Reactive and Event-based Systems

Assignment 3: Publish/Subscribe and Streaming and Jolie with updated Exercercise 1.2 - not required

Thomas T. Hildebrandt

Software, Data, People & Society Section
Department of Computer Science
Copenhagen University, Denmark

Part 1:

In Fig. 1, we show a message broker for a Call Center as given in the lecture. At the call center, every event is associated with a case ID. There are three event producers: The Phone (publishing an event to the broker whenever an outbound or inbound call related to a case ID happens), the email server (publishing an event to the broker whenever an inbound or outbound email related to a case ID happens) and an Email client (publishing an event to the broker whenever an email related to a case ID is handled, eg. read. The format of the topics are `CS/"case ID"/"Activity")` and the message contain additional parameters of the activity, e.g. the start and end-time and the name of the call center employee performing the activity. That is, the message `CS/9/Call Outbound, X)` from the phone represents that an outbound call is made on case 9 with parameters X.

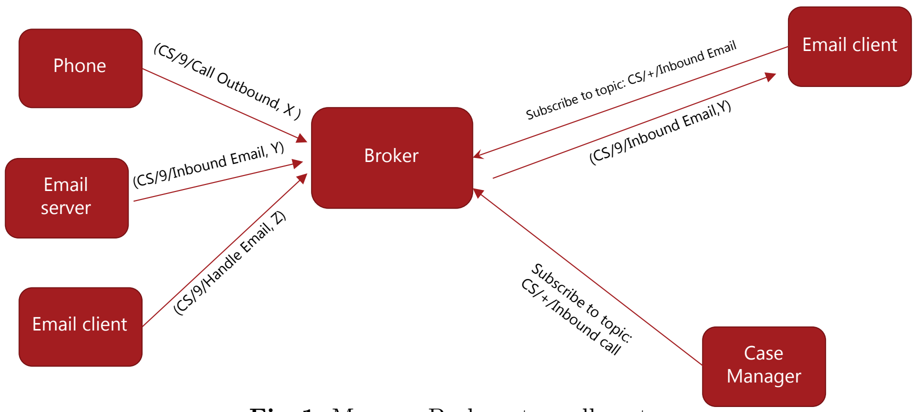

Fig. 1. Message Broker at a call center.

Below we modelled an abstract Broker, a publisher and a subscriber in the pi-calculus. For the sake of making extended the terms simpler, the calculus has been extended with global variables that can store sets of names and a `foreach` that allows iterating over a set of names and a `print` that prints out a received message.

$$\begin{array}{ll} \mathit{Broker} =_{def} & \texttt{let } A = \emptyset \texttt{ in}(\\ & !\texttt{subscribe}(x).A = A + \{x\}.\texttt{0} \\ & \mid !\texttt{mess}(m).(\texttt{foreach } a \in A \texttt{ do } \overline{a}(m)).\texttt{0} \\ &) \\ \mathit{Subscriber} =_{def} & (\nu a).\overline{\texttt{subscribe}}(a).!a(x).\texttt{print}x.\texttt{0} \\ \mathit{Publisher} =_{def} = & \overline{\texttt{mess}}(m1).\overline{\texttt{mess}}(m2).\texttt{0} \end{array}$$

Exercises

1.1 How would you draw that the Case Manager can also send events to the Broker recording the activity `Handle Case`.

1.2 Consider the process $(\nu \texttt{subscribe}, \texttt{mess})(\texttt{Broker}|\texttt{Subscriber}|\texttt{Subscriber}|\texttt{Publisher})$. Explain in your own words what it means. In particular, explain the use of fresh names νa. Try to make an example transition system where the two subscribers first do their actions (synchronizing with the Broker) and then the publisher makes its two actions (synchronizing with the Broker) with the broker also doing its actions in between.

Part 2:

Below is an implementation of a server process subscribing to the call center log (called Disco Example Log) published at the HIVEMQ broker.

```
include "mosquitto/interfaces/MosquittoInterface.iol"
include "console.iol"

execution {concurrent}

inputPort Server {
    Location: "local"
    Protocol: sodep
    Interfaces: MosquittoReceiverInteface
}

outputPort Mosquitto {
    Interfaces: MosquittoInterface
}

embedded {
    Java:
        "org.jolielang.connector.mosquitto.MosquittoConnectorJavaService" in Mosquitto
}
```

```
init {
    request << {
        brokerURL = "tcp://broker.hivemq.com",
        subscribe << {
            topic = "pmcep/Disco Example Log/#"
        }
        // I can set all the options available from the Paho library
    }
    setMosquitto@Mosquitto (request)()
}

main {
    receive (request)
    println@Console("topic :     "+request.topic)()
    println@Console("message :   "+request.message)()
}
```

using the jar files

`JolieMosquittoConnector.jar`

and

`org.eclipse.paho.client.mqttv3-1.1.2-20170804.042534-34.jar`

which should both be placed in a folder called `lib` in the same location as the subscriber source file.

Exercises

2.1 Run the subscriber. It should (after some warnings) receive events from the broker on the call center example with topics at the format `pmcep/Disco Example Log/CaseX/Activity`, where X is a number denoting the ID of the case and `Activity` is the activity, e.g. `Inbound Call`.
2.2 Run the subscriber in two shells. Explain what happens and how this is different than for a message queing system, actors or dataflow networks.
2.3 Implement two new subscribers that uses the topic filters to only listens to respectively Inbound Email and the Inbound call activities as the Email client and the Case manager in Fig 1 above.
2.4 Try to implement a subscriber that counts how many time each kind of activity happens.