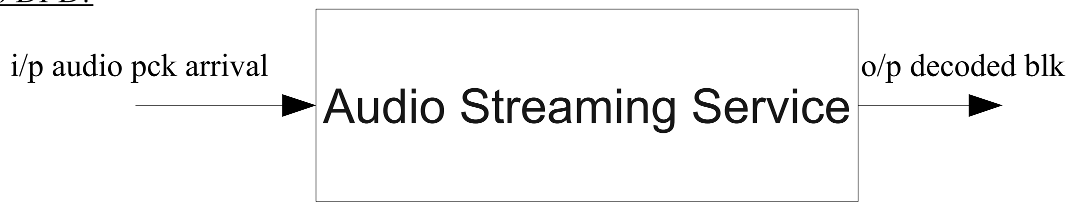
EE5903-CA2-Report

Srikanth Ayalasomayajulu-A0077017

Introduction: This report represents the development of a component for a real-time audio streaming service which receives audio streaming packets from a server, decodes and processes them into audio blocks in turn sends them to the audio devices like head phones

Level 0 DFD:



Given Specifications and Description:

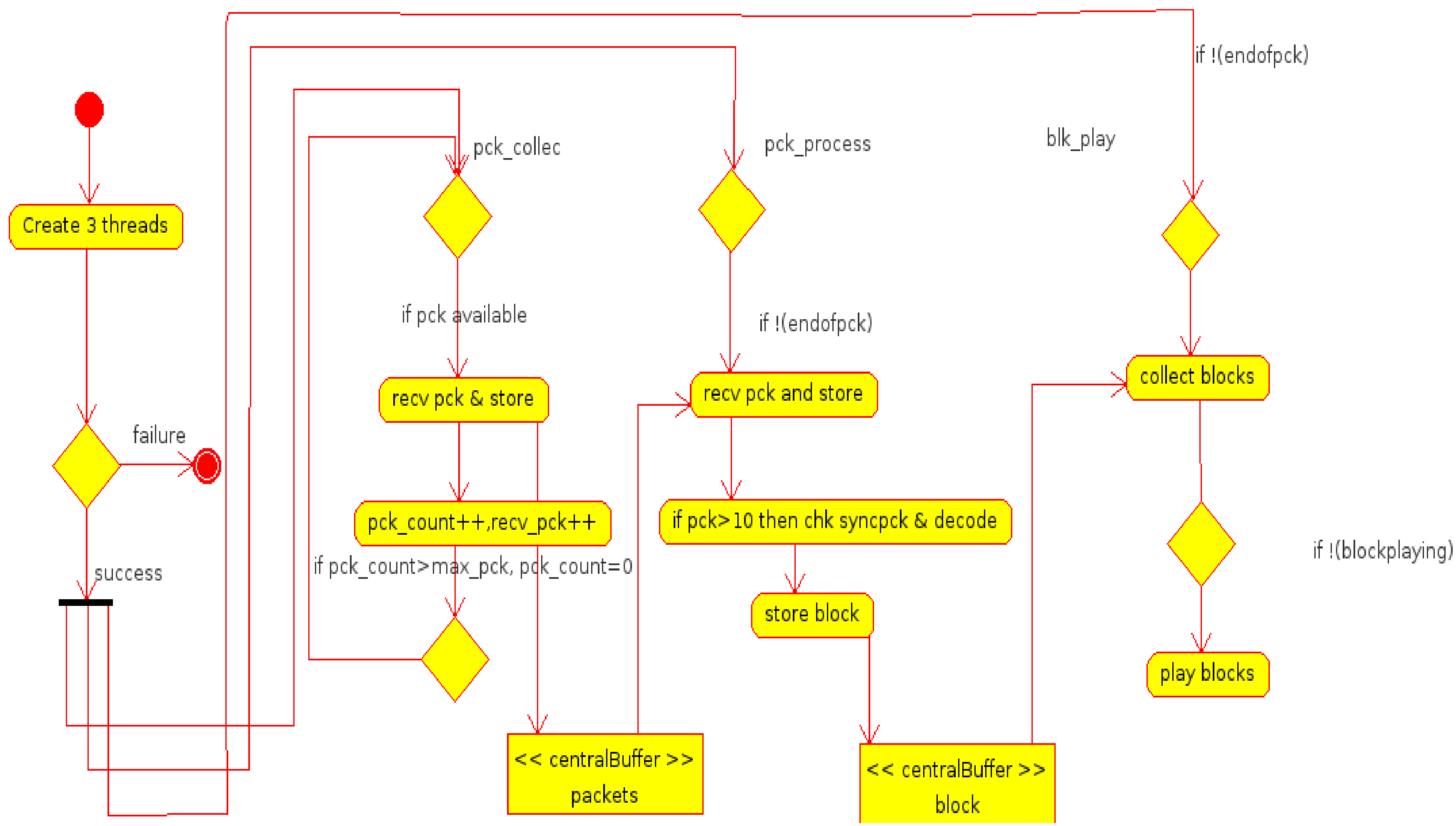
1. Given that arrival of packets are poison distributed whose cumulative distribution function(CDF) can be given as $e^{-\lambda}$. $(\lambda t^n)/n!$ Where λ is the arrival rate and t is the time so that number of packets that arrived in a given time interval can be given as $\rho= \lambda*t$.
2. Given that packets arrive with a minimum gap of 2ms between them and average arrival rate is 100 packets/sec.
3. 10 packets are required per audio block in which 10^{th} packets has to be a sync packet for successful decoding and 25ms is required to decode a block
4. Only one block is buffered at any point of time to ensure smooth audio play.

Description of implemented design:

- ✔ In the design, 3 threads are created to packet collection(*ipthrd*), packet processing to decode packets (*procthrd*) and play of the blocks(*opthrd*).
- ✔ Local buffers for both packets and block are created which can able to store 500 packets and 50 blocks.
- ✔ Two Mutex shared variables `mutexip, mutexproc` are used to handle synchronization between threads.
- ✔ Sync packets are checked at the end of every tenth packet and if it is sync packet, all the before 10 packets including sync packet is decoded and dropped other wise.
- ✔ All the buffers are resetted once they are reached to their maximum storage values.
- ✔ Reasonable sleeps are used to provide smooth play of the audio.
- ✔ Appropriate Joins are used to terminate the threads.

Activity Diagram:

The activity diagram is placed below in landscape mode.


Create 3 threads
failure
success
pck_collec
if pck available
recv pck & store
pck_count++,recv_pck++
if pck_count>max_pck, pck_count=0
<< centralBuffer >>
packets
pck_process
if !(endofpck)
recv pck and store
if pck>10 then chk syncpck & decode
store block
<< centralBuffer >>
block
blk_play
if !(endofpck)
collect blocks
if !(blockplaying)
play blocks