Some Comments on Working Groups

- Embedded (general) -> Philip Ballister
- FPGA/DSP/GPU Co-Processing -> Justin Ford
- GPP/VOLK -> Nathan West
- Improving GNU Radio Experience -> M. Braun

Message Passing API
&
PHY/MAC with USRP and GNU Radio

John Malsbury, Ettus Research

Asynchronous Message Passing API

- Asynchronous Message Passing
- Publish/Subscriber Interface
- Use Cases
  - Both control and data plane
  - Low-rate upstream sync/control
  - Packet assembly/disassembly
- Polymorphic Data Types
- New Connect Function for Async Messages

Message Operation

Ettus Research

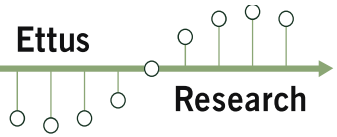
- Publish/Subscribe Model
  - Input ports are subscribed to a single output port
  - Blocks publish async messages on an output port
  - Scheduler inserts message into the input queues
  - Block subscription handled with the msg_connect()
- Register the port in block's __init__ function
- Specify msg_handler for input ports (optional but recommended)
- Two Methods to receive messages
  - Declare a message handler
  - Poll input queue in the work() function of a block (if it has one)

Creating a Port

Ettus Research

```
73  # ////////////////////////////////////////////////////////////////////////
74  #                     Simple MAC w/ ARQ
75  # ////////////////////////////////////////////////////////////////////////
76
77  class easyMAC(gr.basic_block):
78      """
79      docstring for block easyMAC
80      """
81      def __init__( self,addr,timeout,max_attempts):
82          gr.basic_block.__init__(self,
83              name="easyMAC",
84              in_sig=None,
85              out_sig=None)
86
87          #other init code
88
89          #message i/o for radio interface
90          self.message_port_register_out(pmt.intern('to_radio'))
91          self.message_port_register_in(pmt.intern('from_radio'))
92          self.set_msg_handler(pmt.intern('from_radio'),self.radio_rx)
93
94          #message i/o for app interface
95          self.message_port_register_out(pmt.intern('to_app'))
96          self.message_port_register_in(pmt.intern('from_app'))
97          self.set_msg_handler(pmt.intern('from_app'),self.app_rx)
98
99          #message i/o for ctrl interface
100         self.message_port_register_out(pmt.intern('ctrl_out'))
101         self.message_port_register_in(pmt.intern('ctrl_in'))
102         self.set_msg_handler(pmt.intern('ctrl_in'),self.ctrl_rx)
103
```

Message Connect




```
# Connections
##################################################
self.connect((self.digital_ofdm_tx_0, 0), (self.digital_ofdm_rx_0, 0))
self.connect((self.digital_ofdm_rx_0, 0), (self.blocks_tagged_stream_to_pdu_0, 0))
self.connect((self.blocks_pdu_to_tagged_stream_0, 0), (self.digital_ofdm_tx_0, 0))

##################################################
# Asynch Message Connections
##################################################
self.msg_connect(self.blocks_random_pdu_0, "pdus", self.easyMAC_virtual_channel_encoder_0, "in")
self.msg_connect(self.blocks_message_strobe_0, "strobe", self.blocks_random_pdu_0, "generate")
self.msg_connect(self.blocks_message_strobe_0_0, "strobe", self.blocks_random_pdu_0_0, "generate")
self.msg_connect(self.easyMAC_virtual_channel_encoder_0, "out", self.easyMAC_easyMAC_0_0, "from_app")
self.msg_connect(self.blocks_random_pdu_0_0, "pdus", self.easyMAC_easyMAC_0_0, "ctrl_in")
self.msg_connect(self.easyMAC_easyMAC_0_0, "to_radio", self.blocks_pdu_to_tagged_stream_0, "pdus")
self.msg_connect(self.blocks_tagged_stream_to_pdu_0, "pdus", self.blocks_message_debug_0, "print_pdu")
```

GRC XML File

Ettus Research

```
#message i/o for radio interface
self.message_port_register_out(pmt.intern('to_radio'))
self.message_port_register_in(pmt.intern('from_radio'))
self.set_msg_handler(pmt.intern('from_radio'),self.radio_rx)

#message i/o for app interface
self.message_port_register_out(pmt.intern('to_app'))
self.message_port_register_in(pmt.intern('from_app'))
self.set_msg_handler(pmt.intern('from_app'),self.app_rx)

#message i/o for ctrl interface
self.message_port_register_out(pmt.intern('ctrl_out'))
self.message_port_register_in(pmt.intern('ctrl_in'))
self.set_msg_handler(pmt.intern('ctrl_in'),self.ctrl_rx)
```

```
24  <sink>
25    <name>from_radio</name>
26    <type>message</type>
27  </sink>
28  <sink>
29    <name>from_app</name>
30    <type>message</type>
31  </sink>
32  <sink>
33    <name>ctrl_in</name>
34    <type>message</type>
35  </sink>
36
37  <source>
38    <name>to_radio</name>
39    <type>message</type>
40  </source>
41  <source>
42    <name>to_app</name>
43    <type>message</type>
44  </source>
45  <source>
46    <name>ctrl_out</name>
47    <type>message</type>
48  </source>
49 </block>
50
```

Message Handler vs work()

- Using a Message Handler
  - Msg functionality only called if a message arrived
  - Msg-only blocks “sleep” until a message arrives
  - Best if you are doing msg-to-msg exchange

- Polling in work()
  - Useful if you are converting from msg input to stream output
  - Risk of blocking on the msg queue
  - May reduce performance of DSP opeations in work()
  - Scheduler calls work() if:
    - Input/output buffer requirements are met
    - Message arrives at the input queue

Example of a Message Handler

Ettus Research

```
96          self.message_port_register_in(pmt.intern('from_app'))
97          self.set_msg_handler(pmt.intern('from_app'),self.app_rx)
```

```
227
228     def app_rx(self,msg):
229         try:
230             meta = pmt.car(msg)
231             data =  pmt.cdr(msg)
232         except:
233             raise NameError("easyMAC - input not a PDU")
234
235         if pmt.is_u8vector(data):
236             data = pmt.u8vector_elements(data)
237         else:
238             raise NameError("Data is not u8 vector")
239
240         meta_dict = pmt.to_python(meta)
241         if not (type(meta_dict) is dict):
242             meta_dict = {}
243
244         #double check to make sure correct meta data was in pdu
245         if 'EM_USE_ARQ' in meta_dict.keys() and 'EM_DEST_ADDR' in meta_dict.keys():
246             #assign tx path depending on whether PMT_BOOL EM_USE_ARQ is true or false
247             if(meta_dict['EM_USE_ARQ']):
248                 self.queue.put( (data,meta_dict) )
249             else:
250                 self.tx_no_arq(( data,meta_dict) ,USER_IO_PROTOCOL_ID)
251         else:
252             raise NameError("EM_USE_ARQ and/or EM_DEST_ADDR not specified in PDU")
253
254         self.run_arq_fsm()
255
256     def ctrl_rx(self,msg):
257         self.run_arq_fsm()
258
```

Handling Input in work()

Ettus Research

```
50
51    int
52    pdu_to_tagged_stream_impl::work(int noutput_items,
53                    gr_vector_const_void_star &input_items,
54                    gr_vector_void_star &output_items)
55    {
56      char *out = (char *)output_items[0];
57      int nout = 0;
58
59      // if we have remaining output, send it
60      if (d_remain.size() > 0) {
61    nout = std::min((size_t)d_remain.size()/d_itemsize, (size_t)noutput_items);
62    memcpy(out, &d_remain[0], nout*d_itemsize);
63    d_remain.erase(d_remain.begin(), d_remain.begin()+nout);
64    noutput_items -= nout;
65    out += nout*d_itemsize;
66      }
67
68      // if we have space for at least one item output as much as we can
69      if (noutput_items > 0) {
70
71    // grab a message if one exists
72    pmt::pmt_t msg(delete_head_nowait(PDU_PORT_ID));
73    if (msg.get() == NULL)
74      return nout;
75
76    // make sure type is valid
77    if (!pmt::is_pair(msg)) // TODO: implement pdu::is_valid()
78      throw std::runtime_error("received a malformed pdu message");
79
```

File /media/john/81542486-825a-42eb-afdc-7f3f5cbf5c8a/home/john/src/gr-easyMAC/apps/top_block.py opened(10).

File /home/john/src/gnuradio-3.7.x/gr-blocks/lib/wavfile.cc opened(11).

File /home/john/src/gnuradio-3.7.x/gr-blocks/lib/pdu_to_tagged_stream_impl.cc opened(12).

sel: 24L INS TAB mode: Unix (LF) encoding: UTF-8 filetype: C++ scope: gr::blocks::pdu_to_tagged_stre

Publishing a Message

Ettus Research

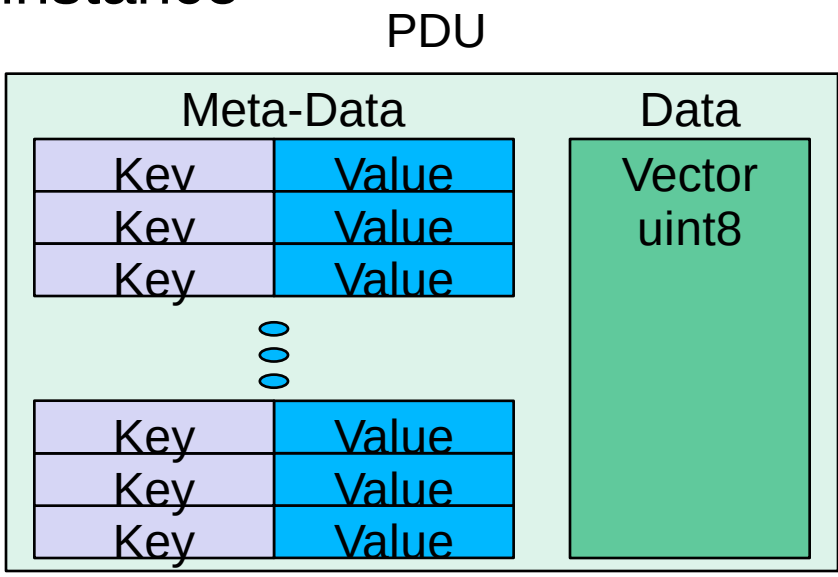
```
156
157     def output_user_data(self,pdu_tuple):
158
159         data = pdu_tuple[0][5:]
160
161         data = pmt.init_u8vector(len(data),data)
162
163         #pass through metadata if there is any
164         meta = pmt.to_pmt(pdu_tuple[1])
165
166         self.message_port_pub(pmt.intern('to_app'),pmt.cons(meta,data))
167
```

Other Notes About Messages

- work() function no longer required if you are only using messages
- Per unit of data, async. message passing typically involves more overhead than a stream
- Better for passing around frames or doing control plane operations
- Not recommended for passing samples

The Protocol Data Unit (PDU)

- Yet another PMT type
- Data and meta-data in a single PMT instance
- Examples
  - Adaptive rate systems
  - Adaptive encoding
  - Multi-channel
  - Per-Packet Control
- Contents
  - Payload – uint8 vector
  - PMT Dictionary – Key/Value Pairs



Constructing a PDU

Ettus Research

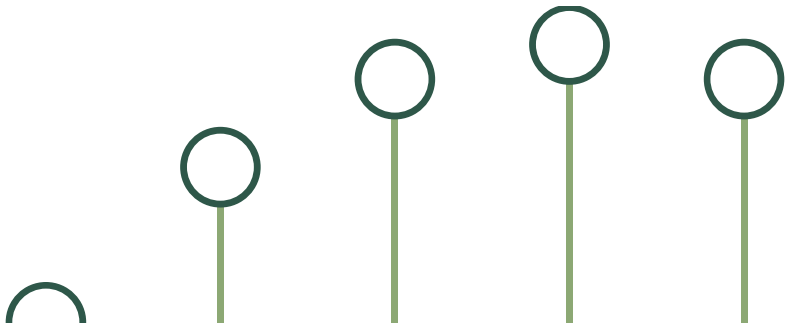
```
142
143     #transmit data through non-arq path
144     def tx_no_arq(self,pdu_tuple,protocol_id):
145         self.send_pkt_radio(pdu_tuple,self.pkt_cnt_no_arq,protocol_id,ARQ_NO_REQ)
146         self.pkt_cnt_no_arq = ( self.pkt_cnt_no_arq + 1 ) % 255
147         return
148
149     #transmit data - msg is numpy array
150     def send_pkt_radio(self,pdu_tuple,pkt_cnt,protocol_id,control):
151
152         #create header, merge with payload, convert to pmt for message_pub
153         data = (pkt_cnt,self.addr,pdu_tuple[1]['EM_DEST_ADDR'],protocol_id,control) + pdu_tuple[0]
154         data  = pmt.init_u8vector(len(data),data))
155
156         meta = pmt.to_pmt(pdu_tuple[1])
157
158         #construct pdu and publish to radio port
159         pdu = pmts.cons(meta,data)
160
161         #publish to msg port
162         self.message_port_pub(pmt.intern('to_radio'),pdu)
163         return
164
165
166     #transmit data through arq path
167     def tx_arq(self,pdu_tuple,protocol_id):
168         self.send_pkt_radio(pdu_tuple,self.pkt_cnt_arq,protocol_id,ARQ_REQ)
169         return
170
```

Reading a PDU

Ettus Research

```
243
244     def app_rx(self,msg):
245         try:
246             meta = pmt.car(msg)
247             data =  pmt.cdr(msg)
248         except:
249             raise NameError("easyMAC - input not a PDU")
250
251         if pmt.is_u8vector(data):
252             data = pmt.u8vector_elements(data)
253         else:
254             raise NameError("Data is not u8 vector")
255
256         meta_dict = pmt.to_python(meta)
257         if not (type(meta_dict) is dict):
258             meta_dict = {}
259
260         #double check to make sure correct meta data was in pdu
261         if 'EM_USE_ARQ' in meta_dict.keys() and 'EM_DEST_ADDR' in meta_dict.keys():
262             #assign tx path depending on whether PMT_BOOL EM_USE_ARQ is true or false
263             if(meta_dict['EM_USE_ARQ']):
264                 self.queue.put( (data,meta_dict) )
265             else:
266                 self.tx_no_arq(( data,meta_dict) ,USER_IO_PROTOCOL_ID)
267         else:
268             raise NameError("EM_USE_ARQ and/or EM_DEST_ADDR not specified in PDU")
269
270         self.run_arq_fsm()
271
```

pre-cog/gr-mac/gr-easyMAC



History

- "Pre-Cog"
  - GNU Radio 3.6 and gr-extras
  - Demo:
    - USRP Features
      - Timed-bursts
      - Timed tuning commands
    - GNU Radio Features
      - Message Passing
      - Python Blocks
      - Stream Tags
  - Random Access, FHSS, TDMA

- gr-mac
  - Upgrade to 3.7.0
  - Demo during the hackfest

- Other MAC-related work
  - Moritz Fischer – Sliding Window ARQ
  - KIT-CEL – Fast Frequency Hopping Prototype (GR 3.6 + gr-extras)

Additional Driver Features

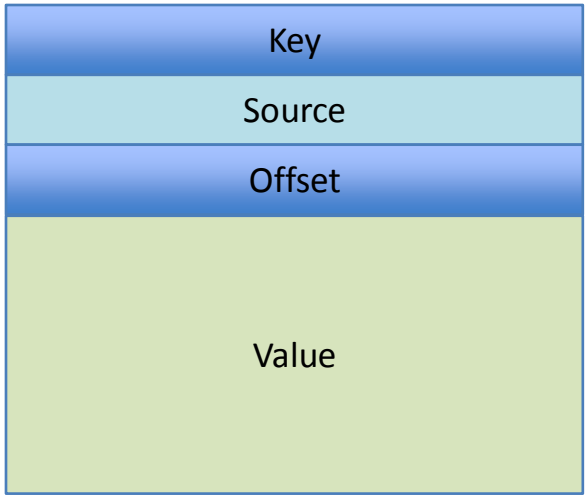
- Burst Transmission
  - Implemented with sample metadata
    - sob, time_spec, eob
  - Allows half-duplex operation - fast T/R Switching
  - Precise transmission alignment in network (TDMA)

- Timed Commands
  - Tuning, GPIO, etc.
  - Frequency hopping, PLL re-sync (SBX), T/R ctrl

- These are exposed in GNU Radio

PMT Example – USRP Burst

Ettus Research



Used by blocks to identify PMT

Specifies origin(block) of PMT

Specifies sample position of tag

Type: Can by many types
"payload data"

Tag, Offset = n
Key = 'tx_time'
Value = [sec frac_sec]

Tag Offset = n
Key = "tx_sob"
Value = pmt.PMT_T

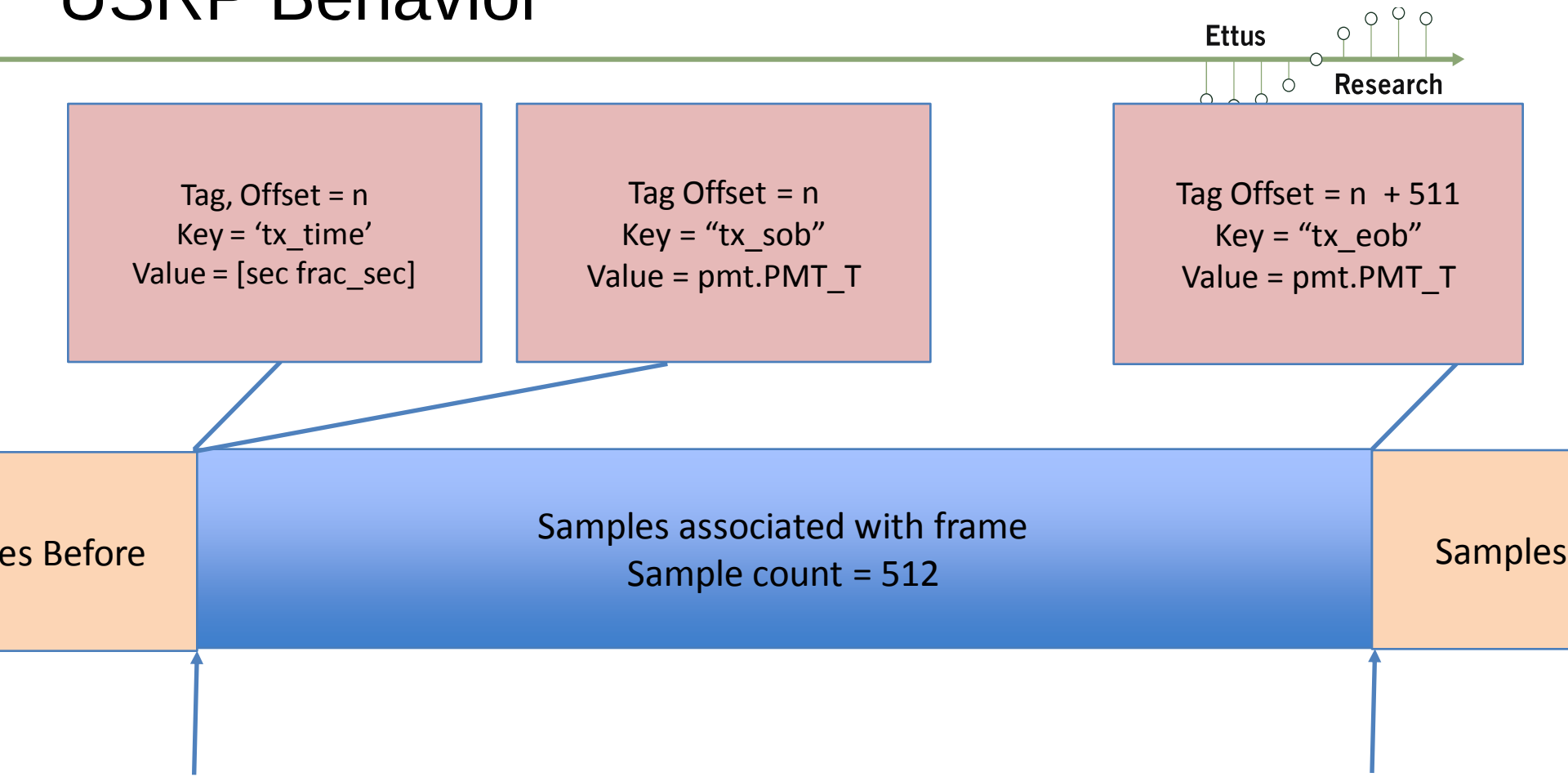
Tag Offset = n + 511
Key = "tx_eob"
Value = pmt.PMT_T

ples Before

Samples associated with frame
Sample count = 512

Sample

USRP Behavior



- Must be sent to USRP some time before actual tx_time
- At tx_time, USRP will set switches to tx and start streaming

Upon receiving eob, USRP will:

- stop tx streaming
- Set T/R switch to Rx port

Example – Getting Time from USRP

Ettus Research

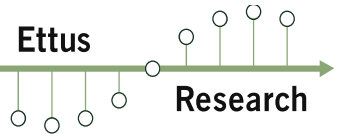
```
226
227         #read all tags associated with port 0 for items in this work function
228         tags = self.get_tags_in_range(0, nread, nread+ninput_items)
229
230         #lets find all of our tags, making the appropriate adjustments to our timing
231         for tag in tags:
232             key_string = pmt.pmt_symbol_to_string(tag.key)
233             if key_string == "rx_time":
234                 self.samples_since_last_rx_time = 0
235                 self.current_integer,self.current_fractional = pmt.to_python(tag.value)
236                 self.time_update = self.current_integer + self.current_fractional
237                 self.found_time = True
238             elif key_string == "rx_rate":
239                 self.rate = pmt.to_python(tag.value)
240                 self.sample_period = 1/self.rate
241                 self.found_rate = True
242
243         #determine first transmit slot when we learn the time
244         if not self.know_time:
245             if self.found_time and self.found_rate:
246         #get current time
247         else:
248             self.time_update += (self.sample_period * ninput_items)
249
250
```

Example – Tx Burst USRP

Ettus Research

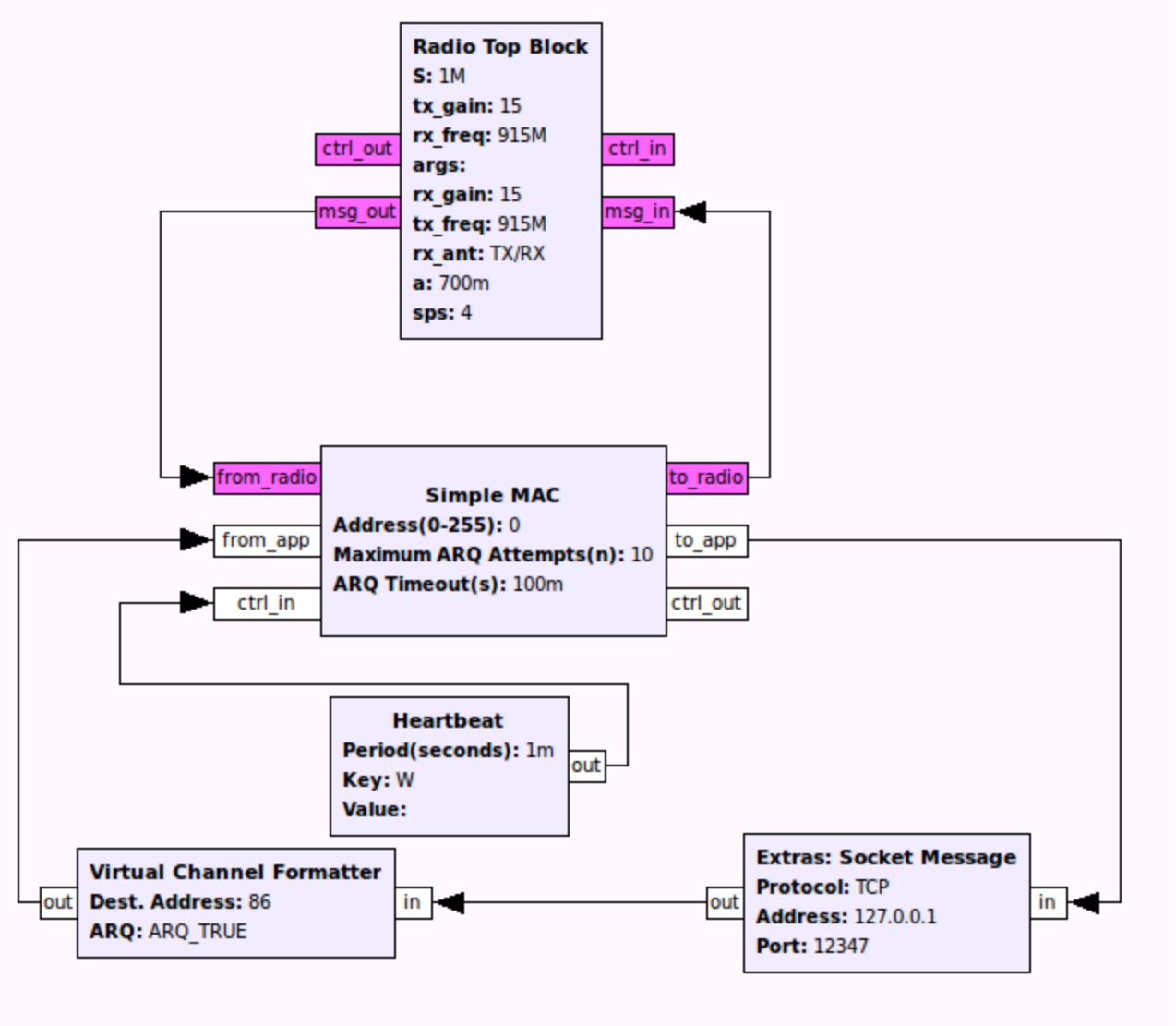
```
113
114         if len(self._pkt) == 0 :
115             item_index = num_items #which output item gets the tag?
116             offset = self.nitems_written(0) + item_index
117             source = pmt.pmt_string_to_symbol("framer")
118             if self.has_tx_time:
119                 key = pmt.pmt_string_to_symbol("tx_sob")
120                 self.add_item_tag(0, self.nitems_written(0), key, pmt.PMT_T, source)
121                 key = pmt.pmt_string_to_symbol("tx_time")
122                 self.add_item_tag(0, self.nitems_written(0), key, pmt.from_python(self.tx_time), source)
123
124             if self.more_frame_cnt == 0:
125                 key = pmt.pmt_string_to_symbol("tx_eob")
126                 self.add_item_tag(0, offset - 1, key, pmt.PMT_T, source)
127             else:
128                 self.more_frame_cnt -= 1
129
```

gr-precog



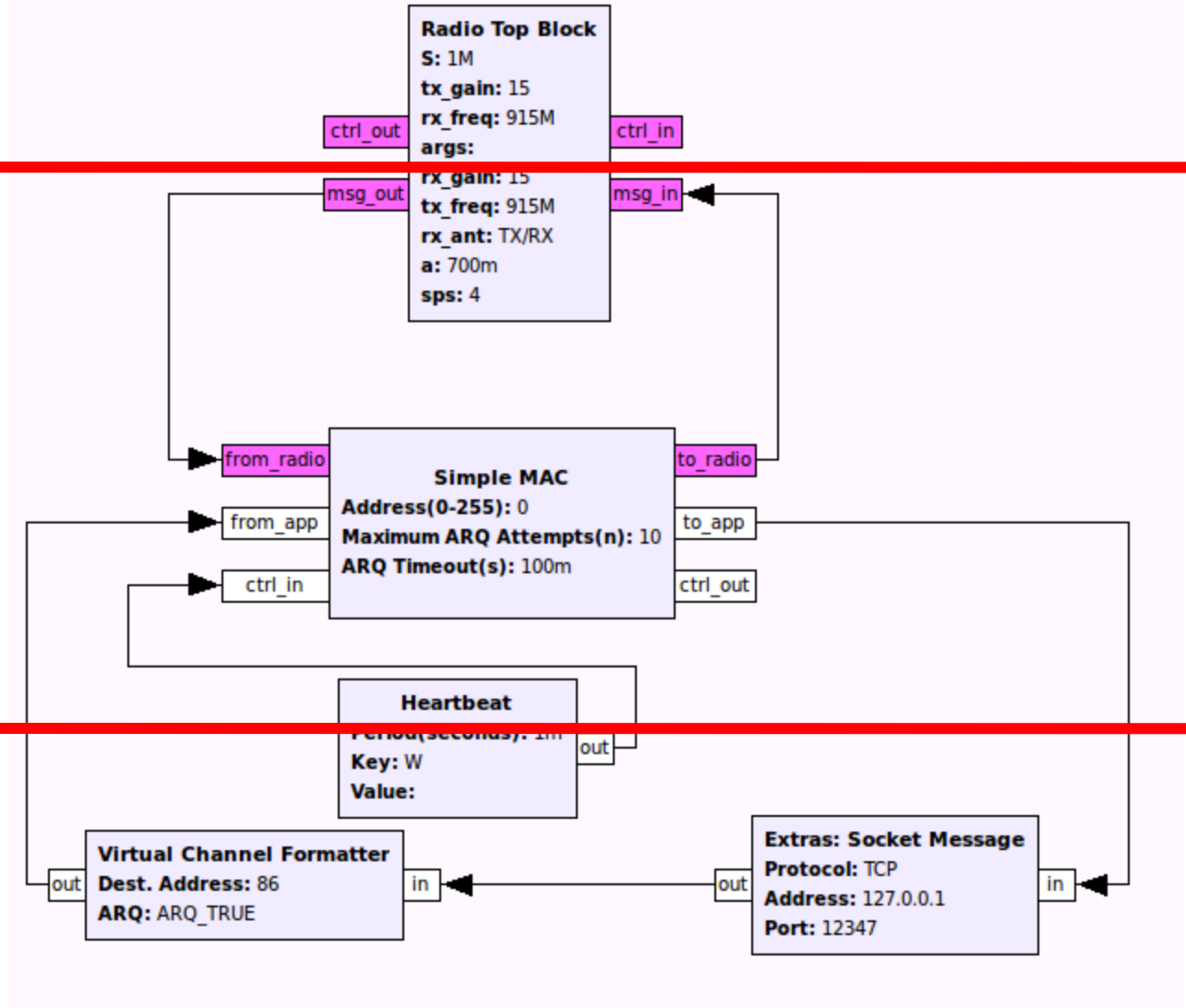
- A first set of prototype blocks built on gr-extras, and “blob” concept
- Demonstrate basic MAC functionality WITHIN GNU Radio
  - This is basic, but developers could do much more sophisticated things…
- Added capabilities
  - Frame agnostic bursting (ie. tight control of tx timing)
  - Simple upper-level MAC with ARQ
  - TDMA Implementation
  - FHSS Transceiver
  - Easy Expansion w/ Layered Protocols
- Upcoming release will use 3.7.0-compatible API and new message passing API

simple_trx.grc

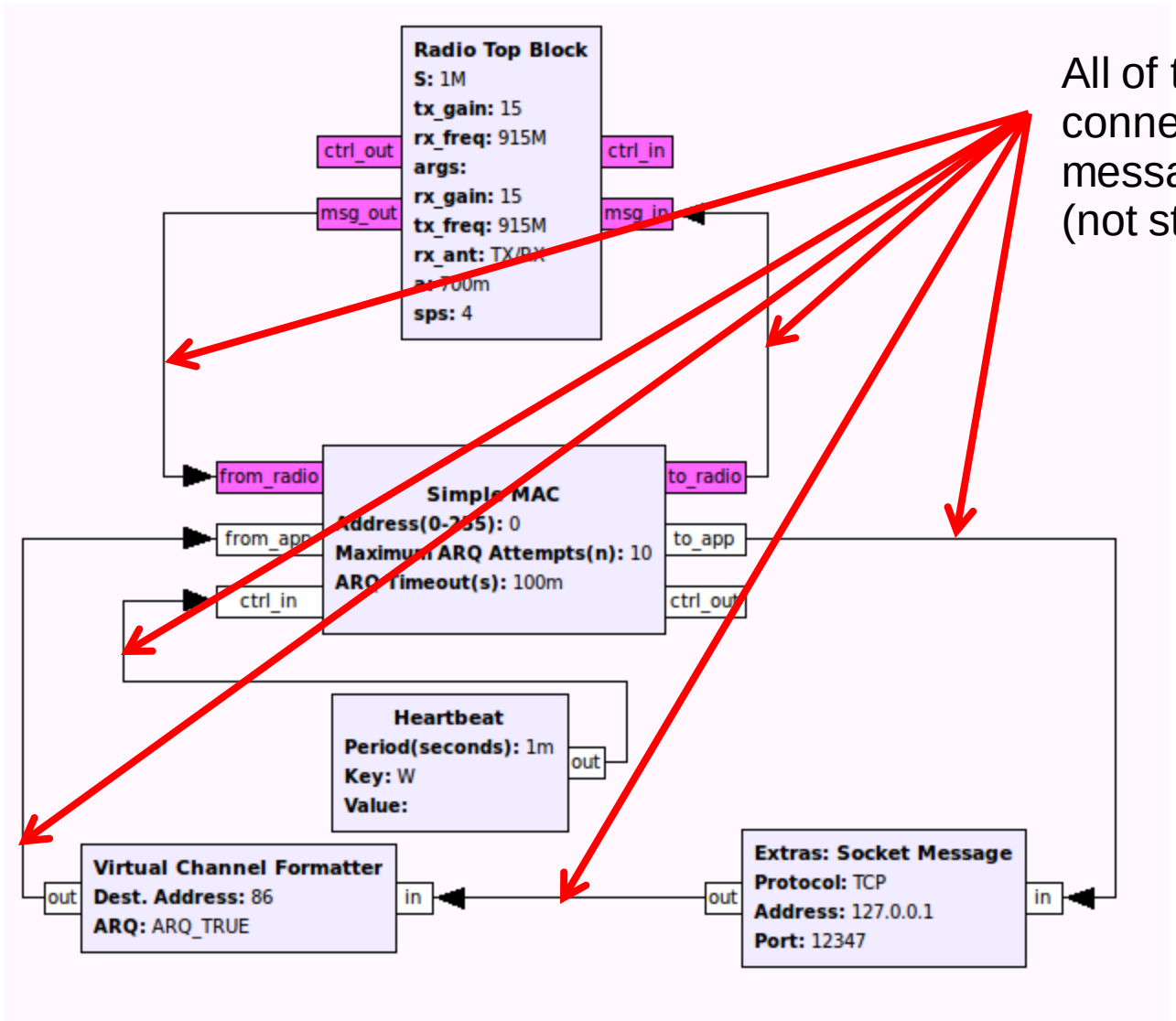


Protocol Layering

Ettus Research

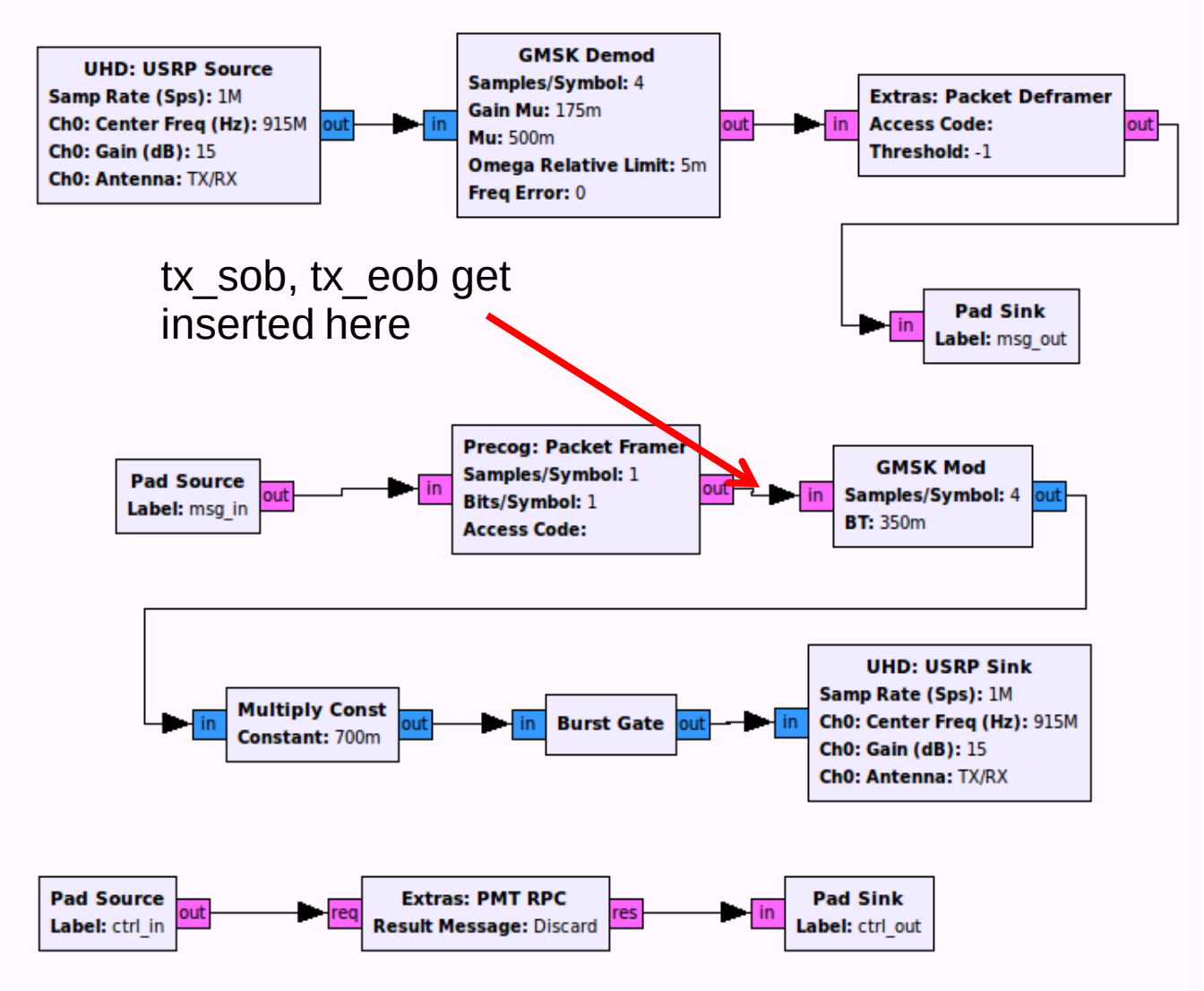


Message Based

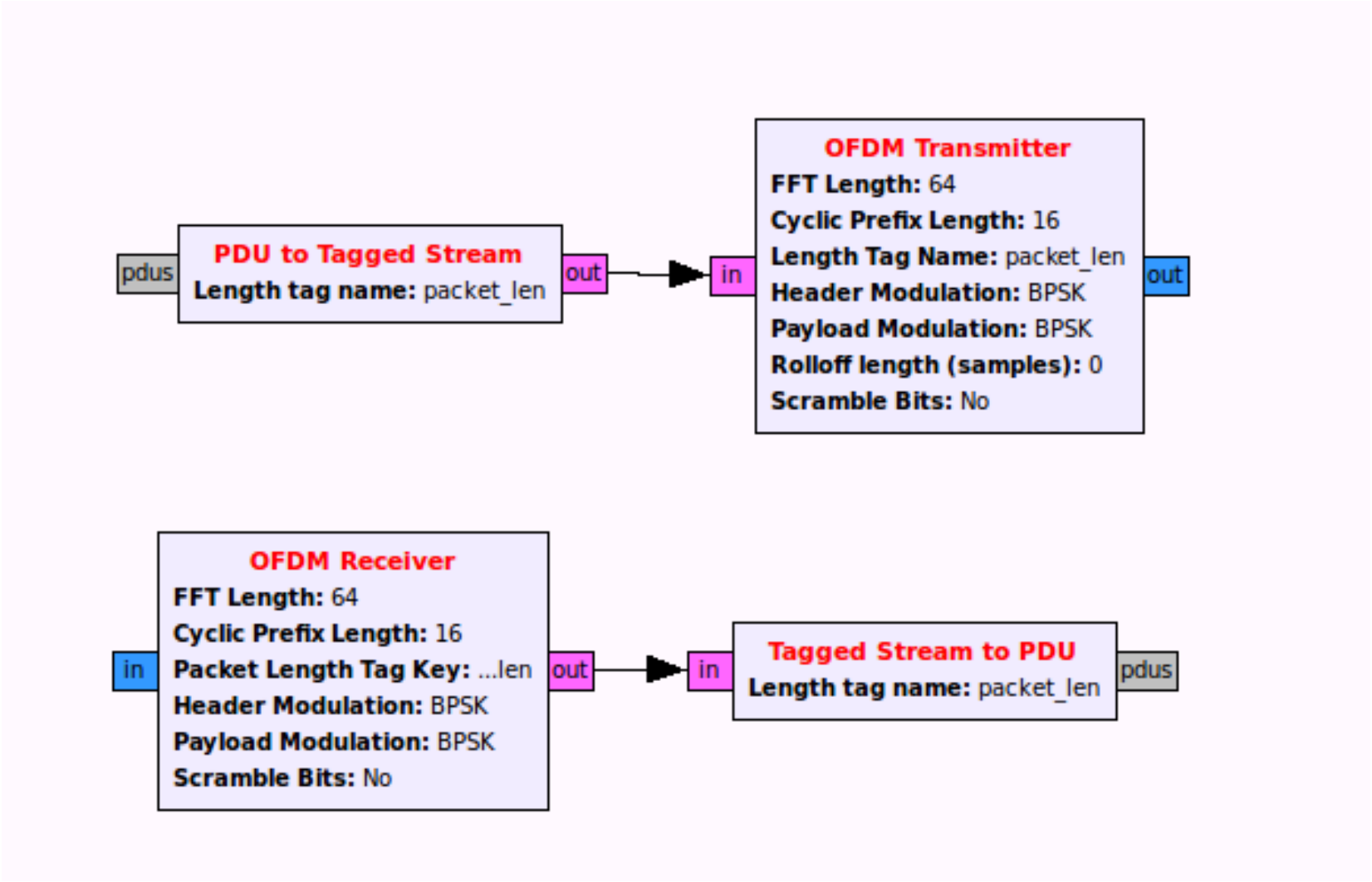


All of these connections are with messages (not streams)

PHY – Inside “Radio Top Block”

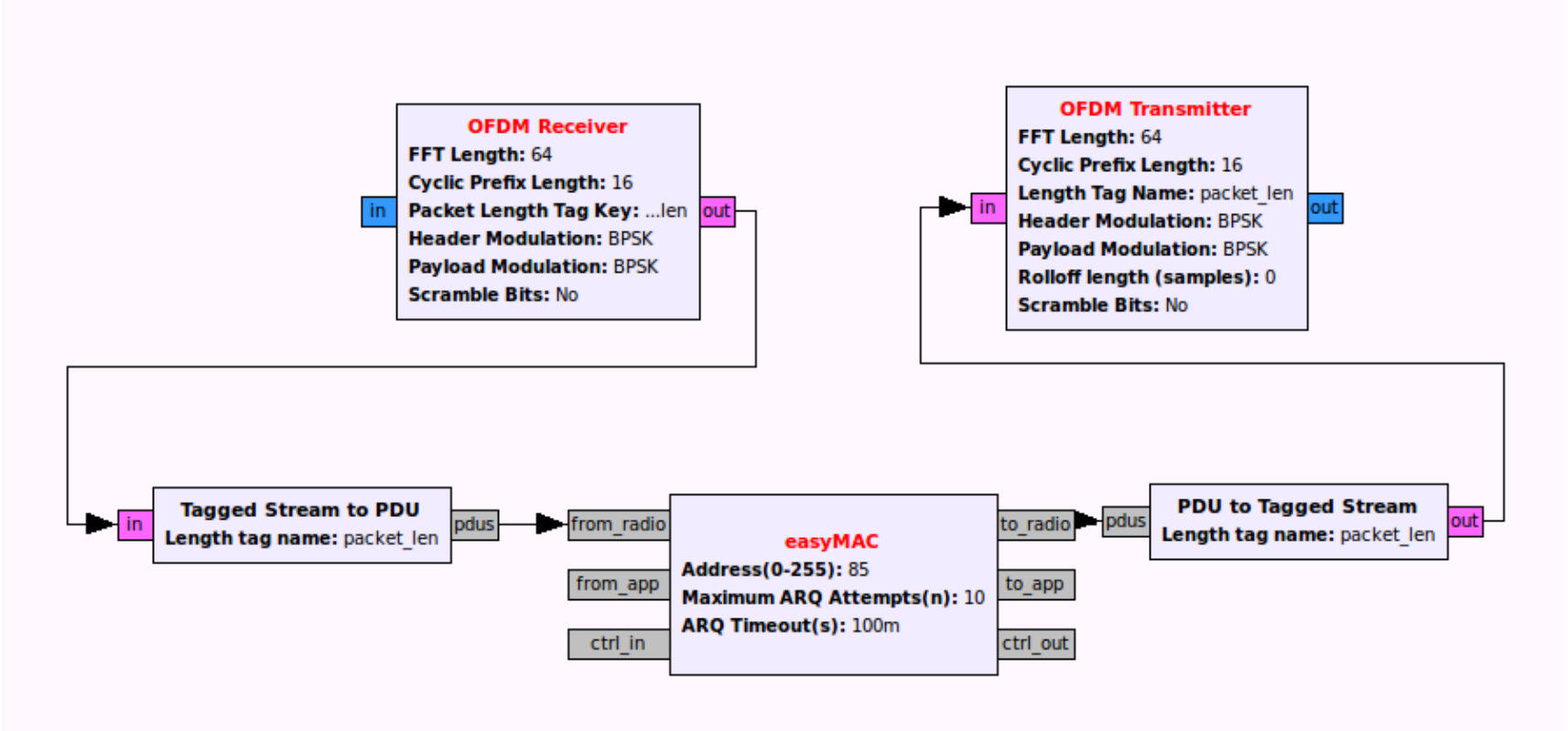


PDU<->Stream


pdus
PDU to Tagged Stream
Length tag name: packet_len
out
in
OFDM Transmitter
FFT Length: 64
Cyclic Prefix Length: 16
Length Tag Name: packet_len
Header Modulation: BPSK
Payload Modulation: BPSK
Rolloff length (samples): 0
Scramble Bits: No
out
in
OFDM Receiver
FFT Length: 64
Cyclic Prefix Length: 16
Packet Length Tag Key: ...len
Header Modulation: BPSK
Payload Modulation: BPSK
Scramble Bits: No
out
in
Tagged Stream to PDU
Length tag name: packet_len
pdus

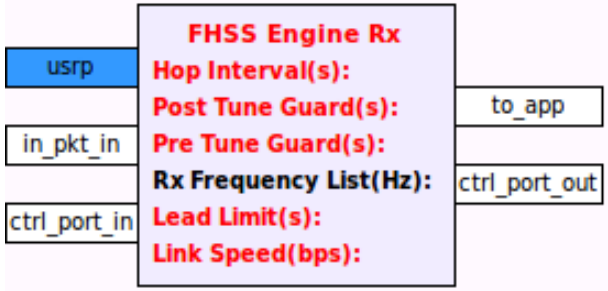
Connected to and OFDM PHY

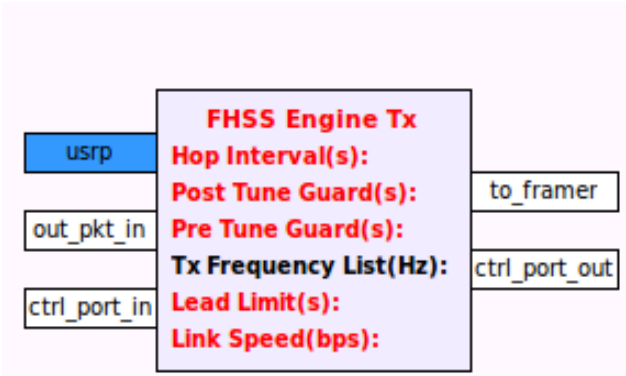
Ettus Research

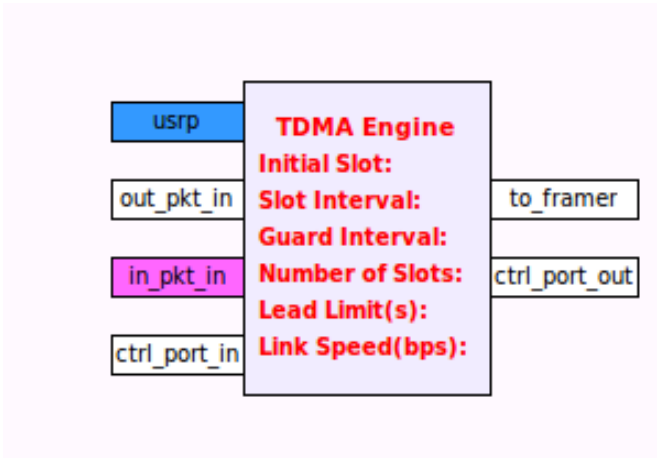


Other Implementations

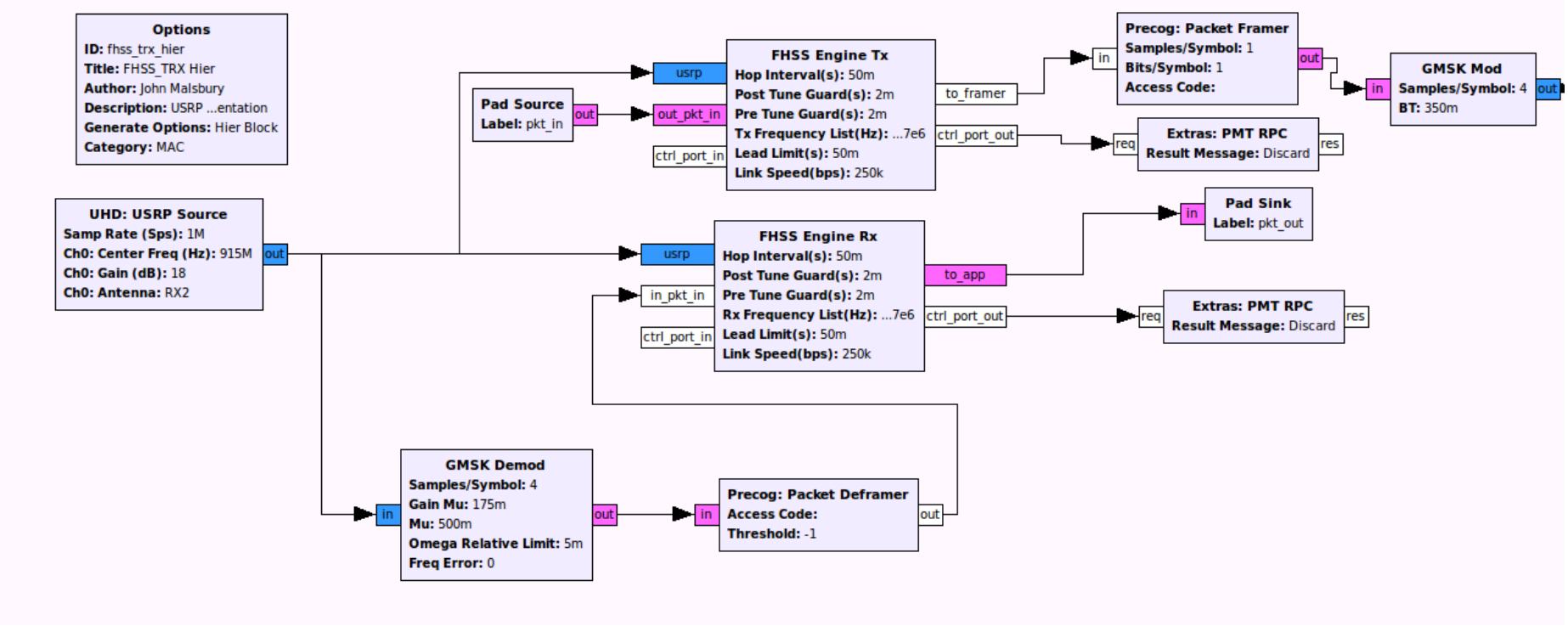
Ettus Research







FHSS Top Block



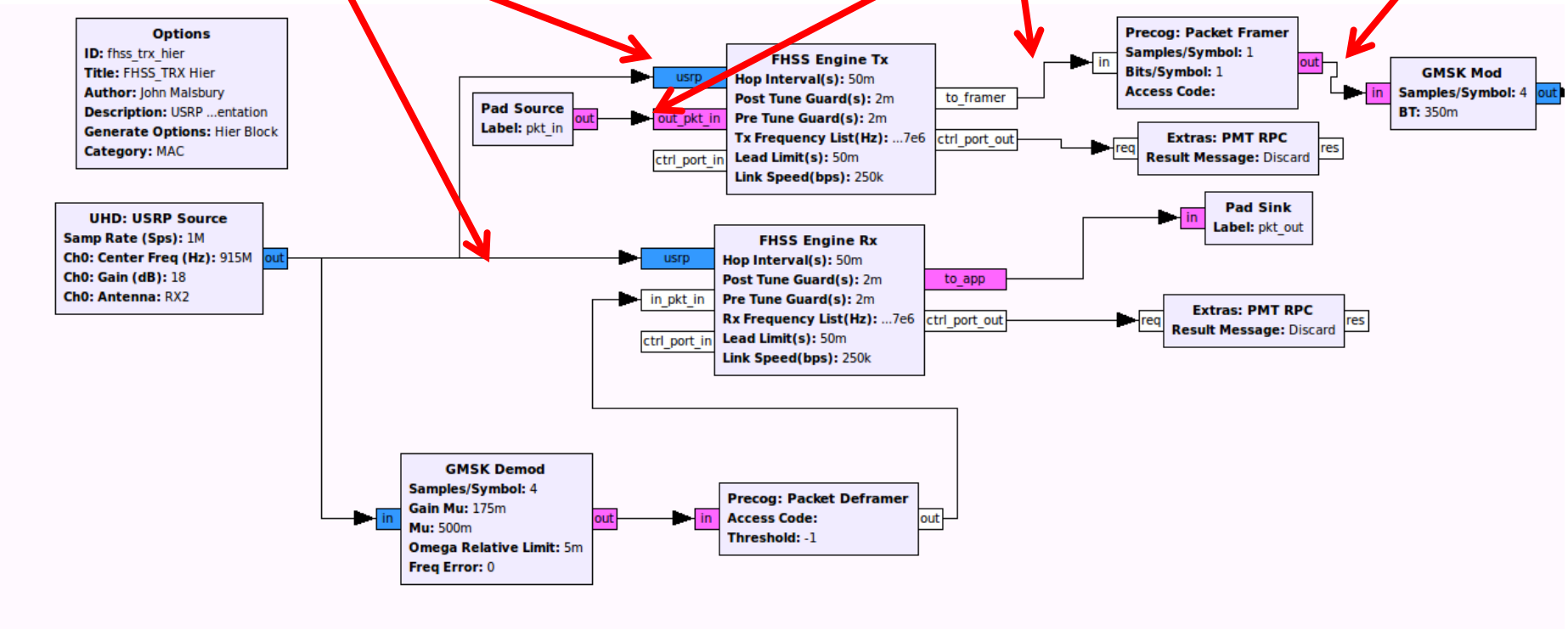
FHSS Top Block

Ettus Research

rx_time, rx_rate,
rx_freq tags

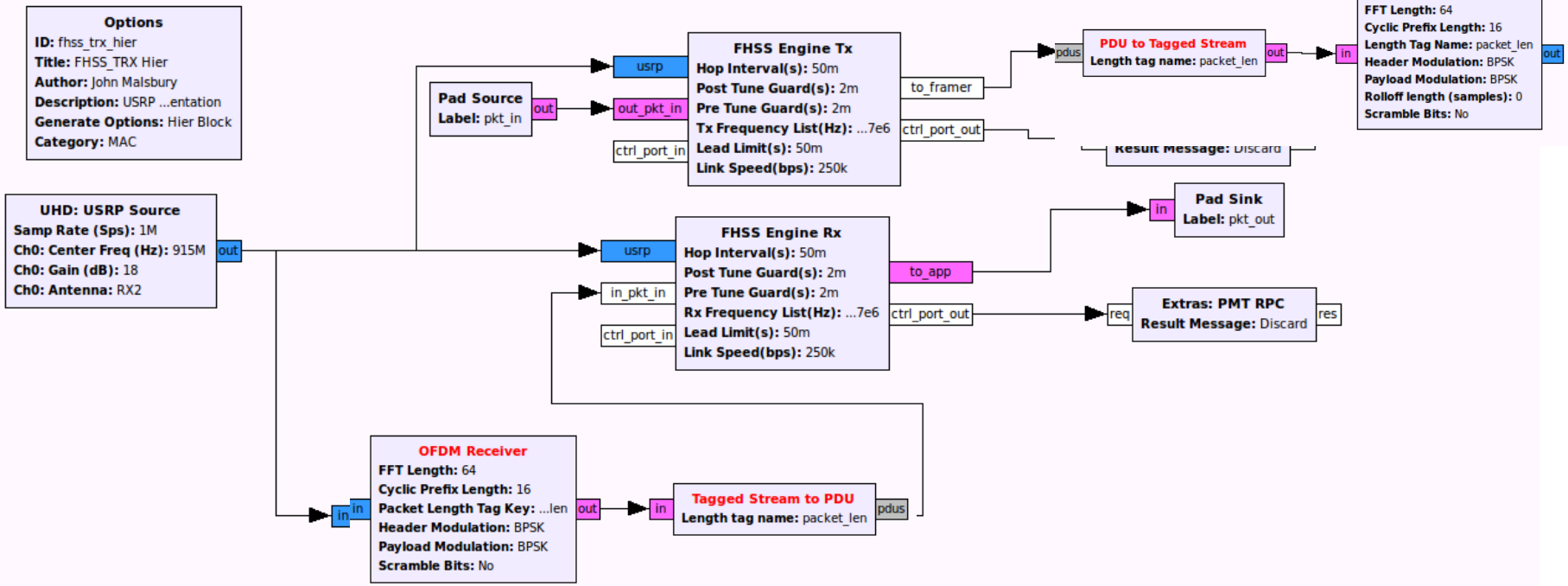
Packets + meta_data
(ie. a PDU)

tx_sob, tx_time,
tx_eob



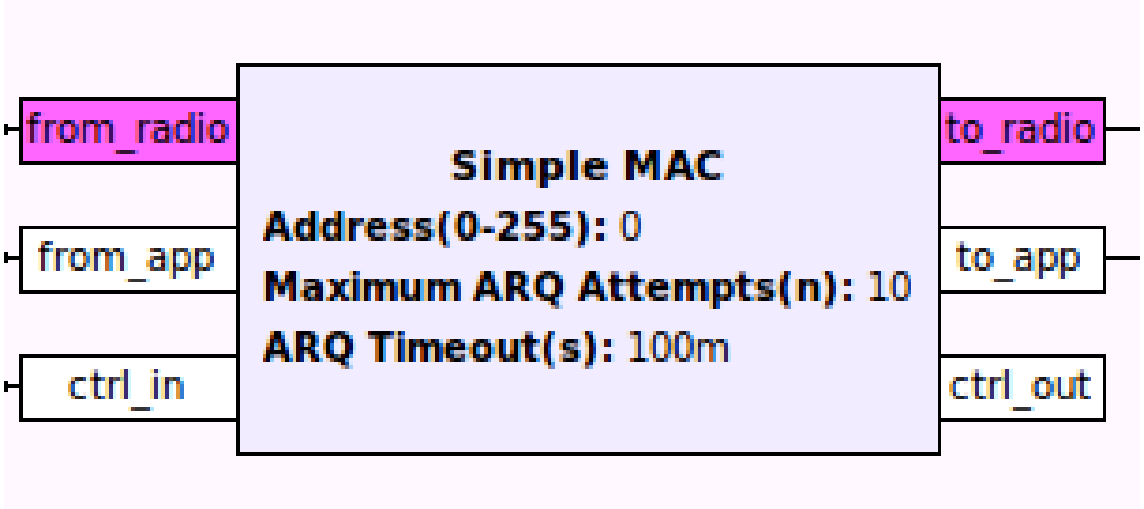
FHSS w/ OFDM?

Ettus Research



gr-precog

Ettus Research



Code for MAC

Ettus Research

Applications Places System 5:26 PM john

simple_mac.py - /home/john/src/clean_install/pre-cog/python - Geany

File Edit Search View Document Project Build Tools Help

Documents

~/src/cle.../python
simple_mac.py
~/src/u...xamples
tx_bursts.cpp
txfreqhop.cpp
~/src/uh...d/types
stream...md.hpp

stream_cmd.hpp | txfreqhop.cpp | tx_bursts.cpp | simple_mac.py

```
220                     self.arq_expected_sequence_number =  ( incoming_pkt[PKT_INDEX_CNT] + 1 ) % 255
221                 else:
222                     if not (self.no_arq_expected_sequence_number == incoming_pkt[PKT_INDEX_CNT]):
223                         self.no_arq_sequence_error_cnt += 1
224                         #print self.no_arq_sequence_error_cnt
225                         #print self.no_arq_sequence_error_cnt,incoming_pkt[PKT_INDEX_CNT],self.no_arq_expected_sequence_number
226                     self.no_arq_expected_sequence_number =  ( incoming_pkt[PKT_INDEX_CNT] + 1 ) % 255
227
228                 incoming_protocol_id = incoming_pkt[PKT_INDEX_PROT_ID]
229
230                 #check to see if this is an ACK packet
231                 if(incoming_protocol_id == ARQ_PROTOCOL_ID):
232                     if incoming_pkt[5] == self.expected_arq_id:
233                         self.arq_channel_state = ARQ_CHANNEL_IDLE
234                         self.pkt_cnt_arq = ( self.pkt_cnt_arq + 1 ) % 255
235                     else:
236                         print 'received out of sequence ack',incoming_pkt[5],self.expected_arq_id
237
238                 #do something with incoming user data
239                 elif(incoming_protocol_id == USER_IO_PROTOCOL_ID):
240                     if not self.throw_away:
241                         self.output_user_data(incoming_pkt)
242                     self.throw_away = False
243
244                 else:
```

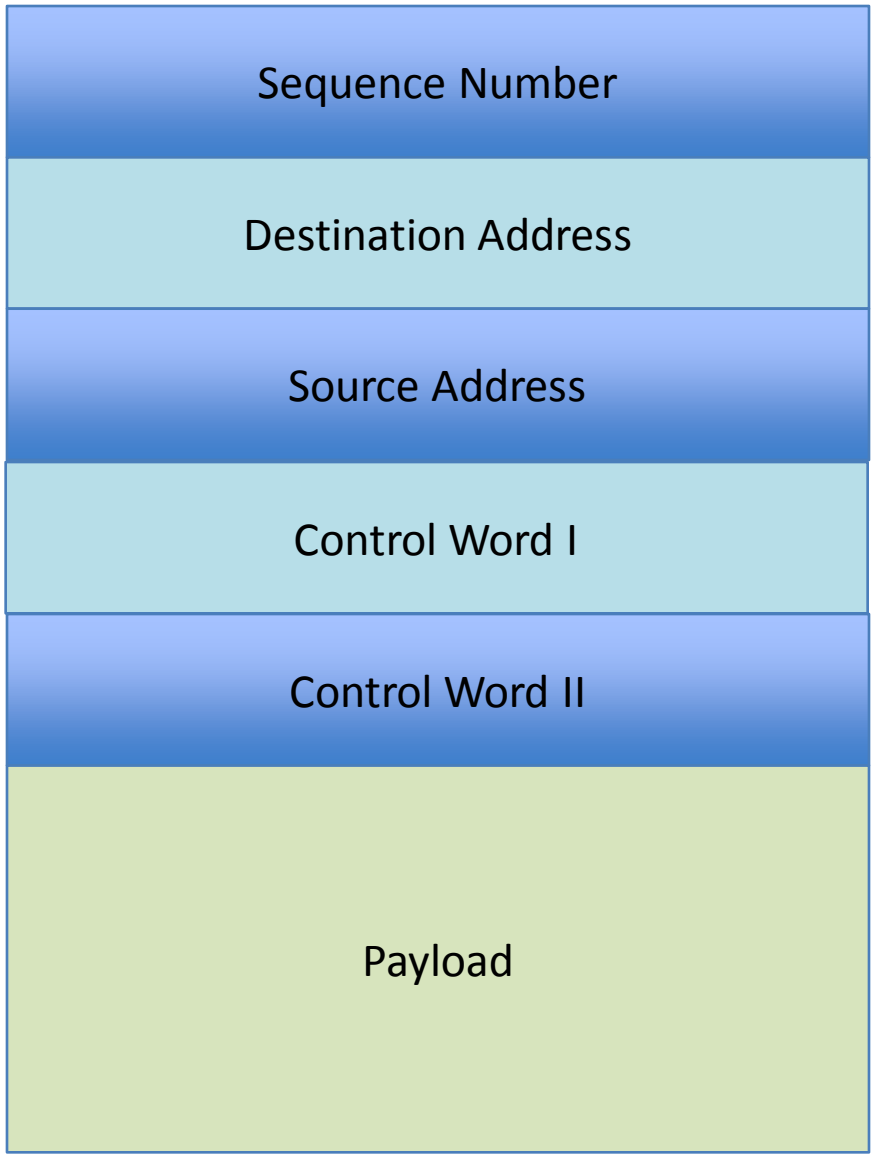
Status
Compiler

16:27:31: This is Geany 0.20.
16:27:31: File /home/john/src/uhd/host/include/uhd/types/stream_cmd.hpp opened(1).
16:38:16: File /home/john/src/uhd/host/examples/txfreqhop.cpp opened(2).
16:41:15: File /home/john/src/uhd/host/examples/tx_bursts.cpp opened(3).
17:25:41: File /home/john/src/clean_install/pre-cog/python/simple_mac.py opened(4).

line: 220 / 283 col: 2 sel: 0 INS SP mode: Unix (LF) encoding: UTF-8 filetype: Python scope: unknown

john@erllc1: ~/src/cle... | simple_mac.py ~/hom... | dyspan_ppt.odp - Libr...

Simple MAC – Frame Structure



Sequence Number for ARQ and dropped-frame detection

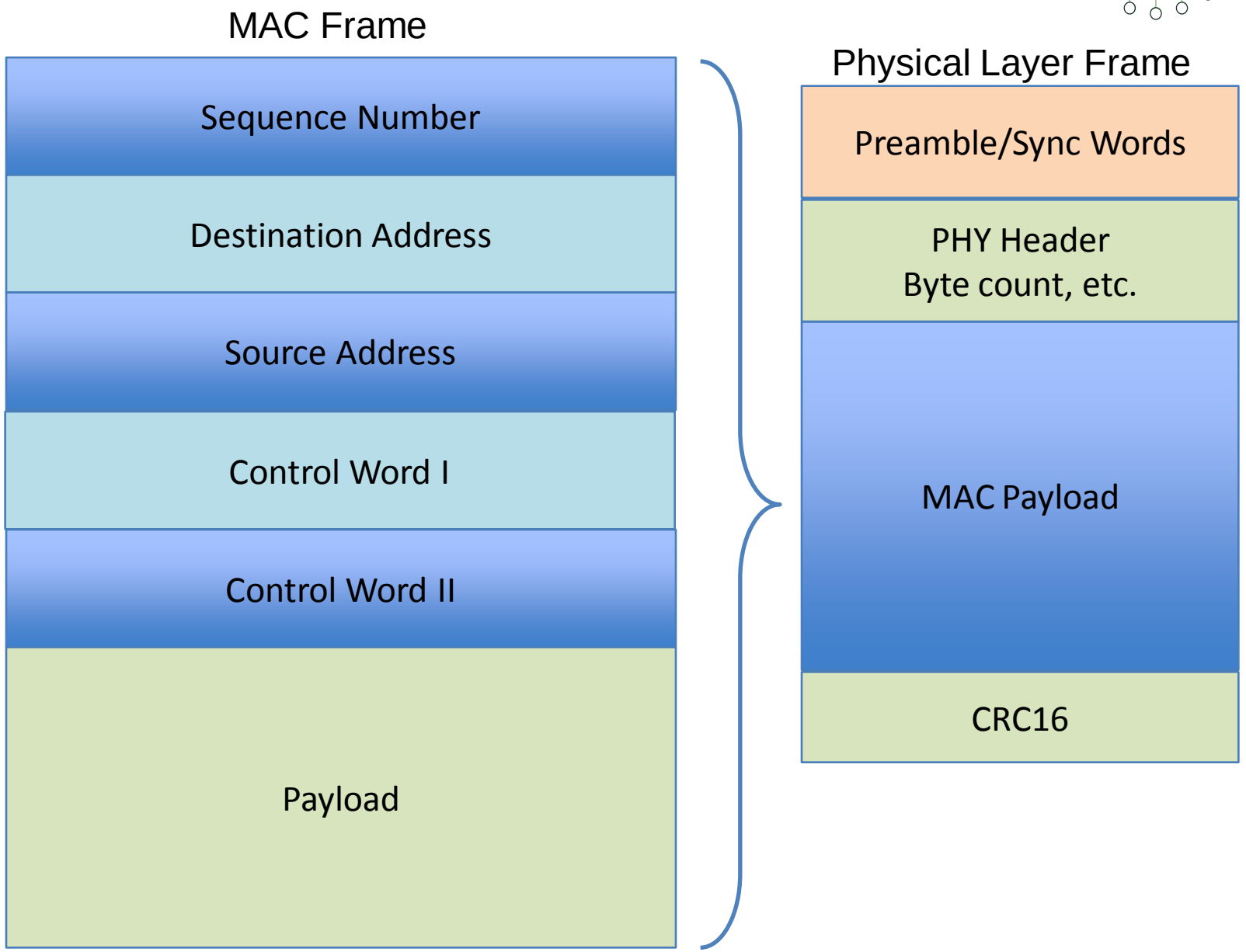
Address of radio we're sending this msg to.

Radio's address ("my address")

Control words: settings include ARQ, protocol id, FEC settings, etc.

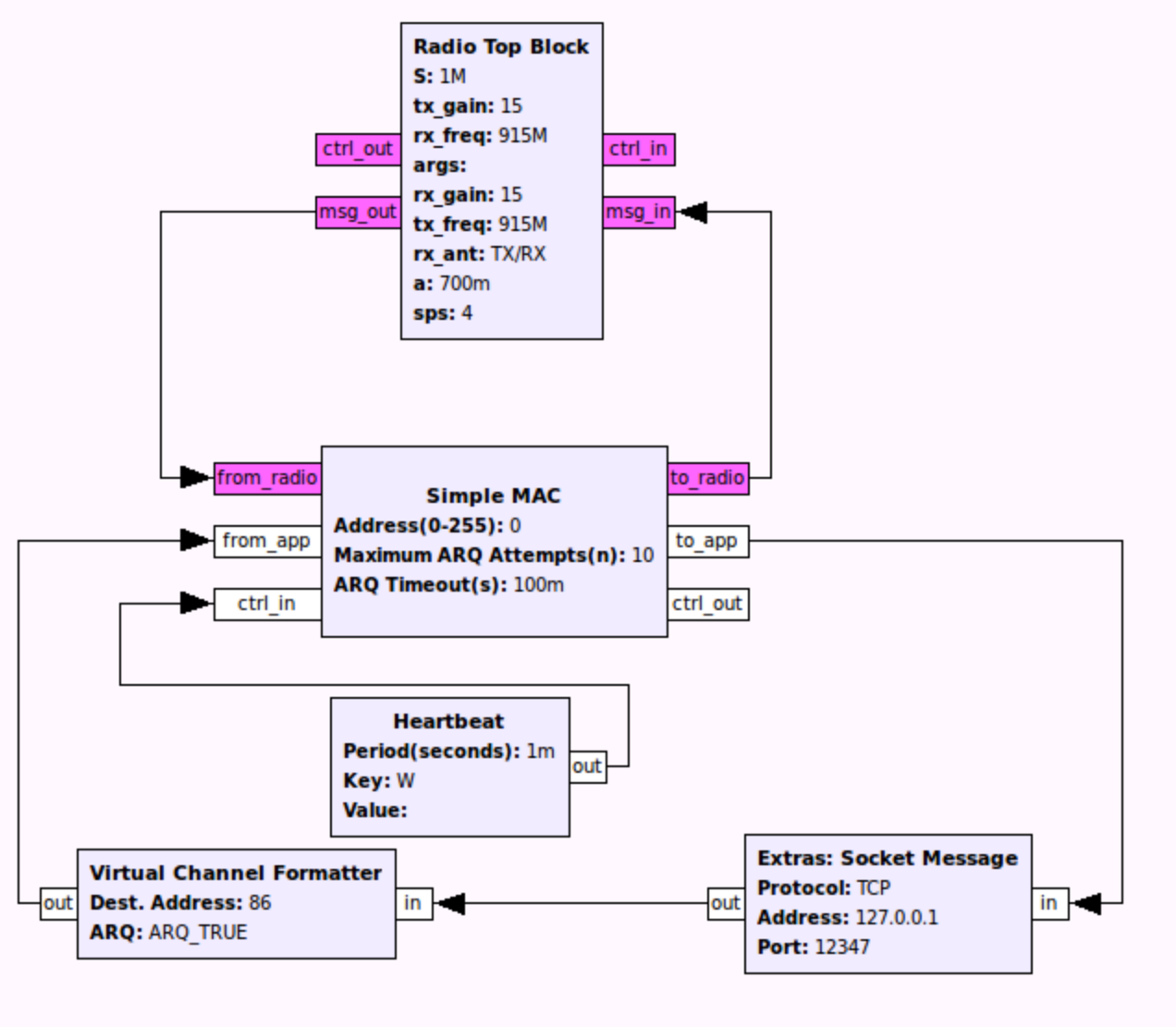
This is our payload, it can be represented as a "blob" and is in our implementation

Example Program – Frame Structure



gr-precog

Ettus Research



gr-precog

Ettus Research

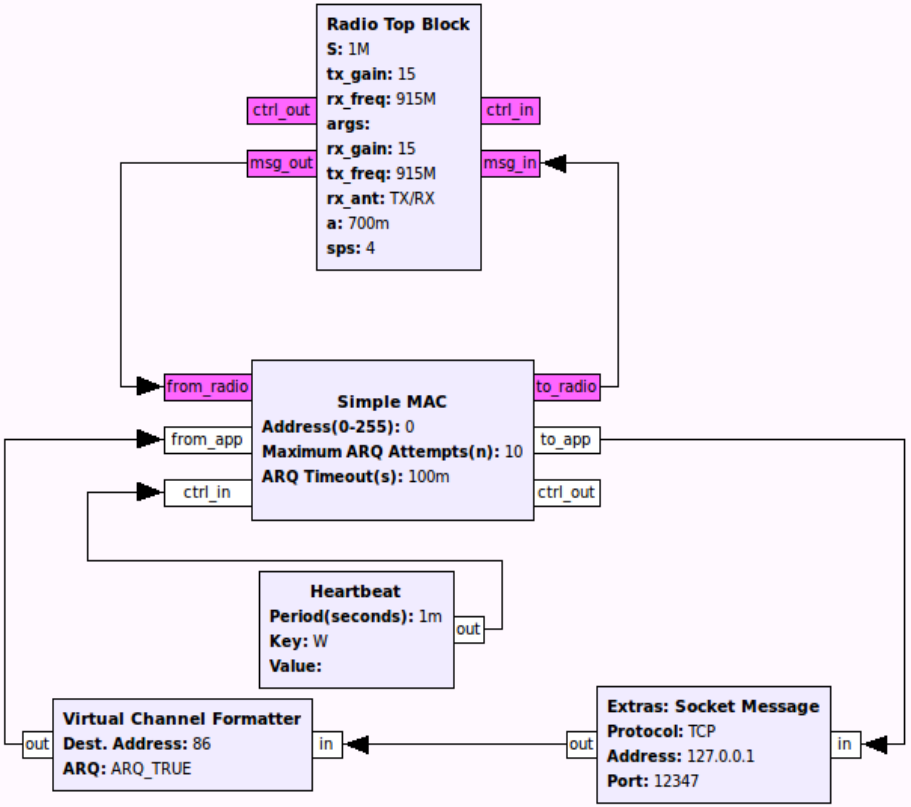
gr-prec

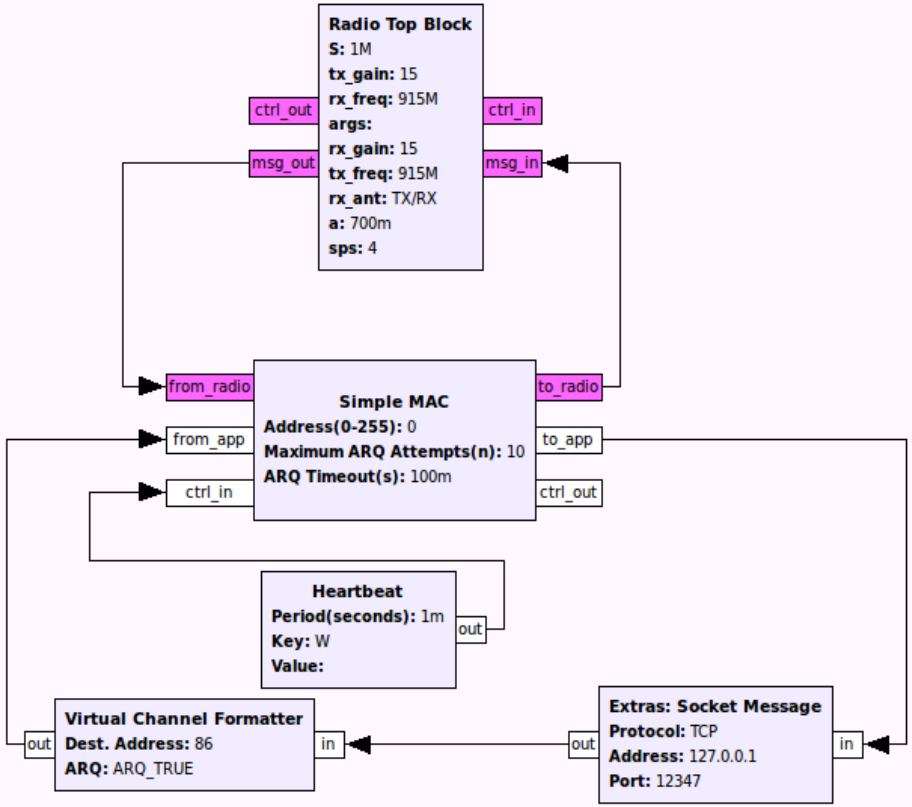
search

```
33      """
34      def __init__(self,channel,arq):
35          gr.basic_block.__init__(self,
36              name="virtual_channel_encoder",
37              in_sig=None,
38              out_sig=None)
39
40          self.channel = channel
41          self.arq = arq
42
43          self.message_port_register_out(pmt.intern('out'))
44          self.message_port_register_in(pmt.intern('in'))
45          self.set_msg_handler(pmt.intern('in'),self.format)
46
47      def format(self,msg):
48          data = pmt.cdr(msg)
49          meta = pmt.car(msg)
50          if not pmt.is_u8vector(data):
51              raise NameError("Data is no u8 vector")
52
53          meta_dict = pmt.to_python(meta)
54          if not (type(meta_dict) is dict):
55              meta_dict = {}
56
57          #lets append some metadata to the pdu
58          if self.arq == ARQ_TRUE:
59              meta_dict['EM_USE_ARQ'] = True
60          else:
61              meta_dict['EM_USE_ARQ'] = False
62
63          meta_dict['EM_DEST_ADDR'] = self.channel
64
65          #convert dictionary back to a pmt
66          meta = pmt.to_pmt(meta_dict)
67
68          self.message_port_pub(pmt.intern('out'),pmt.cons(meta,data))
69
```

gr-precog

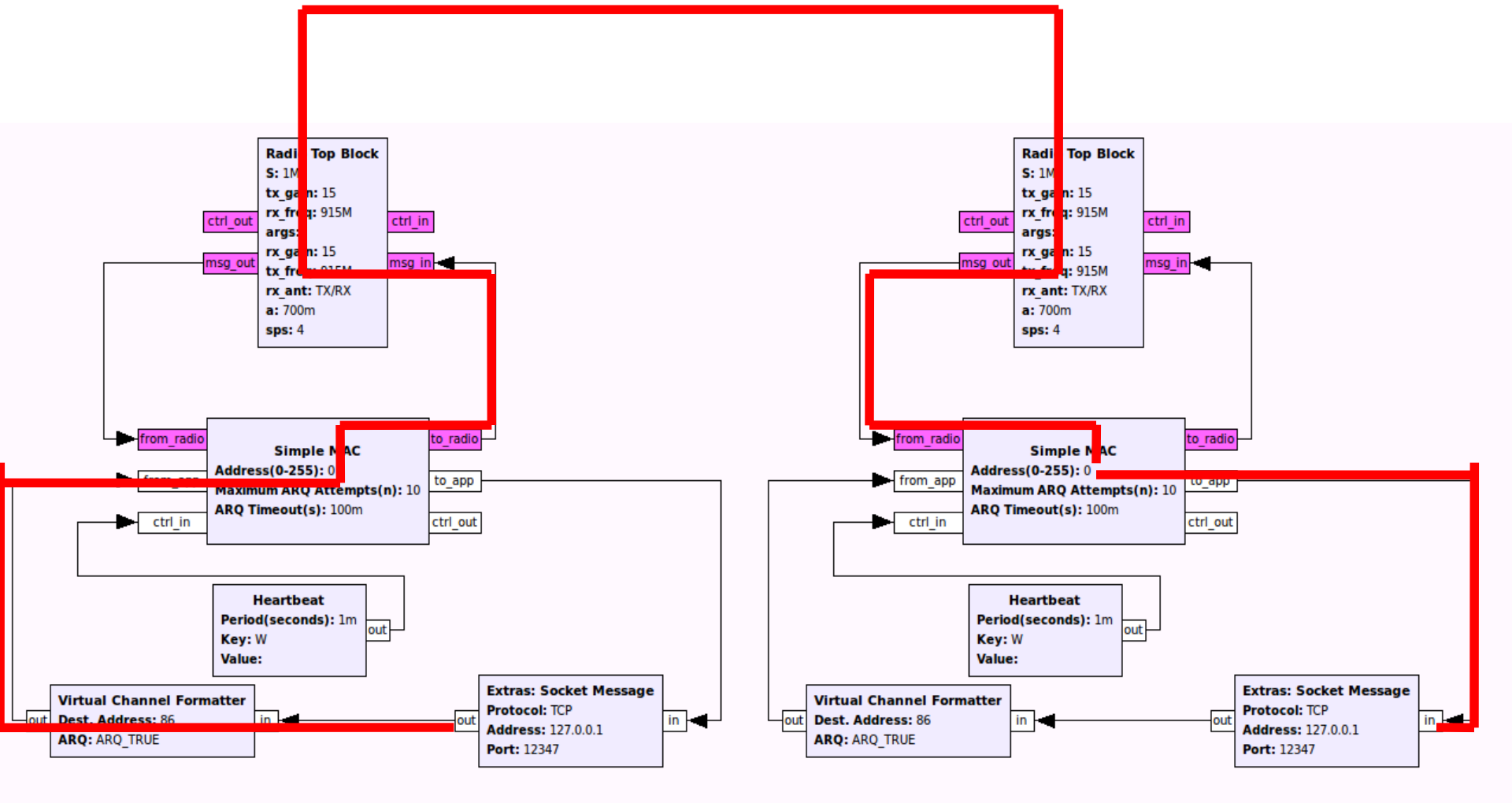
Ettus Research



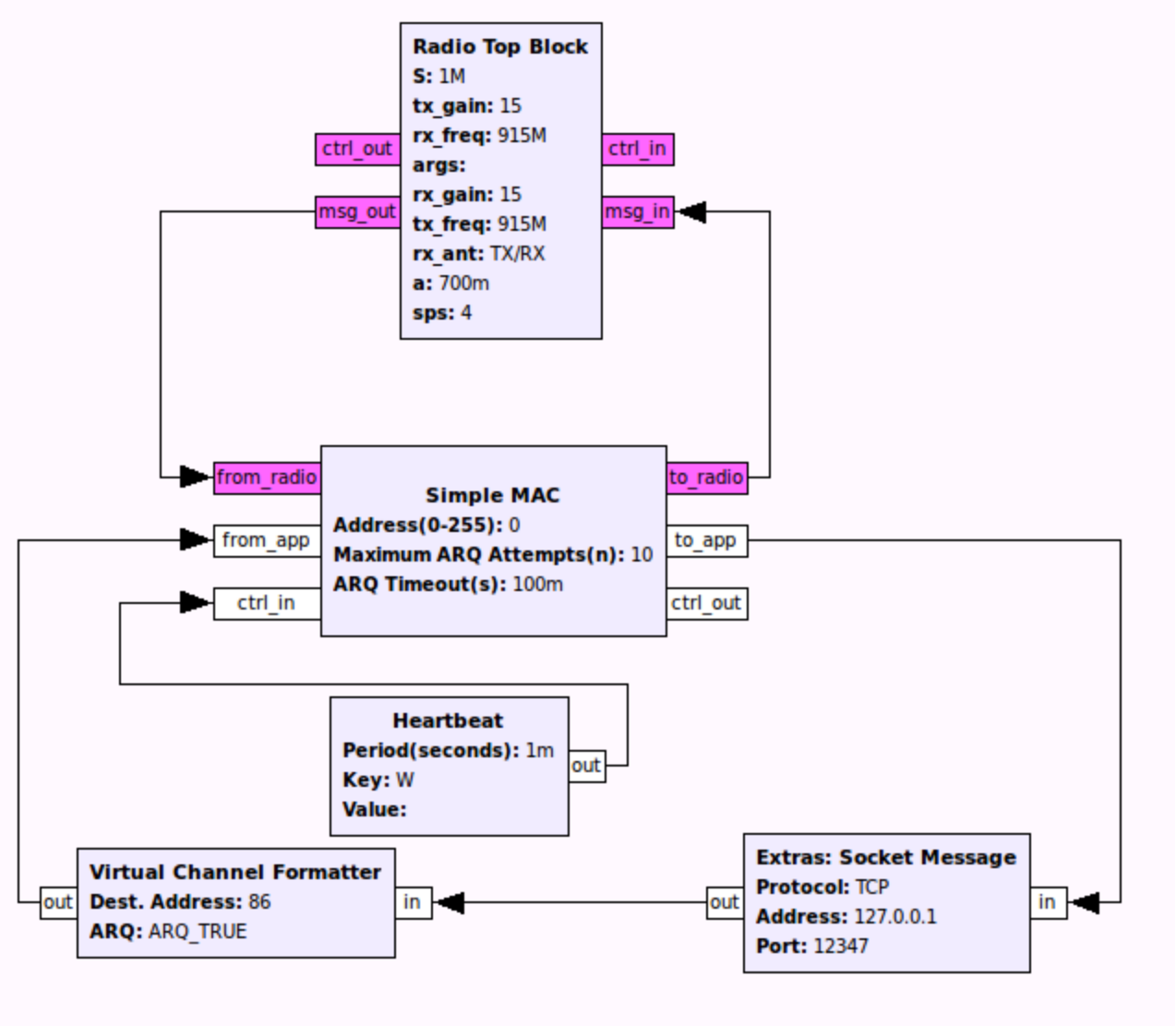


gr-precog

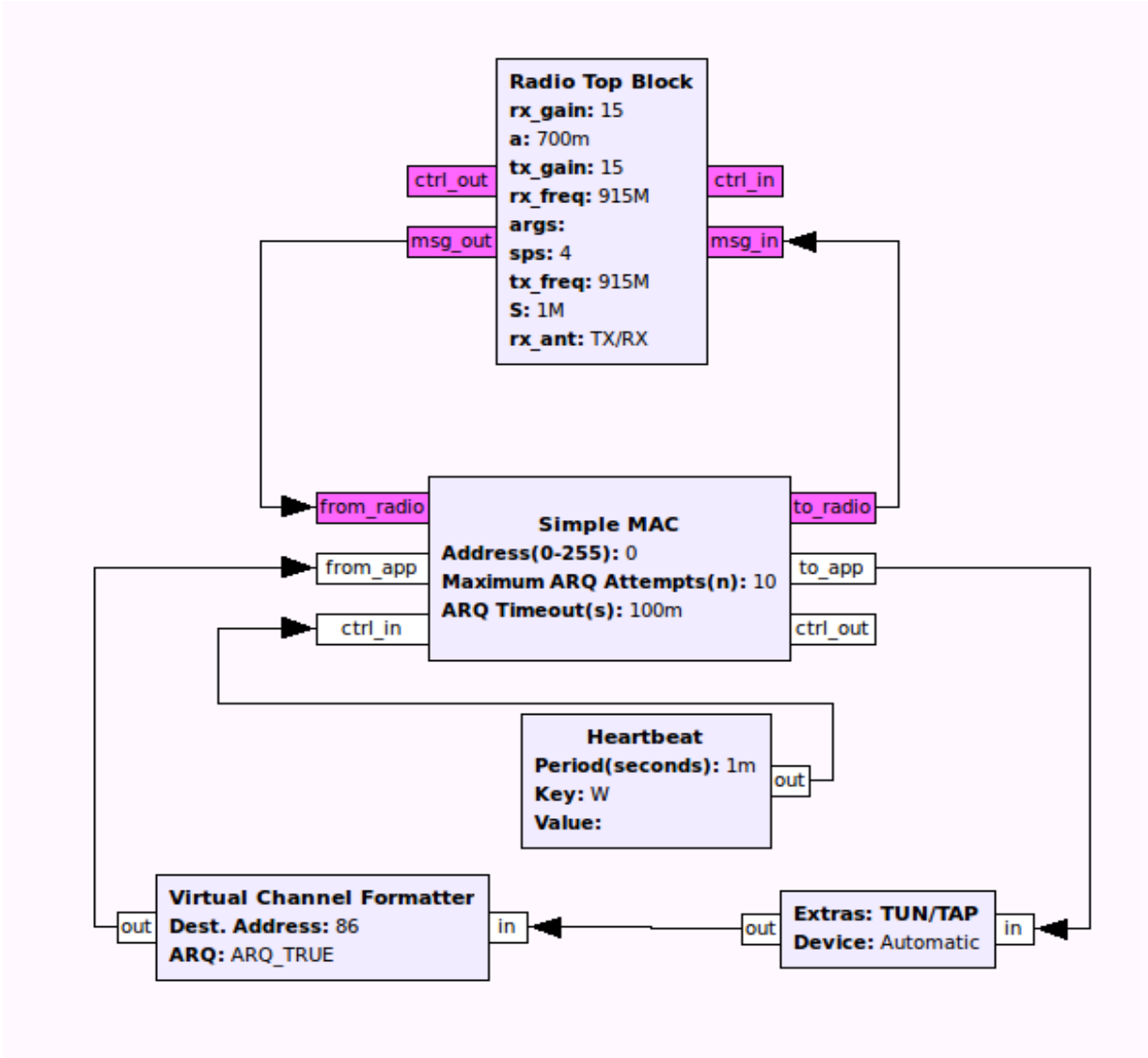
Ettus Research



simple_trx.grc

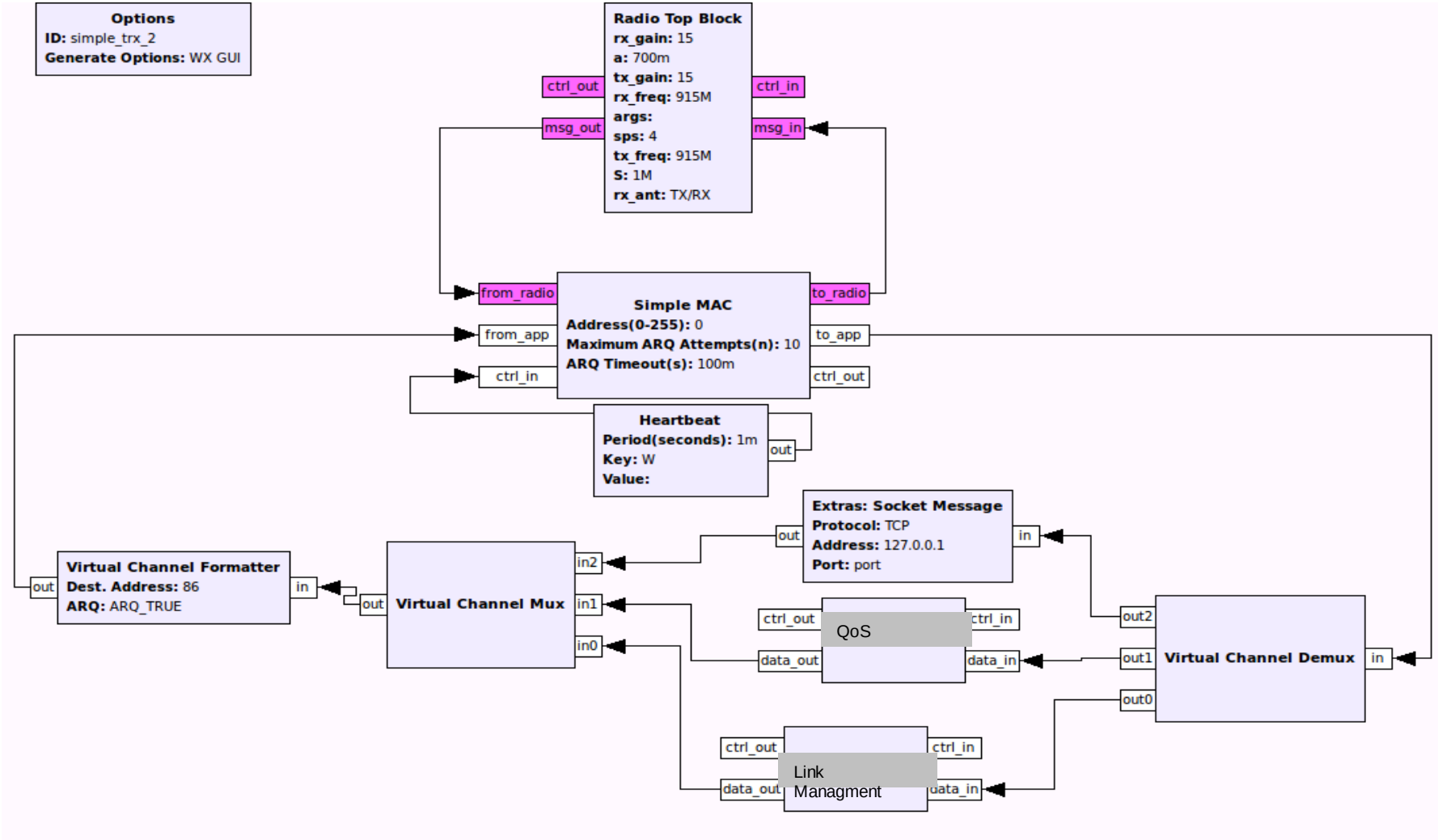


Tunnely.py Equivalent



Upper-Layer Protocols

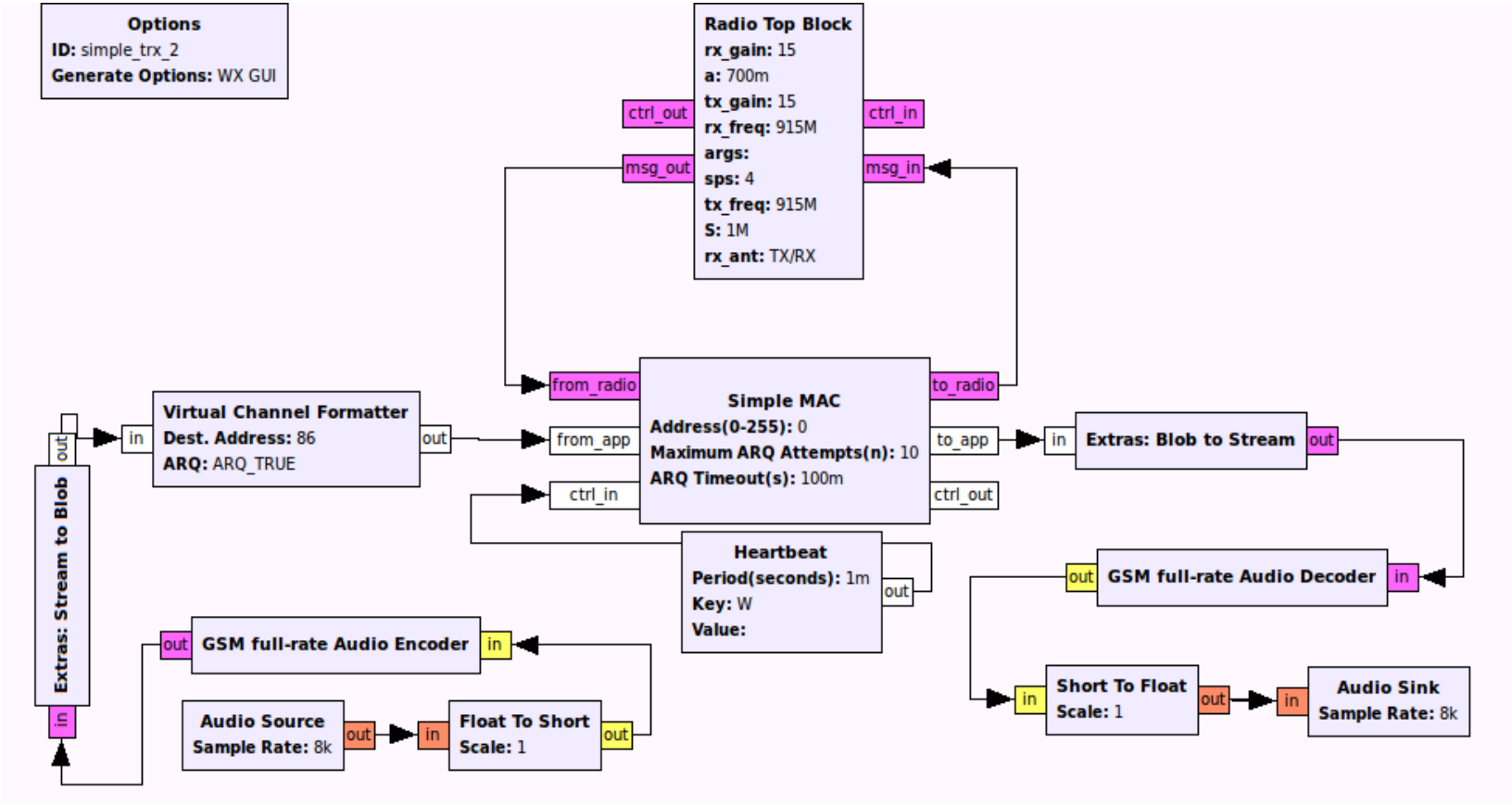
Ettus Research



QoS

Or any other kind of data...

Ettus Research



Conclusion

Ettus Research

- A lot of new, useful features:
  - Message Passing
  - PDUs
  - OFDM PHY
- Additional work:
  - Finish upgrade to 3.7.0
  - Upgrade documentation
  - Integrate sliding-window ARQ
  - C++
  - More optimal FHSS setup
  - FPGA-based PHY w/ PCIe