IETF 103 Bangkok ROLL-BIER Design Team

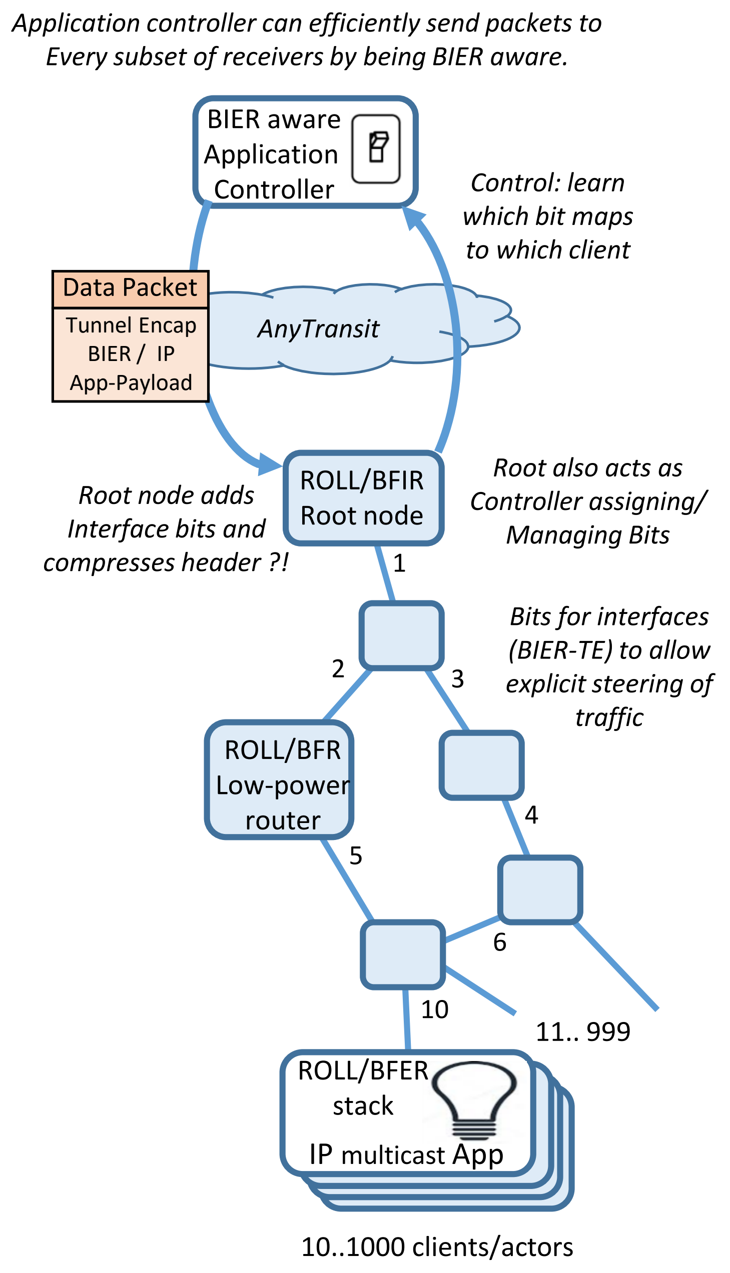
Toerless Eckert (Huawei), <tte@cs.fau.de>

BIER in ROLL Vision

Design Team

- Consider joining/collaborating in BIER design team in ROLL-WG:
- Email: roll-bier-dt@ietf.org (normal subscribe)
- https://trac.ietf.org/trac/roll/wiki/roll-bier-dt (to be filled)
- Issues: tte@cs.fau.de

- What could be cool about this (if design team decides to do it) ?
- End-to-end BIER (with TE) in low-power networks (e.g.: building control)
  - Example: Application Controller sends BIER packet to subset of clients (lightbulbs)
  - Each client is BFER (has a bit)
  - Every packet can address a separate subset of actors through bitstring
  - Only controller app needs to be BIER aware. Receivers can think its just IP multicast.
- BIER TE bits to save power/memory
  - Routers are low-power (memory/CPU). Do not want to keep large routing table (1000 lightbulbs). Links are low power too.
  - Every interface has a bit. Routers only need to route on bits to directly connected downstream neighbors.
- No ASIC constraints. Everything is software
  - Headers/Bitstring can be compressed (loss free, lossy (bloom) to support long bitstrings.
  - Should result in header more compact than existing ROLL/RPL headers even for unicast: Only hop-by-hop bits sets to one receiver: Would also be used for unicast forwarding.



Application controller can efficiently send packets to Every subset of receivers by being BIER aware.
BIER aware Application Controller
Control: learn which bit maps to which client
Data Packet
Tunnel Encap
BIER / IP
App-Payload
AnyTransit
Root node adds Interface bits and compresses header ?!
ROLL/BFIR Root node
Root also acts as Controller assigning/ Managing Bits
1
2
3
Bits for interfaces (BIER-TE) to allow explicit steering of traffic
ROLL/BFR Low-power router
4
5
6
10
11.. 999
ROLL/BFER stack
IP multicast App
10..1000 clients/actors

Status of dt

- Slow start but getting more organized
  - Try weekly meeting via IETF webex, Wed. 7 PM GMT
  - until we can get work items assigned to individual stakeholders
    - Getting participants up to speed
  - Will redo role call for meeting time/cycle after IETF103
- Started to put slide deck (this) and notes etc. into github
  - www.guthub.com/toerless/roll-bier (no ROLL WG github repo group ?)
- Worked through bunch of arch level details
  - But in middle of writing them down

Relevant docs

- draft-thubert-roll-bier
  - Proposed arch of BIER via ROLL
    - Core – not IP Multicast overlay, not L2.5 encap
    - Intends to support BIER/BIER-TE x explicit/bloom-filter mode of operations
- draft-ietf-roll-ccast
  - BIER for ROLL with bloom filters. Expect separate (IP multicast) overlay to handle false positives
- draft-thubert-6lo-bier-dispatch
  - L2.5 proposed encap for BIER packets
  - Comparable layer/function to MPLS/Ether BIER encap (RFC8296) ?!
    - Aka: would not use RFC8296 because in 6LO networks it all about compression
- draft-thubert-roll-unaware-leaves
  - Not covering BIER, but introduces type of nodes we want to support via roll-bier too.

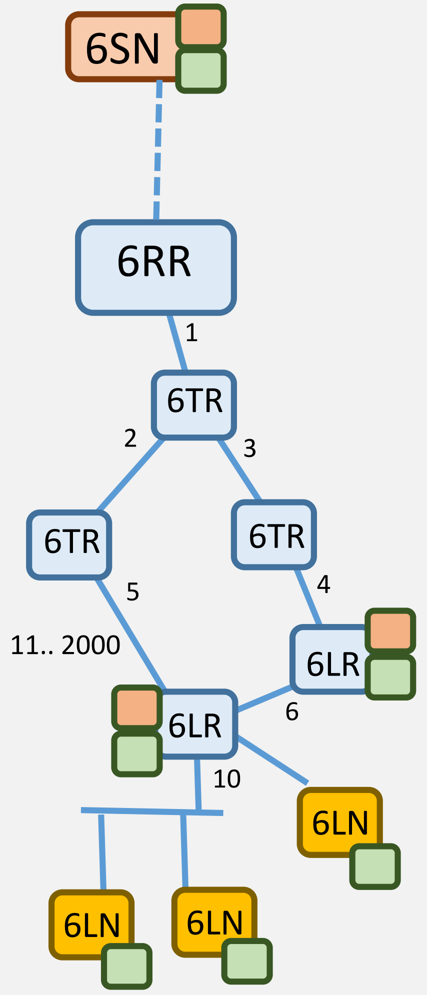
- draft-other-apologies-forgetting-you
  - Please help complete this list

Framework/ Terminology

ROLL+BIER+6LoRH

- Tunnel Mode
    - BIER, BIER-TE: Bits assigned to 6LR
    - BIER-TE: Bits also assigned to adjacencies/6TR
    - No bits assigned to 6LN
- Services
    - BIER 6SN-> 6LR + BA
    - IP unicast 6SN -> 6LR/6LN + IA
- 6LoRH + BIER
    - Compress/uncompress unicast/multicast packets
    - 6RR ... 6LR
    - 6LR ... 6LN ???

Framework



Terminology

🟧	*6SN = Server Node / Application Not running ROLL/BIER able to send Multicst packets with BIER bitstring to target set of destinations explicit*
🟦	*RPL + BIER routers Roles: 6RR = root. needs to support 6LoRH for BIER 6TR = transit – no bit in BIER mode 6LR = leaf (LoWPAN) Router bit in BIER mode, connects to 6LN needs to support 6LoRH BIER*
🟨	*6LN = Lightweight Node no knowledge of ROLL/BIER MAY want to support 6LoRH with BIER extensions for compression*
🟧	*BA = BIER aware app able to send/rcv packets with bitstring, no IP multicast needed*
🟩	*IA = IP unicast/multicast app. No BIER awareness*

Slide v1.0

Continuing to limit work scope by eliminating “below the line / do not work” options

- Only consider IP multicast payload for ROLL-BIER multicast now
  - Required for 6LN (non ROLL-BIER capable) nodes
  - For 6LR receivers we do primarily care that we can address them directly from the sender via bitstring, but not so important to get rid of potentially unnecessary IP Multicast header
    - IP Multicast header should be compressed also by 6LoRH ?!
- No new “faked” IP packets (see later slide)
- Only “transport mode” to 6LR, only “tunnel mode” to 6LN
  - See explanations later
- Only consider BIER-TE semantic, not BIER ?
  - TBD. Maybe we can start defining common forwarding and add BIER control plane later (when seen necessary)

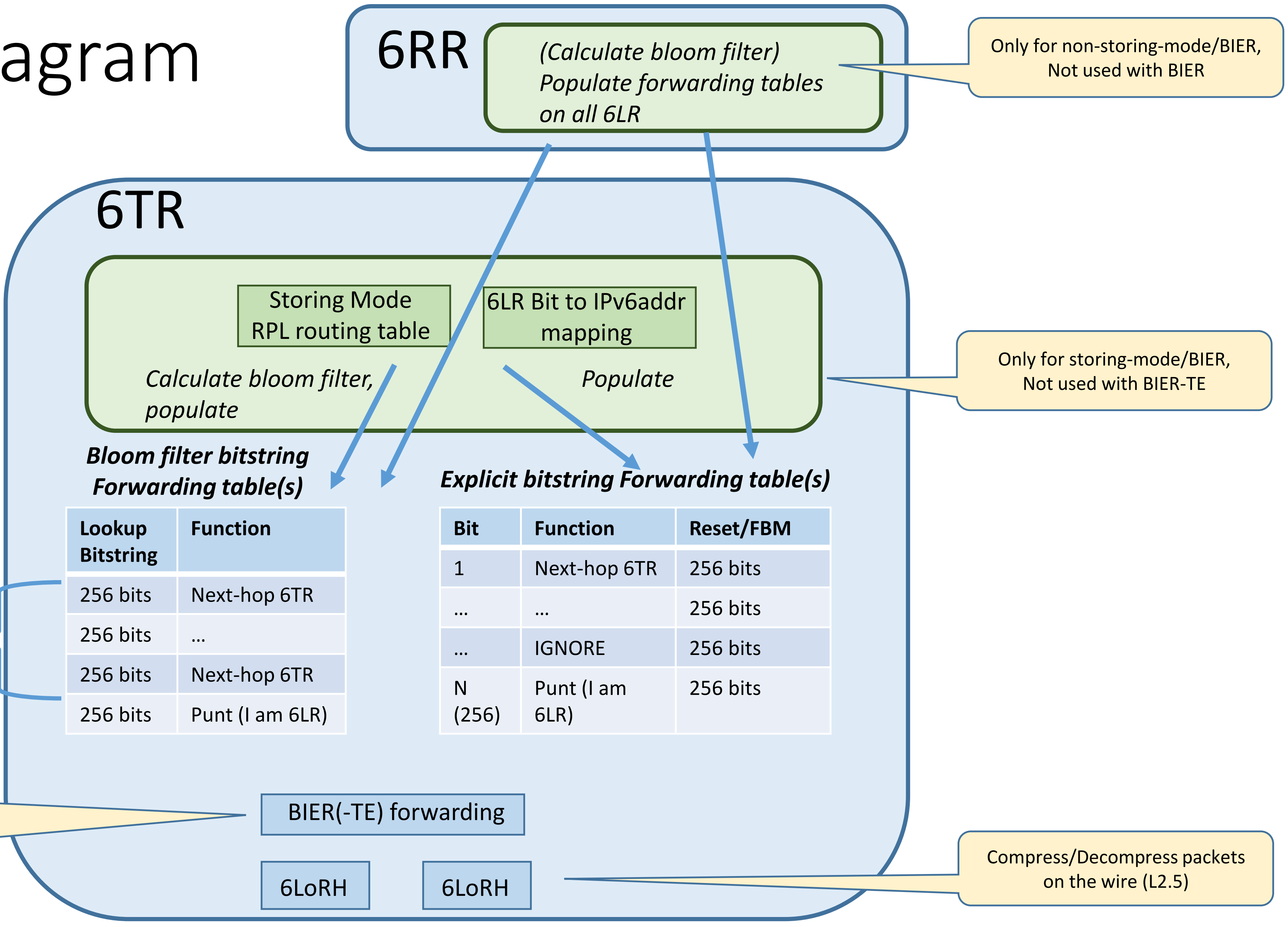
Details, Stack options

- First option: “Tunnel Mode” to 6LN, “Transport mode to 6LR”. Common header
  - Lowest layer is 6LoRH with BIER bitstring. Also all RPL artefact.
  - Next: 6LoPAN compressed IP packet (RFC6282)
- Transport Mode to 6LR:
  - 6LoRH indicates this mode, compres/uncompress destination address
  - Save 6LoPAN state for Dst address and compression field in 6LoPAN header (compressed state)
- Tunnel Mode to 6LN:
  - *Simple IP packet without any RPL/BIER artefacts.*
    - *High compression of IP address through state on every RPL hop for src/dst addresses (stateful)*
    - *Stateless – assumes MAC address derived IPv6 address.*
      - *But only works single L2 hop because it relies on the L2 destination “MAC”.*
    - *Aka: Stateful compression + BIER is big benefit because with BIER only the 6RR and 6LR would need to have state for the addresses because 6TR would just forward based on bits.*
    - *Need to show in slides how BIER benefit applies here:*
      - *No changes required for 6LoPAN to get desired benefit*
    - *FUTURE: define details of “transport mode” options.*

6RR/6LR diagram

Forwarding table does
Not need to distinguish
Between BIER/BIER-TE

Difference just in how
Next-hop entries/FBM for
Bits are created



Bloom filter bitstring Forwarding table(s)

Lookup Bitstring	Function
256 bits	Next-hop 6TR
256 bits	…
256 bits	Next-hop 6TR
256 bits	Punt (I am 6LR)

Explicit bitstring Forwarding table(s)

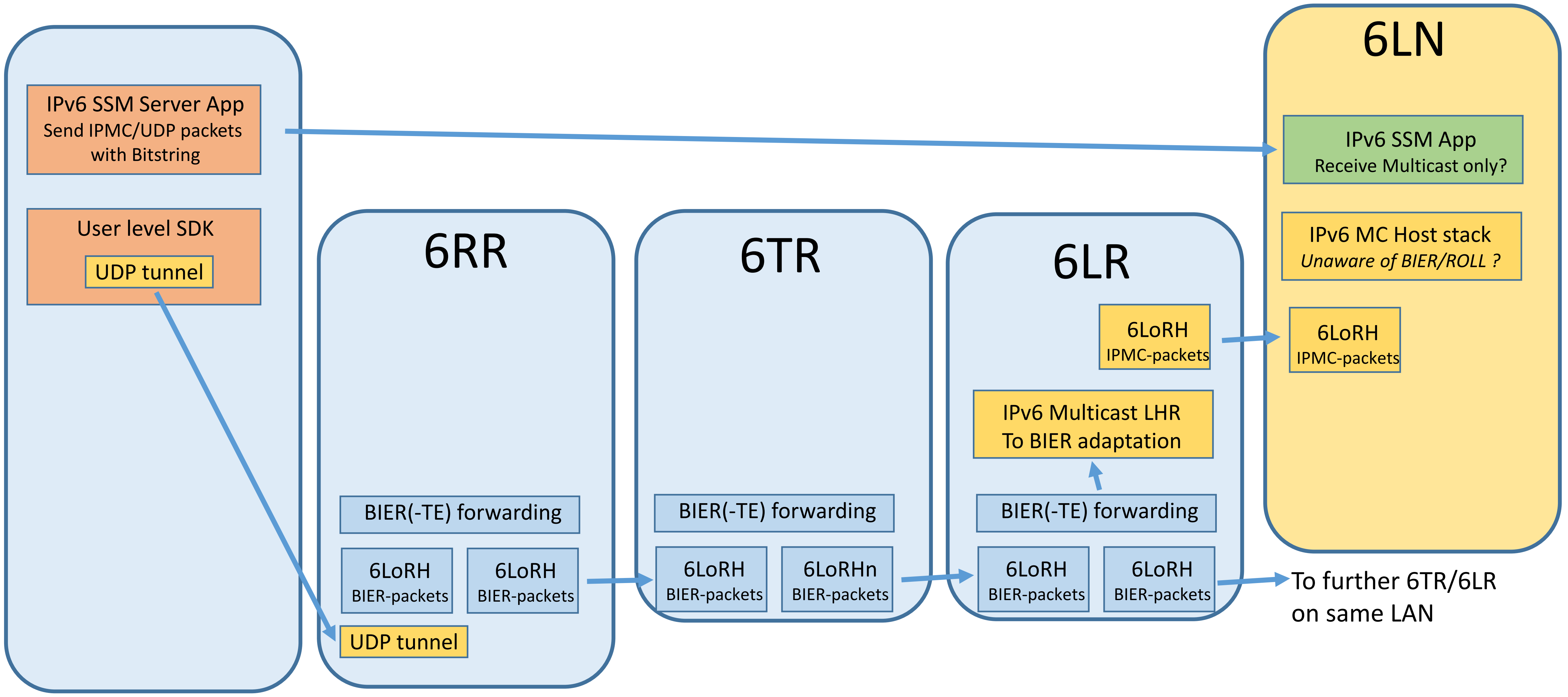
Bit	Function	Reset/FBM
1	Next-hop 6TR	256 bits
…	…	256 bits
…	IGNORE	256 bits
N (256)	Punt (I am 6LR)	256 bits

Storing mode (BIER):
#6LR entries
Non-storing mode (BIER):
#adjacent 6LR/6TR

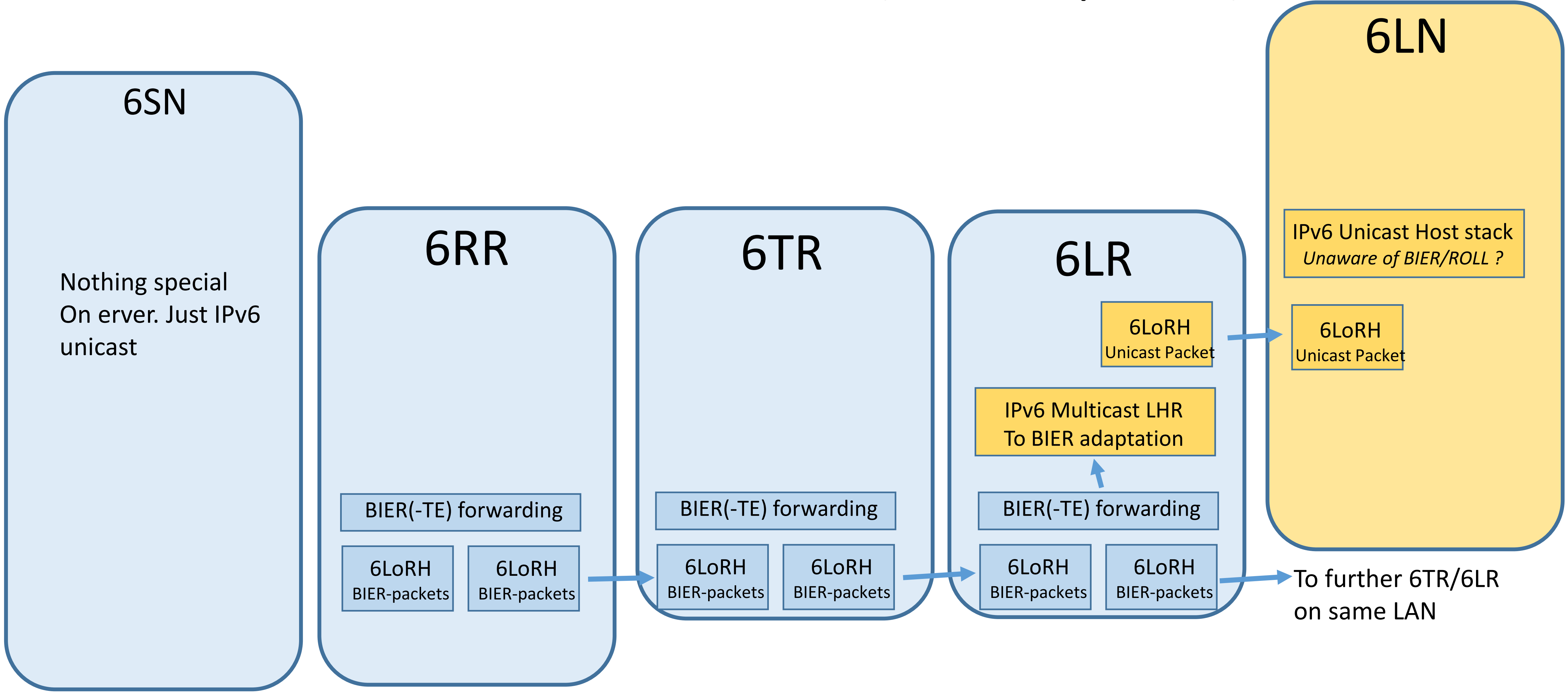
+1 if we are 6LR

One forwarding rule for bloom-filter, one for explicit bitstring. Distinction BIER/BIER-TE is just in how forwarding table gets populated

IP multicast over ROLL-BIER



IP unicast over ROLL-BIER (incomplete)



Bit-Reset/
Bloom Filter considerations (not reviewed)

- With bloom filter bitstring, bits can not be reset
- BIER logic
  - Leads to duplicates. Other radio-network issues can also lead to duplicates, so maybe not a big issue – radio networks MUST be prepared to deal with duplicates
  - Routing protocol microloops are avoided by reset. If RPL can have microloops they would only be protected against by TTL expiry (eg.: > 100 packet replications possible).
  - False positives in bloom filter can create more duplicates and more replicated packet when there is a routing microloop
- With BIER-TE
  - No routing protocol involved = no IGP microloops.
  - False positives can create duplicates AND looping (like badly set bits by BIER-TE controller can create loops too)
- Solutions ?
  - Have ROLL-BIER domain TTL (6LoRH TTL ?) that is set to be as small as possible: longest path to any receiver. Calculated by root node ?!
    - With common network diameter < 8 ?! This should be good enough ?! (4 bits enough to encode ?)
  - Duplicate elimination by packet-ID ? Probably takes too much memory…
- What to do ?
  - Reset bits when using explicit bitstring, not-reset only when using bloom-filter bitstring ?!
  - Figure out we can live without resets for all modes ?

BIER consideration (not reviewed)

- With bitstring length N, we need N * N bits for FBM – reset bits
    - With bloom filters, we can not have FBM (reset bits). But we try to resolve that issue
    - Do we want to forego FBM for non-bloom filter ?
    - Less problems to solve than with bloom filter: No false positives! Just more duplicates, routing microloop duplicates.
- Simple option ?
    - Make FBM a “SHOULD” – low-end devices could optimiz them away.
- Quantify benefit of explicit BIER bitstring (not BIER-TE)
    - Unicast: Do we save anything over existing storing mode with 6LoRH ? (header already well compressed)
        - Maybe there is NO reason to use BIER bitstring for unicast (only BIER-TE)
    - Multicast: assume existing storing mode network (aka: overhead of unicast routes acceptable)
        - BIER avoids to introduce another multicast routing protocol (PIM, RPL-multicast state,…)
        - More efficient use of bits (no bits used for adjacencies, just 6LR

BIER consideration (2, not reviewed)

- Option position summary ? All need to be validated/quantified.
- Non-storing mode networks
  - Use BIER-TE == non-storing. Unicast+multicast.
  - Can further minimize state avoiding FBM memory
  - Smaller networks use explicit bitstring. Larger ones use bloom filter.
- Networks with storing mode routers
  - Use BIER, maybe just multicast. No benefit? to use for unicast.
  - Smaller networks use explicit bitstrings, larger ones use bloom filter.

Bitstring / BIER considerations

- Explicit BIER-TE bitstring allows to save memory on 6TR/6LR:
  - Do not need RPL routing entries for all 6TR/6LR.
- With BIER, we do need RPL routing table for all 6TR/6LR
  - What do we save (why BIER) ? Unclear / need to better caracterize.
  - Comparison complex ? Because we also have to take savings from 6LoRH into account.
- Aka: Do we need to consider Explicit Bitstring + BIER ?
  - Forwarding plane can be the same BIER/BIER-TE. Eliminating BIER just reduces control plane work to consider (RPL -> create BIER forwarding table entries).
  - BIER/BIER-TE forwarding more similar than outside of ROLL:

Open

- With BA (bier aware application) we could send non-IPv6 payload. Any benefit in that ?
  - No ? 6LoRH would compress IP header away so there is no benefit eliminating IP ??
- Unicast between 6LR/6LN ? Unicast 6LR/6LN towards 6RR ?
  - How do we deal with that ?
- IEF Multicast likes SSM.
  - SSM could help to address 6SN (which server needs to know about IGMP memberships). Part of

Refuse

Refuse - Does not work

- Does not work: Multicasting unicast packets
  - Aka: unicast packet replicated via bitstring to more than one destination
  - Could recreate packet with each destinations unicast dest-addr, but:
    - No architecturally clean solution to do so – e.g.: UDP checksum calculation
  - Make destination address a “unicast-group-address” – IPv6 unicast address same on all nodes
    - Avoid problems like UDP checksum
    - But would just reinvent IP multicast service with IP unicast addresses (which we will have).
    - No added value.

Refuse - Below the line (now)

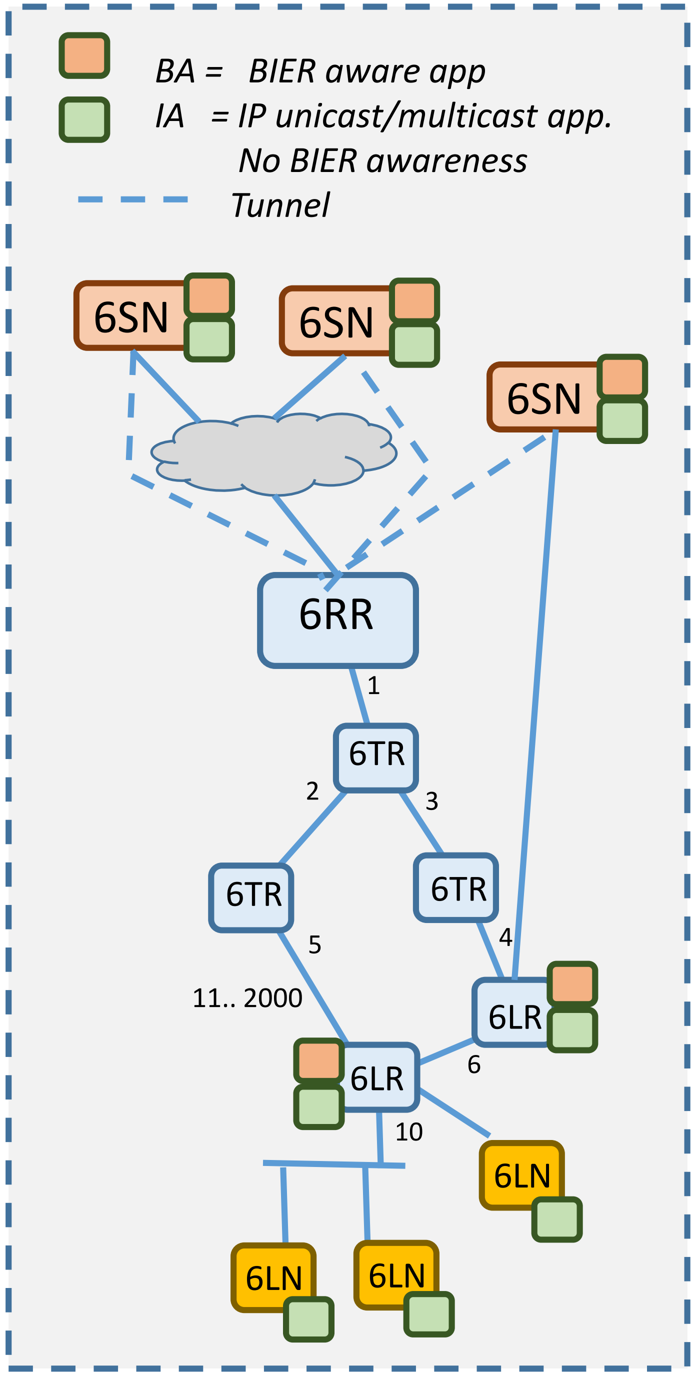
- Using BIER to carry non-IPv6 packets
  - Idea was: BIER packets can directly carry any payload, no IPv6-multicast heade required, eliminate overhead/complexity of IPv6
  - Claim: 6LoRH can compress the IPv6 headers so much that there is not enough initial value in designing header stacks to eliminate IPv6.
  - But: We need to make sure 6LoRH will also equally well compresses the IPv6 multicast header
  - Also eliminates need to figure out how to deal with false-positives for non-IPv6—multicast BIER packets
- Below the line: Transport mode
  - In transport mode, 6LN nodes will have bits assigned to them in the bitstring. This introduces the need for them to be more aware of BIER, e.g.: introduce more BIER awareness into 6LoRH all the way into 6LN and receiver applications.
  - For unicast
    - Will work fine but below line until we have evidence that it could provide better compression than tunnel mode with 6LoRH+BIER compression.
  - Transport mode for multicast
    - Just too much work trying to figure out in first round how to build a lightweight node that ALSO has to be aware of BIER bitstrings.

TO BE DISCUSSED

IP Multicast layer

IP Multicast layer overview

- Primary target for IP Multicast in ROLL-BIER
  - Efficient sending/replication of multicast packets from IP multicast sender (app) on 6SN to IP multicast receiver (app) on 6LN/6LR
  - App on 6SN should be able to explicitly indicate set of receivers. Key benefit not possible with BIER not possible with standard IP multicast.
    - Example from design-team slide: send “ON”/“OFF” message to subset ot light-bulbs that should be switched “ON”/”OFF”
- Not needed: Send IP multicast packets from 6LR/6LN
    - Application client/server mode:
    - Application has one or more server (6SN) sending IP multicast through tunnel into 6RR. Clients (6LR/6LN) can just receive IP multicast.
    - Client->client multicast “emulated” by client sending unicast to Server and then server sends IP multicast
      - Client may receive its own IP multicast packet back, so application need to be able to recognize/filter packets from “itself”
      - Client on 6LR can be excluded from bitstring by 6SN, so this additional filtering only required for 6LN



IP Multicast layer: SSM only

- Majority of IETF Multicast experts prefer SSM IP Multicast over classical ASM IP Multicast
  - Proposal follows that direction
  - SSM: receivers send (S,G) membership instead of (G) membership (MLDv2).
  - S = sender (Server) IP unicast address.
- Benefits
  - If we want to avoid implement directly client->client IP multicast in ROLL-BIER, we need Client/Server model, only servers allowed to send IP multicast, client unicasting to server if they need to be able to send IP multicast (previousslide). This is exactly the model how IP multicast would be done with SSM.
  - SSM avoid need to coordinate IP multicast group addresses across applications (potentially big operational issue).
  - When SSM IP multicast application on a sender wants to create a new SSM stream, it simply needs to allocate an SSM IP multicast group that is unused on this sender (like allocating an TCP port unused for a new unicast service).
  - Client discovery of SSM (S,G) ? Use same scheme a for Unicast server discovery in the absence of IP multicast (e.g.: DNS). As necessary, extend discovery to also discover IP Multicast group (e.g.: Put Server IP unicast address and group-address into DNS. E.g.: group-address in TXT record).
  - Ideal: New IP multicast application on 6SN can allocate new local IPv6 address (independent from IPv6 address used by other IP multicast app running on same 6SN)
    - Benefit: Applications that can have their own SSM IP Multicast sender IPv6 addresses can use static/well-known IPv6 Multicast group application – no need for group discovery by receivers.
    - Best method for local address allocation ? IPv6 privacy addresses ?... ?

IP Multicast layer: MLDv2

- Why do we even need MLDv2 ?
- Not for 6LR: Receiver application on 6LR will receive multicast packet because of bit in BIER bitstring. No IGMP/IP Multicast needed.
- Receiver on 6LN will receive link-layer multicast packet across access-subnet with potentially many 6LN.
  - (S,G) IP multicast header indicates whether packet is of interest to receiver
  - 6LR needs to know whether to send packet to a particular subnet. Some signaling needed to learn this from receivers. No need to reinvent wheel: MLDv2 does this.
  - Receivers need to open link-layer filter (eg: destination) to receive multicast and send packets up to right application. OS-level MLDv2/multicast-socket-API does this. No need to reinvent the wheel.
- BUT!! Above reasoning to reuse/not-modify 6LR->6LN IP multicast (SSM) is ONLY true if IP Multicast packet actually contains IPv6 Multicast header.
  - If we want compressed packet on 6LR->6LN link, then the above rules may not be true: If new OS-level code is needed on 6LR/6LN to support 6LoRH compressed IP multicast packets, then we should re-evaluate what the most pragmatic way is to get a working solution.
  - Unclear what exists today wrt. 6LoRH/IP-multicast/compressed-packets.

IP Multicast layer: proposal (1) - INCOMPLETE

- 6LN:
  - Signaling: SSM subset of RFC3810 (MLDv2), host side
    - Note: This should also be subset of “lightweight IGMPv3/MLDv2” (RFC5790)
  - Only need to be able to receive, not send IP multicast
  - 6LoRH (extension?) to receive compressed IP multicast packets
- 6LR:
  - Signaling: SSM subset of RFC3810 (MLDv2), router side
  - (Modified/Extended) SSM subset of RFC4605 (IGMP/MLD proxy routing)
    - 6LR aggregates SSM receiver state from all subnets: 6LN + internal virtual subnet for IP Multicast receiver application on 6LR itself.
    - 6R sends aggregated membership state to 6SN (join/leave): HOW? (later slide)
    - Any BIER packet with 6LR “bit” set or 6LR comb filter match will be passed to MLD proxy routing forwarding as coming ”from upstream”
    - NO receiving of IP multicast packets from downstream nodes (6LR, 6LN) – just discard
- 6TR:
  - Do not participate in P multicast layer. Just forward BIER. Become 6LR when BIER bitmask/comb-filter entry exists for them. Every 6LR is also always 6TR (forwards BIER independent of processing IP Multicast)
- 6RR:
  - Just like 6TR except that it also must be able to receive tunneled IP Multicast + (pseudo) BIER header from 6SN.

IP Multicast layer: issue

- Simple: let 6SN “just” send SSM IP multicast. No BIER improvements
  - Receipt of IP multicast ONLY determined by MLDv2 membership from receivers (6LN / 6LR).
  - Foregoes application benefits of using BIER (sender determines receivers)
- Problem: Can not get BIER benefit for 6LN without non-heuristic bitstring transport mode”
  - Give BIER bits to 6LN, carry bitstring all the way to 6LN
  - Can not use heuristic likely not useful here: need to be able to eliminate false positives at application level.
- Design for ONLY SSM IP multicast different / simpler as if we want to introduce application layer BIER benefits to application

IP Multicast ONLY model

- 6LR send aggregate MLD membership (join/leave) to RR.
- 6LR keeps (S,G) -> { 6LRi } state.
- Keep table of Bits(6LRi) -> bitstring:
  - bitstring with one bit (non-bloom, BIER)
  - bitstring with > 1 bits (non-bloom, BIER-TE):
    - BIER bit for 6LR plus bits for each hop towards it.
  - bitstring with > 1 bits (bloom BIER/BIER-TE). Bloom compression bitstring result for 6LR (BIER) plus Bloom compression bitstring result for each hop towards it (BIER-TE).
- 6SN just send IP multicast to RR via some tunnel
  - RR may perform RPF check (source address must be 6SN unicast address).
  - Then encaps/forwards BIER/BIER-TE:
    - Loop up (S,G) of packet, Ors Bits(6LRi) for all { 6LRi }. Sends packets
- At minimum same type of hop-bits determination for BIER-TE / non-storing mode as used for unicast non-storing mode today. Just encoded as direct/bloom-filter bits.

IP Multicast + 6SN BIER model

- 6SN should indicate set of destinations instead of “just” MLDv2 join state
- As long as we assume we always use IPv6 multicast packet / MLD, the set of destinations indicated by 6SN must be subset of the “joiners”
  - Simple app example: Each application just has ONE IPv6 multicast group, all application members on 6LR / 6LN join to that group. But 6SN only indicates subset of those application clients for each packet.
  - Does not work for 6LN without transport mode (prior slide).
- Most simple? Hybrid solution (introduce BIER benefit to 6SN with minimum additional work ?)
  - RR forwards (S,G) -> { 6LR } state changes to 6SN. Use of SSN makes this easy: Each 6SN only gets updates for those (Si,G) with its own Si.
  - 6SN send IP Multicast packet with additional BIER pseudo-header
    - Indicating subset of 6LR to which packet should go
  - 6RR then only OR’s bitstrings for this subset as opposed to full { 6LR } it maintains
- INCOMPLETE

Next steps ?!

- Push discussion about BIER interface for 6SN to BIER-WG ?!
  - ROLL should not come up with somethin ROLL specific that does not have to be ROLL specific – better try to find general purpose solution in BIER (ROLL defining requirements)
- SSM IP multicast (ONLY) solution
  - ROLL: (optional) ? compression for 6LoRH extension on access LAN
  - “Tunnel” from 6SN node to 6RR - to send IP Multicast packets from 6SN to 6RR
    - Two cases ?
    - A) 6SN is inside 6LoRH network: simple 6LoRH header extension: one bit to indicate “packet goes to root”, but also encode IP Multicast Group address.
    - B) 6SN is “behind backbone”: Need an actual IP tunnel (path between 6SN/6RR not natively supporting 6LoRH). Just encapsulate same 6LoRH packet inside ?
  - MLDv2 proxy operations on 6LR
    - 6LR -> 6RR covered by ‘draft-ietf-bier-mld-01.txt’, but discovery of 6RR and constraints (only receive, not sending) and 6LR->6LN not covered.