AVA

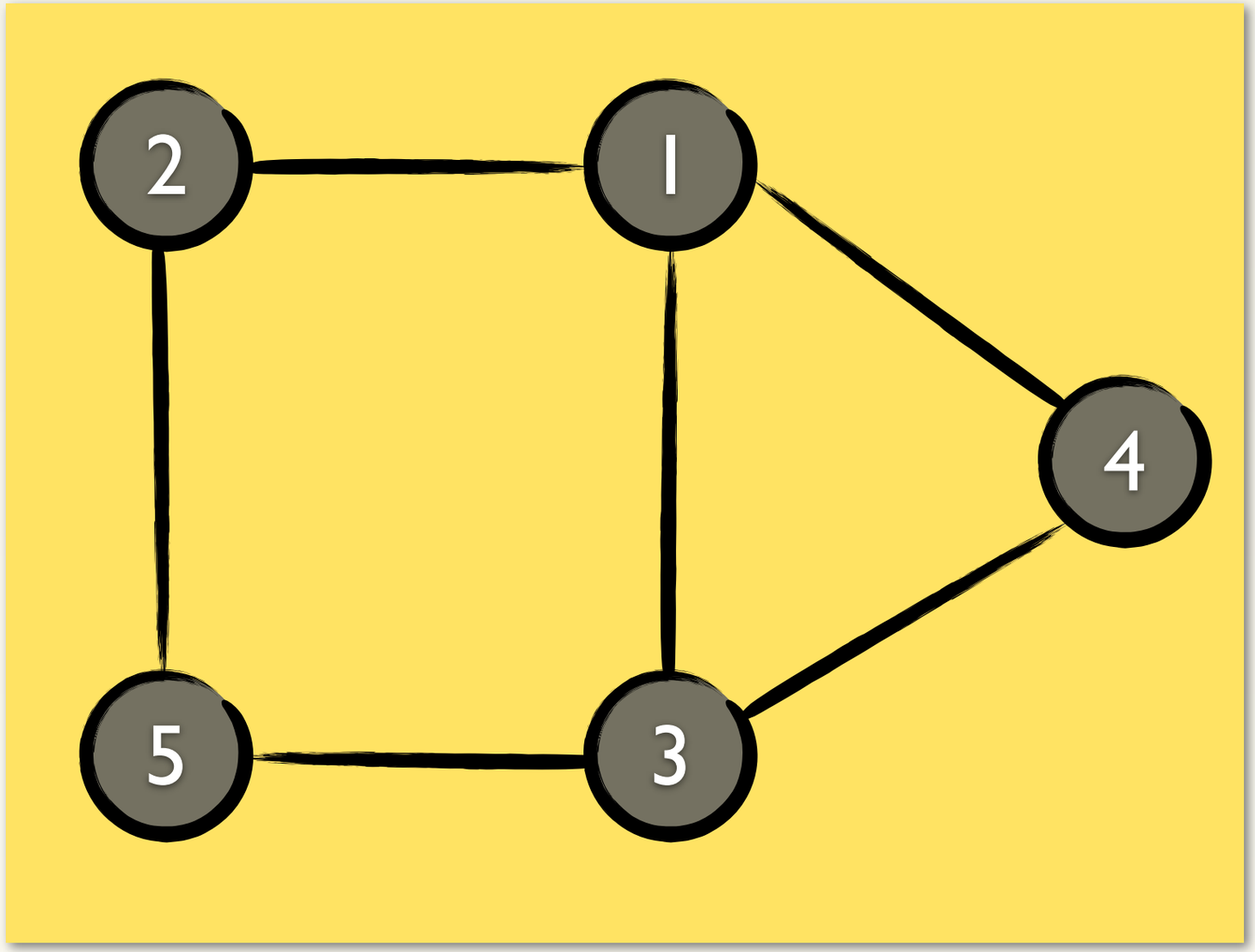
Exercise 2

Money transaction

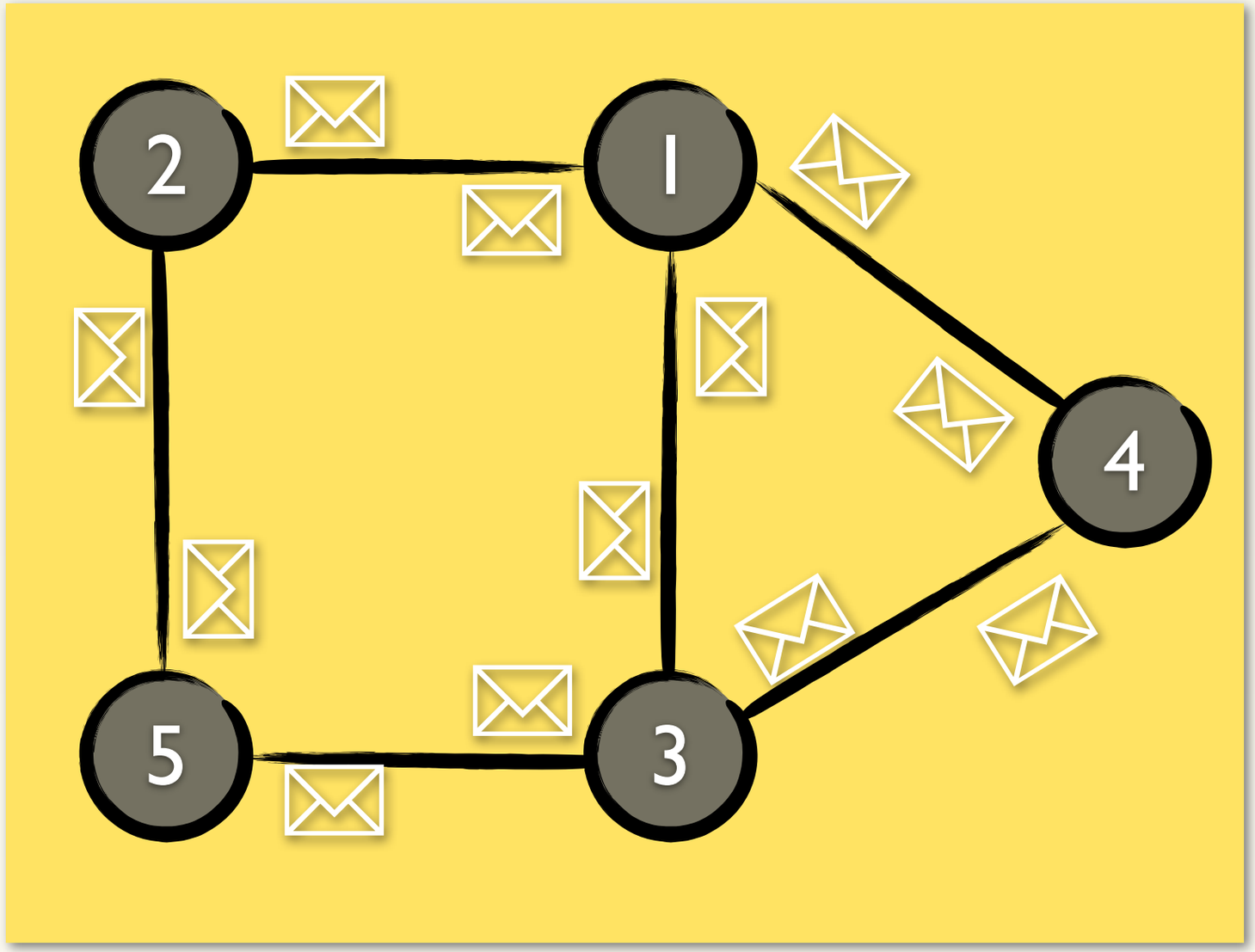
&

Observer

Scenario

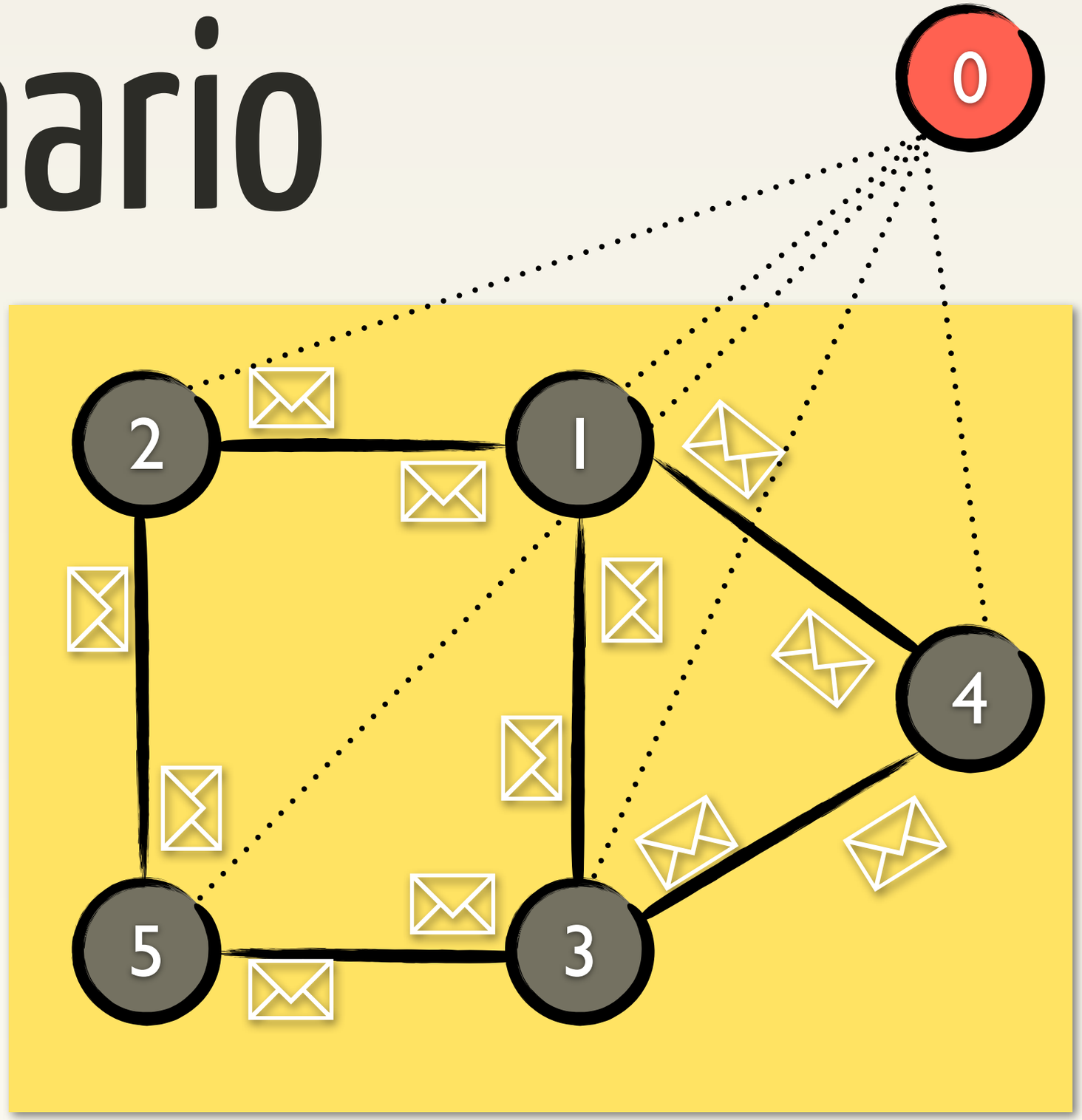


Scenario



Moneytransaction

Scenario



Messagecount

Classes

Bot

- name
- neighbors

- sendMsg
- handleSocket
- msgcount

MoneyBot

- moneyMsgCounter
- accountBalance

- moneytransaction

ObserverBot

- sendMsgSum
- receivedMsgSum

- observe
- msgcount

Messages

{"sender":"2000", "destination":"2001","species":"initiator", "action":"moneytransaction","moneyAmount":"20"}

sender	destination	species	action	data
		•initator •robot •observer	•msgcount •moneytransaction	•moneyAmount •sendMsgs •receivedMsgs

Ruby & JSON

```ruby
msgHash = {
    'sender'=> 2001,
    'destination'=> 2002,
    'action'=> 'moneytransaction',
    'species'=> 'robot'
}


msgString = JSON.generate(msgHash)
parsedMsgHash = JSON.parse(msgString)
```

TCP Socket

```
socket = TCPSocket.open('localhost', PORT)
socket.puts("#{msg}\r\n")
socket.close()
```

TCP Server

```
server = TCPServer.new(PORT)
loop do
    Thread.start(server.accept) do |socket|
        handleSocket(socket)
    end
end
```

Handle Requests

```
def handleSocket(socket)
    begin
        requestString = socket.gets
        requestHash = JSON.parse(requestString)
        sender = requestHash['sender']
        action = requestHash['action']
        if(isForMe(requestHash))
            self.send(action, requestHash)
        end
    rescue StandardError => e
        logError(e);
    ensure
        socket.close
    end
end
```

Handle Actions

```
def msgcount(requestHash)
    #handle msgcount action here...
end

def moneytransaction(requestHash)
    #handle moneytransaction action here...
end
```

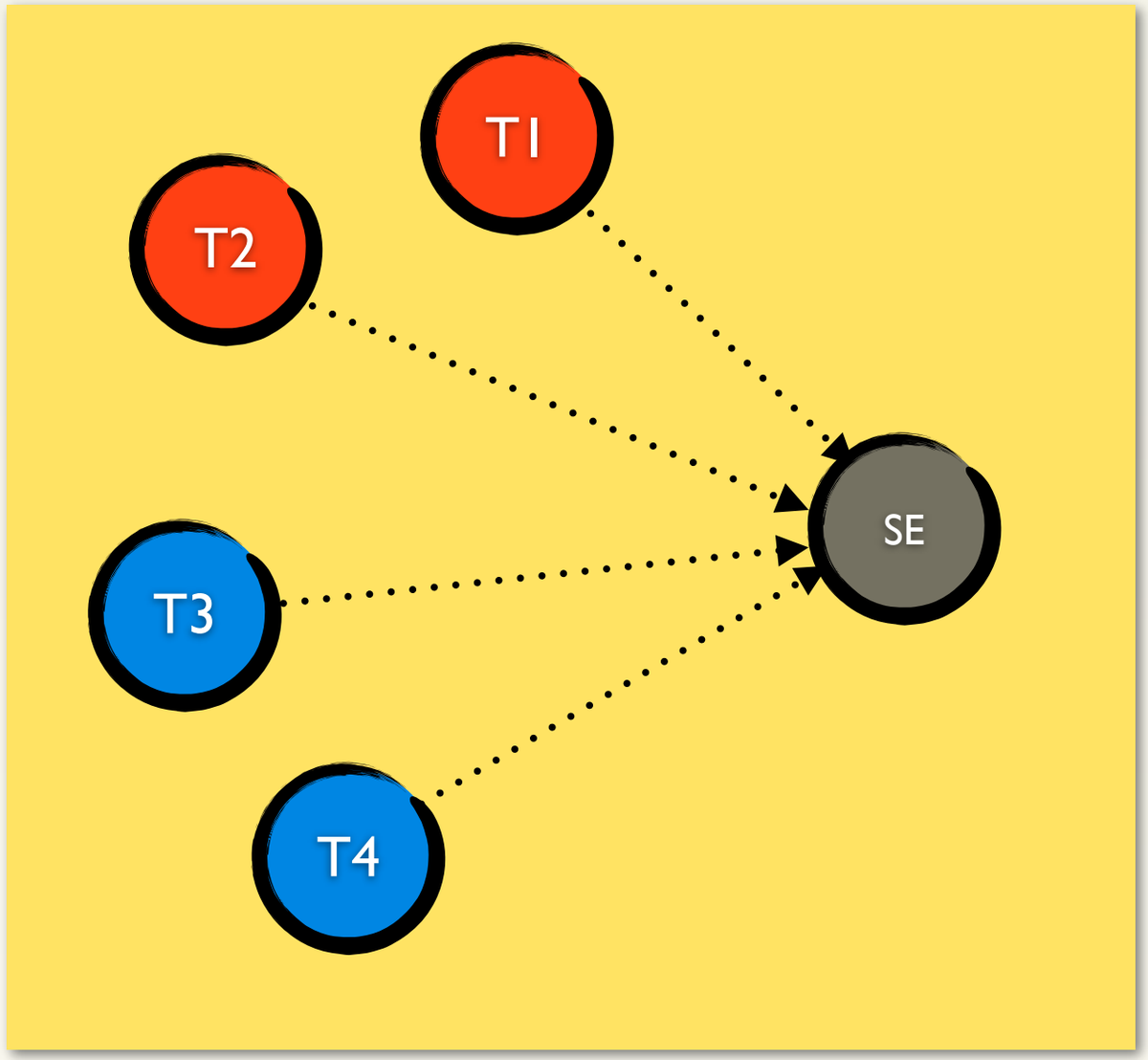
Demo

Stock Exchange

Scenario

Aggressive Traders

Relaxed Traders



Stock Exchange

Classes

Stock Exchange

- marketprice
- scatter

- updateMarketPrice
- marketprice
- buystocks
- sellstocks

Bot

- name
- neighbors

- sendMsg
- handleSocket
- msgcount

Classes

Trader

- accountBalance
- stocks

- marketprice
- buystocks
- sellstocks

RelaxedTrader

- determineStocksToBuy
- determineStocksToSell

Aggressive Trader

- determineStocksToBuy
- determineStocksToSell

Messages

{"sender":"2000", "destination":"2001", "species":"robot", "action":"marketprice", "marketPrice":63”}

sender	destination	species	action	data
		•initator •robot •observer	•marketprice •buystocks •sellstocks	•marketPrice •money •stocks

Relaxed Trader

```
def determineStocksToBuy(max)
    if(@currentMarketPrice < avgMarketPrice())
        return max
    else
        return 0
    end
end

def determineStocksToSell(max)
    if(@currentMarketPrice > avgMarketPrice())
        return max
    else
        return 0
    end
end
```

Aggressive Trader

```
def determineStocksToBuy(max)
    if(@currentMarketPrice > @marketPrices[@marketPrices.size() -2])
        return max
    else
        return 0
    end
end

def determineStocksToSell(max)
    if(@currentMarketPrice < @marketPrices[@marketPrices.size() -2])
        return max
    else
        return 0
    end
end
```

Demo