Learning Object-Oriented Programming, Design and TDD with Pharo

Stéphane Ducasse

March 11, 2019

Layout and typography based on the sbabook LaTeX class by Damien Pollet.

Contents

Illustrations

CHAPTER 1

Network simulator solutions

1.1 Packets

```
KANetworkPacket class >> from: sourceAddress to: destinationAddress
    payload: anObject
    ^ self new
        initializeSource: sourceAddress
        destination: destinationAddress
        payload: anObject
```

```
KANetworkPacket >> initializeSource: source destination: destination
    payload: anObject
    sourceAddress := source.
    destinationAddress := destination.
    payload := anObject
```

```
KANetworkPacket >> sourceAddress
    ^ sourceAddress
```

```
KANetworkPacket >> destinationAddress
    ^ destinationAddress
```

```
KANetworkPacket >> payload
    ^ payload
```

1.2 Nodes

```
KANetworkNode >> initializeAddress: aNetworkAddress
    address := aNetworkAddress
```

```
KANetworkNode >> address
    ^ address
```

1.3 Links

```
KANetworkLink >> initializeFrom: sourceNode to: destinationNode
    source := sourceNode.
    destination := destinationNode.
```

```
KANetworkLink >> source
    ^ source
```

```
KANetworkLink >> destination
    ^ destination
```

```
Object subclass: #KANetworkNode
    instanceVariableNames: 'address outgoingLinks'
    classVariableNames: ''
    category: 'NetworkSimulator-Core'
```

```
KANetworkNode >> hasLinkTo: anotherNode
    ^ outgoingLinks
        anySatisfy: [ :any | any destination == anotherNode ]
```

1.4 Sending a packet

```
KANetworkLink >> isTransmitting: aPacket
    ^ packetsToTransmit includes: aPacket
```

1.5 Transmitting a packet

```
KANetworkLink >> transmit: aPacket
    "Transmit aPacket to the destination node of the receiver link."
    (self isTransmitting: aPacket)
        ifTrue: [
            packetsToTransmit remove: aPacket.
            destination receive: aPacket from: self ]
```

```
Object subclass: #KANetworkNode
    instanceVariableNames: 'address outgoingLinks arrivedPackets'
    classVariableNames: ''
    category: 'NetworkSimulator-Core'
```

```
KANetworkNode >> initialize
    outgoingLinks := Set new.
    arrivedPackets := OrderedCollection new
```

```
KANetworkNode >> hasReceived: aPacket
    ^ arrivedPackets includes: aPacket
```

1.6 The loopback link

```
KANetworkNode >> initialize
    loopback := KANetworkLink from: self to: self.
    outgoingLinks := Set new.
    arrivedPackets := OrderedCollection new
```

```
KANetworkNode >> linksTowards: anAddress do: aBlock
    "Simple flood algorithm: route via all outgoing links.
    However, just loopback if the receiver node is the routing
     destination."
    anAddress = address
        ifTrue: [ aBlock value: self loopback ]
        ifFalse: [ outgoingLinks do: aBlock ]
```

1.7 Modeling the network itself

```
KANetworkTest >> buildNetwork
    alone := KANetworkNode withAddress: #alone.

    net := KANetwork new.

    hub := KANetworkNode withAddress: #hub.
    #(mac pc1 pc2 prn) do: [ :addr |
        | node |
        node := KANetworkNode withAddress: addr.
        net connect: node to: hub ].

    net
        connect: (KANetworkNode withAddress: #ping)
        to: (KANetworkNode withAddress: #pong)
```

```
KANetwork >> initialize
    nodes := Set new.
    links := Set new
```

```
KANetwork >> connect: aNode to: anotherNode
    self add: aNode.
    self add: anotherNode.
    links add: (self makeLinkFrom: aNode to: anotherNode) attach.
    links add: (self makeLinkFrom: anotherNode to: aNode) attach
```

1.8 Looking up nodes

```
KANetwork >> nodeAt: anAddress ifNone: noneBlock
    ^ nodes
        detect: [ :any | any address = anAddress ]
        ifNone: noneBlock
```

1.9 Looking up links

```
KANetwork >> linkFrom: sourceAddress to: destinationAddress
    ^ links
        detect: [ :anyLink |
            anyLink source address = sourceAddress
                and: [ anyLink destination address =
    destinationAddress ] ]
        ifNone: [
            NotFound
                signalFor: sourceAddress -> destinationAddress
                in: self ]
```

1.10 Packet delivery with forwarding

```
KANetworkHub >> forward: aPacket from: arrivalLink
    self
        linksTowards: aPacket destinationAddress
        do: [ :link |
            link destination == arrivalLink source
                ifFalse: [ self send: aPacket via: link ] ]
```

Bibliography