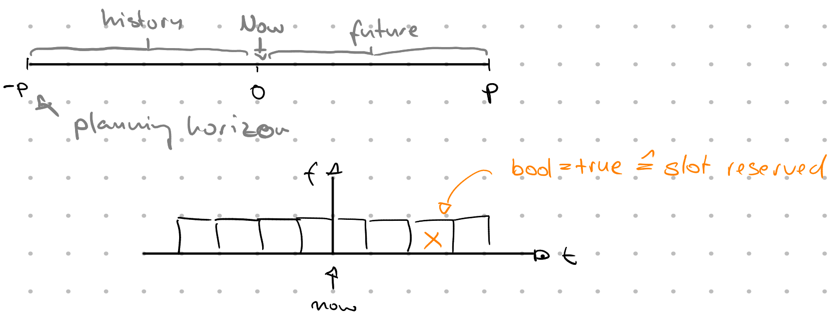
MC-SOTDMA Implementation Diary

Frequency Channel → center freq & bandwidth & is_p2p & is_blacklisted

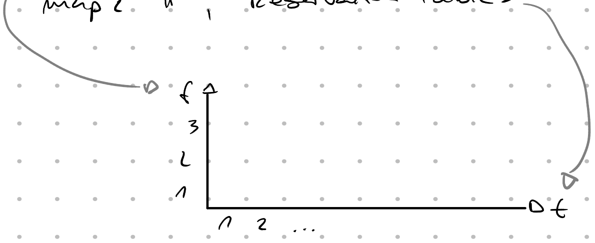
should move to a channel blocklist!

Reservation Table → std::vector<bool> utilization_vec

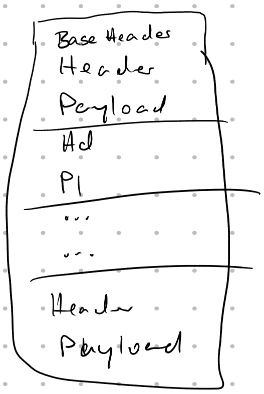


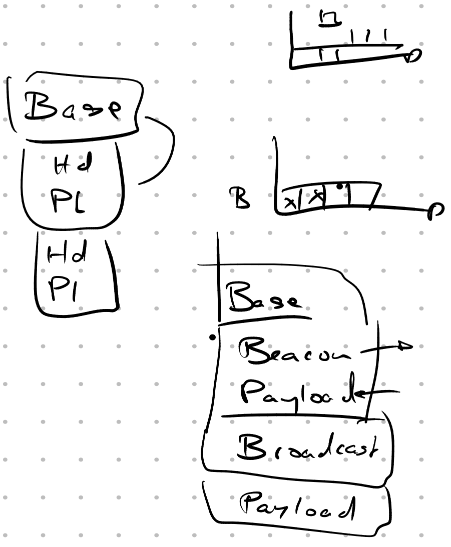
⇒ supports
isIdle(slot) isIdle(range_start, length)
findEarliestIdleRange(start, length)
mark(slot, utilized)
update(num_slots) → shifts vector to left

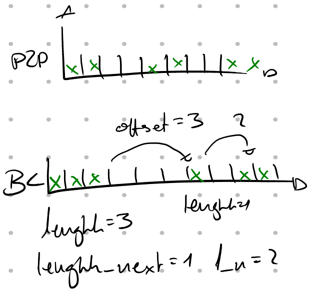
Reservation Manager → map< freq, Frequency Channel>
map< " , Reservation Table>



Frame Format →







Base Header

ICAO ID	27
offset	8
length_curr	4
timeout	8
type	3
CRC	16
	3
Σ	70

+ length_next 4

Beacon

type		
Position	12 + 14 + 12	} 65
CPR odd/even	1	
#hops to GS	5	
pos_quality	2	

P2P usage as payload?

Broadcast

type

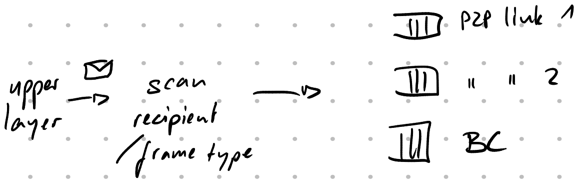
no additional

Unicast

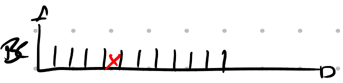
type		3
destination	ICAO ID	27
ARQ:		
arq	1/0	1
seqno	?	8
ack_no		8
ack_slot		8

proposal as payload!

Workflow



new BC packets → BC slot selection → slot arrives → concatenate as many packets from the queue as possible



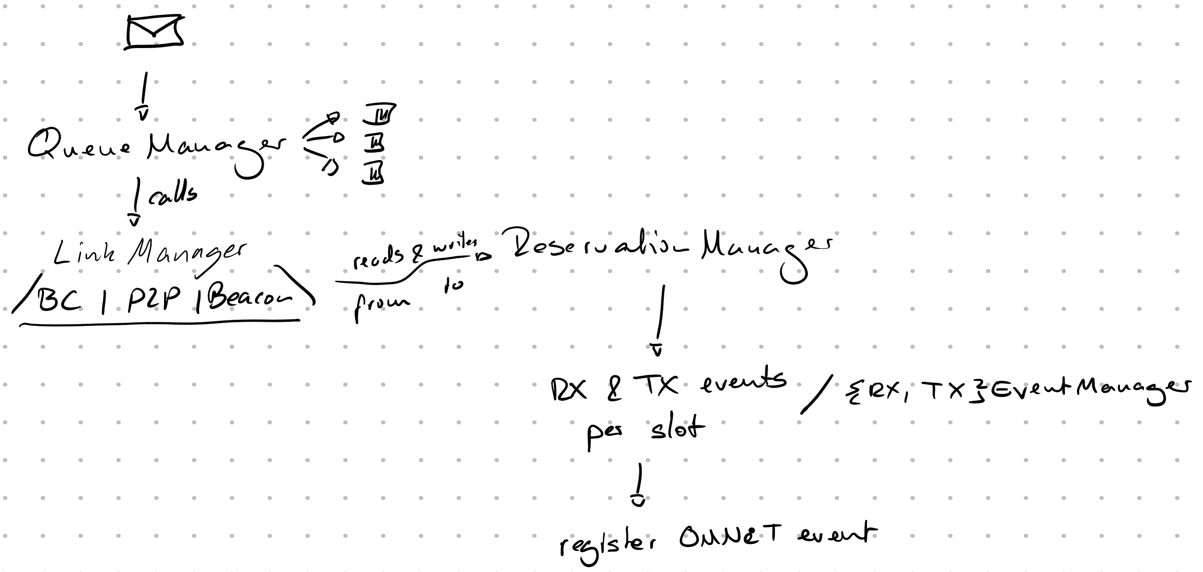
→ if more data → advertise next slot
→ don't

beacon slot @ init → arrives → put P2P map + as much broadcast data as possible
→ beacon slot selection

new P2P link req. → request ↓ BC → sent → keep track of handshake

→ mark all proposal slots as RX → rcvd ✓
→ lost → rtx (w/ counter)

P2P slot arrives → concatenate as much as possible → send



Possible Header Combinations

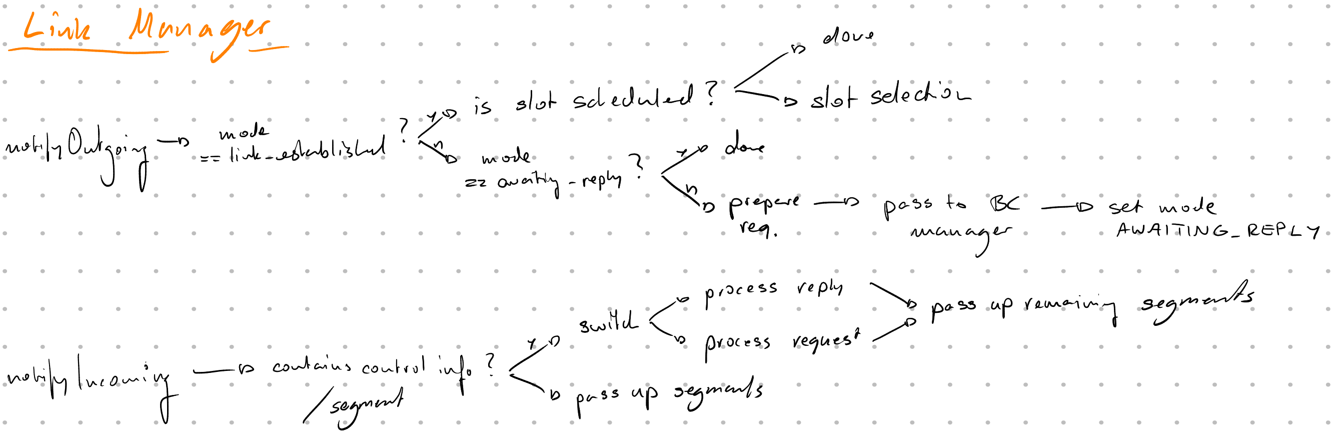
①
Base
Beacon
Broadcast | n Unicasts

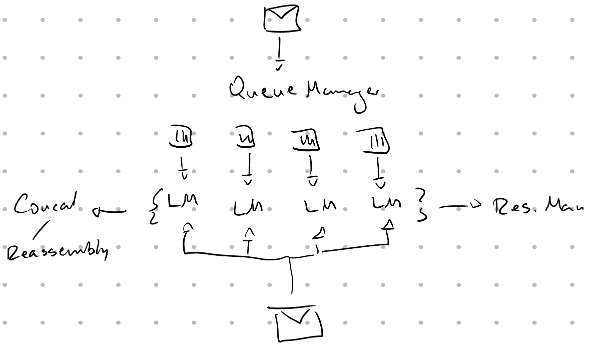
②
Base
Broadcast
n Unicast

③
Base
Unicast

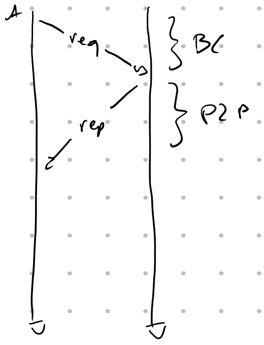
④
? Base
Unicast@BC

Link Manager





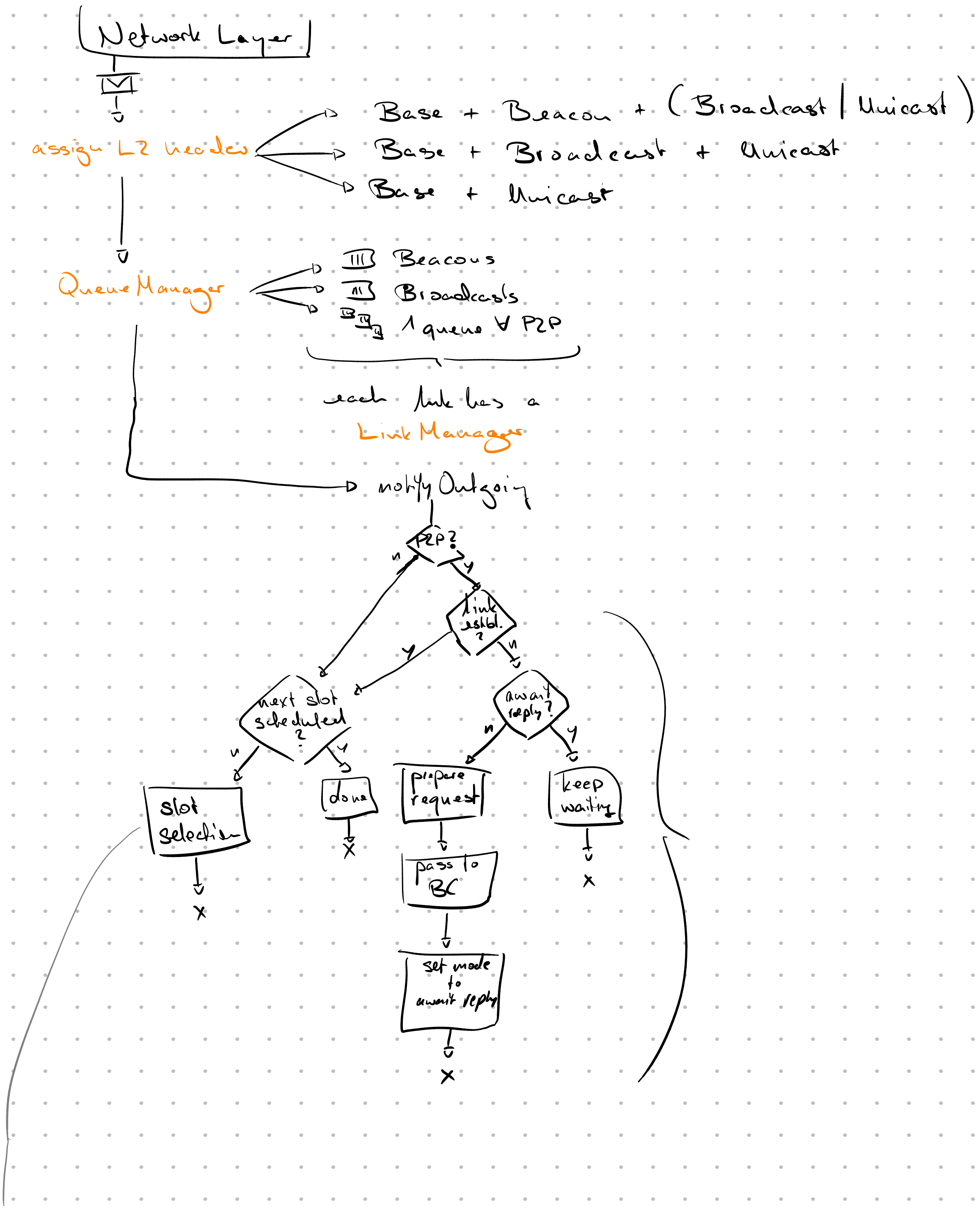
Link Establishment Request Creation

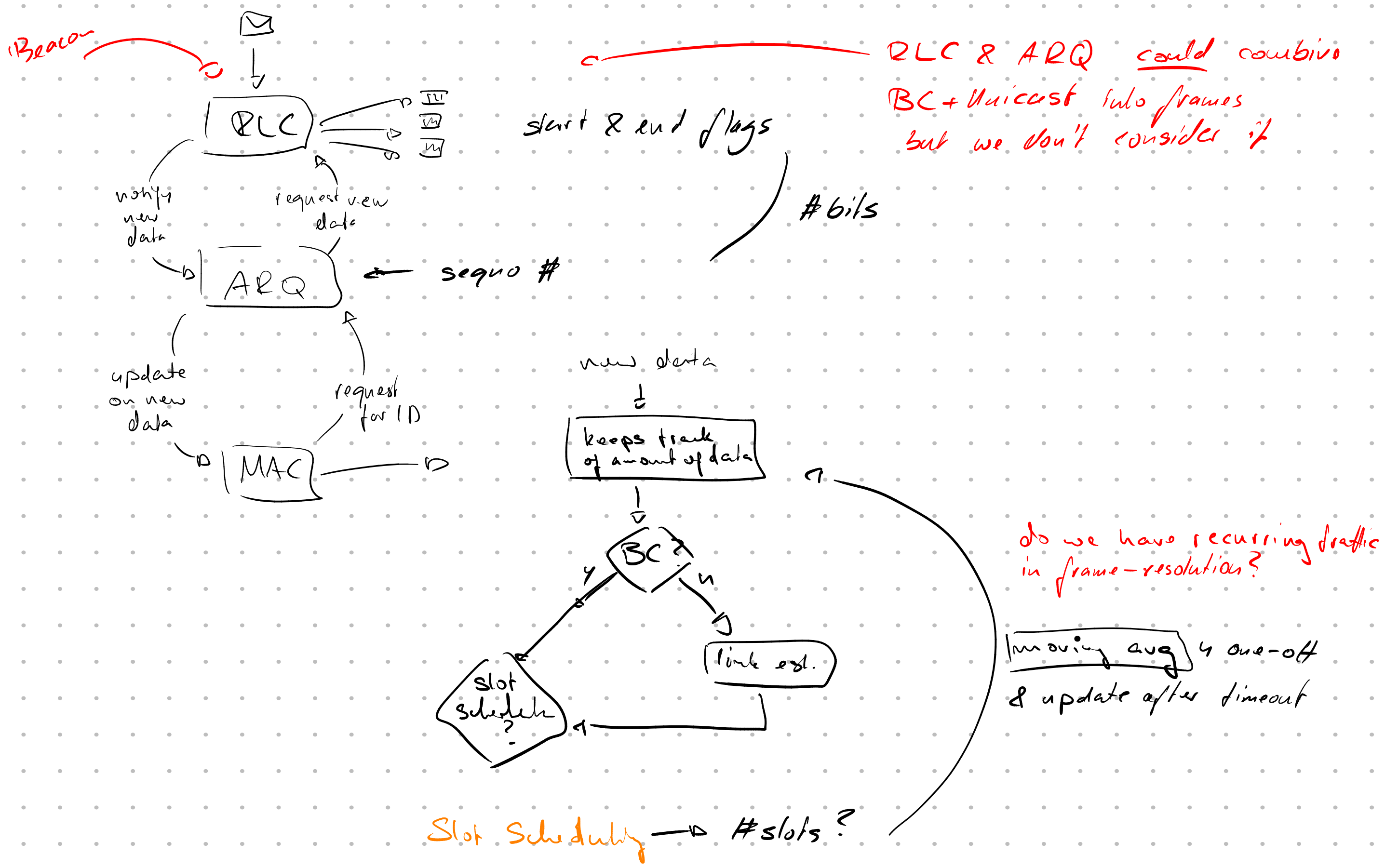


req has header unicast after a broadcast header and the proposal as payload

proposal {(freq, slot, length)}
#proposals → configurable

Current State of Affairs 13.11.2020

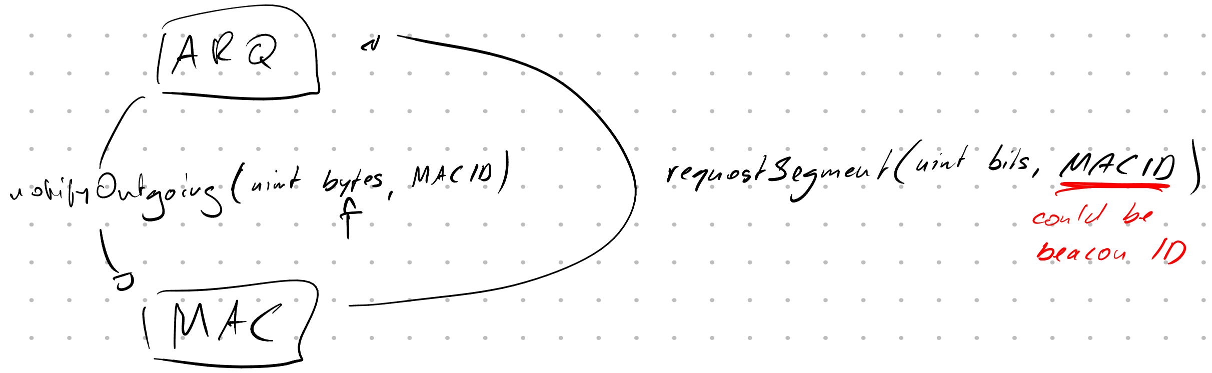




RLC & ARQ could combine
BC + Unicast into frames
but we don't consider it

do we have recurring traffic
in frame-resolution?

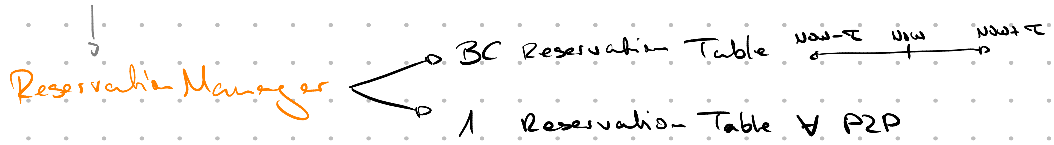
moving avg ↳ one-off
& update after timeout

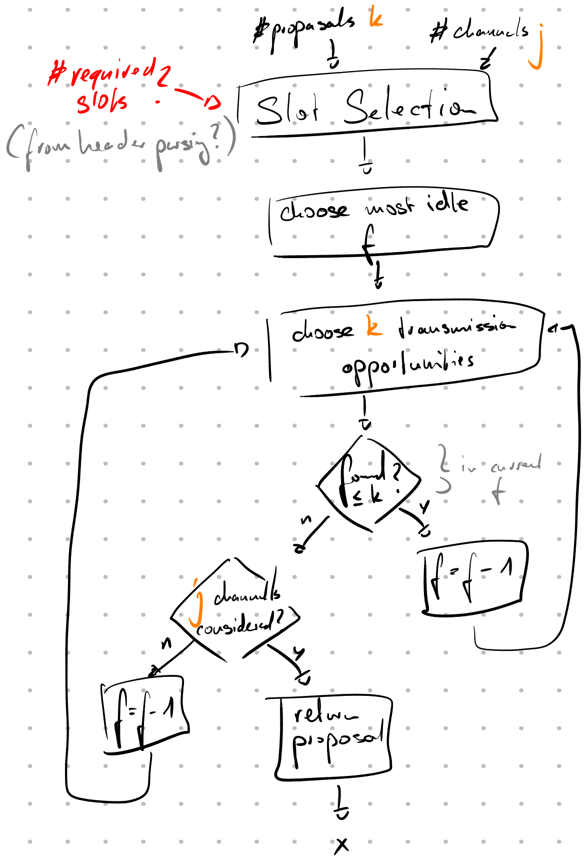

ARQ
notifyOutgoing(uint bytes, MACID)
f
MAC
requestSegment(uint bits, MACID)

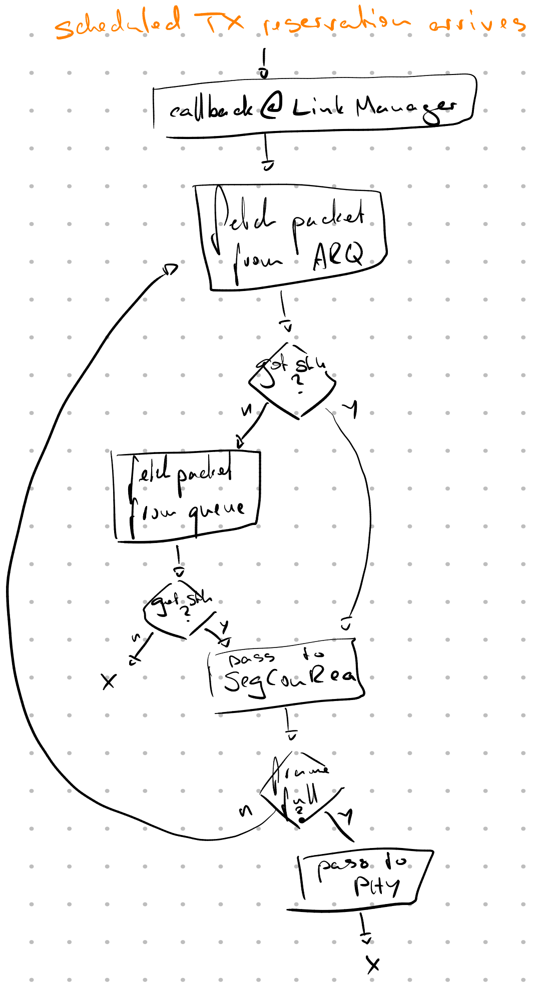
could be
beacon ID

20
40
60
...
500
470

moving avg


Reservation Manager
BC Reservation Table
now−τ
now
now+τ
1 Reservation Table ∀ P2P

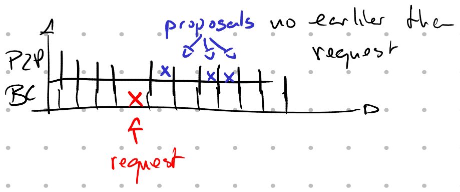

#proposals k
#channels j
#required slots
(from header parsing?)
Slot Selection
choose most idle f
choose k transmission opportunities
found ≤ k?
in current f
n
y
f = f − 1
j channels considered?
n
y
f = f − 1
return proposal
X

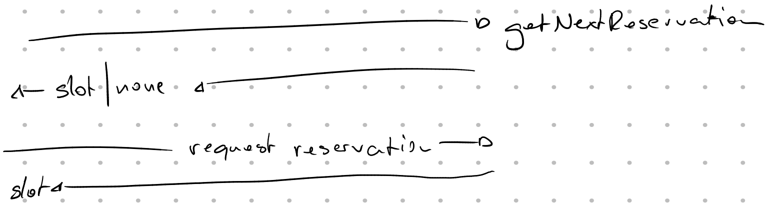

scheduled TX reservation arrives
callback @ Link Manager
fetch packet from ARQ
got sth?
n
y
fetch packet from queue
got sth?
n
y
X
pass to SegConReq
frame full?
n
y
pass to PHY
X

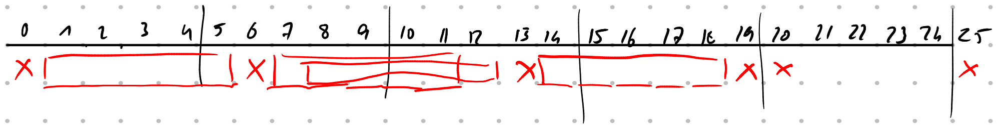
LinkManager
P2P

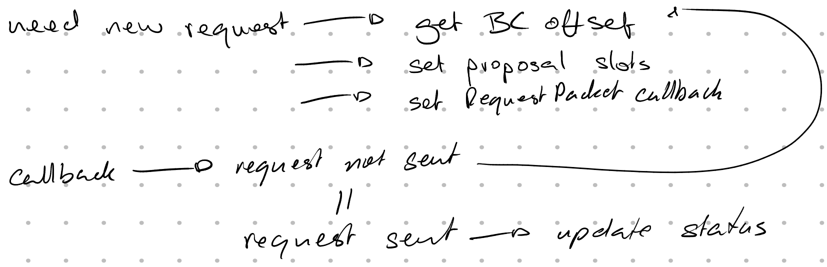
LinkManager
BC

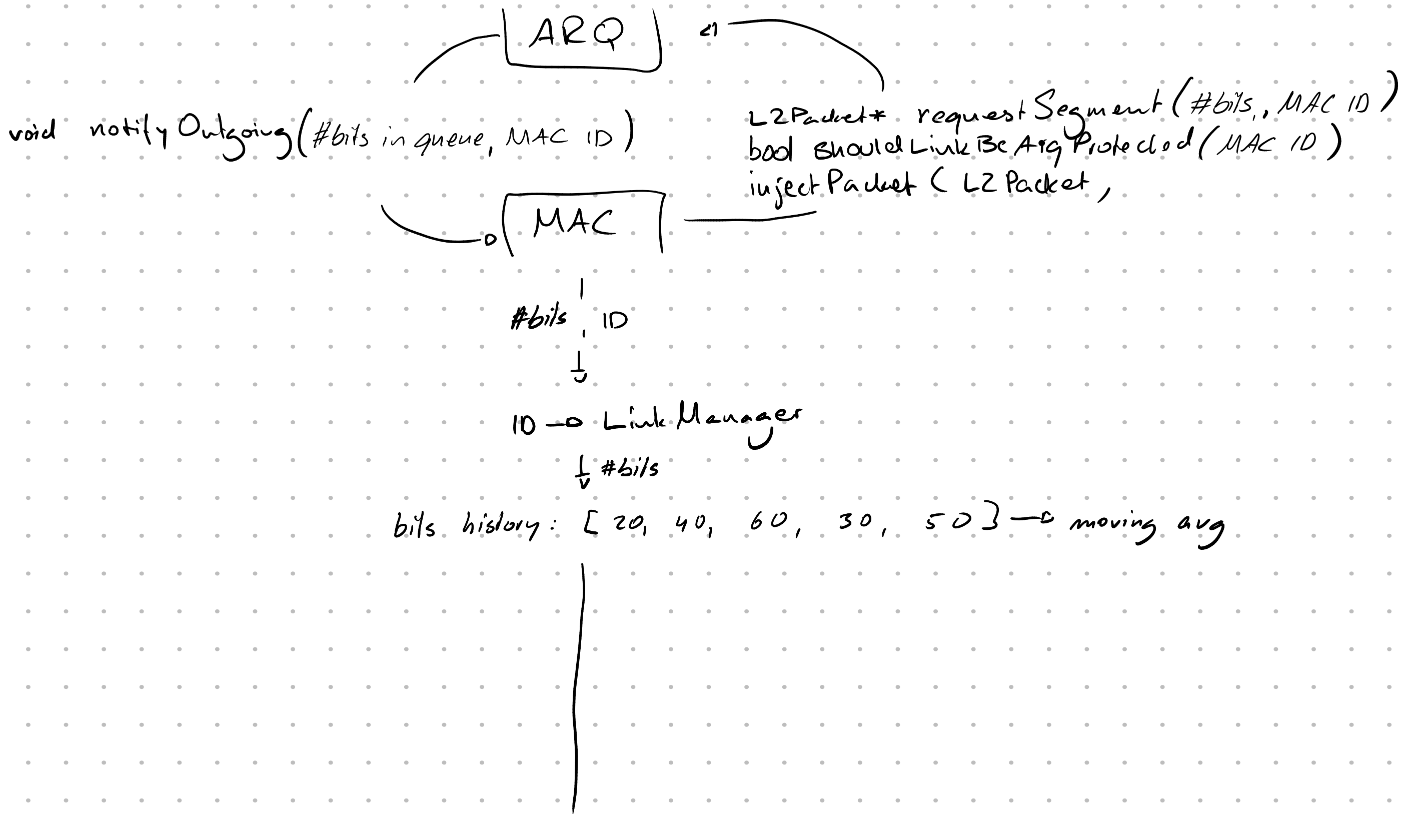
→ new request
→ proposal
→ slots → earliest slot?



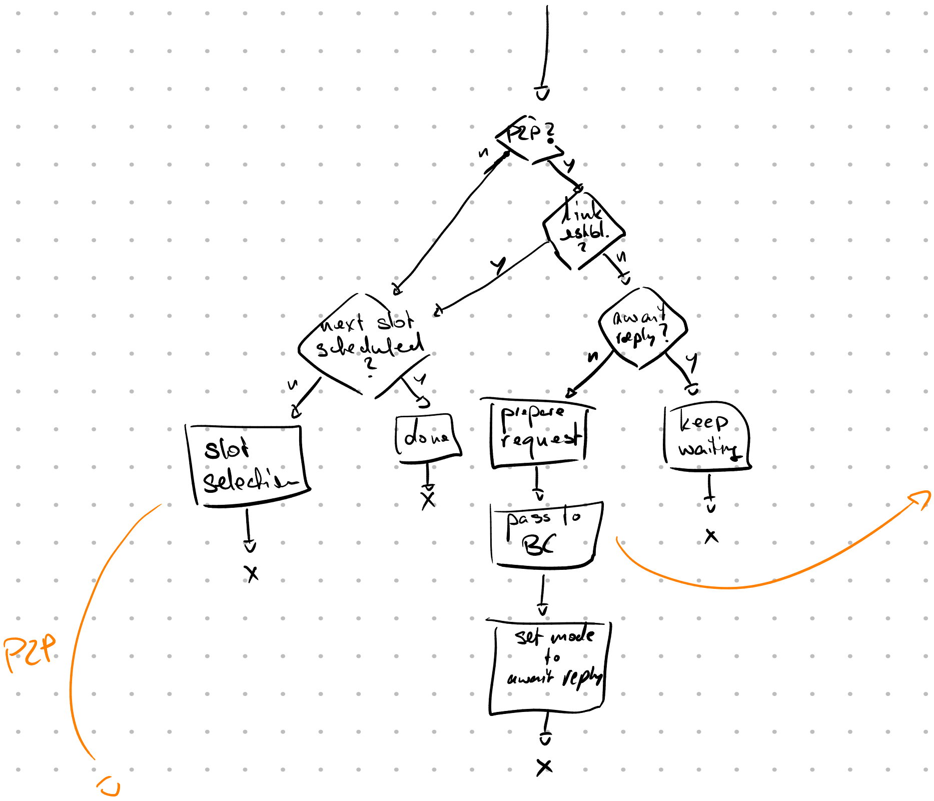








```
void notifyOutgoing(#bits in queue, MAC ID)
L2Packet* requestSegment(#bits, MAC ID)
bool shouldLinkBeArqProtected(MAC ID)
injectPacket( L2Packet,
```



request injected into RLC

↓

request is in BC queue

↓

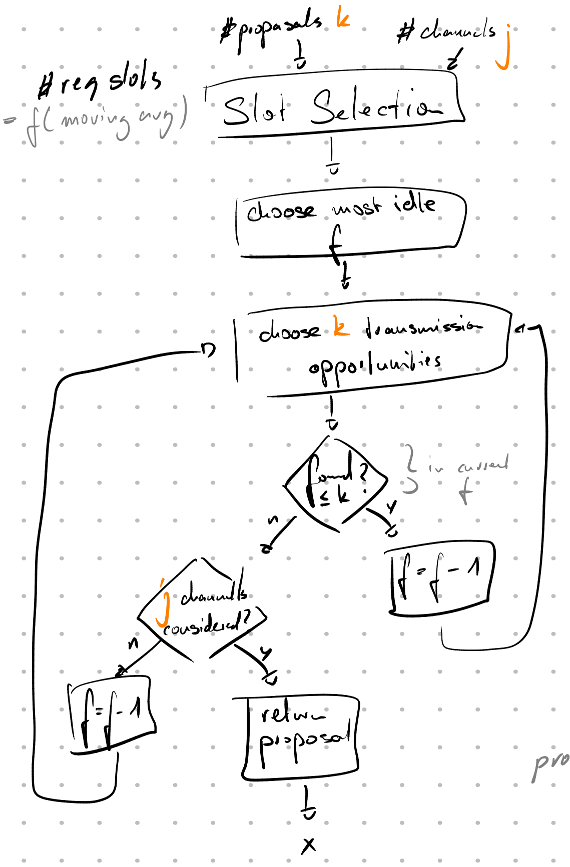
slot arrives

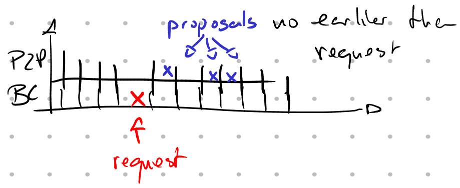
↓

callback to original P2P Link Manager

↓

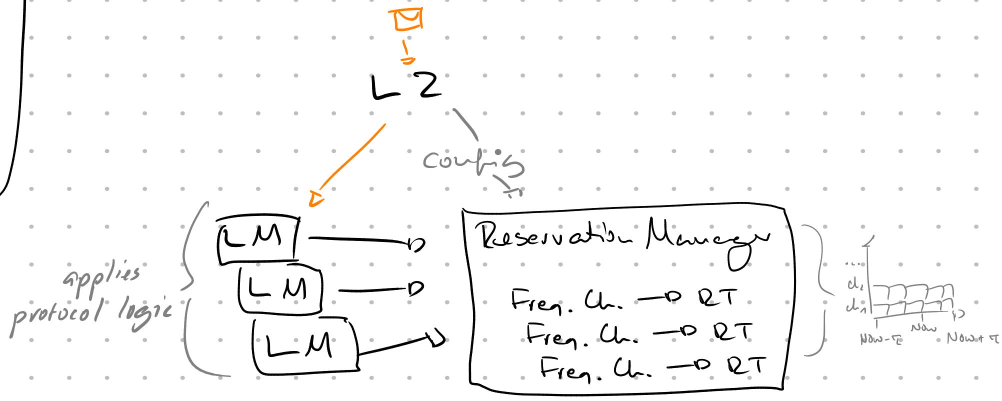
compute proposal and put into request (which must've reserved space beforehand)





TODO BC SS

Beacon SS



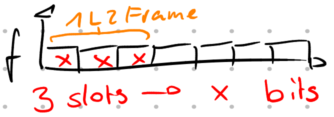
Slot arrives → LM is notified

→ PHY is queried for #bits that can be sent

→ ARQ is queried for data

→ passed down to PHY

L2 Frame



Base Header

66b
27b ICAO ID
8b offset
4b length
8b timeout
3b frame type } all other
16b CRC } headers too

Broadcast

23b
4b next_length

Beacon

65b
39b position
5b #GS hops
2b pos quality

Unicast

79b + SREJ
1b ARQ ?
8b seqno tx
8b seqno rx
8b arq slot
8b SREJ
27b dest ICAO ID

Link Request

79b + 8b #
+ 8b #
same as unicast
+ proposal size

Link Reply

79b
same as unicast

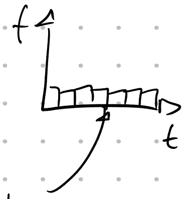
Base + Unicast

(1)
Base
Beacon
Broadcast | n Unicasts

(2)
Base
Broadcast
n Unicast

(3)
Base
Unic

dB. update
→ Slot arrives → Reservation's MacID
→ LM.onTxSlot → requestSeg
→ pass down



① slot arrives → request segment
→ notify LinkManager → set header fields
→ pass down

~~don't need callback~~
~~just do header parsing !~~ do need it, req → BC

② other direction

↑
MAC update reservations
↑
PHY

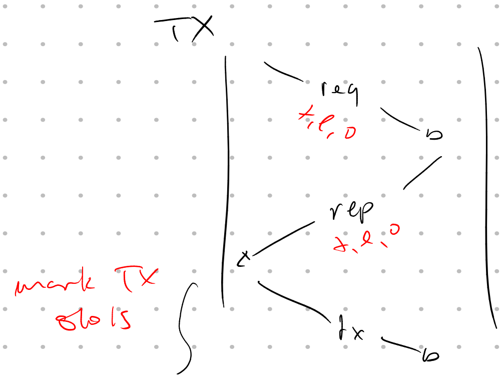
③ BC & Beacon Slot Selection
④ Setting headers

---

notifyOutgoing → requestNewLink → prepareRequest → setCallback
→ inject RLC
→ notifyOutgoing → schedule BC slot → mark

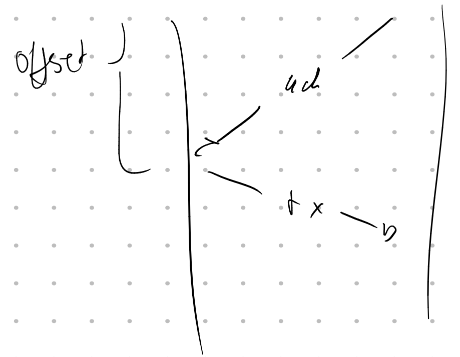
→ onTx → send

→ receiveFromLower → LM.rfl → parseHeaders
→ processBase
→ processRequest → prepareReply → forwardReply
→ scheduleReply TX@
mark RX @ offsets too



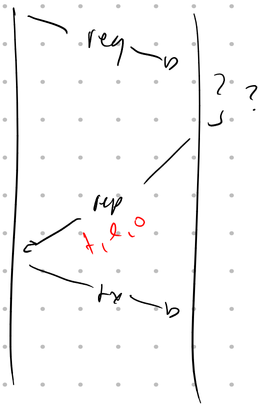
timeout, length, offset

timeout for TX!
length: other users can't decode w/o knowing it
offset: dyn 4 BC fixed 4 P2P



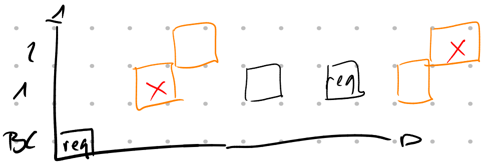
=> decrement timeout

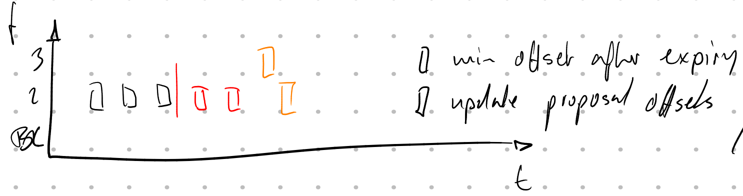
data -> RX => mark RX for t, f, o



proposal -> select

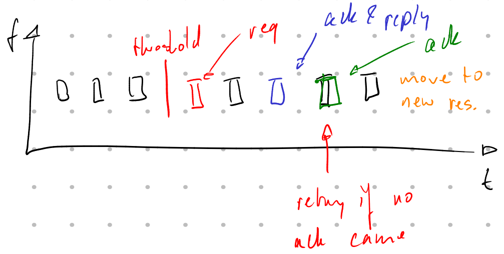
Link Renewal





reply in ACK

ignore RX when established



unack'd: ~~either BC negotiation or force ACK-like reply~~

min offset to after expiry

1. send req & await_reply
2. process incoming packets w.r.t. renewal

   -> detect too-late reply -> trigger next try
3. get reply

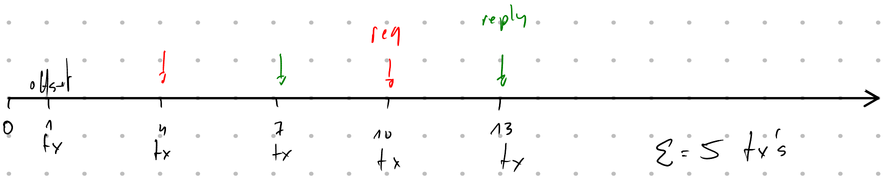
   -> prepare ACK

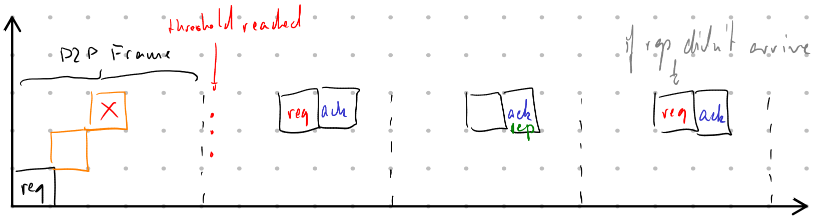
   -> save chosen reserv.
4. send ACK
5. expiry -> move to new reservation

(a) LM must know if bidirectional

(b) ACK expect. in unicast header if unid.

(c) schedule control frames from header processing





✓ TX schedule: leave 1 frame for reply

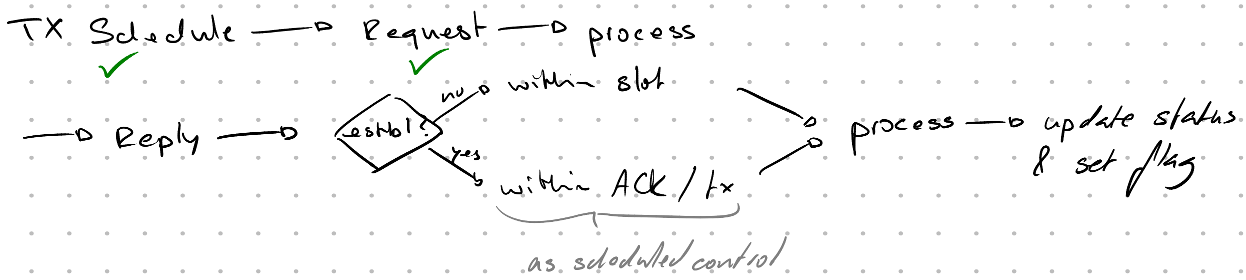
✓ process should be resettable

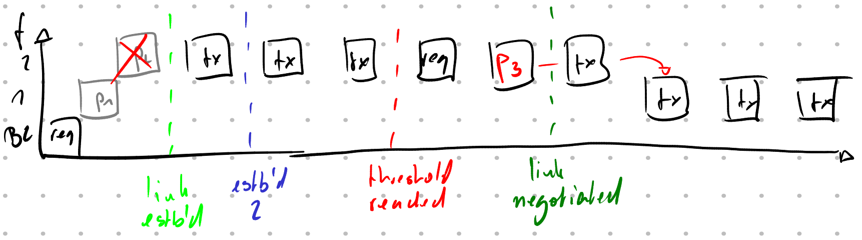
header parsing should forward replies to the process

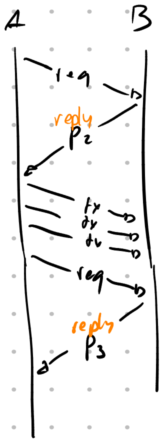
✓ process must calculate slot after which a new req is required

RX needs an ACK too (bit that indicates a new link is agreed)

actual change must be put into beacon







Tests

① est'd link, both have reservations & status

② expiring → both have status & tx has control msg

③ expiring + send req → status is reply_sent, RX has control msg

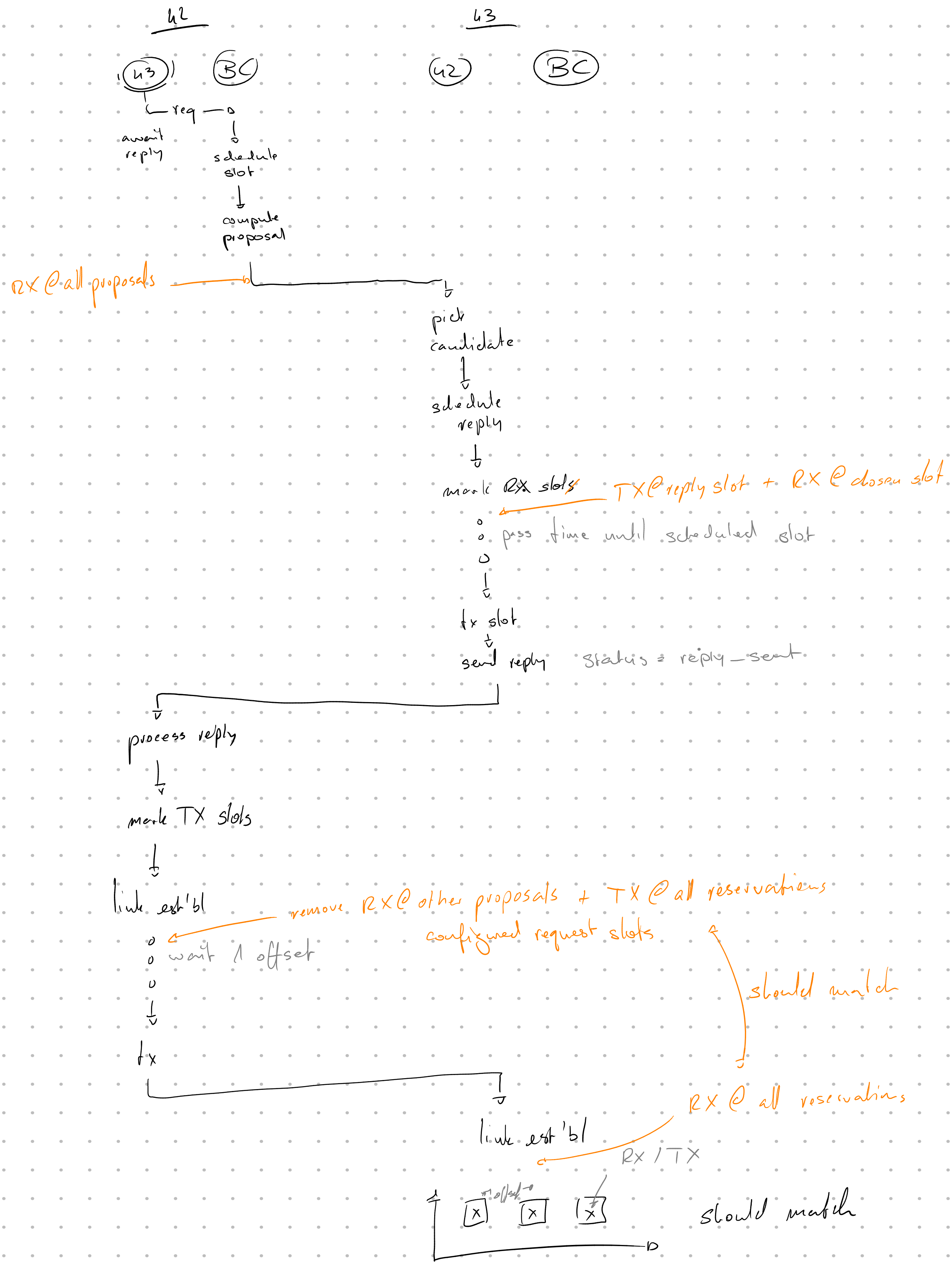
④ expiring + send rep → status is negotiated @ RX & TX

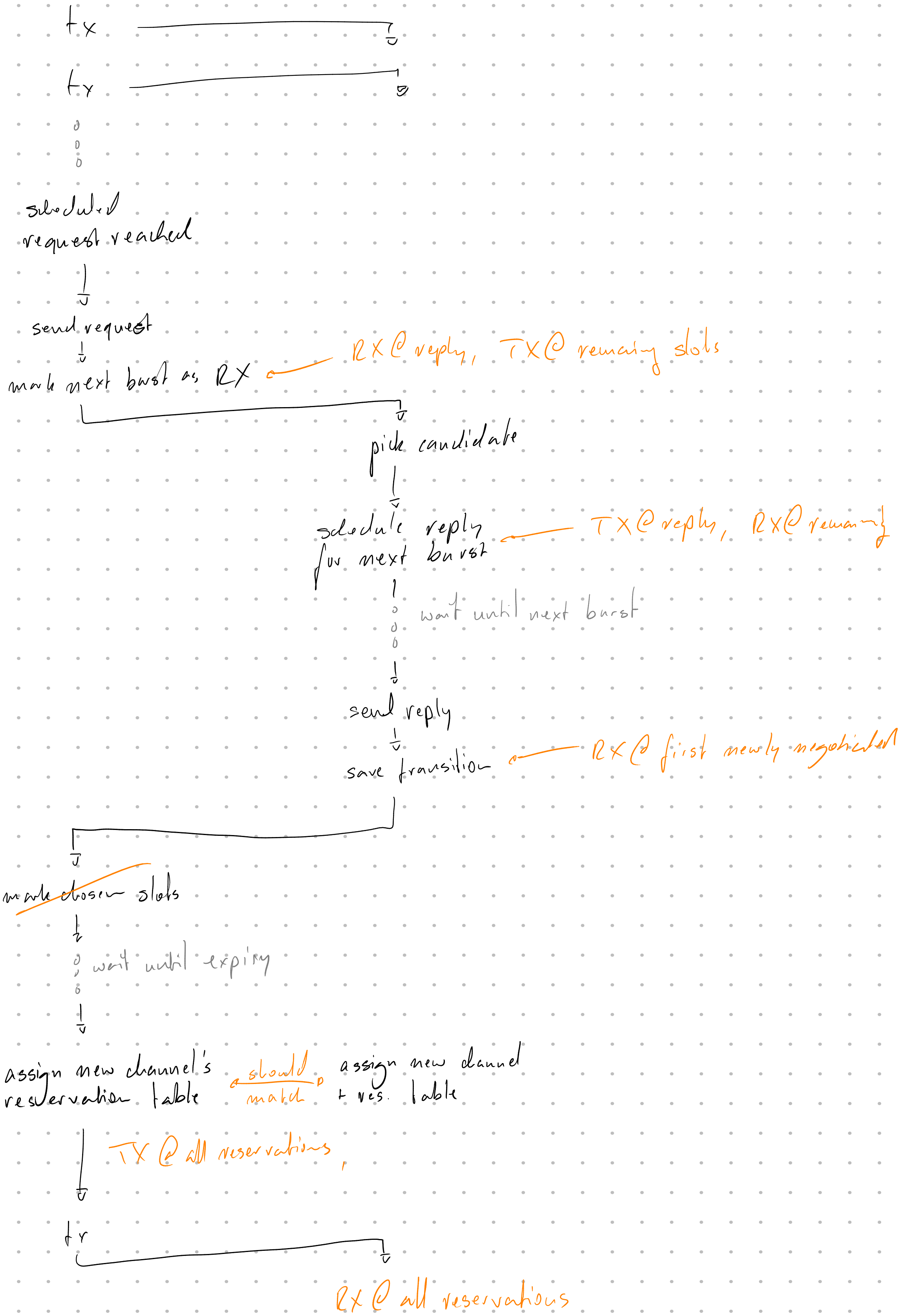
⑤ expiring + lost req → send 2nd try after not getting a reply (repeat for #tries)

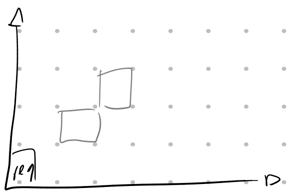
⑥ expiring + lost rep —▷ send new req until #tries

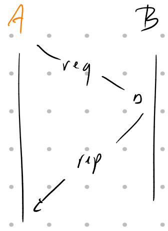
6.1 success
6.2 final fail => status is unestablished

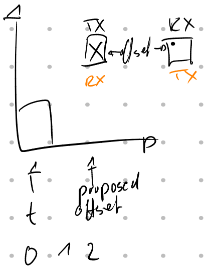
⑦ expired + new link

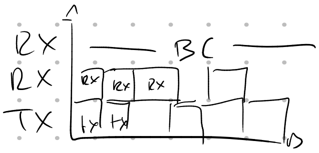


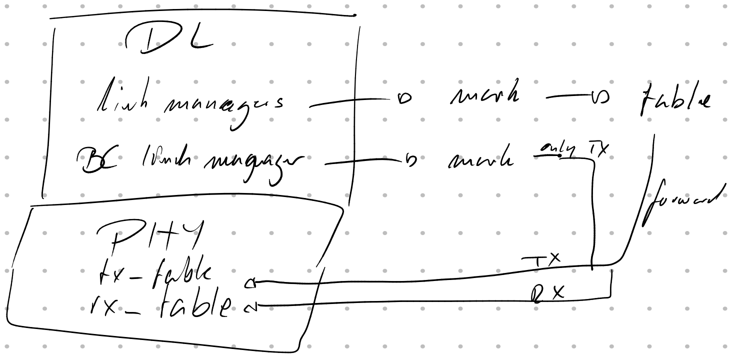

tx
tx
scheduled request reached
send request
mark next burst as RX
RX @ reply, TX @ remaing slots
pick candidate
schedule reply for next burst
TX @ reply, RX @ remaing
wait until next burst
send reply
save transition
RX @ first newly negotiated
~~mark chosen~~ slots
wait until expiry
assign new channel's reservation table
should match
assign new channel + res. table
TX @ all reservations,
tr
RX @ all reservations

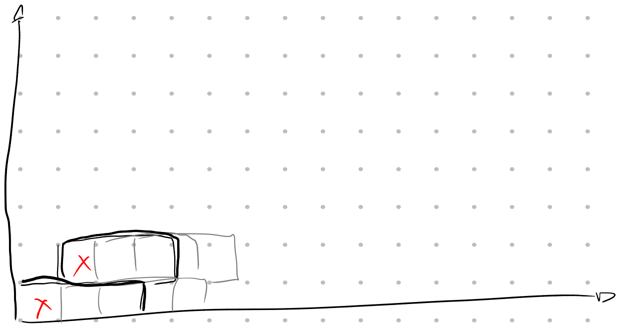

req

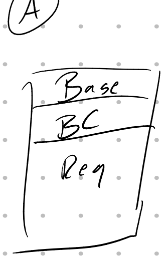

A
B
req
rep


TX
RX
offset
RX
TX
t
proposed offset
0 1 2


RX
RX
TX
BC
RX
RX
RX
TX
TX


DL
link managers
mesh
table
BC link manager
mesh
only TX
forward
PHY
tx_table
rx_table
TX
RX


X
X


A
Base
BC
Req

User ①

LM ②

LM BC

User ②

LM ①

generate request

pass to

update status "awaiting reply"

schedule slot

. . .

wait for slot

compute proposal

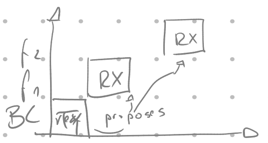
mark RX slots ∀ resources ∈ proposal

transmit

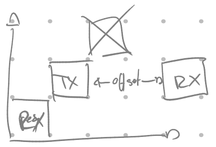
pick viable candidate

schedule reply

mark TX @ reply slot
RX @ reply slot + 1 offset



@ ①



@ ②

. . .

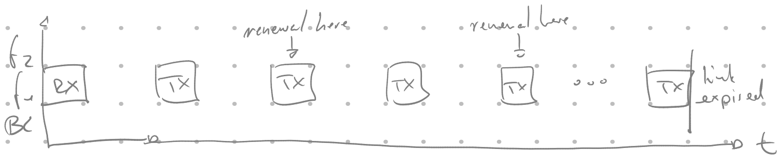
wait for slot

transmit

clear pending RXs
assign channel to link
reset timeout
mark all TX reservations
configure renewal request slots
update status "link established"

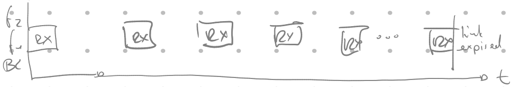
. . .

wait until next burst



@ ①

transmit



@ ②

update status "link established"
mark all RX reservations

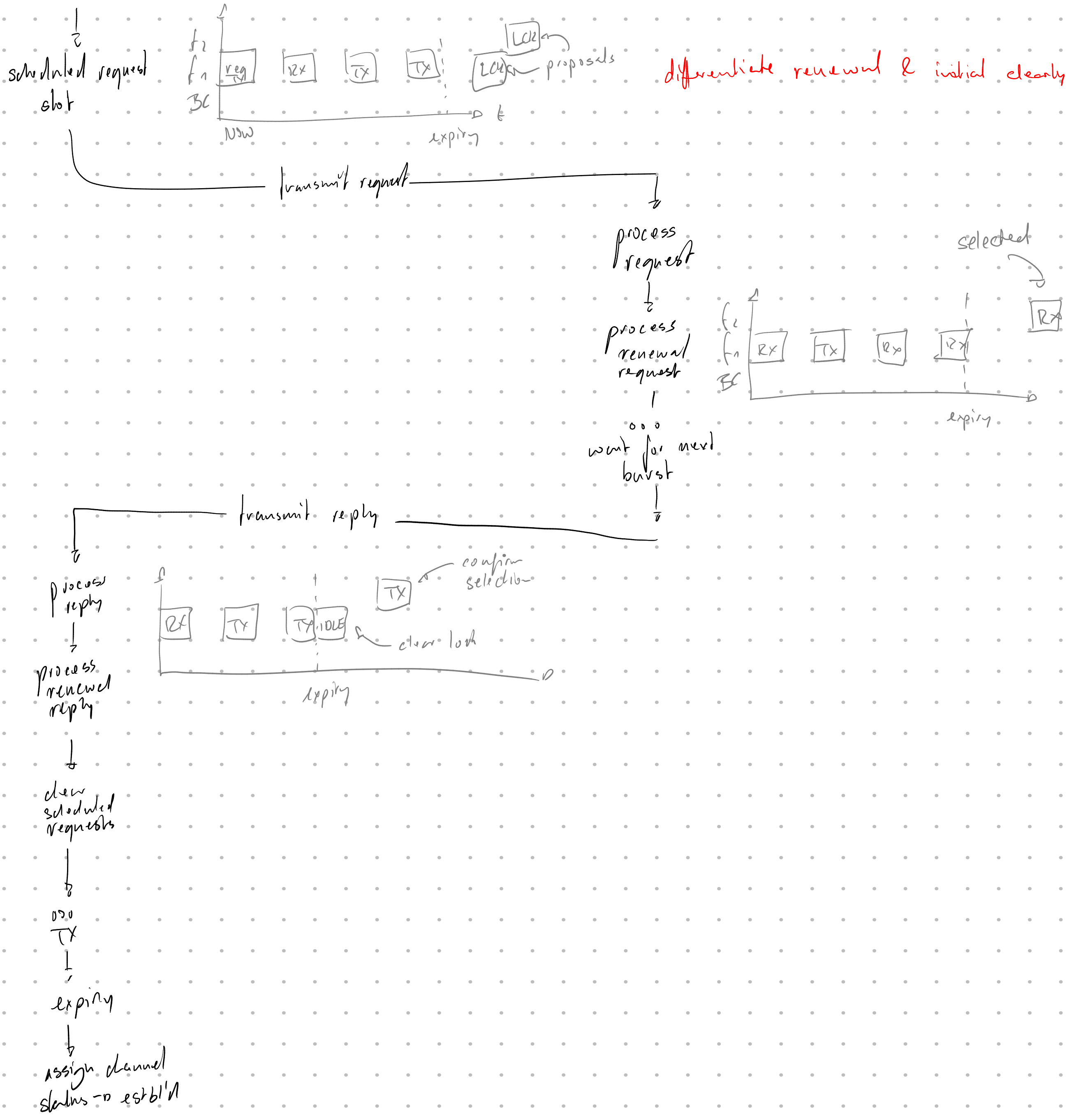
. . .
wait until next burst

mark all RX reservations

. . .
wait until next burst

mark all RX reservations

. . .



differentiate renewal & initial clearly