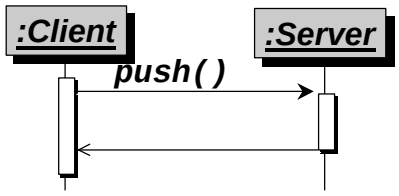
Advanced Communication

Motivation

- Requests we have seen have
  - two parties (client and server object0
  - trigger execution of one operation
  - have at-most-once reliability
- Other forms of requests are useful
- We therefore look at different forms of
  - Requests Synchronization
  - Request Multiplicity
  - Request Reliability

Request Synchronization

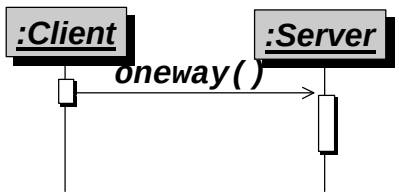
- *OO-Middleware: synchronous requests.*
- *Synchronous requests might block clients unnecessarily*



- *Non-synchronous forms:*
  - *oneway requests*
  - *deferred synchronous requests*
  - *asynchronous requests*

Oneway Requests

- *Return control to client as soon as request has been taken by middleware*
- *Client and server are not synchronized*
- *Use if*
  - *Server does not produce a result*
  - *Failures of operation can be ignored by client*



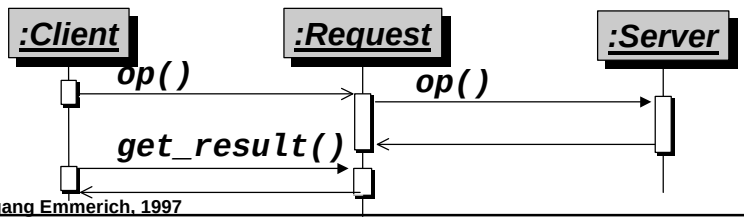
Oneway requests in CORBA

- Declared statically in the interface definition of the server object
- IDL compiler validates that
  - operation has a void return type
  - does not have any out or inout parameters
  - does not raise type specific exceptions
- Example:

```
interface Team {
  oneway void mail_timetable(in string tt);
};
```

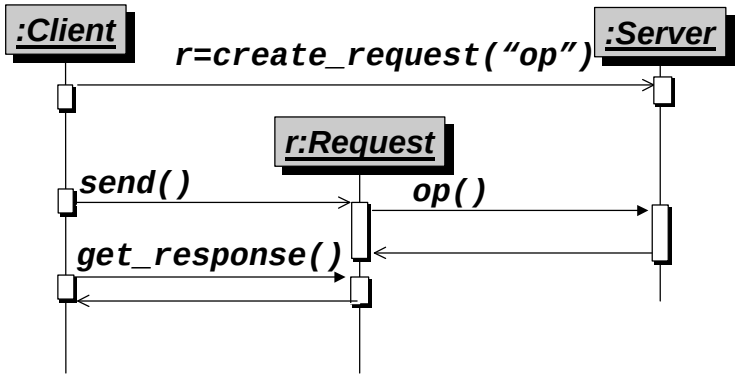

Deferred Synchronous Requests

- Return control to client as soon as request has been taken by middleware
- Client initiates synchronization
- Use if
  - Requests take long time
  - Client should not be blocked
  - Clients can bear overhead of synchronization



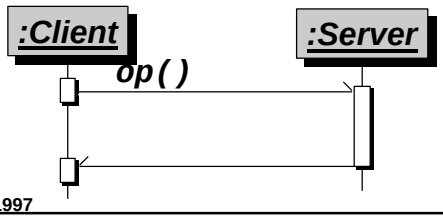
CORBA Deferred Sync. Requests

- Determined at run-time through Dynamic Invocation Interface.
- By invoking `send()` from a `Request` object.



Asynchronous Requests

- Return control to client as soon as request has been taken by middleware
- Server initiates synchronization
- Use if
  - Requests take long time
  - Client should not be blocked
  - Server can bear overhead of synchronization



Asynchronous Requests by Threads

- *Client has interface for callback*
- *Perform request in a newly created thread*
- *Client continues in main thread*
- *New thread is blocked*
- *Requested operation invokes callback to pass result*
- *New thread dies when request is complete*

Example in Java

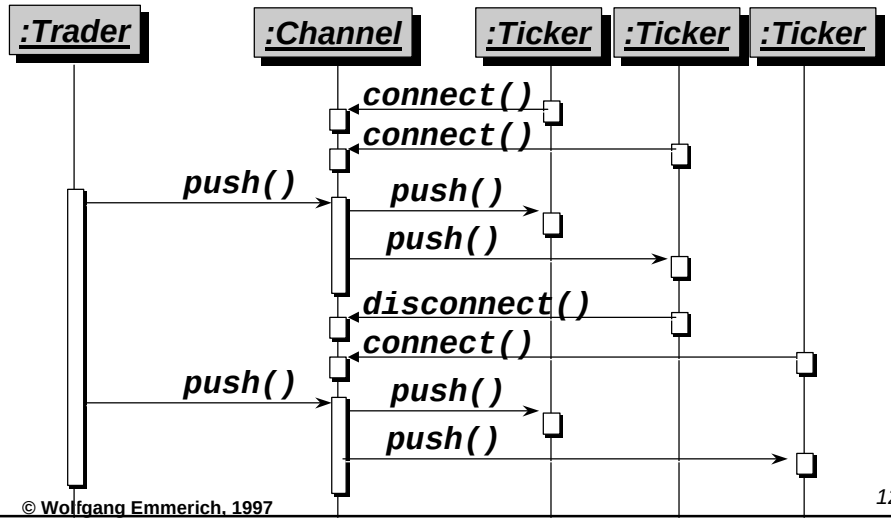
```
interface Callback {
  public void result(String s);
}
class PrintSquad implements Callback {
 public void Print(Team team, Date date){
  A=newAsyncReqPrintSquad(team,date,this);
  A.start()
 }
 public void result(String s){
  System.out.print(s)
 }
}
class AsyncReqPrintSquad extends Thread {
 Team team; Date date; Callback call;
 AsyncReqPrintSquad(Team t, Date d, Callback c) {
  team=t;date=d;call=d;
 }
 public void run() {
  String s=team.AsString(date);
  call.result(s);
 }
}
```

Request Multiplicity

- **OO Middleware: peer-to-peer requests**
  - Two components: client and server
  - One operation execution
  - Non-anonymous
- **Other forms:**
  - More than two components (group requests)
  - More than one operation (multiple requests)

Group Communication

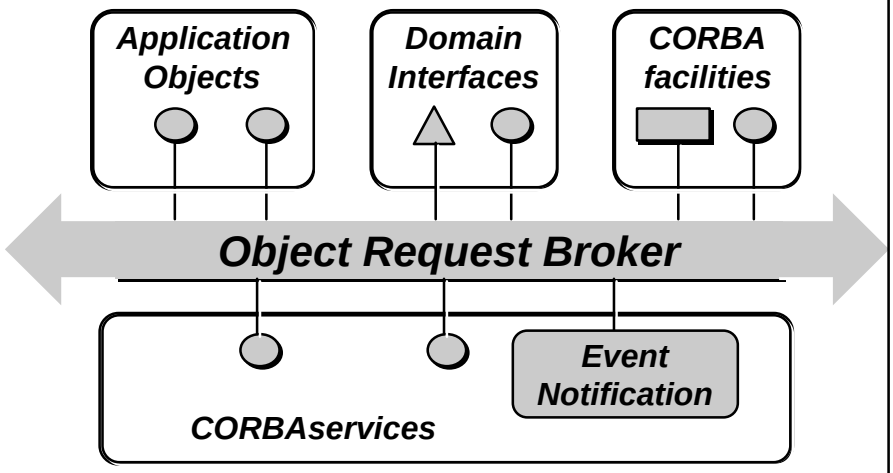
- **Example: Stock Exchange Ticker**



Group Communication Principles

- **Group communication informs a group of components about a particular event.**
- **Two roles:**
  - **Event producer**
  - **Event consumer**
- **Producers and consumers do not know each other.**
- **Two forms of request:**
  - **push-type: producer initiates communication**
  - **pull-type: consumer initiates communication**

CORBA Event Notification Service



Multiple Requests

- *Triggers n operation executions in one request.*
- *Usually deferred synchronous*
- *Defined at run-time.*
- *Advantages:*
  - *Smaller overhead on client side*
  - *Process results as they become available*

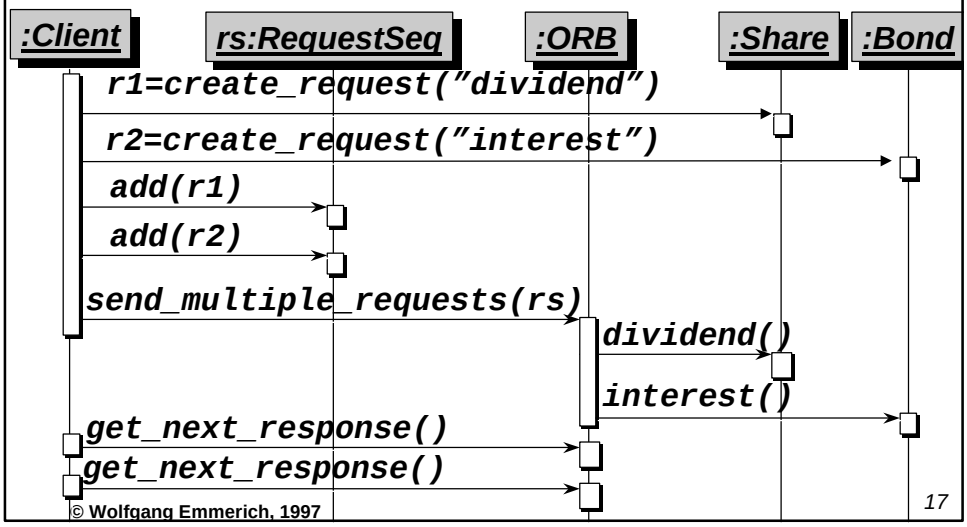
Multiple Requests in CORBA

- *Dynamic Invocation Interface supports multiple deferred synchronous requests*

```
module CORBA {
 interface ORB {
  typedef sequence<Request> RequestSeq;
  Status send_multiple_requests (
     in RequestSeq targets
  )
  Status get_next_response(in Flags f);
 };
};
```

Multiple Request Example

- Revenue from a portfolio of assets



Request Reliability

- OO Middleware: At-most-once semantics
- Other degrees of request reliability:
  - Peer-to-peer
    - exactly once
    - atomic
    - at-least-once
    - maybe
  - Group and multiple requests
    - k-reliability
    - totally ordered
    - best effort