UDP Sample Code

Goal

- Implement very simple code that shows vp8 being used in a video conferencing type application.

The App

- Cross platform
  - (SDL for rendering, libvidcap for capture)
  - DSHOW / V4L2 there as well
  - SDL / DirectX for rendering
- Short (total 2k lines of code)
- Non production (ie lacks robustness)
- Work across the real internet (no Natting etc)
- Shows off VP8 features

Parameters

- Fec,
- Bitrate
- Frame Size and Frame Rate
- UDP loss behavior (recovery / resend)
- Simulated loss behavior
- Codec parameters (minq, maxq, bitrate fluctuation etc)

Sample GrabCompressAndSend Command Lines

./grabcompressandsend –c 12 -t 2000 –b 10 –q 45 –d 30 –i test.cpk

Set up to capture and transmit video leaving about 70% of the cpu free, have a pretty high threshold, set the min quantizer to 10, max to 45 drop the first 30 frames to allow the image to stabilize, and transmit to the machine test.cpk.

Sample ReceiveDecompressAndPlay Command Line

./receivedecompressandplay –w 640 –h 480 –f 30 –b 200 –n 6 –d 5 –l 20 –t 800 –i 50 –c 4

Set up to receive, decompress and play video. Request 640x480@30fps from the other side, with xor packets after 5 packets, simulate 2% loss, and give up on any packets after 800 ms, requesting resends every 50 ms up to 4 times.

Grab Compress and Send Data Structures

- Normal RTP packet with data
- Count till redundant Packet
- Packet starts frame?
- Packet ends Frame
- Frame Type (Normal, Key, Recovery 1 Recovery 2)

Packet

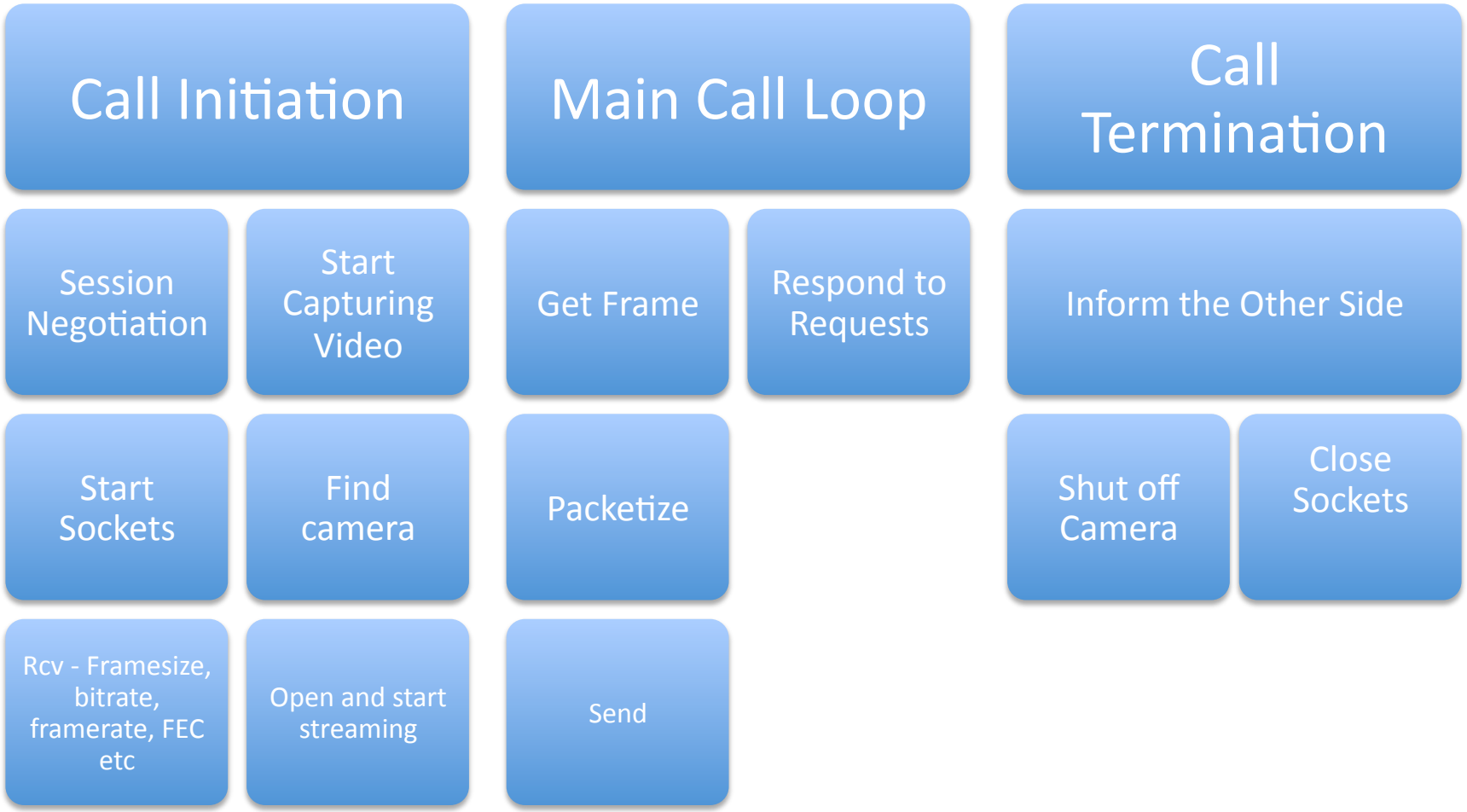
- Circular buffer of Size PSM using mask to Index
- Used to resend packets or create xor packets

Packet Store

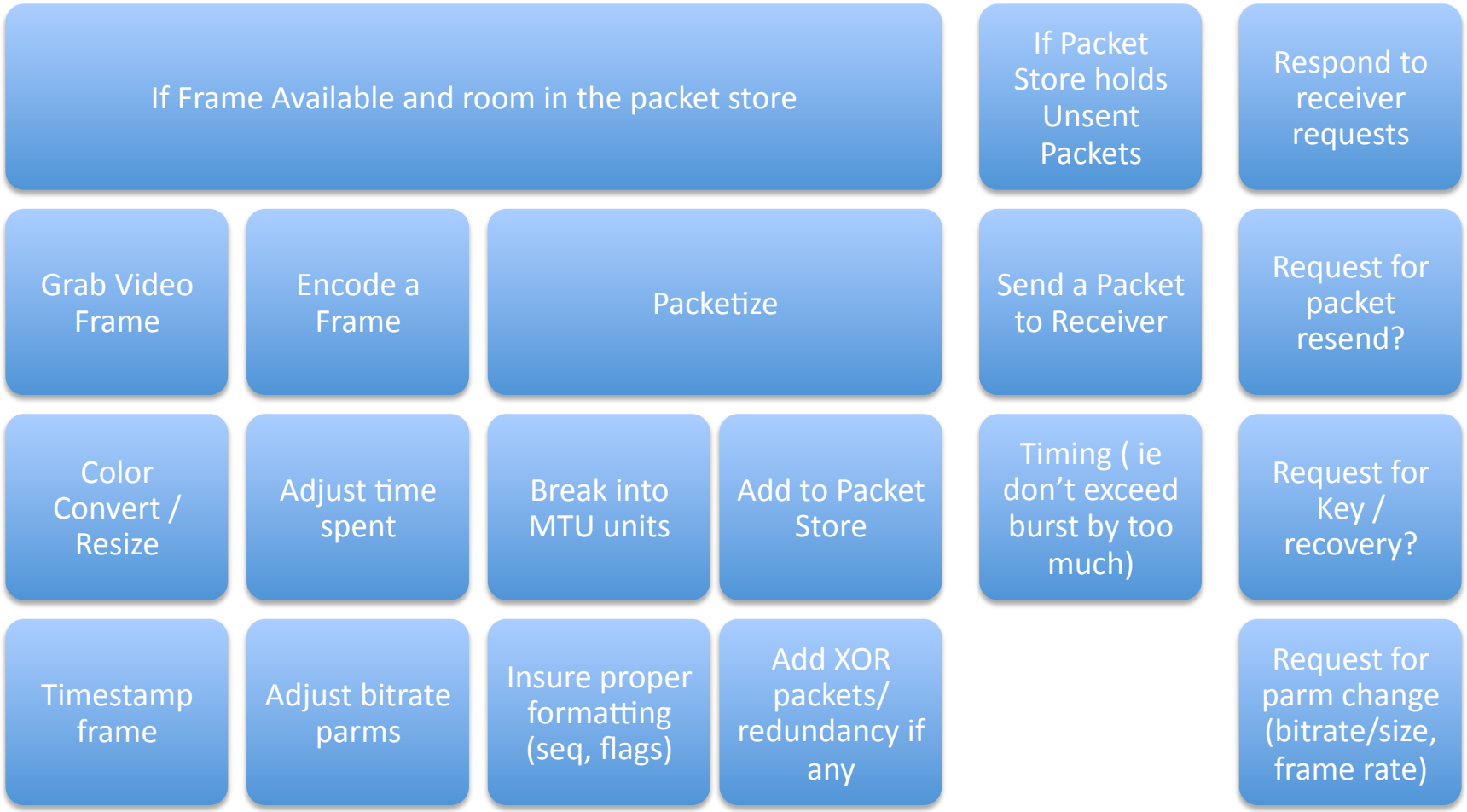
- FEC Data Type (Num/ Denom)
- FEC Position Index
- Packet Store Add / Write Position

Packetizer

Grab Compress and Send Tasks Overview



Grab Compress and Send Data Main Call Loop In More Detail



Recovery Requests

If missing seq is older than oldest recovery frame	If missing seq newer than newest recovery frame	If missing seq newer than the other recovery frame
Send Key	Send frame referring to and updating that recovery frame	Send frame referring to and updating that recovery frame (its now newest)

Receive Decompress and Play Data Structures

Packet

- Normal RTP packet with data
- Count till redundant Packet
- Packet starts frame?
- Packet ends Frame
- Frame Type (Normal, Key, Recovery 1 Recovery 2)

Packet Store

- Circular buffer of Size PSM using mask to Index
- Used to reorder packets and rebuild them using fec in the case of lost packets

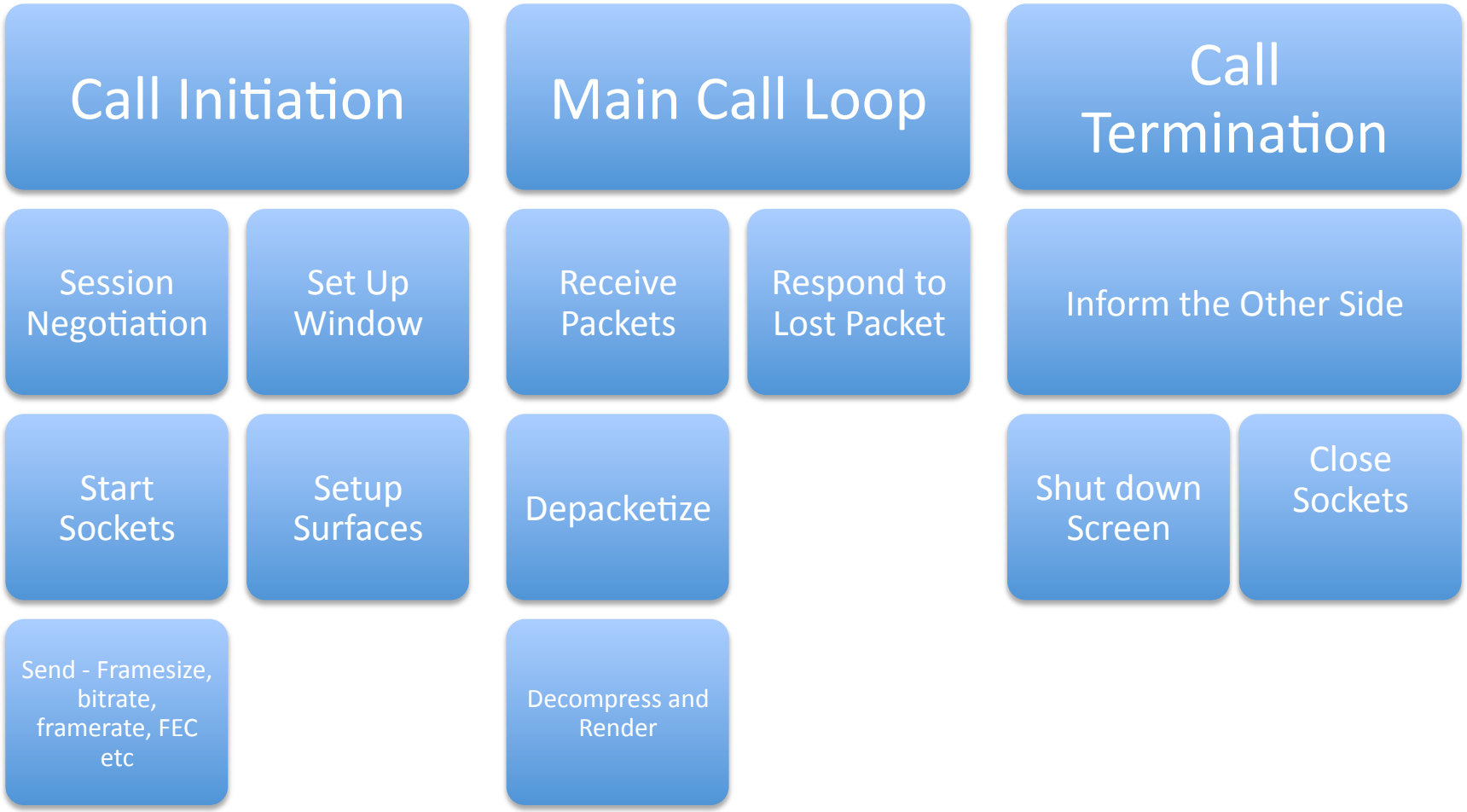
Depacketizer

- Skip Store Write Position
- Packet Store Add / Write Position
- Oldest Sequence we are waiting to play
- Last Frame Shown's Time Stamp

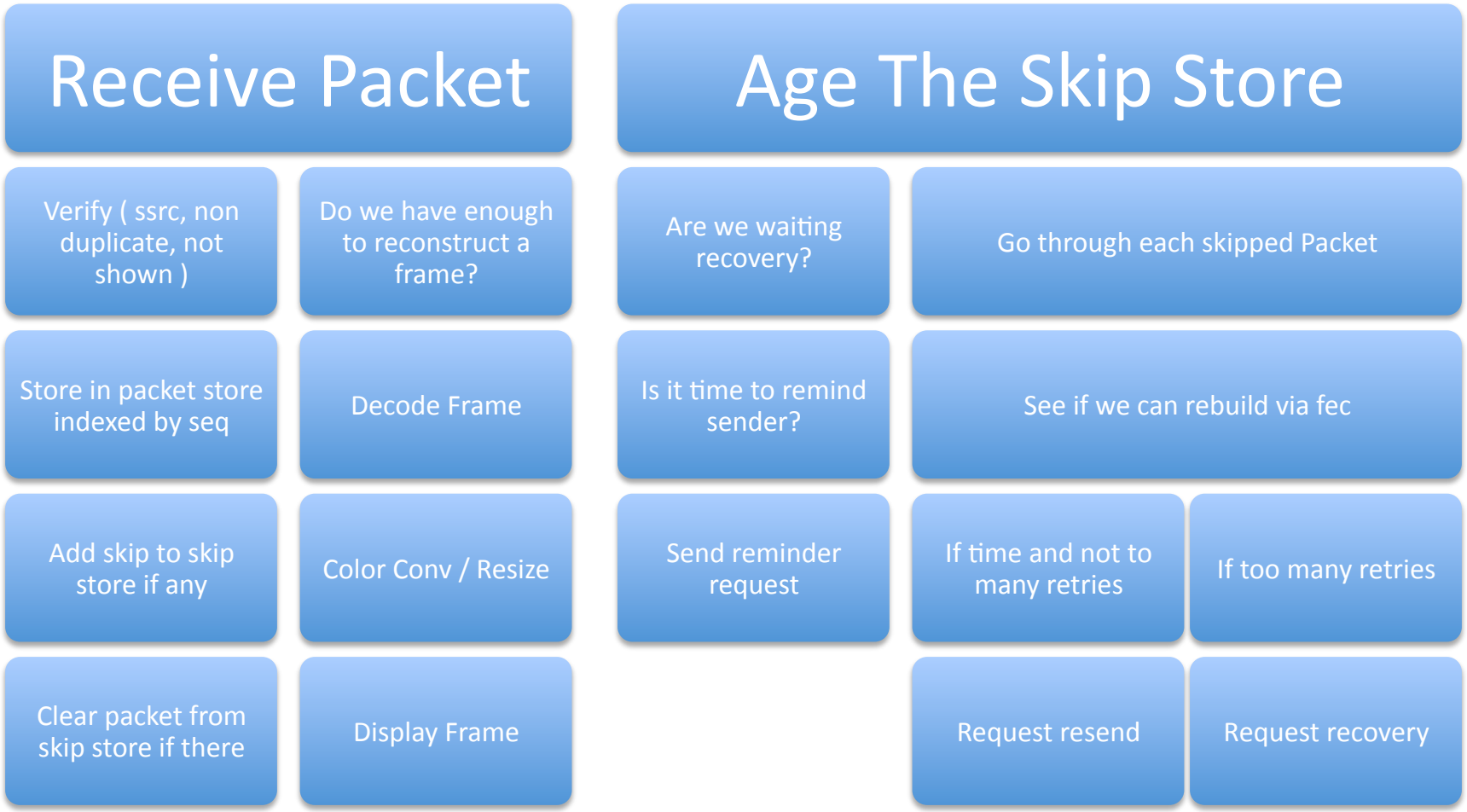
Skip Store

- Sequence number lost
- Retry count (if any)
- How long its been skipped
- Given Up
- Received

Receive Decompress and Play Tasks Overview



Receive Decompress and Play Main Loop



To DO

- Use Temporal scalable frame patterns
- Simulate multiparty conditions
- Simulate Lag
- Test on other platforms
- Output stats (Fps, bitrate, SNR)