RelayMQ++

A Quick Introduction

Christian Scheideler, Alexander Setzer, Thorsten Götte

WS 16/18

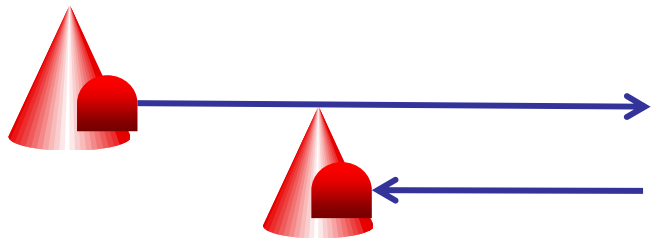
Advanced Distributed Algorithms and Datastructures
Paderborn University

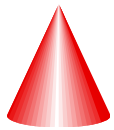
Contents

Basic Concepts

TCM Model

Abstraction:



- Processes  (at AL and TCL)
- Relays  (managed by the TCL)
- Processes have references (→) to local relays

Main Classes and Methods

RelayRef

```
1 typedef RelayRef
```

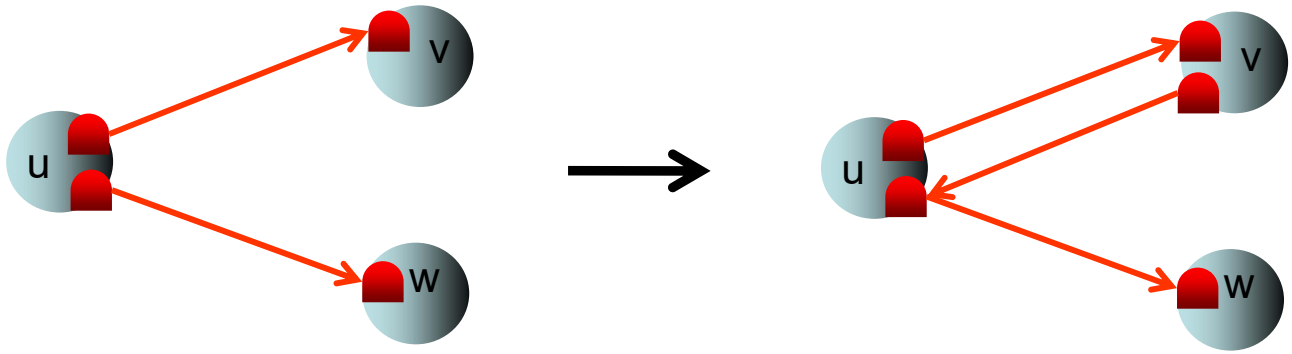
- Can send messages to a Subject
- Has no direct c'tor
- Can be shared via sending a message
- Not intended to be subclassed

RelayRef->Send

```
/**
 * Sends a string and references
 *
 * Returns true on success
 */
bool Send(string msg,
          vector<RelayRef> refs);
```

TCM Model

Recall the safe introduction rule:



Instead of introducing w to v, u can only introduce its relay to w to v.

RelayRef->Send

```
RelayRef to_v = ...;
RelayRef to_w = ...;

to_v->Send("INTRODUCE",
           {to_w});
```

RelayRef->CompareTo

```
/**
 * Checks if Relays have the
 * same target
 */
bool CompareTo(RelayRef otherRef);
```

RelayRef->Close

```
/**
* Closes Relay,
  s.t. no more messages are
  received or forwarded
*/
void Close();
```

Subject

```
class Subject: public Relay
```

- Can receive messages
- Can create additional incoming relays
- Monitors its Relays
- Intended to be subclassed for custom behavior

Subject->onMessage

```
/**
 * Exectuted whenever there is
 * a message
 * for the Subject
 */
virtual bool
onMessage(RelayRef receiver,
          string msg,
          vector<RelayRef> refs);
```

Subject->onTimeout

```
1   /**
     * Executed if
3    * a) there is no I/O
     * b) at least 1s has passed
5    */
    bool onTimeout();
```

Subject->onClose

```
/**
* Called if the Relay is closed
   remotely
*/
void onClose(RelayRef);
```

Subject->Stop

```
/**
* Stops the Subject.
*
* Note: Make sure that
* all Relays are closed !!!
*/
void Stop();
```

Subject->CreateNewIncoming

```
/**
* Create new IncomingRelay for
   this Subject
*/
RelayRef CreateNewIncoming();
```

Note: This c'tor makes sure that messages are received

ApplicationContext

```
1  class ApplicationContext;
```

- Handles I/O in the background
- Calls onMessage and onTimeout
- Manages your Subjects

ApplicationContext::Create

```
/**
 * Creates a new Subject of class T
 * and passes Args to its c'tor
 */
template<class T, typename... Args>
RelayRef Create<T>(Args... args);
```

ApplicationContext::Create

- Use this to create your Subjects
- Do NOT use the normal c'tor
- Returns a RelayRef and NOT the subject instance

ApplicationContext::ExecuteLater

```
/**
 * Executes the given function
   later
 */
TicketHandle ExecuteLater(function
   <void()> fun);
```

ApplicationContext::ExecuteLater

- Use this to simulate time
- Do NOT use `sleep(long millis)`

ApplicationContext

```
int main(void){
  ApplicationContext::Init();

  //Bootstrap your App here
  auto uno= Create<MySubject>();
  auto dos= Create<MySubject>(uno);

  //This method blocks until Ctrl+C
  ApplicationContext::Start();
}
```

Setup

Setup

1. Download VirtualBox or VMWare Workstation Player, e.g. here **`www.vmware.com/`**
2. Download preconfigured machine from **`http://web.cs.upb.de/ti/relaymq-machine.tar.gz`**
3. Register for Clion student license at **`www.jetbrains.com/student/`**

Example Program

Example

Open CLion

Thanks for your attention!

Questions?