- Enterprise middleware:
  - Focus on business logic
  - How to make distribution transparent?

Example: Book store



Example: Book store

- Book details
  - ISBN, author, ... and stock
- Store operations:
  - Search books
- Shopping cart operations:
  - Add to cart
  - Buy content of cart (check-out)

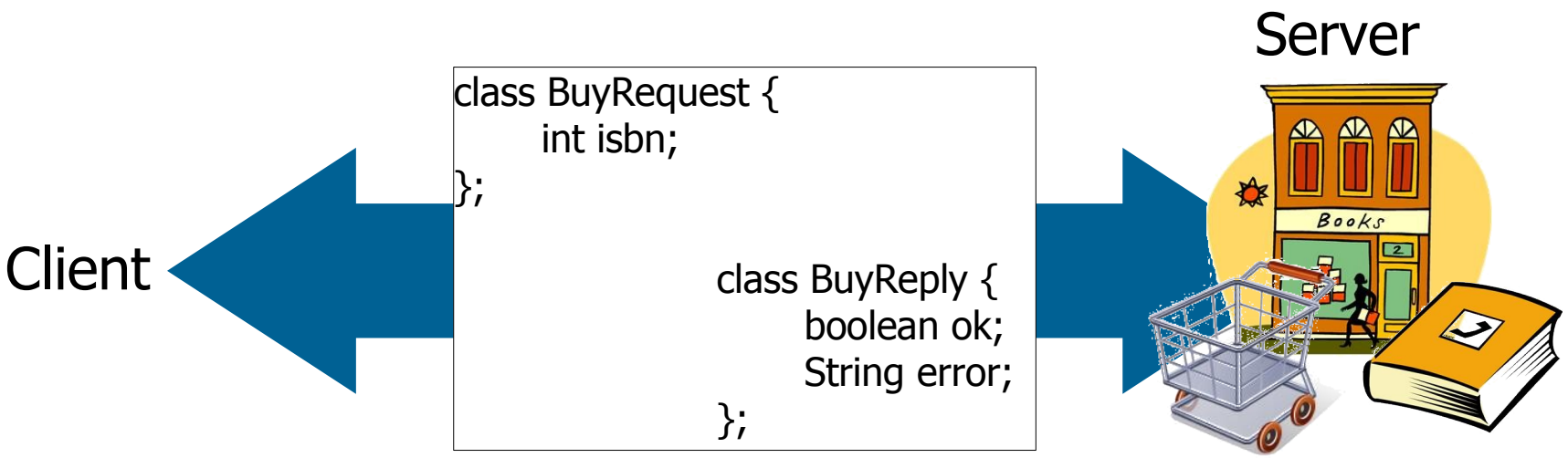
Example: Book store

- Model as Java interfaces/classes:
  - Book:
    - Attributes: ISBN, title, author, ...
  - Store:
    - Persistent book collection, search logic
  - Cart:
    - Transient book collection, check-out logic
- Use by a typical client:

```
Store s = ... ;
Cart c = ... ;
c.add(s.search(...));
c.add(s.search(...));
c.buy();
```

Client/server

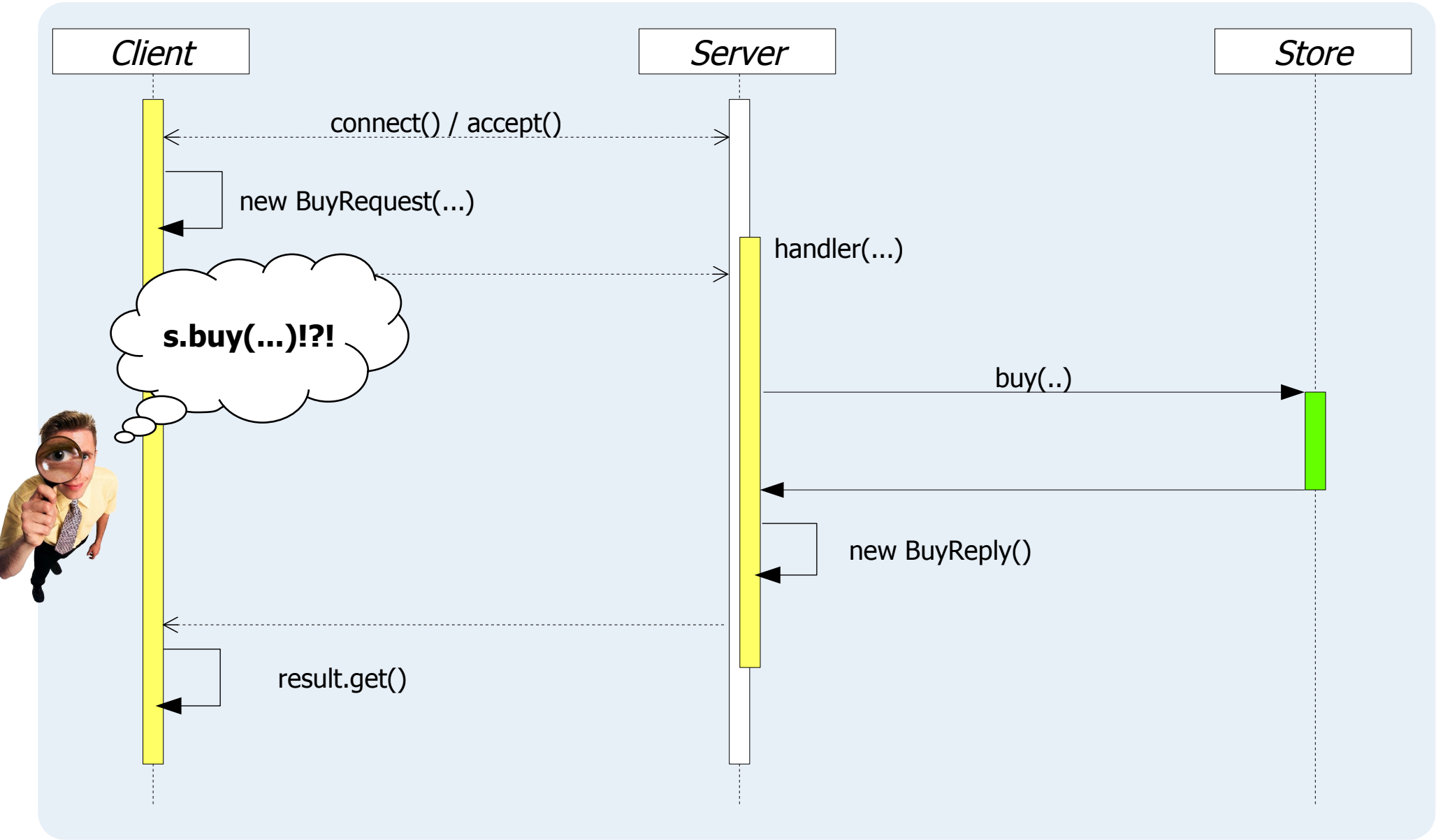
- Messages for each request and reply
- Building blocks: Communication and marshalling



Client/server

- Client sends request and waits for reply
- Server waits for request, executes it and sends reply
- Pairing replies to requests:
  - Only one operation pending in the connection
  - Tag requests, copy tag to reply, lookup request when reply arrives

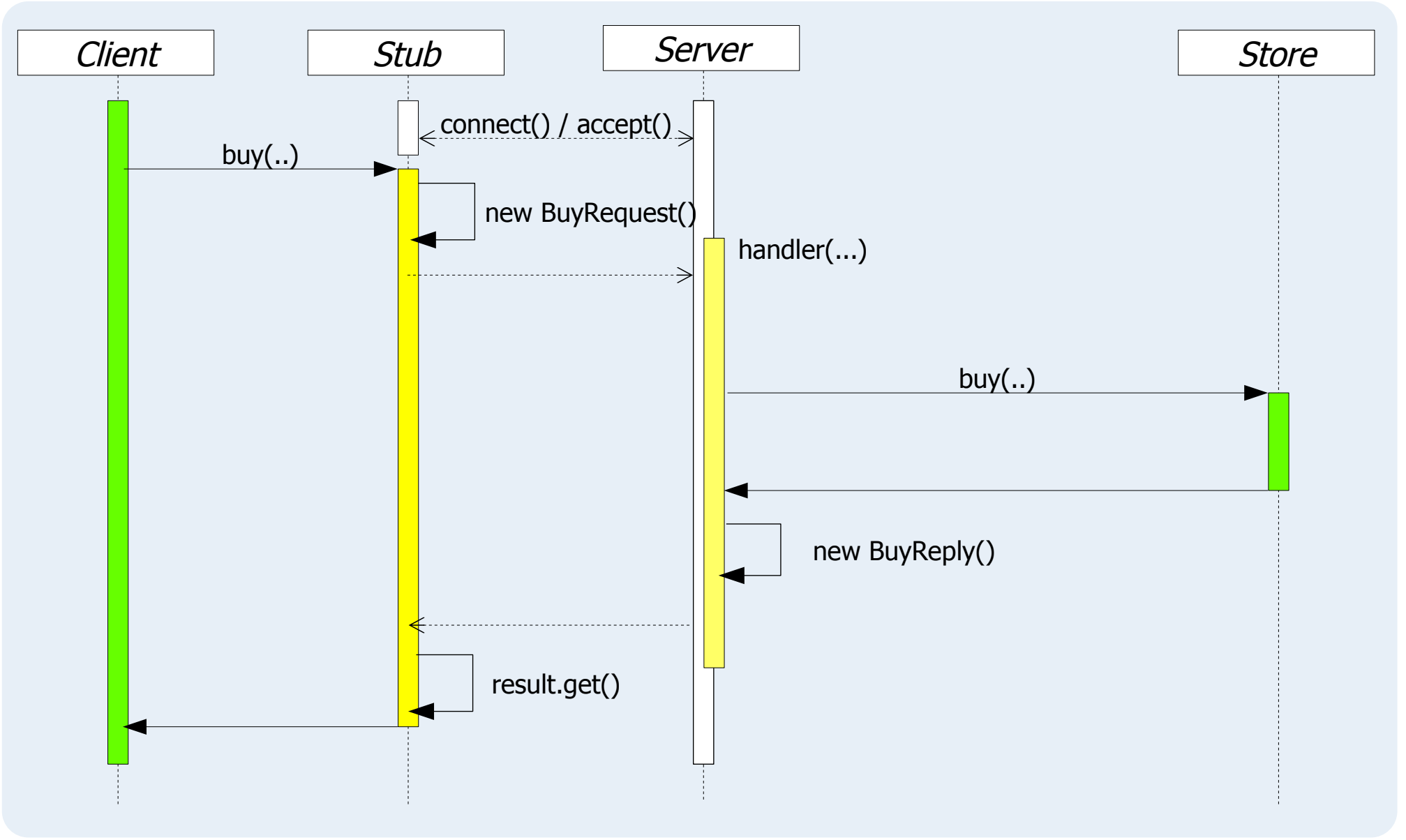
Example



Challenges

- Client aware of distribution
- Flat interface

Refactoring to stub



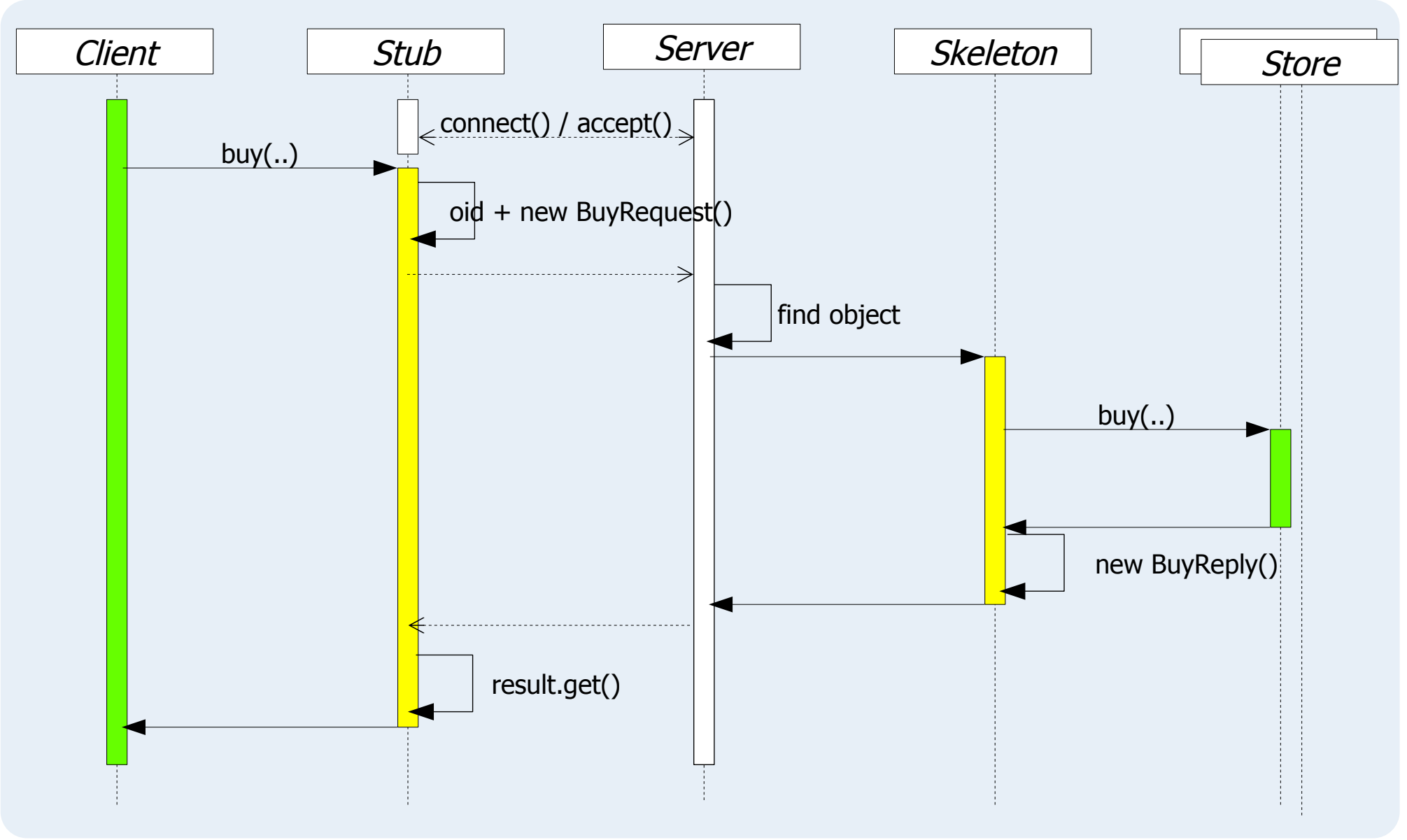
Challenges

- ~~Client aware of distribution~~
- Flat interface

Object identifier

- In the client, add an object identifier to requests
- In the server:
  - Keep a map of identifiers to objects
  - Lookup target object
  - Invoke method on target object

Refactoring to stub + skeleton



Object Reference

- Information necessary to create a stub:
  - Location (e.g., host and port)
  - Identifier
  - Object type
- "Reference" in a distributed system

Object Reference

- Created when exporting a servant object:
  - Get location from server
  - Create identifier and add to map
- Embedded in both stubs and servants
- Sent automatically when:
  - Returning a remote object (stub or servant)
  - Passing a remote object (stub of servant) as a parameter

Distributed Objects Runtime

- Main operations:
  - Export object: creates object reference and adds object to server map
  - Import object: creates stub for object reference
- Internally:
  - Handles connections
  - Keeps objects in server map

Garbage collection

- Locally referenced objects are never discarded by the garbage collector
- When exported, objects are referenced by the server map:
  - Exported objects would never be discarded!
- Synchronous system: use timed leases
- Asynchronous system: observe object references

Summary

- Distributed objects hide distribution in object-oriented systems
- Not completely hidden:
  - Object creation
  - References vs value
- Not hidden and not solved:
  - Atomicity regarding concurrency and faults