Report on Reliable Transfer Protocol

Name: Nagarjuna pamu

Roll No: B11069

Language used : Scala (object oriented and functional hybrid)

Library: Akka (for asynchronous IO)

Akka provides asynchronous IO library and also provides Actors support

1)Why Actors ?(Actors can be used as STATE MACHINES)

Used actors in this project, Actors are basically objects which can send messages, they can execute methods and have some state.

Actors implement message passing and are good for implementing FINITE STATE MACHINES.

FINITE STATE MACHINES are good for implementing protocol because they involve state changes upon receiving a message and actors are very good models for implementing state machines.

2)Why Asynchronous IO ?

In usual synchronous IO , one has to wait for the message blocking the thread and once the message arrives we can process it. But in case of asynchronous IO program (here actor) will be notified once the message arrives. Asynchronous IO is non blocking and gives lots of performance in large projects.

Its a push model instead of pull model of computation.

Am passionate about using asynchronous IO , anyways this project does not need asynchronous IO ,but I believe asynchronous IO should be part of programming.

3)Working of the Project ?

AKKA IO is a asynchronous non-blocking IO library

AKKA is a open source toolkit and runtime for simplifying building concurrent applications on JVM and AKKA io is the asynchronous IO library part of the akka toolkit

1)Actor

Actor can be used for implementing FINITE STATE MACHINES

Actor is a object which can do three things
1)communicate(send messages)
2)store(have state)
3)process(can execute code)

UDP sender is an Actor
UDP receiver is also an Actor

This Project has following files

package com.nagarjuna_pamu.data_sender

Receiver.scala => Its an Actor which receives acks from the data receiver

Sender.scala => Its an Actor which sender data packets to the data receiver

package com.nagarjuna_pamu.data_receiver

Receiver.scala => Its an actor which receives data packets from the data sender

Sender.scala => Its an Actor which sends ack packets to the data receiver

package com.nagarjuna_pamu

Main.scala => Bootstrap program to start all the actors

Params.scala => Contains all parameters such as window size, timeout time etc

Utils.scala => Contains all methods for decoding , encoding packets and decoding , encoding packets .

Basic working of the protocol is as following:

Data Sender

1. Choose sequence number randomly , Send window number of packets and wait for the ack
2. if ack is received
3. check the ack
   - if ack contains correct sequence number of the first packet of the window
   - check the message body for errors
   - check the bitmap for loss of the packets (if loss occurred greater than recommended resend current window)
   - if ack is OK , send next window of packets or else send lost packets
   - Slide the window based on packets received from the left using bitmap.
4. If ack is not received timeout
   - resend the current window again.
5. If timeouts more the max tries.
   - quit

Data Receiver

1. Packets are received
   - take the first packet sequence number and store
   - make a bitmap array of size equal to window and store 1 for received and 0 for lost packet based on first sequence number
   - store the message

2. After filling the bitmap of window size , prepare the ack packet using the given specification
3. send the ack(periodically send the acks).
4. Quit after particular number of timeouts .