Localized Optimizations over Path Segments

IETF 106, LOOPS side meeting 2019-11-19

Note Well

- You may be recorded
- Be nice, and be professional
- The IPR guidelines of the IETF apply:
  see **http://ietf.org/ipr** for details.

Agenda

- What is LOOPS (25)
- Technical discussion about specific items (3x15)
- Next steps (5)

What is LOOPS?

Localized Optimizations over Path Segments

IETF 106, LOOPS side meeting 2019-11-19
(Slides compiled by Carsten Bormann)

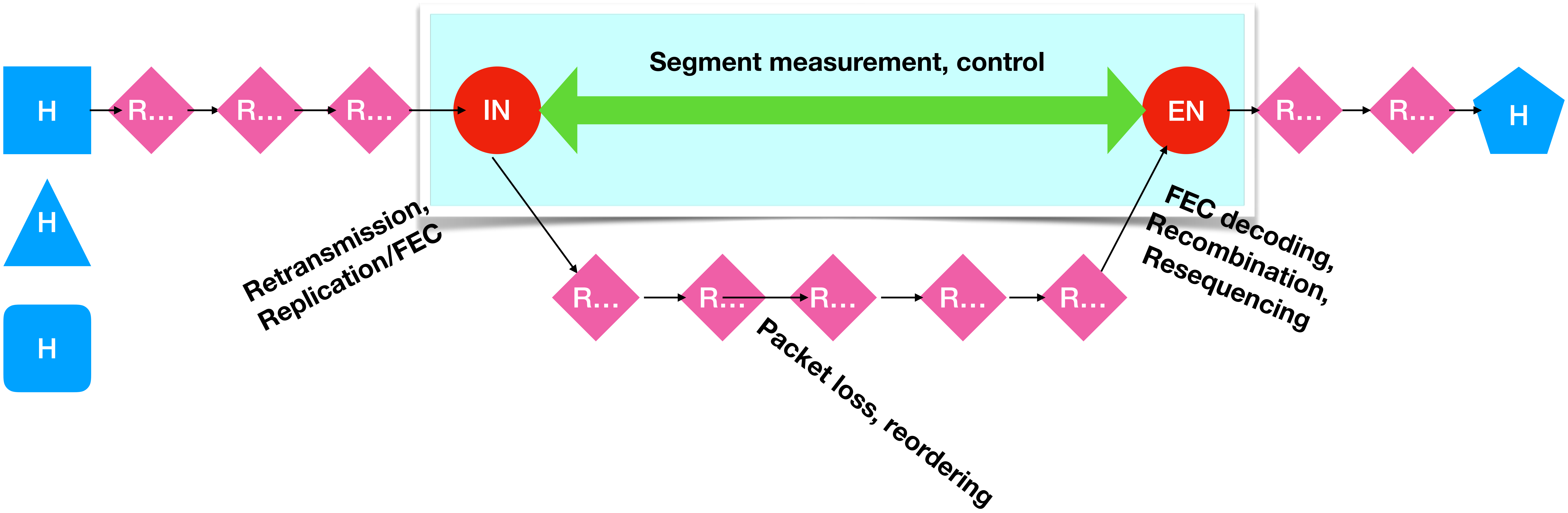
Packet loss

Packet loss

Packet loss

Tail loss!

LOOPS Opportunity



Recover Packets

Locally

Reduce end-to-end packet loss

Recover locally, **where needed**, with low latency

In the **network**

Host participation not required

Don’t look
Don’t touch

Works with any kind of IP packets

How to recover?

Piggyback (Tunnel), separate Packets

Tunnel

- **Retransmission**
  - Reverse information needed: ACK/NACK
  - Forward information: sequence numbering (if needed)
- **Forward Error Correction** (redundancy)
  - Can use dynamic selection of block size/rate: measurement input
  - "Retransmission" also possible by adding FEC

- Aim for low setup overhead
- Keep most setup out of protocol ("controller model")

How not to blow up the Internet

- Concealing losses removes important congestion signal
  - End-hosts would ramp up to higher rates, increase congestion

- Need **congestion feedback**
  - Preferred: ECN
  - Fallback: Selective dropping (selective recovery, actually)
- Host transport protocol improvements will help improve LOOPS performance, but are not prerequisite to obtaining benefit

Elements of LOOPS

- Information model for local **recovery**: in-network retransmission/FEC
  - Can be encapsulated in a variety of formats; define some of those
- Local **measurement**: e.g. segment forward delay/variation
  - To set recovery parameters
  - To determine if loss was caused by congestion
- Congestion **feedback**:
  ECN (or drops) to inform end hosts about congestion loss

Freezer (not currently in scope)

- Multipath
- Measurement along string of LOOPS pairs (“almost e2e”)
- MTU handling, fragmentation, aggregation, header compression
- Selection of one or more specific tunnel encapsulation or measurement formats (beyond “sketches” showing it can be made to work)

LOOPS vs. transport protocols

- LOOPS is separate from the end-to-end transport protocol
  - Hands-off approach: don't meddle
  - Do not assume the end-to-end protocol is out to help us, either
  - No direct control over sending rate (cc feedback only)
- LOOPS should not just be a classical transport protocol
  - Residual loss is OK
  - More choices: Tight interaction with the path segment being optimized

Where “transport protocol” intuition may not even work

- Relatively controlled/managed environment;
  setup mechanism assumed (can supply parameters so not everything needs to be high dynamic range)
- No full reliability intended; remaining gaps are OK (and at some point must leave the focus of attention)
  - Setup might set upper bound for overhead volume (e.g., 10 %), can well be “risky” in the way that this is used
- Tunnels usually have packets in flight (possibly a large number); tail processing rarely invoked (but may still be desired); don’t need overly conservative RTO

Documents out there

- Use cases and problem statement: “LOOPS (Localized Optimizations on Path Segments) Problem Statement and Opportunities for Network-Assisted Performance Enhancement” <draft-li-tsvwg-loops-problem-opportunities>
- Protocol: “LOOPS Generic Information Set” <draft-welzl-loops-gen-info>
- One of the Encapsulations: “Embedding LOOPS in Geneve” <draft-bormann-loops-geneve-binding-00.txt>

- Charter proposal for a LOOPS WG <https://github.com/loops-wg/charter>
- LOOPS mailing list loops@ietf.org

Related work (see IETF105 BOF)

- Encapsulations: Many (e.g., NVO3 for Geneve; GUE; GRE?)
- FEC: NWCRG for e.g., sliding window FEC, encapsulation techniques
- Tunnel congestion Feedback (TSVWG)
- Also: measurement work, IOAM;
  knowledge about behavior of transport protocols (TCP, QUIC)
  adaptation layer retransmission work (6Lo Fragment Recovery)

Sliding Window FEC

- Sliding windows fit quite well to LOOPS application
  (Can also use traditional block formats)

- Various drafts for FEC scheme and specific embeddings in NWCRG and TSVWG, e.g.,

  - "Sliding Window Random Linear Code (RLC) Forward Erasure Correction (FEC) Schemes for FECFRAME" <draft-ietf-tsvwg-rlc-fec-scheme-16.txt>

  - "Forward Error Correction (FEC) Framework Extension to Sliding Window Codes" <draft-ietf-tsvwg-fecframe-ext-08.txt>

(F)AQ (1)

- So this is only about encrypted traffic?
    - Any traffic is welcome, we just don’t try to peek beyond L3 info
- So how do you know which packets are worth recovering?
    - Today we don’t. If more L3 marking becomes available, we’d use it.
- How do you transport your measurement-related information?
    - Forward info: In encapsulation extension (e.g., with sequence number). Reverse info: The same way we transport the ACK channel. Depends on encapsulation.

(F)AQ (2)

- How do you avoid spending more for LOOPS encapsulation than the performance enhancement is worth?
  - LOOPS will need some management that is weighing this (and doing the pair setup in the first place)
  - Can dynamically switch off and on (e.g., based on monitoring)
- How to relay congestion for non-ECN-capable transports?
  - Dropping. Or, actually, not even requesting a retransmission when a congestion event would be relayed anyway.

Slides based on

LOOPS Generic Information Set

draft-welzl-loops-gen-info-00
LOOPS BoF
IETF 105 - Montreal

Michael Welzl, U. Oslo
Carsten Bormann, ed., U. Bremen TZI

Why look at this draft now?

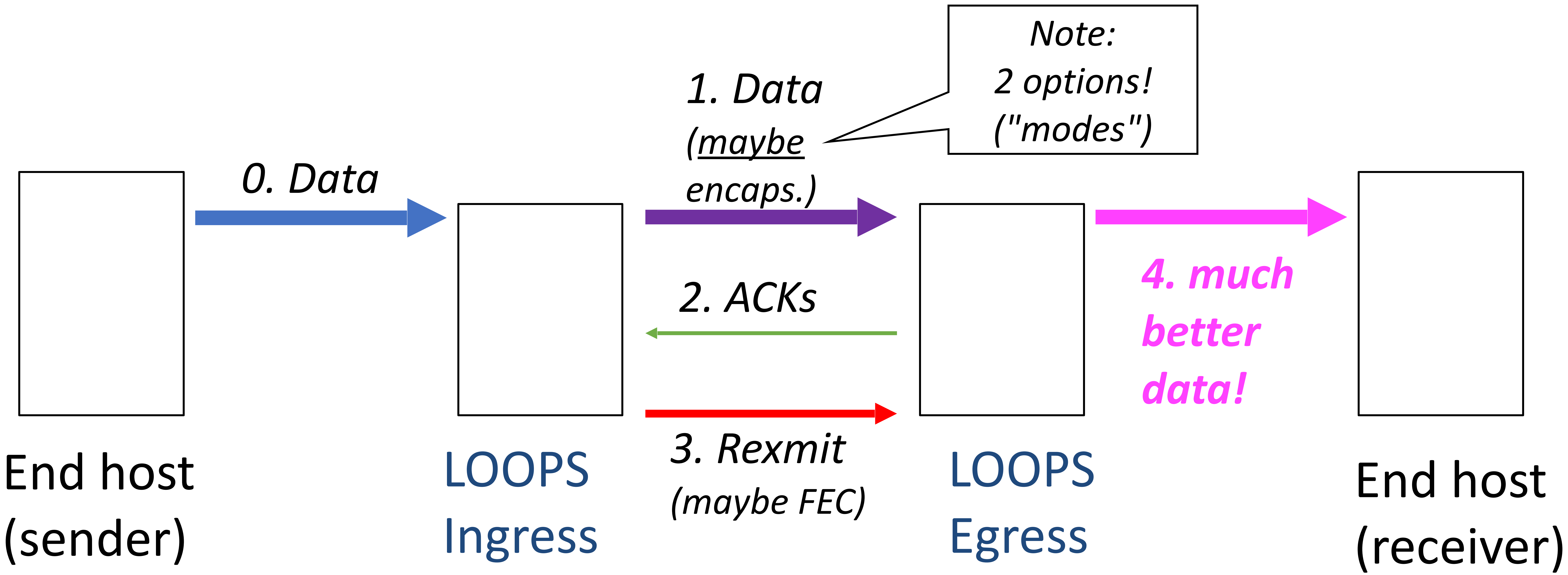
- "The present document is a strawman for the set of information that would be interchanged in a LOOPS protocol, without already defining a specific data packet format."

→ an overview of how a LOOPS protocol could work

… as an existence proof, and to aid visualization

~~We are not picking alternatives today.~~ We are starting to collect reasons for decisions, to prepare the WG work.

Context



Note:
2 options!
("modes")
0. Data
1. Data
(maybe
encaps.)
2. ACKs
3. Rexmit
(maybe FEC)
4. much
better
data!
End host
(sender)
LOOPS
Ingress
LOOPS
Egress
End host
(receiver)

Problems to address

- From previous slide:
  - Loss detection/retransmission
  - FEC control

- Also: detect congestion on ingress-egress segment
  - Measurement / congestion detection
  - Congestion signaling to end hosts if congestion was detected

- Next: some concrete ideas on how to deal with these problems
  - Just a strawman (two, actually)!

Tunnel mode

1. Ingress forwards

- Encapsulate: not tied to any specific overlay protocol
  - Document contains sketches of bindings to GUE and Geneve
- We don't try to understand data after the IP header
  - Hence, need to add our own Packet Sequence Number (PSN)
- Some more information added
  - Tunnel type
  - "ACK desirable" flag (asks for feedback block 1, next slide)
  - Anything needed by FEC

2. Egress answers

- Block 1 (optional, only upon "ACK desirable" — timely feedback)
  - PSN being acknowledged
  - Absolute time of reception for the packet acked (PSN)

- Block 2 (optional — eventual feedback)
  - an ACK bitmap (based on PSN)
  - a delta indicating the end PSN of the bitmap

- Can be interspersed and repeated
- Can be piggybacked on a reverse direction data packet or sent separately
- Usually aggregated in some useful form

3. Ingress retransmits

- Detects need for rexmit via ~~NACK or RTO~~ timing
  - Make decision based on congestion

  → Use ECN if possible

  → Calculate latency variation from timestamps in feedback blocks 1

- ... Or, rather than "just rexmit", send FEC repair packets

4. Egress forwards

- De-FEC

- Inform end hosts about congestion if needed
  - Might be able to distinguish "real" congestion from, e.g., corruption loss
  - ECN much preferred as an outgoing signal!

Summary: information exchanged

- Forward: encapsulation, containing...
  - Packet Sequence Number (PSN)
  - Tunnel type
  - "ACK desirable" flag (asks for feedback block 1, next slide)
  - Anything needed by FEC

- Backward: optional blocks type 1 (timely) and 2 (eventual)...
  (can be piggybacked, aggregated, interspersed, repeated, ...)
  - Block 1 (optional, only upon "ACK desirable” — timely)
    - PSN being acknowledged
    - Absolute time of reception for the packet acked (PSN)
  - Block 2 (optional — eventual)
    - an ACK bitmap (based on PSN)
    - a delta indicating the end PSN of the bitmap

Transparent mode

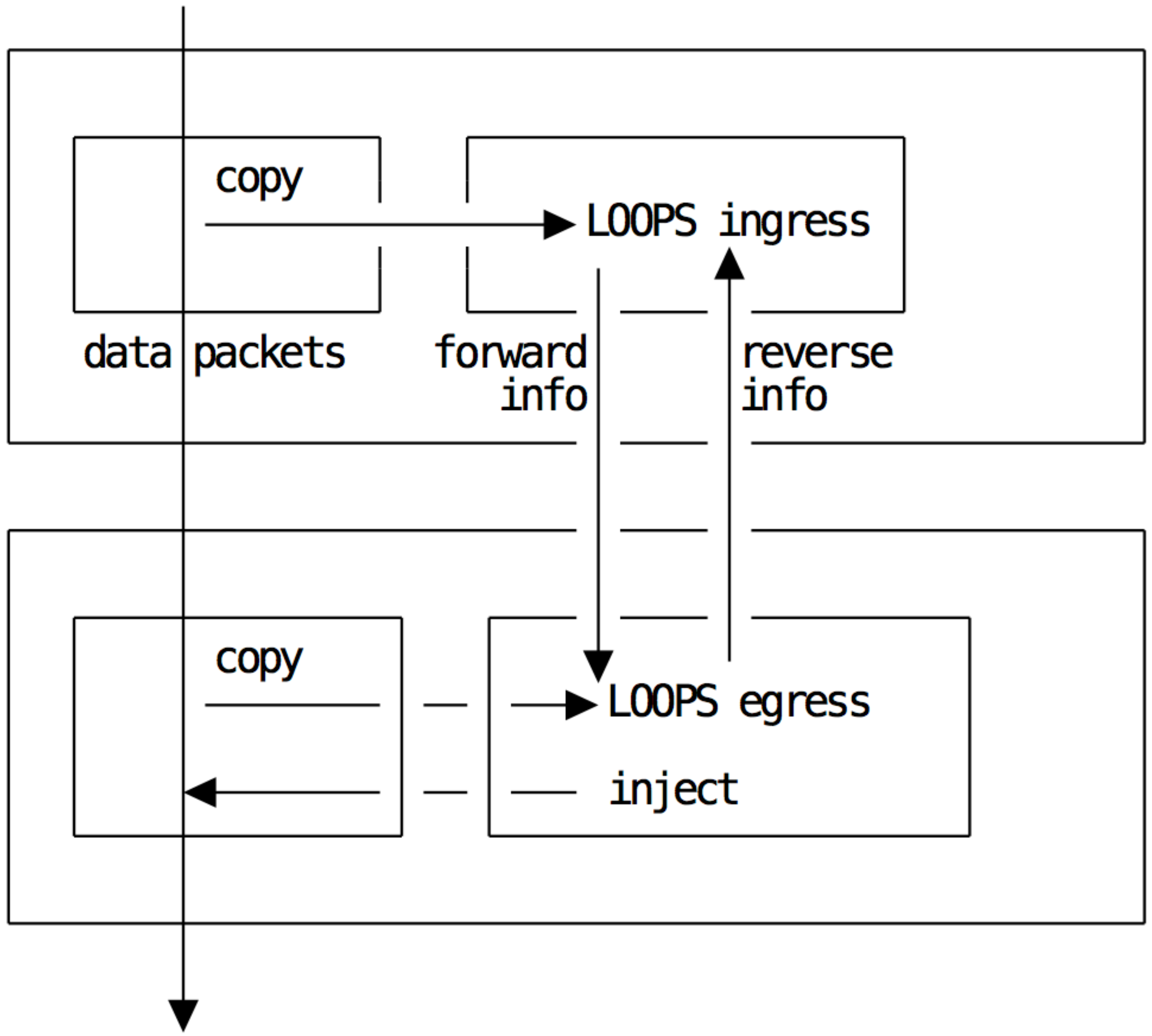
A bit more radical... describing least intrusive method here:

"Never delay and don't even tunnel"

Just discussing rexmit; FEC could also be done

1. Ingress forwards

- Just forward
- Also, keep a copy of packets, with a hash for identification
  - From immutable header fields
  - May need to include data beyond IP



2. Egress answers

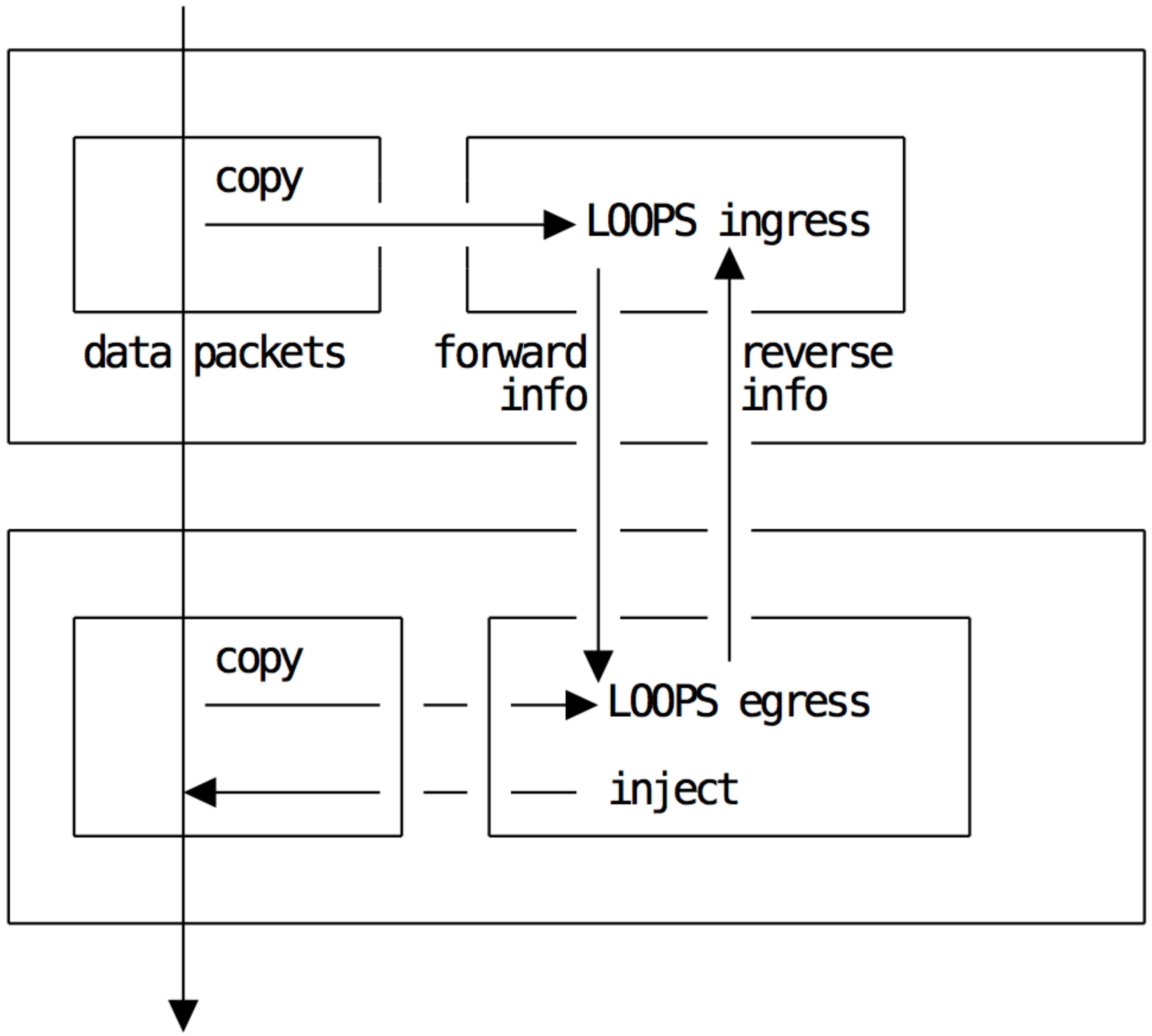
- ACK everything; no NACK possible
  - Same hash calculation as ingress
- ACK format similar to tunnel mod

3. Ingress retransmits

- Detects need for rexmit via timing only
  - Decide based on congestion estimation as before
- Cost of hash collisions is low: misses retransmit opportunity.

4. Egress forwards

- That's all it does. There will be re-ordering.



Summary: information exchanged

- Forward: nothing extra
- Backward: roughly as before - optional blocks type 1...
  (can be piggybacked, aggregated, interspersed, repeated, ...)
  - Block 1
    (*limited in some way: was optional, only upon "ACK desirable" for tunnel mode, but egress doesn't get this information in transparent mode*)
    - PSN being acknowledged
    - Absolute time of reception for the packet acked (hash)
  - Block 2 (hmm)
    - an ACK Bloom filter?

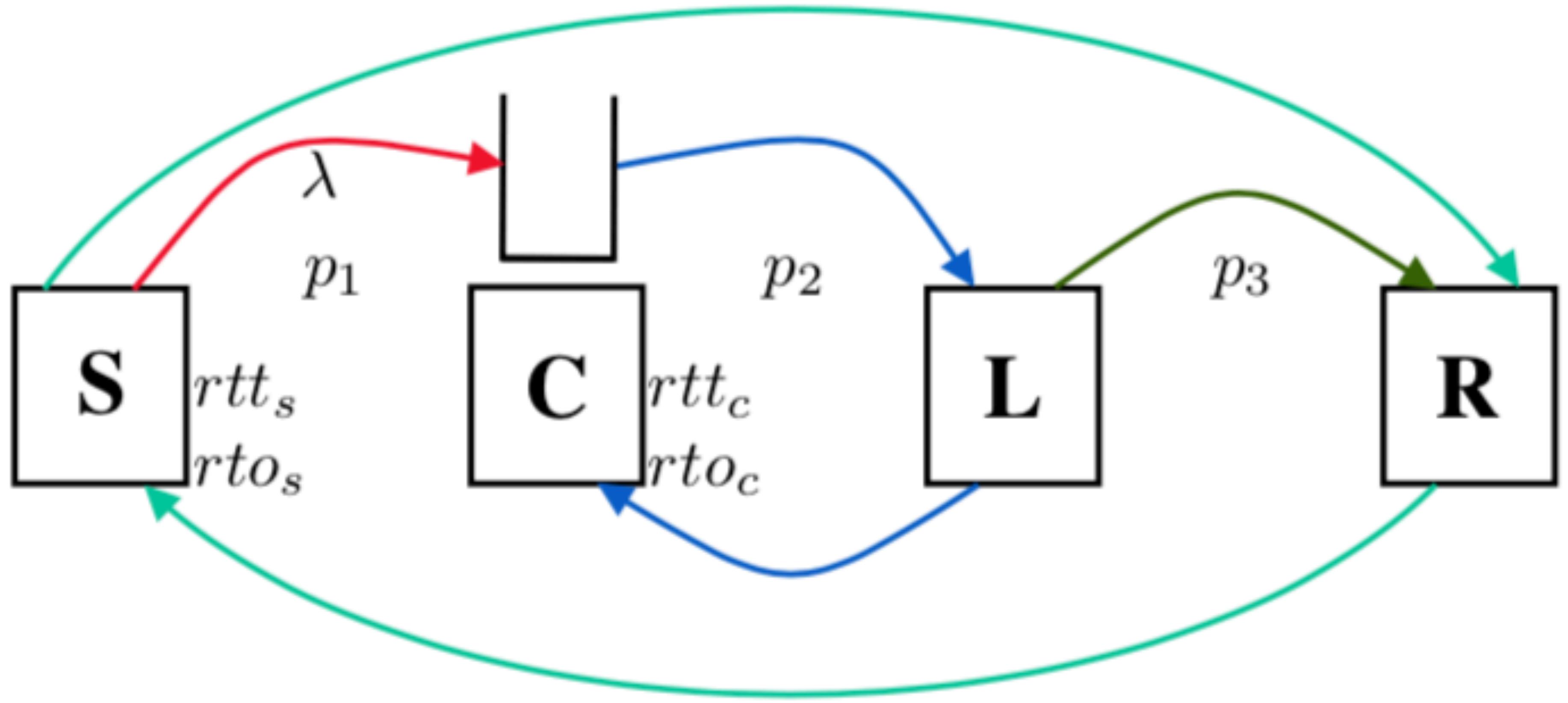
Conclusion

- Spectrum of possibilities, from "full-fledged" to min-intrusive
  - Various trade-offs between these options
- In all cases: LOOPS can be very beneficial when the LOOPS segment RTT is much shorter than the e2e RTT
  - Wireless NICs use this fact
- Some packet drops really cause pain
  - LOOPS can help

Tail loss!

Recent work:
Markov chain analysis of a LOOPS ingress cache

- System constraints: "transparent mode"
  - sender S emits packets with rate λ
  - cache C additionally stores a copy of packets
  - loss detector L confirms reception of each packet
  - C retransmits if timer rto_c expires
  - R acknowledges packets to S
  - p_i: packet loss probability in segment i



- Continuous-Time Markov Chain (CTMC) with finite states
  - Cache can be modelled as an M/D/1/N queuing model
  - Poisson arrival, deterministic service time

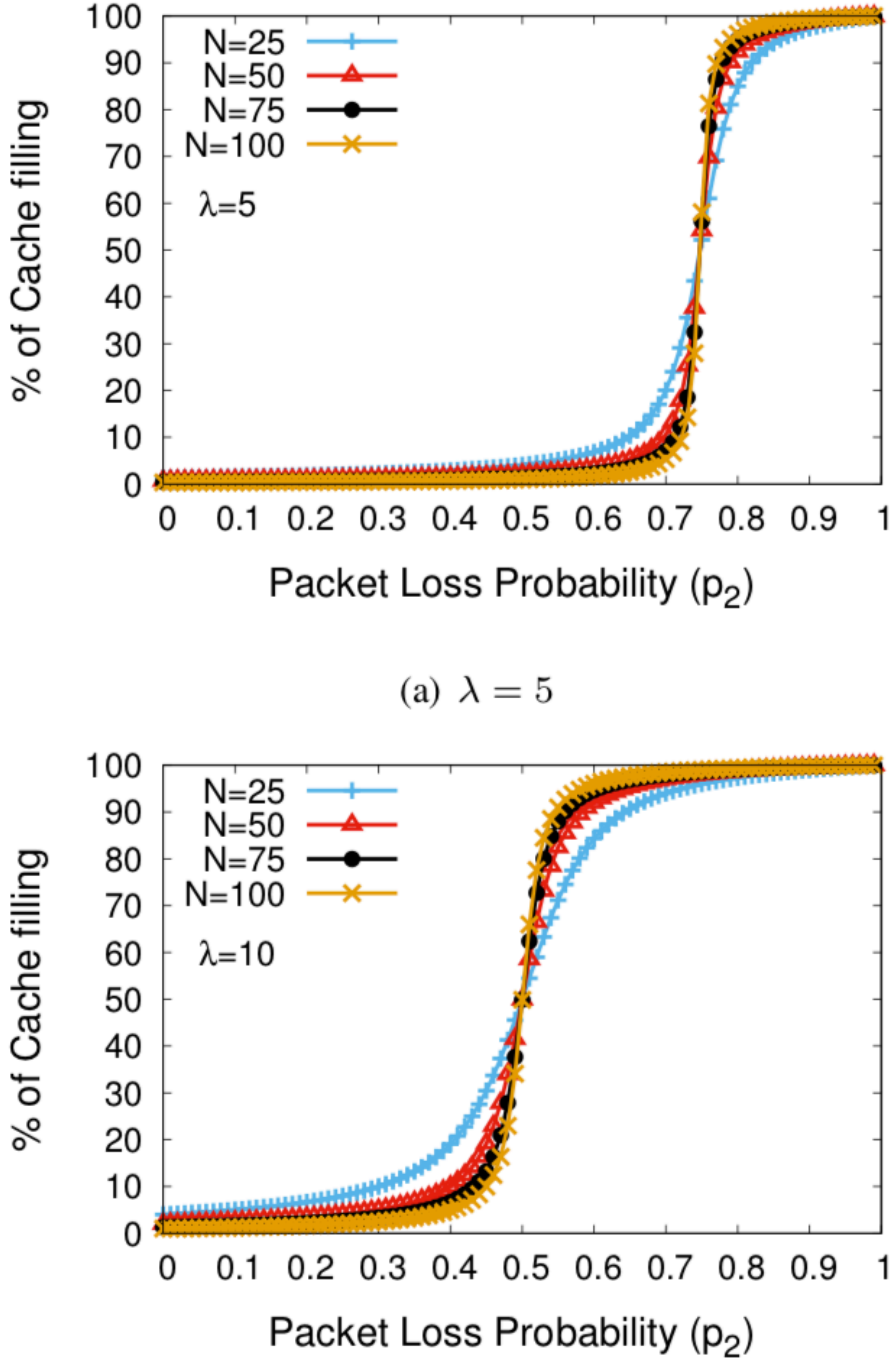
- *Runa Barik, Michael Welzl, Peyman Teymoori, Safiqul Islam, Stein Gjessing: "Performance Evaluation of In-network Packet Retransmissions using Markov Chains", accepted for publication, CNC'20 workshop, proceedings of IEEE ICNC 2020, HI USA 2020.*

Results

- Major influencing factors: p_2, rtt_c/rto_s
- Non-linear dependence between cache filling level and packet loss probability p_2

$p_1 = 0.01$, $p_3 = 0.01$, $rtt_s = 0.1$, $rto_s = 0.2$, $rtt_c = 0.05$, $rto_c = 0.1$, initial $p_b = 0.01$

- Cache size has the least impact on its utilization: irrespective of the size, it can be fully utilized by higher packet loss probabilities
- LOOPS reduces retransmission rate and expected caching time at S



(a) $\lambda = 5$

(b) $\lambda = 10$

Discussion items:

- Acks and retransmission:
  How does acknowledgement information + measurements flow to enable retransmission decisions (whether, when, how often)
- Encapsulation:
  Using encapsulation level formatting vs. our own (e.g. TLV, flag-based)
  Discuss Geneve as a starting point
- FEC framework:
  What is the benefit of new-fangled (e.g., RLC/sliding window) over traditional FEC (e.g., RaptorQ)

Acks and retransmission

- LOOPS defines a protocol: What information do the two ends make available to each other when
- Retransmission decision can be local matter, if protocol supplies needed information
  - Traditional DUPACK-style: Any (small number of) packets that are ahead of the current order indicate a loss
  - Time-based: tolerate reordering; use more information than just

Acks and retransmission

- Design so far:
  - PSN numbers payload packets (not LOOPS packets)
  - Block 1 ACKs (timely information): Latency & RTT measurement
    - **Ingress** decides whether this should be returned (ACK desirable bit)
    - Never requested for retransmissions (avoid ACK ambiguity)
  - Block 2 ACKs (aggregate information): ACK bitmaps
    - Basis for retransmission decisions
    - **Egress** decides when to acknowledge and how much, how often

Acks and retransmission: What to achieve?

- Local retransmission most useful if it preempts end-to-end retransmission
  ➔ retransmission should happen quickly
  - But not so quickly that it creates spurious local retransmissions
  - Assumption: deadline for retransmission a couple RTTs after initial
  - One implementation strategy: Single retransmission only
- Potential goal: Robustness to some packet reordering on the path segment
- Class 1:
  - in average, significantly more than one packet is in flight
  - In average, loss recovery << 10 %

Acks and retransmission: Detect holes

- with reordering, hole-based retransmission schemes (e.g., Dupack) will cause some spurious retransmissions
  - end-to-end: too bad.
    - CC effect (reduction of CC window) will be canceled out quickly by spurious retransmission detection.
    - end-to-end delivery still faster with local recovery.
    - RACK etc. will ease this consideration over time.
  - LOOPS: Maybe don’t do that, then?
- How does a time-based retransmission scheme look like for LOOPS?

Time-based recovery (Strawman): RACK-inspired equivalent for LOOPS?

- Divide time into slots, e.g., milliseconds (or ~ an order of magnitude under min_RTT; possibly defined in setup protocol)
- For each packet, remember slot when last sent
  - and whether already retransmitted (no longer used for measurement)
- Use aggregate ACKs to mark some slots as “mostly arrived”
- Subtract a reordering window, based on how hard that boundary is
- Pace out older unacknowledged packets at retransmission pace
- Flush slot when all acknowledged or beyond retransmission deadline

Encapsulation

- LOOPS focuses on the “generic information” interchanged
- Need to map to specific encapsulation protocols.
  These have:
  - Required fields (that may map to LOOPS information)
  - Existing options (that may map to LOOPS information)
  - Extension points (TLVs, flag-based)

Encapsulation

- LOOPS information may go into
  - Required fields, existing options (possibly after some mapping)
  - Extension points (via one-to-one mapping)
  - Blobs that go into an extension point (e.g., Geneve “LOOPS option”)
    - Need internal structure for representing those data:
      - LOOPS-specific approach, vs.
      - Mirror some of the encapsulation protocol’s approaches

Encapsulation: Geneve Strawman

- draft-bormann-loops-geneve-binding-00.txt
- Goes for single LOOPS option:
  - Geneve:
    4-byte aligned, increments of 4 bytes
  - Flags for PSN/ACK, timestamp blob/echo, reception time, …
    (and a few flag-only flags, e.g., ACK desirable)
  - Separate flag for (single) non-4-byte element: Block2
- Unfortunately, the items themselves then are not 4-byte aligned

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Flags              |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
|                                                               |
~                        Flag Based Data                        ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Geneve Overview

- Tunnel protocol primarily used in DC to support virtualization
- UDP based: X over UDP over IP
- Data plane extensibility: by TLVs
- Pure data plane spec. Can leverage many control planes.
- Two categories
  - Tunnel endpoint
    - Originate/terminate packets
    - Can insert/delete/parse options
    - **LOOPS should work on these points**
  - Transit device
    - Can interpret options
    - Must not insert/delete/change options
    - Must not modify Geneve headers

Geneve Format

Outer IP hdr	Out UDP hdr (dst port=6081)

Length of options, in 4B multiples, max 63 = 252B options

Control pkt. Consumed by tunnel endpoints

Critical option present. Tunnel endpoints must parse. If not support, drop pkt. If not set, may strip options and forward

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver|  Opt Len  |O|C|    Rsvd.  |          Protocol Type        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Virtual Network Identifier (VNI)       |    Reserved   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Variable Length Options                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Protocol data appearing after Geneve hdr [ETYPES]

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Option Class         |      Type     |R|R|R| Length  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Variable Option Data                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Namespace for the 'Type' field.

Length of the option, in 4B multiples excluding option header (4B)

```
0 1 2 3 4 5 6 7 8
+-+-+-+-+-+-+-+-+
|C|    Type     |
+-+-+-+-+-+-+-+-+
```

C: Critical option.

Proposed LOOPS option - map LOOPS to tunnel protocol

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|C|   LType     |
+-+-+-+-+-+-+-+-+
```

LType: LOOPS Mode.
0 – retransmission mode
64 – FEC mode

LOOPS namespace

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Option Class         |      Type     |R|R|R| Length  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Variable Option Data                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Flags             |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
|                                                               |
~                      Flag Based Data                          ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Variable Option Data Format in Geneve LOOPS Option

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|I|R|D|S|T|E|A|R|             |B|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

See next page

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|I|R|D|S|T|E|A|R|             |B|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Flags in LOOPS option

Used on their own

- I: Initial Packet Sequence Number (PSN) flag; may be set by the LOOPS ingress to notify the egress about using a new initial PSN.
- R: Initial PSN Received flag; echo of I flag provided by the LOOPS egress.
- D: ACK Desired flag; set by the LOOPS ingress if it wants the egress to generate an acknowledgement immediately upon receiving a particular packet.

Used with additional 32-bit in “flag based data” field

- S: PSN flag; indicates a PSN data block is carried. It must be set when a packet payload is present. It must not be set if the packet is a pure LOOPS ACK packet, i.e. when no payload is included in the packet.
- T: Timestamp flag; indicates a Timestamp data block is carried
- E: Echoed Timestamp flag.
- A: ACK number flag; indicates presence of a BLOCK 1 ACK info
- R: Reception time flag: May only be set if A is set. Indicates that an absolute reception time is carried.

Used with additional variable-length block in “flag based data” field (least bit)

- B: Block 2 ACK info flag; indicates presence of a BLOCK 2 ACK info

Design choices and open questions

1. Single option or Multiple options
   - Single: Efficient, compact
2. Pure ACK
   - Set O bit in Geneve header
   - A control packet without payload
3. VNI:
   - mandatory in Geneve
   - A specific VNI can be used for LOOPS enabled traffic, or for a particular tenant's LOOPS enabled traffic, or
4. Alignment: Geneve requires 4 byte alignment. PSN length?
5. Move some flags to "Type" field of generic option header?
6. Length of Timestamp & Echoed Timestamp?

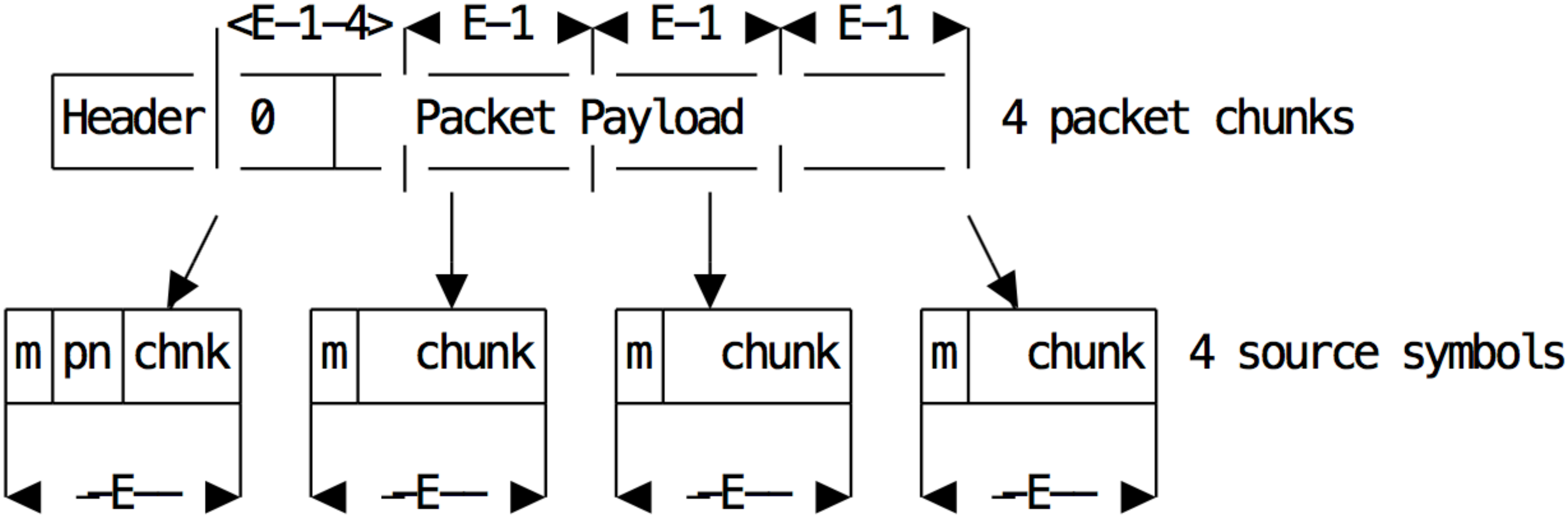
FEC approach

- Support multiple classes of FEC schemes, e.g.:
  - Very simple parity (as in SMPTE 2022)
  - Fountain Codes (e.g., RaptorQ)
  - Sliding Window schemes (e.g., RLC)
- Assume all codes are systematic (needed for transparent mode)
  - Except for transparent mode, augment payload packets by FEC indices
- Possibly add special handling for larger-than-tunnel-MTU packets
- Add repair packets with repair information

FEC approach

- LOOPS can provide:
  - forward: place for FEC indices, separate format for repair packets
  - return: Block 2 acknowledgements, or aggregate loss rate feedback
- Assumption: large size variance of payload packets (avg 400..700 B)
  - Payload packets are divided up before being funneled into FEC
  - Not necessarily related to the way they are sent forward
  - Any piggybacking for repair segments?
    Recombining/splitting of payload packets (also for MTU reasons)?

From draft-roca-nwcrg-rlc-fec-scheme-for-quic-02:



FEC: Design choices

- Classes of FEC schemes (that can be handled equivalently by LOOPS)
  - What are the FEC indices to be added to payload packets?
    (Tunnel: right there; Transparent: separately)
- Do we put in some MTU mitigation (breaking up payload packets)?
  Piggy-backing runts/short packets/repair symbols?
- Feedback:
  - For controlling FEC rate — what is the time scale?
  - For filling in repair packets?
- Details of the construction of FEC input and repair packets;
  how are reconstructed packets put together again?

Next Steps

While we are not a WG…

- Continue on, *working* like a WG
  - Explore design space, maybe holding back on tough decisions for now
- Continue improving the set of documents, possibly adding FEC document
  - Identify authors and reviewers
- Employ github.com/loops-wg and loops@ietf.org for coordination
- Review charter proposal at github; react to AD input on this
- Aim for being a WG at IETF 107 (Vancouver, March 2020)