SYSC 5104 Methodologies for Discrete Event Modelling and Simulation

Assignment1:

Image Transfer Protocol Simulator

Mengyao Wu 101006768

Xinrui Zhang 101027055

Carleton University

Oct. 13th, 2020

Table of Contents

[Part I Motivation and Introduction 1](#_Toc54446885)

[Image Sender 2](#_Toc54446886)

[Network 3](#_Toc54446887)

[Receiver 3](#_Toc54446888)

[Part II Formal Specifications 3](#_Toc54446889)

[For atomic models 3](#_Toc54446890)

[Packet Generator 3](#_Toc54446891)

[Packet Forwarder 4](#_Toc54446892)

[Retransmission Controller 6](#_Toc54446893)

[Network Delay 8](#_Toc54446894)

[Switch Buffer 9](#_Toc54446895)

[Receiver 9](#_Toc54446896)

[For coupled models 10](#_Toc54446897)

[Packet Sender 10](#_Toc54446898)

[Image Sender 10](#_Toc54446899)

[Network 11](#_Toc54446900)

[ITP 11](#_Toc54446901)

[Testing Strategies 11](#_Toc54446902)

[Part III Model Implementation and Testing 12](#_Toc54446903)

[Atomic Models 12](#_Toc54446904)

[PacketGenerator 12](#_Toc54446905)

[PacketForwarder 14](#_Toc54446906)

[Retransmission (Retransmission controller) 17](#_Toc54446907)

[NetworkDelay 20](#_Toc54446908)

[SwitchBuffer 21](#_Toc54446909)

[Receiver 23](#_Toc54446910)

[Coupled Model 24](#_Toc54446911)

[PacketSender 24](#_Toc54446912)

[ImageSender 27](#_Toc54446913)

[Network 30](#_Toc54446914)

[ITP top module 31](#_Toc54446915)

[Discussion 34](#_Toc54446916)

# Part I Motivation and Introduction

A person does not inspect images pixel by pixel. Instead, humans scan an image based on its overall pixel distribution. As a result, in the case of an image transfer over the internet, it is not necessary to receive every single packet correctly. However, if too many consecutive chunks of data are lost, the receiver might lose important information. It is possible to develop a protocol which allows bounded or controlled packet loss so the received image will not lose a large consecutive chunk of data.

With a fixed amount of data to transfer and no packet loss, the longer the packet size, the faster the transmission since there will be fewer packets to be sent. However, in a congested network environment, switch or router buffer might be approaching to its limit. As a result, a larger packet will be less likely to fit into the available buffer and it will be more likely to be dropped. The drop of a large packet will result in large chunk of image data loss on the receiver side. This will require retransmission hence in turn more overhead added. We want to find the optimal packet size under certain network conditions to enable faster transfer while preserving acceptable quality of image.

To solve the problem, we designed an ITP (Image Transfer Protocol) which allows variable packet size during image transfer. A DEVS model called “ITP Simulator” will be created to simulate its behavior. The simulation aims to find the optimal packet size for the developed image transfer protocol under certain congested network conditions. The protocol will determine how much consecutive portion of image data might be lost based on the ACK sequence number received. If the consecutive data loss is too big, a re-transmission must be done. By iterating through various packet sizes, a potential optimal packet size might be found in simulations.

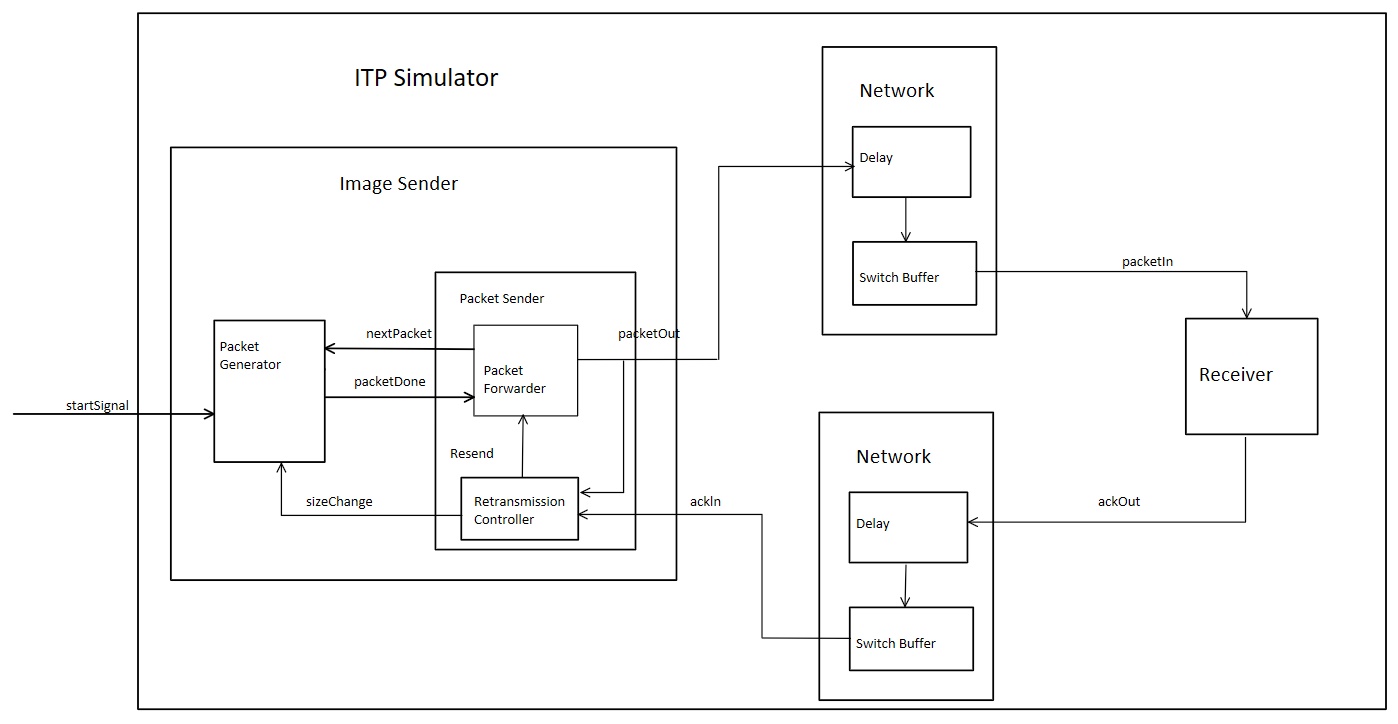


Figure 1: A hierarchical view of the ITP Simulator

As shown in figure 1, there are three high level components in the ITP model: Image Sender, Network and Receiver. Image Sender is decomposed into Packet Sender and Packet Generator. Packet Sender is further decomposed into Packet Forwarder and Retransmission Controller. Network is decomposed into Delay and Switch Buffer module.

## Image Sender

Packet Generator generates packets based on state variable like sequence number, packet size and image data to be sent. It starts with Start Signal, generates, and sends a packet to Packet Sender. Upon receiving signal nextPacket, it generates another packet based on most up to date state variable. Packet Generator also receives sizeChange signal from Packet Sender and update its state variable.

Inside Packet Sender, there are Packet Forwarder and Retransmission Controller. Packet Forwarder receives packet from Packet Generator and sends packet out to network as well as retransmission controller. After forwarding, it sends NextPacket signal to Packet Generator. It also receives Resend signal, which contains packets to be retransmitted, and retransmits those packets. Retransmission Controller receives a copy of packet sent out by Packet Sender, buffers them and wait for ACK signal. After certain period, it (our ITP protocol) calculates what packets need to be retransmitted and send Resend signal to Packet Forwarder. It might also send out SizeChange signal to Packet Generator.

## Network

Receives packet or ACK packet, pass it through Delay and Switch Buffer module. Delay module will delay the packet for some time. Switch Buffer will have random buffer size available and compare it to the size of incoming packets. If packet size bigger than available buffer size, packets dropped. Otherwise, send it to Receiver.

## Receiver

Receives a packet and send out ACK.

# Part II Formal Specifications

## For atomic models

The formal specifications <S, X, Y, δint, δext, λ, ta> for the atomic models are defined as follows:

### Packet Generator

S = {sigma = ∞, phase = passive, totalDataSize = 0, totalGeneratedSize = 0, packetSize = default, send = False, currentPacket\_size = 0, currentPacket\_seq = 0, sequence = 0}

X = {startSignal, nextPacket, sizeChange}

Y = {packetDone}

Parameters = {default = 5}

δint (s) {

case phase

active:

if (send){

if (totalGeneratedSize < totalDataSize) {

if ((totalGeneratedSize + packetSize) <= totalDataSize) {

currentPacket\_size = packetSize

}

else {

packetSize = totalDataSize - totalGeneratedSize

currentPacket\_size = packetSize

}

currentPacket\_seq = sequence

sequence ++

totalGeneratedSize += packetSize

}

else

phase = passive //simulation terminated here

}

passive:

/\* Never happens \*/

sending = False

sigma = INFINITY

}

δext (s, X,e) {

case phase

passive:

if (StartSignal){

phase = active

totalDataSize = StartSignal

sending = True

assert (packetSize < totalDataSize)

currentPacket\_size = packetSize

currentPacket\_seq = sequence

sequence ++

totalGeneratedSize += packetSize

sigma = 0

}

else{

/\* Never happens \*/

}

active:

if (SizeChange) {

packetSize == SizeChange

sigma = 0

}

else if (NextPacket){

sending = True

sigma = 0

}

}

λ(s){

if (active && sending)

send currentPacket to port out

}

ta(s) = sigma

### Packet Forwarder

S = {sigma = ∞, phase = passive, currentPacket = None, LastSeqNum = 0, signalNextPacket = false, resending = false, resendReceiving = false, resendIndex = 0, resendBuffer[bufferSize] = 0, currentSequence = 0}

X = {resendIn, packetIn}

Y = {packetOut, nextPacket}

parameters: {forwardTime = 5, bufferSize = 10, timeOut = 600}

δint (s) {

if(resending and not resendReceiving){

if(resendIndex <= 0){

resendIndex = 0

resending = false

signalNextPacket = true

currentPacket = Packet\_t(0,0)

sigma = timeOut // wait 600 ms to signal generator after retransmission done

}else{

resendIndex--

currentPacket = resendBuffer[state.resendIndex]

sigma = forwardTime

}

}else if (sending){

sending = false

sigma = INFINITY

}else{

signalNextPacket = false

sigma = INFINITY

}

}

δext (s,X,e) {

if(! sending){

if(X == packetIn){

currentPacket = packetIn

currentSequence = currentPacket.sequence

signalNextPacket = false

sending = true

sigma = forwardTime

}else if (X == resendIn){

currentSequence = 0

signalNextPacket = false

currentPacket = resendIn

if(currentPacket.size != 0){

resendReceiving = true

resendBuffer[resendIndex] = currentPacket

resendIndex++

sigma = INFINITY

}else{

if(resendReceiving)

resending = true

resendReceiving = false

sigma = 0

}

}

}else{

/\* Never happens \*/

}

}

λ(s){

if (currentPacket.size != None)

send currentPacket to port packetOut

if ( (currentSequence != LastSeqNum) && ! resending && ! resendReceiving)

send int(1) to port nextPacket

}

ta(s) = sigma

### Retransmission Controller

S = {sigma = ∞, phase = passive, packetBuffer[bufferSize] = {}, ackBuffer[bufferSize] = {}, resendBuffer[bufferSize] = {}, packetIndex = 0, ackIndex = 0, resent = false, resending = false, currentPacket = None, resendIndex = 0, chunkLossLimit = 0.0, dataLoss = 0,dataTotal = 0, size = 0}

X = {ackIn, packetIn}

Y = {resend, sizeChange}

parameters: {bufferSize = 10, timeOut = 600, signalPacket = Packet\_t{0,0}, forwardTime = 5}

δint (s) {

case phase

active && ! resent:

if not resending{

resending = true

dataLoss = 0

dataTotal = 0

for i in each packetIndex {

dataTotal += packetBuffer[i].size

check if each packet in the ackIndex is received

if(! received){

put the packet in the resendBuffer[resendIndex]

resendIndex++

dataTotal += packetBuffer[i].size

}

}

}

ackIndex = 0

packetIndex = 0

lossRatio = dataLoss/dataTotal

if(lossRatio <= chunkLossLimit){

currentPacket = signalPacket

resent = true

compare loassRatio and chunkLossLimit then update the size

else

size = packetBuffer.size

if((resendIndex - 1 )< 0){

resent = true

currentPacket = signalPacket

}else{

currentPacket = resendBuffer[resendIndex -1]

resendIndex --

}

}

sigma = forwardTime

passive:

phase = passive

resendIndex = ackIndex = packetIndex = 0

resent = resending = false

currentPacket = signalPacket

sigma = INFINITY

}

δext (s,X,e) {

case phase

passive:

sigma = INFINITY

if(X == packetIn){

if(packetIn.sequence == 1){//marks new cycle

packetIndex = 0

ackIndex = 0

}

packetBuffer[packetIndex] = packetIn

if packetBuffer is full{

resent = false

phase = active

sigma = timeOut

}

packetIndex++

}else if (X == ackIn){

ackBuffer[ackIndex] = ackIn

ackIndex++

}

active:

if(X == packetIn)

assert(false) //should not happen during retransmission

else if(X == ackIn){

ackBuffer[ackIndex] = ackIn

ackIndex++

sigma = sigma - e

}

}

λ(s){

if (active && resending){

send currentPacket to port resend

send size to port sizeChange

}

}

ta(s) = sigma

### Network Delay

S = {sigma = ∞, phase = passive, currentPacket = None, packetDelayPair[MAX\_QUEUE] = {}, totalDelayAhead = 0, delay = 0}

X = {in}

Y = {out}

Parameters = {MAX\_QUEUE = 100, networkDelay = 50}

δint (s) {

case phase

busy:

if empty(packetDelayPair)

phase = passive

sigma = INFINITY

else

currentPacket, sigma = Get(packetDelayPair) //(Packet, delay)

shift one item to left in packetDelayPair

passive:

/\* Never happens \*/

}

δext (s,X,e) {

case phase

passive:

currentPacket = in

sigma = networkDelay

phase = busy

busy:

sigma = sigma - e

totalDelayAhead = 0

calculate totalDelayAhead by adding all delays in the PacketQueue

if (totalDelayAhead > networkDelay){

assert("Error, totalDelayAhead > initialDelay")

}

delay = networkDelay - totalDelayAhead

add((Packet,delay), PacketQueue)

}

λ(s){

send currentPacket to port out

}

ta(s) = sigma

### Switch Buffer

S = {sigma = ∞, phase = passive, currentPacket = None, currentBufferSize = 0}

X = {in}

Y = {out}

Parameters ={MAX\_BUFFER\_SIZE = 40}

δint (s) {

case phase

active:

phase = passive

sigma = INFINITY

passive: /\* Never happens \*/

}

δext (s,X,e) {

case phase

passive:

currentPacket = in

cast the random (0,1) to range(0, MAX\_BUFFER\_SIZE)

if(currentBufferSize > currentPacket.size){

phase = active

sigma = 0 // call internal transition function right away

}else

Sigma = INIFINITY

active: /\* Never happens \*/

}

λ(s){

send currentPacket to port out

}

ta(s) = sigma

### Receiver

S = {sigma = ∞, phase = passive, currentACK = 0, currentSize = 0}

X = {in}

Y = {out}

δint (s) {

case phase

active:

phase = passive

sigma = ∞

passive: /\* Never happens \*/

}

δext (s,e,X) {

case phase

active:

phase = active

currentPacket = Packet

sigma = 0

passive:

phase = active

currentPacket = Packet

sigma = 0

}

λ(s) {

sequence = currentPacket.sequence

currentACK = sequence

send currentACK to port out

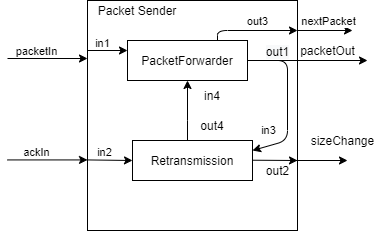
}

ta(s) = sigma

## For coupled models

The formal specifications <X, Y, M, EIC, EOC, IC, SELECT> for the coupled models are defined as follows:

### Packet Sender

 X = {packetIn, ackIn}

Y = {nextPacket, packetOut, sizeChange}

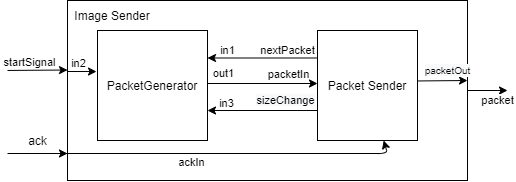
M = {PacketForwarder, Retransmission}

EIC = {(PacketSender.packetIn, PacketForwarder.in1), (PacketSender. ackIn, Retransmission.in2)}

EOC = {(PacketForwarder.out3, PacketSender. nextPacket), (PacketForwarder.out1, PacketSender. packetOut), (Retransmission.out2, PacketSender. sizeChange)}

IC = {(Retransmission.out4, PacketForwarder.in4), (PacketForwarder.out1, Retransmission.in3)}

### Image Sender



X = {startSignal, ack}

Y = {packet}

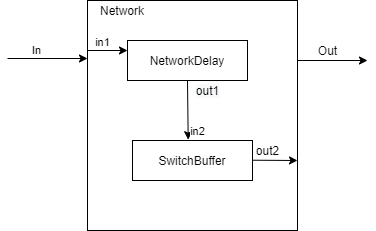
M = {PacketGenerator, PacketSender}

EIC = {(Sender. startSignal, PacketGenerator.in2), (Sender. ack, PacketSender. ackIn)}

EOC = {(PacketSender. packetOut, Sender. packet)}

IC = {(PacketSender. nextPacket, Sender.in1), (PacketSender. sizeChange, Sender.in3), (PacketGenerator.out1, PacketSender. packetIn)}

### Network

X = {In}

Y = {Out}

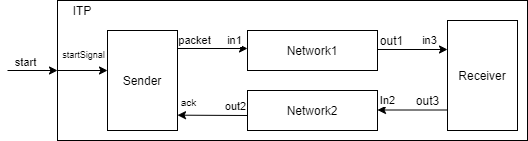
M = {NetworkDelay, SwitchBuffer}

EIC = {(Network.In, NetworkDelay.in1)}

EOC = {(SwitchBuffer.out2, Network.Out)}

IC = {(NetworkDelay.out1, SwitchBuffer.in2)}

### ITP

X = {start}

Y = {}

M = {Sender, Network1, Network2, Reciever}

EIC = {(ITP.start, Sender. startSignal)}

EOC = {}

IC = {(Sender.packet, Network1.in1), (Network1.out1, Reciever.in3), (Reciever.out3, Network2.in2), (Network2. Out2, Sender.ack)}

## Testing Strategies

Part III provides a detailed description for functions of each atomic and coupled models. Ideally, the testing input should challenge any situations that match the functional description of each atomic/coupled model. This would require too much manual input generation and testing. However, each atomic model or coupled model has some dependency on others. The input of an atomic model is also an output of another atomic model. As a result, the testing input for an atomic model is generated with the assumption that its dependent model is behaving correctly. With this assumption, the input distribution should be similar to some atomic models’ output distribution. For the top model, where everything is coupled, it only has one single input which is the start signal. The output messages for top model will be very complex since it contains all the components. It will be tested with 3 typical scenarios with 1 scenario covers at least one complete cycle. Both white and black box testing method are used since the detailed output messages are also checked.

# Part III Model Implementation and Testing

## Atomic Models

### PacketGenerator

It is initialized with passive state. Upon receiving signal startSignal, state is changed to active state. StartSignal has an integer indicates how many data need to be generated and sent out. Generator generates and sends one packet per time to PacketForwarder and waits for the signal nextPacket to come. Once it receives signal nextPacket, it generates and sends out another packet. Generator might also receive sizeChange signal from Retransmission Controller and update the size of packets to be subsequentially generated. Below has startSignal come in at time 500 ms with total data size 300. Default packet size is 10 and it should be changed at the time 2 s as sizeChange signal comes. The sizeChange will be applied to next generated packet (not the generated one). Upon receives nextPacket signal, generator sends out current packet at the same time as there is no delay.

Run scripts: ./PacketGenerator\_test.sh

//startSignal

00:00:00:500 300

// sizeChange signal (default set to 10)

00:00:02 20

//nextPacket signal

00:00:01 1

00:00:01:005 1

00:00:01:010 1

00:00:01:015 1

00:00:01:020 1

00:00:01:025 1

00:00:01:030 1

00:00:01:035 1

00:00:01:040 1

00:00:02:005 1

00:00:02:010 1

00:00:02:015 1

00:00:02:020 1

00:00:02:025 1

00:00:02:030 1

00:00:02:035 1

00:00:02:040 1

//testing output

00:00:00:500 //startSignal with size 300 received

[cadmium::basic\_models::pdevs::iestream\_input\_defs<int>::out: {300}] generated by model input\_reader\_startSignal

00:00:00:500 //first packet generated and sent out with default size 10 and sequence 1

[PacketGenerator\_defs::packetDone: {10 1}] generated by model PacketGenerator1

00:00:01:000 //received nextPacket signal

[cadmium::basic\_models::pdevs::iestream\_input\_defs<int>::out: {1}] generated by model input\_reader\_nextPacket

00:00:01:000 //received nextPacket signal and send out current packet

[PacketGenerator\_defs::packetDone: {10 2}] generated by model PacketGenerator1

00:00:01:005

[cadmium::basic\_models::pdevs::iestream\_input\_defs<int>::out: {1}] generated by model input\_reader\_nextPacket

00:00:01:005

[PacketGenerator\_defs::packetDone: {10 3}] generated by model PacketGenerator1

00:00:01:010

[cadmium::basic\_models::pdevs::iestream\_input\_defs<int>::out: {1}] generated by model input\_reader\_nextPacket

00:00:01:010

[PacketGenerator\_defs::packetDone: {10 4}] generated by model PacketGenerator1

00:00:01:015

[cadmium::basic\_models::pdevs::iestream\_input\_defs<int>::out: {1}] generated by model input\_reader\_nextPacket

00:00:01:015

[PacketGenerator\_defs::packetDone: {10 5}] generated by model PacketGenerator1

00:00:01:020

[cadmium::basic\_models::pdevs::iestream\_input\_defs<int>::out: {1}] generated by model input\_reader\_nextPacket

00:00:01:020

[PacketGenerator\_defs::packetDone: {10 6}] generated by model PacketGenerator1

00:00:01:025

[cadmium::basic\_models::pdevs::iestream\_input\_defs<int>::out: {1}] generated by model input\_reader\_nextPacket

00:00:01:025

[PacketGenerator\_defs::packetDone: {10 7}] generated by model PacketGenerator1

00:00:01:030

[cadmium::basic\_models::pdevs::iestream\_input\_defs<int>::out: {1}] generated by model input\_reader\_nextPacket

00:00:01:030

[PacketGenerator\_defs::packetDone: {10 8}] generated by model PacketGenerator1

00:00:01:035

[cadmium::basic\_models::pdevs::iestream\_input\_defs<int>::out: {1}] generated by model input\_reader\_nextPacket

00:00:01:035

[PacketGenerator\_defs::packetDone: {10 9}] generated by model PacketGenerator1

00:00:01:040

[cadmium::basic\_models::pdevs::iestream\_input\_defs<int>::out: {1}] generated by model input\_reader\_nextPacket

00:00:01:040 //send out packet with sequence# 10

[PacketGenerator\_defs::packetDone: {10 10}] generated by model PacketGenerator1

00:00:02:000 // received sizeChange signal with size 20

[cadmium::basic\_models::pdevs::iestream\_input\_defs<int>::out: {20}] generated by model input\_reader\_sizeChange

00:00:02:000

[PacketGenerator\_defs::packetDone: {}] generated by model PacketGenerator1

00:00:02:005

[cadmium::basic\_models::pdevs::iestream\_input\_defs<int>::out: {1}] generated by model input\_reader\_nextPacket

00:00:02:005

[PacketGenerator\_defs::packetDone: {10 1}] generated by model PacketGenerator1

00:00:02:010

[cadmium::basic\_models::pdevs::iestream\_input\_defs<int>::out: {1}] generated by model input\_reader\_nextPacket

00:00:02:010 //size change take effect on this packet as this is the first packet generated after size change

[PacketGenerator\_defs::packetDone: {20 2}] generated by model PacketGenerator1

00:00:02:015

[cadmium::basic\_models::pdevs::iestream\_input\_defs<int>::out: {1}] generated by model input\_reader\_nextPacket

00:00:02:015

[PacketGenerator\_defs::packetDone: {20 3}] generated by model PacketGenerator1

00:00:02:020

[cadmium::basic\_models::pdevs::iestream\_input\_defs<int>::out: {1}] generated by model input\_reader\_nextPacket

00:00:02:020

[PacketGenerator\_defs::packetDone: {20 4}] generated by model PacketGenerator1

00:00:02:025

[cadmium::basic\_models::pdevs::iestream\_input\_defs<int>::out: {1}] generated by model input\_reader\_nextPacket

00:00:02:025

[PacketGenerator\_defs::packetDone: {20 5}] generated by model PacketGenerator1

00:00:02:030

[cadmium::basic\_models::pdevs::iestream\_input\_defs<int>::out: {1}] generated by model input\_reader\_nextPacket

00:00:02:030

[PacketGenerator\_defs::packetDone: {20 6}] generated by model PacketGenerator1

00:00:02:035

[cadmium::basic\_models::pdevs::iestream\_input\_defs<int>::out: {1}] generated by model input\_reader\_nextPacket

00:00:02:035

[PacketGenerator\_defs::packetDone: {20 7}] generated by model PacketGenerator1

00:00:02:040

[cadmium::basic\_models::pdevs::iestream\_input\_defs<int>::out: {1}] generated by model input\_reader\_nextPacket

00:00:02:040

[PacketGenerator\_defs::packetDone: {20 8}] generated by model PacketGenerator1

### PacketForwarder

PacketForwarder receives one packet per time from PacketGenerator. It then sends out the packet with a fixed forward time of 5 ms. After packet forwarded, it signals the generator for next packet with signal nextPacket. It keeps doing this until it reaches the 10th packet (sequence# 10). It then waits for response from Retransmission Controller. It will receive packets to be resent and store them in a buffer until all resend packets received. The packet received which is size 0 and sequence 0 from Retransmission Controller indicates end of resend packets. It will then send all resend packets one per time with forward time of 5 ms. After last resend packet has been resent, it waits for 600ms before it signals the generator with signal nextPacket.

Run scripts: ./PacketForwarder\_test.sh

//packetIn signal (from generator)

00:00:01 10 1

00:00:02 10 2

00:00:03 10 3

00:00:04 10 4

00:00:05 10 5

00:00:06 10 6

00:00:07 10 7

00:00:08 10 8

00:00:09 10 9

00:00:10 10 10 // first 10 packets

00:01:11 10 1

00:01:12 10 2

00:01:13 10 3

00:01:14 10 4

00:01:15 10 5

00:01:16 10 6

00:01:17 10 7

00:01:18 10 8

00:01:19 10 9

00:01:20 10 10 //second 10 packets

00:01:21 10 1

00:01:22 10 2

00:01:23 10 3

00:01:24 10 4

00:01:25 10 5

00:01:26 10 6

// resendIn signal

00:00:10:600 10 8

00:00:10:700 10 5

00:00:10:800 10 3

00:00:10:900 0 0 // end of resend packets for first 10 packets

00:01:20:800 0 0 //end of resend packets (no resend) for second 10 packets

//test output messages

00:00:01:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 1}] generated by model input\_reader\_packetIn

00:00:01:005

[PacketForwarder\_defs::packetOut: {10 1}, PacketForwarder\_defs::nextPacket: {1}] generated by model PacketForwarder1

Skipped Some Messages

00:00:10:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 10}] generated by model input\_reader\_packetIn

00:00:10:005 // 10th packet sent out by PacketForwarder

[PacketForwarder\_defs::packetOut: {10 10}, PacketForwarder\_defs::nextPacket: {}] generated by model PacketForwarder1

00:00:10:600 // start to receive resend packets

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 8}] generated by model input\_reader\_resendIn

00:00:10:700

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 5}] generated by model input\_reader\_resendIn

00:00:10:800

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 3}] generated by model input\_reader\_resendIn

00:00:10:900 // end of receiving resend packets

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {0 0}] generated by model input\_reader\_resendIn

00:00:10:900

[PacketForwarder\_defs::packetOut: {}, PacketForwarder\_defs::nextPacket: {}] generated by model PacketForwarder1

00:00:10:905 // start to forward resend packets received

[PacketForwarder\_defs::packetOut: {10 3}, PacketForwarder\_defs::nextPacket: {}] generated by model PacketForwarder1

00:00:10:910

[PacketForwarder\_defs::packetOut: {10 5}, PacketForwarder\_defs::nextPacket: {}] generated by model PacketForwarder1

00:00:10:915

[PacketForwarder\_defs::packetOut: {10 8}, PacketForwarder\_defs::nextPacket: {}] generated by model PacketForwarder1

00:00:11:515 //finished forwarding rensend packets, send nextPacket signal to PacketGenerator

[PacketForwarder\_defs::packetOut: {}, PacketForwarder\_defs::nextPacket: {1}] generated by model PacketForwarder1

00:01:11:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 1}] generated by model input\_reader\_packetIn

00:01:11:005

[PacketForwarder\_defs::packetOut: {10 1}, PacketForwarder\_defs::nextPacket: {1}] generated by model PacketForwarder1

Skipped Some Messages

00:01:20:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 10}] generated by model input\_reader\_packetIn

00:01:20:005

[PacketForwarder\_defs::packetOut: {10 10}, PacketForwarder\_defs::nextPacket: {}] generated by model PacketForwarder1

00:01:20:800 // end of resend packets (no resend this time)

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {0 0}] generated by model input\_reader\_resendIn

00:01:20:800 //send nextPacket signal to generator

[PacketForwarder\_defs::packetOut: {}, PacketForwarder\_defs::nextPacket: {1}] generated by model PacketForwarder1

00:01:21:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 1}] generated by model input\_reader\_packetIn

00:01:21:005

[PacketForwarder\_defs::packetOut: {10 1}, PacketForwarder\_defs::nextPacket: {1}] generated by model PacketForwarder1

Skipped Some Messages

### Retransmission (Retransmission controller)

Start with passive state, receives all the packets sent from the PacketForwarder. Packets are sent and received in a cycle of 10 packets. After 10 packets received(buffered) in the Retransmission Controller, wait 600 ms for Ack (600 ms timeout from 10th packet received). After 600 ms timeout, compare packets sent (received by retransmission controller) with Ack received by Retransmission Controller, buffer lost packets. Calculate data loss and compare with acceptable loss limit then it determines if lost packets need to be resent as well as if packet size change needed. If lost packets need to be resent, send them one by one to PacketForwarder with packet size 0 to tell PacketForwarder it has forwarded all the resend packets. After each cycle, go back to passive state.

Run scripts: ./Retransmission\_test.sh

//packetIn

00:00:00:005 10 1

00:00:00:010 10 2

00:00:00:015 10 3

00:00:00:020 10 4

00:00:00:025 10 5

00:00:00:030 10 6

00:00:00:035 10 7

00:00:00:040 10 8

00:00:00:045 10 9

00:00:00:050 10 10

00:00:00:660 10 1

00:00:00:665 10 2

00:00:00:670 10 3

00:00:00:675 10 4

00:00:00:680 10 5

00:00:00:685 10 6

00:00:00:690 10 7

00:00:00:695 10 8

00:00:00:700 10 9

00:00:00:705 10 10

//ackIn

00:00:00:105 10 1

00:00:00:110 10 2

00:00:00:115 10 3

00:00:00:120 10 4

00:00:00:125 10 5

00:00:00:135 10 7

00:00:00:140 10 8

00:00:00:145 10 9 //ackIn missing sequence 6 and 10, data loss is 20, within acceptable loss limit 20

00:00:00:780 10 5

00:00:00:790 10 7

00:00:00:805 10 10//for the second round, only 3 ackIn received, more than loss limit

//testing output

00:00:00:005

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 1}] generated by model input\_reader\_packetIn

00:00:00:010

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 2}] generated by model input\_reader\_packetIn

00:00:00:015

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 3}] generated by model input\_reader\_packetIn

00:00:00:020

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 4}] generated by model input\_reader\_packetIn

00:00:00:025

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 5}] generated by model input\_reader\_packetIn

00:00:00:030

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 6}] generated by model input\_reader\_packetIn

00:00:00:035

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 7}] generated by model input\_reader\_packetIn

00:00:00:040

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 8}] generated by model input\_reader\_packetIn

00:00:00:045

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 9}] generated by model input\_reader\_packetIn

00:00:00:050

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 10}] generated by model input\_reader\_packetIn

00:00:00:105

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 1}] generated by model input\_reader\_ackIn

00:00:00:110

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 2}] generated by model input\_reader\_ackIn

00:00:00:115

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 3}] generated by model input\_reader\_ackIn

00:00:00:120

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 4}] generated by model input\_reader\_ackIn

00:00:00:125

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 5}] generated by model input\_reader\_ackIn

00:00:00:135

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 7}] generated by model input\_reader\_ackIn

00:00:00:140

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 8}] generated by model input\_reader\_ackIn

00:00:00:145

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 9}] generated by model input\_reader\_ackIn

00:00:00:650

[Retransmission\_defs::sizeChange: {}, Retransmission\_defs::resend: {}] generated by model Retransmission1

// data loss is 20, within the acceptable loss limit, no packets to resend, send packet size 0 to PacketForwarder

00:00:00:655

[Retransmission\_defs::sizeChange: {10}, Retransmission\_defs::resend: {0 0}] generated by model Retransmission1

00:00:00:660

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 1}] generated by model input\_reader\_packetIn

00:00:00:665

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 2}] generated by model input\_reader\_packetIn

00:00:00:670

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 3}] generated by model input\_reader\_packetIn

00:00:00:675

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 4}] generated by model input\_reader\_packetIn

00:00:00:680

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 5}] generated by model input\_reader\_packetIn

00:00:00:685

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 6}] generated by model input\_reader\_packetIn

00:00:00:690

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 7}] generated by model input\_reader\_packetIn

00:00:00:695

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 8}] generated by model input\_reader\_packetIn

00:00:00:700

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 9}] generated by model input\_reader\_packetIn

00:00:00:705

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 10}] generated by model input\_reader\_packetIn

00:00:00:780

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 5}] generated by model input\_reader\_ackIn

00:00:00:790

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 7}] generated by model input\_reader\_ackIn

00:00:00:805

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 10}] generated by model input\_reader\_ackIn

00:00:01:305

[Retransmission\_defs::sizeChange: {}, Retransmission\_defs::resend: {}] generated by model Retransmission1

//only 3 ackIn received, data loss is 70. Send sizeChange to PacketGenerator and start to forward resend packets to PacketForwarder

00:00:01:310

[Retransmission\_defs::sizeChange: {5}, Retransmission\_defs::resend: {10 9}] generated by model Retransmission1

00:00:01:315

[Retransmission\_defs::sizeChange: {5}, Retransmission\_defs::resend: {10 8}] generated by model Retransmission1

00:00:01:320

[Retransmission\_defs::sizeChange: {5}, Retransmission\_defs::resend: {10 6}] generated by model Retransmission1

00:00:01:325

[Retransmission\_defs::sizeChange: {5}, Retransmission\_defs::resend: {10 4}] generated by model Retransmission1

00:00:01:330

[Retransmission\_defs::sizeChange: {5}, Retransmission\_defs::resend: {10 3}] generated by model Retransmission1

00:00:01:335

[Retransmission\_defs::sizeChange: {5}, Retransmission\_defs::resend: {10 2}] generated by model Retransmission1

00:00:01:340

[Retransmission\_defs::sizeChange: {5}, Retransmission\_defs::resend: {10 1}] generated by model Retransmission1

// finished forwarding all resend packets, send packet with size 0 to tell PacketForwarder resend is done.

00:00:01:345

[Retransmission\_defs::sizeChange: {5}, Retransmission\_defs::resend: {0 0}] generated by model Retransmission1

### NetworkDelay

It receives packets, sends received packets out with a fixed delay of 50 ms. The packets will come one after another within 50 ms. Due to the DEVS formalism and feature of Cadmium, a Queue is implemented as this atomic module will have more than one packet in delay process (ex. Packets arrive at a rate of 1 packet per 5ms).

Run scripts: ./NetworkDelay\_test.sh

//input packetIn

00:00:00:001 10 1

00:00:00:002 10 2

00:00:00:003 10 3

00:00:00:004 10 4

00:00:00:005 10 5

00:00:00:006 10 6

00:00:00:007 10 7

00:00:00:008 10 8

00:00:00:009 10 9

00:00:00:010 10 10

//test output

00:00:00:001

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 1}] generated by model input\_reader

00:00:00:002

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 2}] generated by model input\_reader

00:00:00:003

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 3}] generated by model input\_reader

00:00:00:004

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 4}] generated by model input\_reader

00:00:00:005

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 5}] generated by model input\_reader

00:00:00:006

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 6}] generated by model input\_reader

00:00:00:007

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 7}] generated by model input\_reader

00:00:00:008

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 8}] generated by model input\_reader

00:00:00:009

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 9}] generated by model input\_reader

00:00:00:010

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 10}] generated by model input\_reader

00:00:00:051 // 50 ms delay compare with receving time at 1 ms

[NetworkDelay\_defs::out: {10 1}] generated by model NetworkDelay1

00:00:00:052

[NetworkDelay\_defs::out: {10 2}] generated by model NetworkDelay1

00:00:00:053

[NetworkDelay\_defs::out: {10 3}] generated by model NetworkDelay1

00:00:00:054

[NetworkDelay\_defs::out: {10 4}] generated by model NetworkDelay1

00:00:00:055

[NetworkDelay\_defs::out: {10 5}] generated by model NetworkDelay1

00:00:00:056

[NetworkDelay\_defs::out: {10 6}] generated by model NetworkDelay1

00:00:00:057

[NetworkDelay\_defs::out: {10 7}] generated by model NetworkDelay1

00:00:00:058

[NetworkDelay\_defs::out: {10 8}] generated by model NetworkDelay1

00:00:00:059

[NetworkDelay\_defs::out: {10 9}] generated by model NetworkDelay1

00:00:00:060

[NetworkDelay\_defs::out: {10 10}] generated by model NetworkDelay1

### SwitchBuffer

It has a random generator inside to simulate the available buffer size. If a packet’s size is smaller than the generated buffer size, packet is sent out. Otherwise, packet dropped. As a result, packets with smaller size should have smaller chance of bein g dropped. The generated random number is evenly distributed and casted to the range of packet size from 0 to 40.

Run scripts: ./SwitchBuffer\_test.sh

//input packetIn

00:00:01 10 1

00:00:02 10 2

00:00:03 10 3

00:00:04 10 4

00:00:05 10 5

00:00:06 10 6

00:00:07 10 7

00:00:08 10 8

00:00:09 10 9

00:00:10 10 10

00:01:11 20 1

00:01:12 20 2

00:01:13 20 3

00:01:14 20 4

00:01:15 20 5

00:01:16 20 6

00:01:17 20 7

00:01:18 20 8

00:01:19 20 9

00:01:20 20 10

//testing output

00:00:00:000

00:00:00:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {}] generated by model input\_reader

00:00:01:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 1}] generated by model input\_reader

00:00:02:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 2}] generated by model input\_reader

//This is the packet being sent out. If not shown as being generated by the SwitchBuffer, it means being dropped.

00:00:02:000

[SwitchBuffer\_defs::out: {10 2}] generated by model SwitchBuffer1

00:00:03:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 3}] generated by model input\_reader

00:00:03:000

[SwitchBuffer\_defs::out: {10 3}] generated by model SwitchBuffer1

00:00:04:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 4}] generated by model input\_reader

00:00:04:000

[SwitchBuffer\_defs::out: {10 4}] generated by model SwitchBuffer1

00:00:05:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 5}] generated by model input\_reader

00:00:05:000

[SwitchBuffer\_defs::out: {10 5}] generated by model SwitchBuffer1

00:00:06:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 6}] generated by model input\_reader

00:00:06:000

[SwitchBuffer\_defs::out: {10 6}] generated by model SwitchBuffer1

00:00:07:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 7}] generated by model input\_reader

00:00:07:000

[SwitchBuffer\_defs::out: {10 7}] generated by model SwitchBuffer1

00:00:08:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 8}] generated by model input\_reader

00:00:08:000

[SwitchBuffer\_defs::out: {10 8}] generated by model SwitchBuffer1

00:00:09:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 9}] generated by model input\_reader

00:00:10:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 10}] generated by model input\_reader

00:00:10:000

[SwitchBuffer\_defs::out: {10 10}] generated by model SwitchBuffer1

//total of 8/10 packets sent out, 20% drop rate with packet size 10

00:01:11:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {20 1}] generated by model input\_reader

00:01:12:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {20 2}] generated by model input\_reader

00:01:12:000

[SwitchBuffer\_defs::out: {20 2}] generated by model SwitchBuffer1

00:01:13:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {20 3}] generated by model input\_reader

00:01:13:000

[SwitchBuffer\_defs::out: {20 3}] generated by model SwitchBuffer1

00:01:14:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {20 4}] generated by model input\_reader

00:01:14:000

[SwitchBuffer\_defs::out: {20 4}] generated by model SwitchBuffer1

00:01:15:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {20 5}] generated by model input\_reader

00:01:16:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {20 6}] generated by model input\_reader

00:01:16:000

[SwitchBuffer\_defs::out: {20 6}] generated by model SwitchBuffer1

00:01:17:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {20 7}] generated by model input\_reader

00:01:18:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {20 8}] generated by model input\_reader

00:01:19:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {20 9}] generated by model input\_reader

00:01:20:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {20 10}] generated by model input\_reader

00:01:20:000

[SwitchBuffer\_defs::out: {20 10}] generated by model SwitchBuffer1

//total of 5/10 packets sent out, 50% drop rate with packet size 20

### Receiver

It receives packets and sends packets (ACK) out right away with no delay.

Run scripts: ./Receiver\_test.sh

//input packetIn

00:00:10 13 1

00:00:30 22 2

00:00:45 53 3

00:00:52 37 4

00:01:25 41 5

00:01:35 94 6

00:01:55 54 7

//testing output

00:00:00:000

00:00:00:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {}] generated by model input\_reader

00:00:10:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {13 1}] generated by model input\_reader

00:00:10:000

[Receiver\_defs::out: {13 1}] generated by model receiver1

00:00:30:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {22 2}] generated by model input\_reader

00:00:30:000

[Receiver\_defs::out: {22 2}] generated by model receiver1

00:00:45:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {53 3}] generated by model input\_reader

00:00:45:000

[Receiver\_defs::out: {53 3}] generated by model receiver1

00:00:52:000 //receive

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {37 4}] generated by model input\_reader

00:00:52:000 //send it once receive it

[Receiver\_defs::out: {37 4}] generated by model receiver1

00:01:25:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {41 5}] generated by model input\_reader

00:01:25:000

[Receiver\_defs::out: {41 5}] generated by model receiver1

00:01:35:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {94 6}] generated by model input\_reader

00:01:35:000

[Receiver\_defs::out: {94 6}] generated by model receiver1

00:01:55:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {54 7}] generated by model input\_reader

00:01:55:000

[Receiver\_defs::out: {54 7}] generated by model receiver1

## Coupled Model

### PacketSender

PacketSender consists of Retransmission Controller and PacketForwarder. This coupled model receives packet from PacketGenerator and ACKs from the network. For every group of 10 packets, it receives one packet per time and sends nextPacket signal to PacketGenerator. At the end of this cycle, it determines if any packets need to be resent based on the ACKs received.

Run scripts: ./PacketSender\_test.sh

//input packetIn (packetDone)

00:00:01 10 1

00:00:02 10 2

00:00:03 10 3

00:00:04 10 4

00:00:05 10 5

00:00:06 10 6

00:00:07 10 7

00:00:08 10 8

00:00:09 10 9

00:00:10 10 10 //first group of 10 packets

00:01:11 10 1

00:01:12 10 2

00:01:13 10 3

00:01:14 10 4

00:01:15 10 5

00:01:16 10 6

00:01:17 10 7

00:01:18 10 8

00:01:19 10 9

00:01:20 10 10 //second group of 10 packets

//input ackIn

00:00:01:500 10 1

00:00:02:500 10 2

00:00:03:500 10 3

00:00:04:500 10 4

00:00:08:500 10 8

00:00:10:500 10 10 //receives 6/10 ACKs, resend needed

00:01:11:500 10 1

00:01:12:500 10 2

00:01:13:500 10 3

00:01:14:500 10 4

00:01:15:500 10 5

00:01:16:500 10 6

00:01:17:500 10 7

00:01:18:500 10 8

00:01:19:500 10 9

00:01:20:500 10 10 //receives 10/10 ACKs

//testing output

00:00:01:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 1}] generated by model input\_reader\_packet\_done

00:00:01:005

[PacketForwarder\_defs::packetOut: {10 1}, PacketForwarder\_defs::nextPacket: {1}] generated by model PacketForwarder1

00:00:01:500

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 1}] generated by model input\_reader\_ack

Skipped Messages

[PacketForwarder\_defs::packetOut: {10 10}, PacketForwarder\_defs::nextPacket: {}] generated by model PacketForwarder1

00:00:10:500 //receives 6/10 ACKs, resend needed

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 10}] generated by model input\_reader\_ack

00:00:10:605

[Retransmission\_defs::sizeChange: {}, Retransmission\_defs::resend: {}] generated by model Retransmission1

00:00:10:610 //too much data loss, size reduced, send sizeChange to PacketGenerator

[Retransmission\_defs::sizeChange: {5}, Retransmission\_defs::resend: {10 9}] generated by model Retransmission1

00:00:10:615

[Retransmission\_defs::sizeChange: {5}, Retransmission\_defs::resend: {10 7}] generated by model Retransmission1

00:00:10:620

[Retransmission\_defs::sizeChange: {5}, Retransmission\_defs::resend: {10 6}] generated by model Retransmission1

00:00:10:625

[Retransmission\_defs::sizeChange: {5}, Retransmission\_defs::resend: {10 5}] generated by model Retransmission1

00:00:10:630

[Retransmission\_defs::sizeChange: {5}, Retransmission\_defs::resend: {0 0}] generated by model Retransmission1

00:00:10:630

[PacketForwarder\_defs::packetOut: {}, PacketForwarder\_defs::nextPacket: {}] generated by model PacketForwarder1

00:00:10:635

[PacketForwarder\_defs::packetOut: {10 5}, PacketForwarder\_defs::nextPacket: {}] generated by model PacketForwarder1

00:00:10:640

[PacketForwarder\_defs::packetOut: {10 6}, PacketForwarder\_defs::nextPacket: {}] generated by model PacketForwarder1

00:00:10:645

[PacketForwarder\_defs::packetOut: {10 7}, PacketForwarder\_defs::nextPacket: {}] generated by model PacketForwarder1

00:00:10:650

[PacketForwarder\_defs::packetOut: {10 9}, PacketForwarder\_defs::nextPacket: {}] generated by model PacketForwarder1

00:00:11:250 //finished retransmission, send nextPacket to PacketGenerator

[PacketForwarder\_defs::packetOut: {}, PacketForwarder\_defs::nextPacket: {1}] generated by model PacketForwarder1

00:01:11:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 1}] generated by model input\_reader\_packet\_done

00:01:11:005

[PacketForwarder\_defs::packetOut: {10 1}, PacketForwarder\_defs::nextPacket: {1}] generated by model PacketForwarder1

00:01:11:500

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 1}] generated by model input\_reader\_ack

00:01:12:000

Skipped Messages

00:01:20:000 // receives 10/10 ACKs

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 10}] generated by model input\_reader\_packet\_done

00:01:20:005

[PacketForwarder\_defs::packetOut: {10 10}, PacketForwarder\_defs::nextPacket: {}] generated by model PacketForwarder1

00:01:20:500

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 10}] generated by model input\_reader\_ack

00:01:20:605

[Retransmission\_defs::sizeChange: {}, Retransmission\_defs::resend: {}] generated by model Retransmission1

00:01:20:610 //no packet loss, double the size to 20, send sizeChange to PacketGenerator

[Retransmission\_defs::sizeChange: {20}, Retransmission\_defs::resend: {0 0}] generated by model Retransmission1

00:01:20:610 //send nextPacket to the PacketGenerator

[PacketForwarder\_defs::packetOut: {}, PacketForwarder\_defs::nextPacket: {1}] generated by model PacketForwarder1

### ImageSender

This coupled model consists of 3 atomic models: PacketGenerator, PacketForwarder and Retransmission. 3 atomic models have already been tested above including the coupled model PacketSender (consists of PacketForarder and Retransmission).

This model receives a startSignal (PacketGenerator receives this signal), extract the total size of data to be sent from the startSignal, send 10 packets one by one as a cycle, waits for Acks with timeout 600ms after all 10 packets sent. It resends the lost packets if data loss is more than acceptable loss limit. Adjust the packet size based on packet loss. (more loss, smaller size and vice versa).

Run scripts: ./ImageSender\_test.sh

#### First Test Case where no resend needed

//input startSignal

00:00:00 100

//input ackIn signal

00:00:00:605 10 1

00:00:00:610 10 2

00:00:00:615 10 3

00:00:00:620 10 4

00:00:00:625 10 5

00:00:00:630 10 6

00:00:00:635 10 7

00:00:00:640 10 8

00:00:00:645 10 9

00:00:00:650 10 10 //all ACKs received

//testing output

00:00:00:000

00:00:00:000

[cadmium::basic\_models::pdevs::iestream\_input\_defs<int>::out: {}] generated by model input\_reader\_start\_signal

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {}] generated by model input\_reader\_ack

00:00:00:000 //received startSignal, total data is 100 size

[cadmium::basic\_models::pdevs::iestream\_input\_defs<int>::out: {100}] generated by model input\_reader\_start\_signal

00:00:00:000 //start to generate and send packets

[PacketGenerator\_defs::packetDone: {10 1}] generated by model PacketGenerator1

00:00:00:005

[PacketForwarder\_defs::packetOut: {10 1}, PacketForwarder\_defs::nextPacket: {1}] generated by model PacketForwarder1

00:00:00:005 //Generate packets

[PacketGenerator\_defs::packetDone: {10 2}] generated by model PacketGenerator1

00:00:00:010 //Send packets

[PacketForwarder\_defs::packetOut: {10 2}, PacketForwarder\_defs::nextPacket: {1}] generated by model PacketForwarder1

00:00:00:010

[PacketGenerator\_defs::packetDone: {10 3}] generated by model PacketGenerator1

00:00:00:015

[PacketForwarder\_defs::packetOut: {10 3}, PacketForwarder\_defs::nextPacket: {1}] generated by model PacketForwarder1

00:00:00:015

[PacketGenerator\_defs::packetDone: {10 4}] generated by model PacketGenerator1

00:00:00:020

[PacketForwarder\_defs::packetOut: {10 4}, PacketForwarder\_defs::nextPacket: {1}] generated by model PacketForwarder1

00:00:00:020

[PacketGenerator\_defs::packetDone: {10 5}] generated by model PacketGenerator1

00:00:00:025

[PacketForwarder\_defs::packetOut: {10 5}, PacketForwarder\_defs::nextPacket: {1}] generated by model PacketForwarder1

00:00:00:025

[PacketGenerator\_defs::packetDone: {10 6}] generated by model PacketGenerator1

00:00:00:030

[PacketForwarder\_defs::packetOut: {10 6}, PacketForwarder\_defs::nextPacket: {1}] generated by model PacketForwarder1

00:00:00:030

[PacketGenerator\_defs::packetDone: {10 7}] generated by model PacketGenerator1

00:00:00:035

[PacketForwarder\_defs::packetOut: {10 7}, PacketForwarder\_defs::nextPacket: {1}] generated by model PacketForwarder1

00:00:00:035

[PacketGenerator\_defs::packetDone: {10 8}] generated by model PacketGenerator1

00:00:00:040

[PacketForwarder\_defs::packetOut: {10 8}, PacketForwarder\_defs::nextPacket: {1}] generated by model PacketForwarder1

00:00:00:040

[PacketGenerator\_defs::packetDone: {10 9}] generated by model PacketGenerator1

00:00:00:045

[PacketForwarder\_defs::packetOut: {10 9}, PacketForwarder\_defs::nextPacket: {1}] generated by model PacketForwarder1

00:00:00:045

[PacketGenerator\_defs::packetDone: {10 10}] generated by model PacketGenerator1

00:00:00:050

[PacketForwarder\_defs::packetOut: {10 10}, PacketForwarder\_defs::nextPacket: {}] generated by model PacketForwarder1

00:00:00:605 //start receiving ACKs from the input file which simulates the network

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 1}] generated by model input\_reader\_ack

00:00:00:610

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 2}] generated by model input\_reader\_ack

00:00:00:615

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 3}] generated by model input\_reader\_ack

00:00:00:620

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 4}] generated by model input\_reader\_ack

00:00:00:625

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 5}] generated by model input\_reader\_ack

00:00:00:630

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 6}] generated by model input\_reader\_ack

00:00:00:635

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 7}] generated by model input\_reader\_ack

00:00:00:640

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 8}] generated by model input\_reader\_ack

00:00:00:645

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 9}] generated by model input\_reader\_ack

00:00:00:650

[cadmium::basic\_models::pdevs::iestream\_input\_defs<Packet\_t>::out: {10 10}] generated by model input\_reader\_ack

[Retransmission\_defs::sizeChange: {}, Retransmission\_defs::resend: {}] generated by model Retransmission1

00:00:00:655 //all ACKs received, no data loss, change packet size to 20

[Retransmission\_defs::sizeChange: {20}, Retransmission\_defs::resend: {0 0}] generated by model Retransmission1

00:00:00:655

[PacketForwarder\_defs::packetOut: {}, PacketForwarder\_defs::nextPacket: {1}] generated by model PacketForwarder1

#### Second Test Case where packets resend is performed

//input startSignal

00:00:00 100

//input ackIn signal

00:00:00:605 10 1

00:00:00:610 10 2

00:00:00:615 10 3

00:00:00:620 10 4

00:00:00:645 10 9

00:00:00:650 10 10 //ACKs for 5,6,7,8 is not received, ACKs for 10 received at the edge of timeout

//testing output

00:00:00:000

Skipped Messages

00:00:00:655 //too much data loss, reduce size to 5, start resending

[Retransmission\_defs::sizeChange: {5}, Retransmission\_defs::resend: {10 10}] generated by model Retransmission1

00:00:00:660

[Retransmission\_defs::sizeChange: {5}, Retransmission\_defs::resend: {10 8}] generated by model Retransmission1

00:00:00:665

[Retransmission\_defs::sizeChange: {5}, Retransmission\_defs::resend: {10 7}] generated by model Retransmission1

00:00:00:670

[Retransmission\_defs::sizeChange: {5}, Retransmission\_defs::resend: {10 6}] generated by model Retransmission1

00:00:00:675

[Retransmission\_defs::sizeChange: {5}, Retransmission\_defs::resend: {10 5}] generated by model Retransmission1

00:00:00:680

[Retransmission\_defs::sizeChange: {5}, Retransmission\_defs::resend: {0 0}] generated by model Retransmission1

00:00:00:680

[PacketForwarder\_defs::packetOut: {}, PacketForwarder\_defs::nextPacket: {}] generated by model PacketForwarder1

00:00:00:685

[PacketForwarder\_defs::packetOut: {10 5}, PacketForwarder\_defs::nextPacket: {}] generated by model PacketForwarder1

00:00:00:690

[PacketForwarder\_defs::packetOut: {10 6}, PacketForwarder\_defs::nextPacket: {}] generated by model PacketForwarder1

00:00:00:695

[PacketForwarder\_defs::packetOut: {10 7}, PacketForwarder\_defs::nextPacket: {}] generated by model PacketForwarder1

00:00:00:700

[PacketForwarder\_defs::packetOut: {10 8}, PacketForwarder\_defs::nextPacket: {}] generated by model PacketForwarder1

00:00:00:705

[PacketForwarder\_defs::packetOut: {10 10}, PacketForwarder\_defs::nextPacket: {}] generated by model PacketForwarder1

00:00:01:305 //retransmission done

[PacketForwarder\_defs::packetOut: {}, PacketForwarder\_defs::nextPacket: {1}] generated by model PacketForwarder1

### Network

Network Model consists of models networkDelay and SwitchBuffer. This coupled model has submodule chained sequentially and packets are flowing in one direction. Since the atomic submodules have been tested already and the coupled model is simple, no testing performed for this coupled Network module.

### ITP top module

The top module couples all atomic modules. It only requires one signal startSignal. The signal specifies the total size of data (image size) to be sent. The result will be recorded in the output messages detailing the behavior of each atomic module. The overall behavior is same as described in part I.

Run scripts: ./ITP\_top\_model\_test.sh

#### First test case: only data of size 10 to be sent (one packet go through the network)

//input startSignal

00:00:00 10

//testing output

00:00:00:000 //startSignal

[cadmium::basic\_models::pdevs::iestream\_input\_defs<int>::out: {10}] generated by model input\_reader\_start\_signal

00:00:00:000 //generator generates packet

[PacketGenerator\_defs::packetDone: {10 1}] generated by model PacketGenerator1

00:00:00:005 //first packet with sequence 1 sent out

[PacketForwarder\_defs::packetOut: {10 1}, PacketForwarder\_defs::nextPacket: {1}] generated by model PacketForwarder1

00:00:00:055

[NetworkDelay\_defs::out: {10 1}] generated by model NetworkDelay1

00:00:00:055

[SwitchBuffer\_defs::out: {10 1}] generated by model SwitchBuffer1

00:00:00:055

[Receiver\_defs::out: {10 1}] generated by model Receiver1

00:00:00:105

[NetworkDelay\_defs::out: {10 1}] generated by model NetworkDelay2

00:00:00:105 //packet (ACK) sent out by SwitchBuffer2 in second network, indicates ACK received by sender

[SwitchBuffer\_defs::out: {10 1}] generated by model SwitchBuffer2

#### Second Test Case: data of size 200 to be sent (need at least two cycles)

Too many messages are generated for this test case, only most important one is selected.

//input startSignal

00:00:00 200

//testing output

00:00:00:000 //startSignal, image(total data) size is 120

[cadmium::basic\_models::pdevs::iestream\_input\_defs<int>::out: {120}] generated by model input\_reader\_start\_signal

00:00:00:000 //first packet generated

[PacketGenerator\_defs::packetDone: {10 1}] generated by model PacketGenerator1

00:00:00:005 //first packet sent out, signal for next packet

[PacketForwarder\_defs::packetOut: {10 1}, PacketForwarder\_defs::nextPacket: {1}] generated by model PacketForwarder1

Skipped Messages

00:00:00:045

[PacketGenerator\_defs::packetDone: {10 10}] generated by model PacketGenerator1

00:00:00:050

[PacketForwarder\_defs::packetOut: {10 10}, PacketForwarder\_defs::nextPacket: {}] generated by model PacketForwarder1

00:00:00:055

[NetworkDelay\_defs::out: {10 1}] generated by model NetworkDelay1

00:00:00:055

[SwitchBuffer\_defs::out: {10 1}] generated by model SwitchBuffer1

00:00:00:055

[Receiver\_defs::out: {10 1}] generated by model Receiver1

Skipped Messages

00:00:00:100

[NetworkDelay\_defs::out: {10 10}] generated by model NetworkDelay1

00:00:00:100

[SwitchBuffer\_defs::out: {10 10}] generated by model SwitchBuffer1

00:00:00:100

[Receiver\_defs::out: {10 10}] generated by model Receiver1

00:00:00:105

[NetworkDelay\_defs::out: {10 1}] generated by model NetworkDelay2

00:00:00:110

[NetworkDelay\_defs::out: {10 2}] generated by model NetworkDelay2

00:00:00:110 //SwitchBuffer2’s output is actually ACK sent to sender

[SwitchBuffer\_defs::out: {10 2}] generated by model SwitchBuffer2

00:00:00:115

[NetworkDelay\_defs::out: {10 3}] generated by model NetworkDelay2

00:00:00:115

[SwitchBuffer\_defs::out: {10 3}] generated by model SwitchBuffer2

00:00:00:120

[NetworkDelay\_defs::out: {10 4}] generated by model NetworkDelay2

00:00:00:120

[SwitchBuffer\_defs::out: {10 4}] generated by model SwitchBuffer2

00:00:00:125

[NetworkDelay\_defs::out: {10 5}] generated by model NetworkDelay2

00:00:00:125

[SwitchBuffer\_defs::out: {10 5}] generated by model SwitchBuffer2

00:00:00:130

[NetworkDelay\_defs::out: {10 6}] generated by model NetworkDelay2

00:00:00:135

[NetworkDelay\_defs::out: {10 7}] generated by model NetworkDelay2

00:00:00:135

[SwitchBuffer\_defs::out: {10 7}] generated by model SwitchBuffer2

00:00:00:145

[NetworkDelay\_defs::out: {10 9}] generated by model NetworkDelay2

00:00:00:145

[SwitchBuffer\_defs::out: {10 9}] generated by model SwitchBuffer2

00:00:00:150

[NetworkDelay\_defs::out: {10 10}] generated by model NetworkDelay2

00:00:00:150

[SwitchBuffer\_defs::out: {10 10}] generated by model SwitchBuffer2 //7 ACKs received, 3 packets lost

00:00:00:655 //start resend and retransmission

[Retransmission\_defs::sizeChange: {10}, Retransmission\_defs::resend: {10 8}] generated by model Retransmission1

00:00:00:655

[PacketGenerator\_defs::packetDone: {}] generated by model PacketGenerator1

00:00:00:660

[Retransmission\_defs::sizeChange: {10}, Retransmission\_defs::resend: {10 6}] generated by model Retransmission1

00:00:00:660

[PacketGenerator\_defs::packetDone: {}] generated by model PacketGenerator1

00:00:00:665

[Retransmission\_defs::sizeChange: {10}, Retransmission\_defs::resend: {10 1}] generated by model Retransmission1

00:00:00:665

[PacketGenerator\_defs::packetDone: {}] generated by model PacketGenerator1

00:00:00:670

[Retransmission\_defs::sizeChange: {10}, Retransmission\_defs::resend: {0 0}] generated by model Retransmission1

00:00:00:670

[PacketGenerator\_defs::packetDone: {}] generated by model PacketGenerator1

[PacketForwarder\_defs::packetOut: {}, PacketForwarder\_defs::nextPacket: {}] generated by model PacketForwarder1

00:00:00:675

[PacketForwarder\_defs::packetOut: {10 1}, PacketForwarder\_defs::nextPacket: {}] generated by model PacketForwarder1

00:00:00:680

[PacketForwarder\_defs::packetOut: {10 6}, PacketForwarder\_defs::nextPacket: {}] generated by model PacketForwarder1

00:00:00:685

[PacketForwarder\_defs::packetOut: {10 8}, PacketForwarder\_defs::nextPacket: {}] generated by model PacketForwarder1

00:00:01:285 //retransmission done, signal for nextPacket

[PacketForwarder\_defs::packetOut: {}, PacketForwarder\_defs::nextPacket: {1}] generated by model PacketForwarder1

00:00:01:285

[PacketGenerator\_defs::packetDone: {10 1}] generated by model PacketGenerator1

00:00:01:290// First packet of second round is sent

[PacketForwarder\_defs::packetOut: {10 1}, PacketForwarder\_defs::nextPacket: {1}] generated by model PacketForwarder1

Skipped Messages

#### Third Test Case: data loss is too much, and packet changed from 10 to 5

Same input as above, run the testing multiple times, pick up the one in which Network drops more packets

[SwitchBuffer\_defs::out: {10 1}] generated by model SwitchBuffer2

00:00:00:110

[NetworkDelay\_defs::out: {10 2}] generated by model NetworkDelay2

00:00:00:110

[SwitchBuffer\_defs::out: {10 2}] generated by model SwitchBuffer2

00:00:00:120

[NetworkDelay\_defs::out: {10 4}] generated by model NetworkDelay2

00:00:00:125

[NetworkDelay\_defs::out: {10 5}] generated by model NetworkDelay2

00:00:00:125

[SwitchBuffer\_defs::out: {10 5}] generated by model SwitchBuffer2

00:00:00:130

[NetworkDelay\_defs::out: {10 6}] generated by model NetworkDelay2

00:00:00:130

[SwitchBuffer\_defs::out: {10 6}] generated by model SwitchBuffer2

00:00:00:135

[NetworkDelay\_defs::out: {10 7}] generated by model NetworkDelay2

00:00:00:135

[SwitchBuffer\_defs::out: {10 7}] generated by model SwitchBuffer2

00:00:00:140

[NetworkDelay\_defs::out: {10 8}] generated by model NetworkDelay2

00:00:00:145

[NetworkDelay\_defs::out: {10 9}] generated by model NetworkDelay2

00:00:00:145

[SwitchBuffer\_defs::out: {10 9}] generated by model SwitchBuffer2 //total of 6 ACKs received, 40% data loss

00:00:00:650

[Retransmission\_defs::sizeChange: {}, Retransmission\_defs::resend: {}] generated by model Retransmission1

00:00:00:655 //packet size changed to 5

[Retransmission\_defs::sizeChange: {5}, Retransmission\_defs::resend: {10 10}] generated by model Retransmission1

# Discussion

The atomic and coupled model testing have verified that the implementation is same as expected. Due to time constraint, delay parameter such as forward time, time out and network delay are fixed. A more realistic simulation should have a bounded random network delay. Such random delay will cause more than one input being received by some atomic models. This requires more refined implementation to deal with selection function. Overall speaking, the implementation has covered the most important features of desired modeling and it will be useful to study the impact of packet size for our image transfer protocol under various network conditions.