RBS (Recursive BitString Structure)
for improved scaling beyond BIER/BIER-TE

BIER-WG, IETF115 London, v1.0

draft-eckert-bier-rbs-00
replaces draft-eckert-bier-cgm2-rbs-00

Toerless Eckert, Futurewei USA (tte@cs.fau.de) (Editor)

Michael Menth, (menth@uni-tuebingen.de),
Xuesong Geng (gengxuesong@huawei.com), Xiuli Zheng (zhengxiuli@huawei.com),
Rui Meng (mengrui@huawei.com), Fengkai Li (lifengkai@huawei.com)

Why ?

“Traffic engineering” very important for applications at scale (e.g.:DCN)

- DetNet application (RFC8578) – bounded latency, jitter, disjoint path/trees (resilience/PREOF) (BIER-TE_
- Capacity optimization (non-ECMP load distribution), lowest-cost tree (steiner) (BIER-TE)
- Per-packet selection of destinations (adaptive streaming etc.) (BIER/BIER-TE)

Current solution (BIER / BIER-TE) scaling issues ?

- Because of “flat bitstrings”
- Flat bitstring was chosen ca. 10 years ago as best solution for BIER, BIER-TE just reused it

  We think hardware can do better - now/in future.

Marketing pitch

(A)We can encode trees more efficient than BIER-TE, maybe even better than BIER destinations

(B)We can make larger headers more scalable to process

(C)We can make this easier to auto-configure than BIER-TE (and BIER).

Likely more experimental ?! Than BIER 10 years ago ? (pending HW validation)

How it works RBS address structure

Compressed Tree “Address” at A

RU-Offset: **Offset for A**	RU-Length: **Length for A**

1. Describes whole tree!!
2. Rtr A examines its BitString (BS A)
   determines the copies to adjacent Rtr to make
3. Sees two bits are set. Creates two packet copies
4. For each copy, RU-Offset/RU-Length adjusted

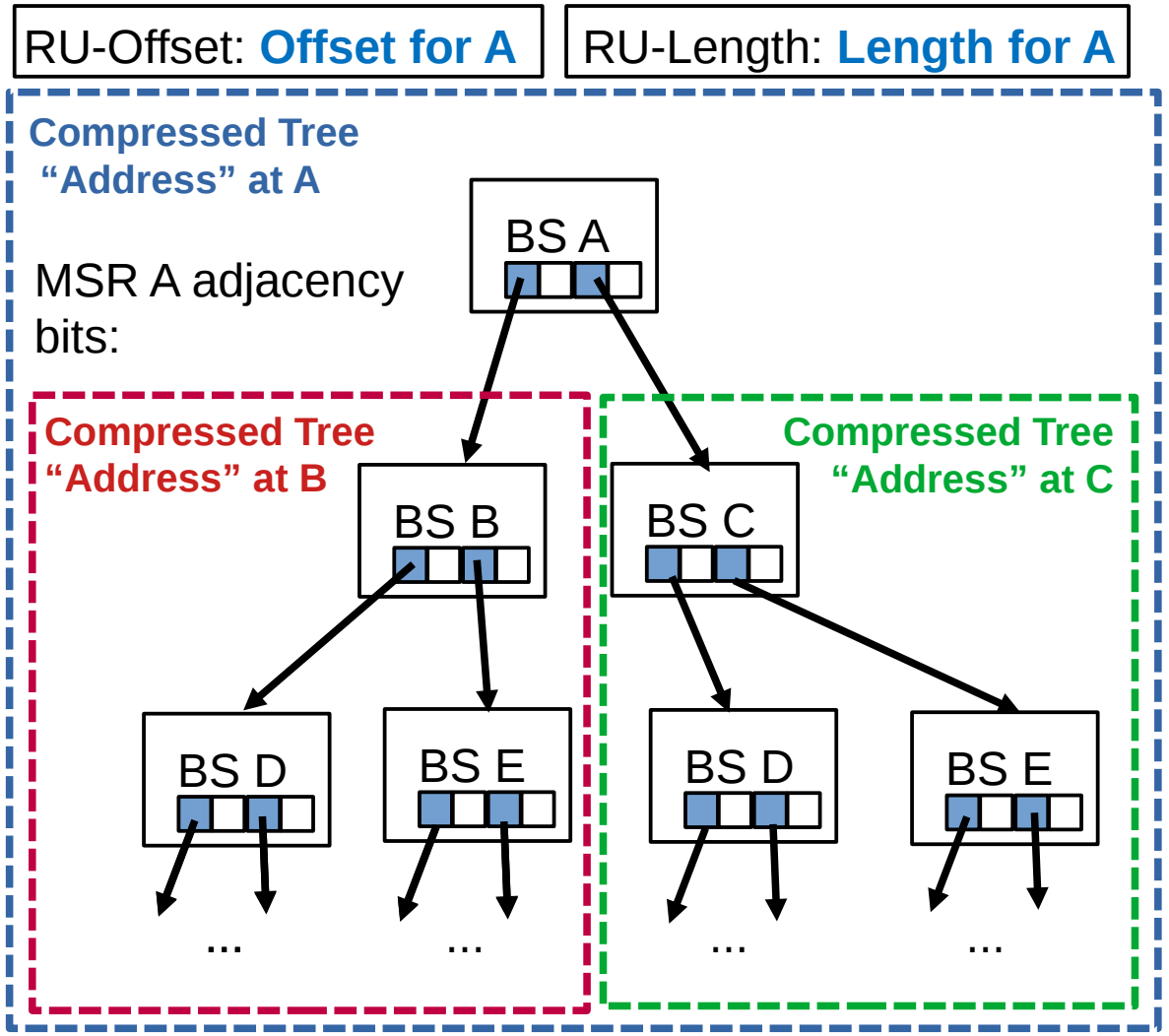
Compressed Tree “Address” at B

RU-Offset: **Offset for B**	RU-Length: **Length for B**

Compressed Tree “Address” at C

RU-Offset: **Offset for C**	RU-Length: **Length for B**

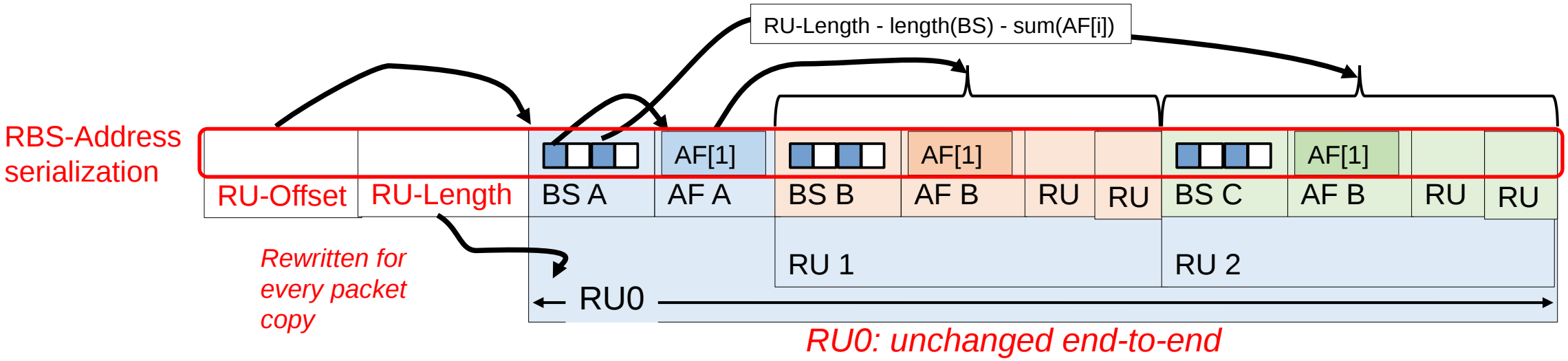
Each routers structure (A, B, C) is called
a Recursive Unit (RU). Tree Root structure is called RU0 (A)



Serialization with hop by hop processing

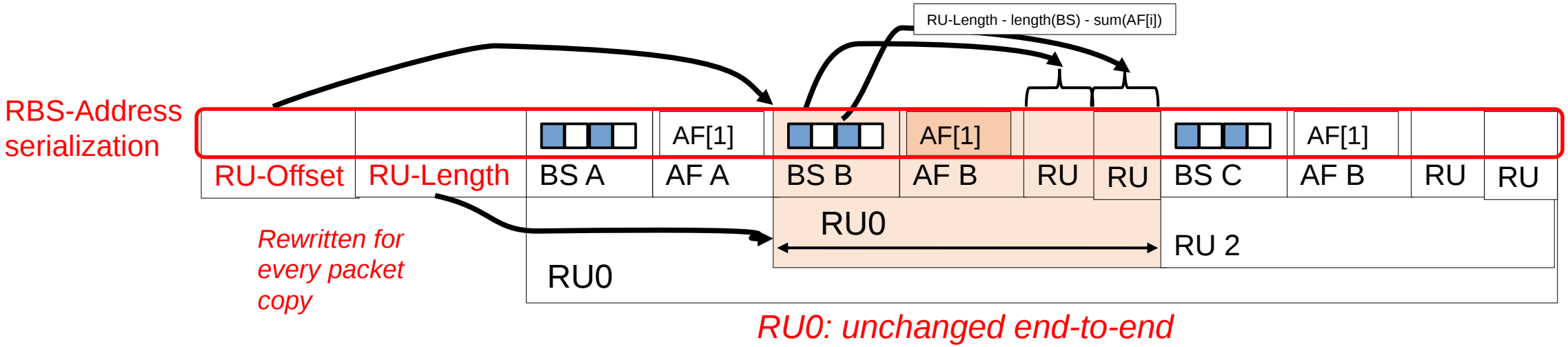
Not showing bits in bitstring without R)ecursive bit and hence no AF/RU associated with the

RBS-address for packet at **Router A**



RU0: unchanged end-to-end

RBS-address for packet at **Router B**



RU0: unchanged end-to-end

Serialization into RFC8296 packet header and cost

Serialized RBS Address: (RU-Length, RU-Offset, RU0)

With RFC8296: use Bitstring field

0-pad to next 2^N size (permitted sizes in RFC8296)

Select RBS vs. BIER, BIER-TE: By SubDomain (as BIER-TE)

Do not need SI in bift-id field – optimize use of field ?

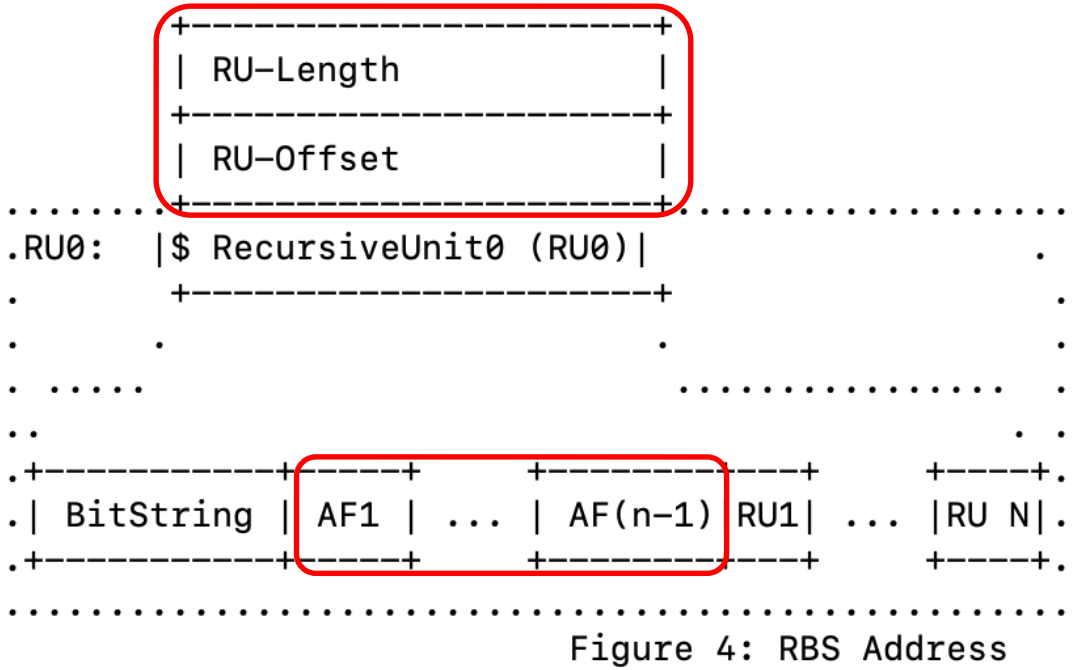


Figure 4: RBS Address

Cost of RBS serialization

Aka: bits who are not “local bitstrings of each rouer”

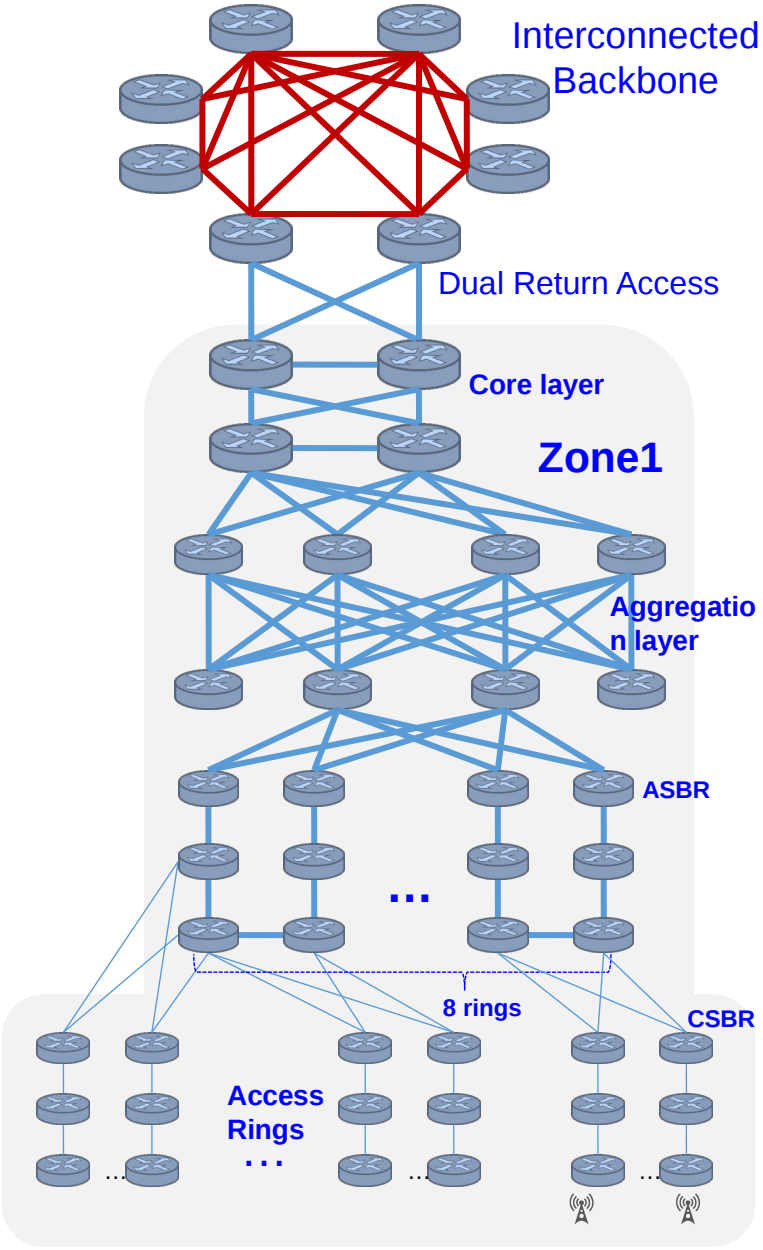
AF1..AF(n-1): length of the RU of the tree-next-hops RU

Draft suggests 8/12 bits length each

RU-Length / RU-Offset

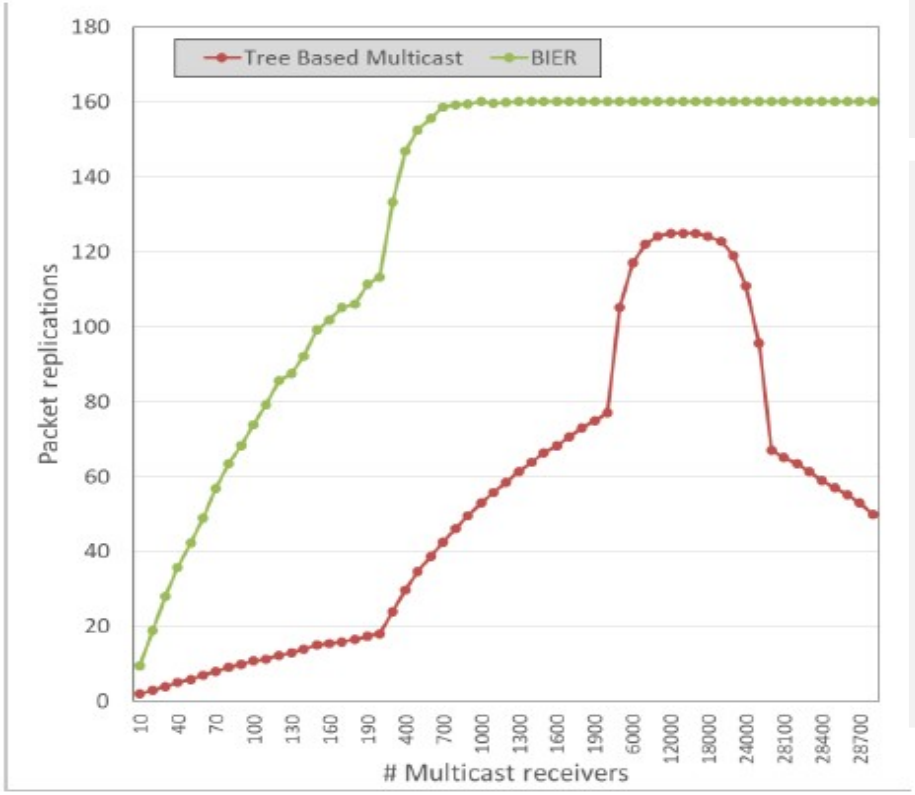
Draft suggests 2 * 12 bits.

Initial performance gain analysis (simulation based)



Topology description:

1. Typical topology of Beijing Mobile in China.
2. All zones dual homing access to backbone.
3. Core layer: 4 nodes full mesh connected
4. Aggregation layer: 8 nodes are divided into two layers, with full interconnection between the layers and dual homing access to the core layer on the upper layer.
5. Aggregation rings: 8 rings, 6 nodes per ring
6. Access rings: 3600 nodes, 18 nodes per ring



Comparison notes:

1. **RBS:** We randomly select egress points as group members, with the total number ranging from 10 to 28800 (for sake of simplicity, we assume merely one client per egress point). The egress points are randomly distributed in the topology with 10 runs for each value, showing the average result in our graphs. The total number of samples is 60
2. **BIER:** We divide the overall topology into 160 BIER domains, each of which includes 180 egress points, providing the total of 28000 egress points.
3. **Simulation:** In order to compare the BIER against the in-packet tree encoding mechanism, we limit the size of the header to 256 bits (the typical size of a BIER header).

Conclusion:

BIER reaches its 160 packet replication limit at about 500 users (BFER), while the in-packet tree encoding reaching its limit of 125 replications at about 12000 users. And the following decrease of replications is caused by the use of node-local broadcast as a further optimization.

For the sake of comparison, the same 256-bit encapsulation limit is imposed on RBS, but we can completely break the 256-bit encapsulation limit, thus allowing the source to send fewer multicast streams.

(A)Replication efficiency for small trees in large network

10,000 BFER, BitStringLength 256

- Traffic to small set of eg.: 10, 20, 30, 40 destinations
- BIER: May still need up to 10, 20, 30, 40 packet copies
  - …because each BFER may be in a different SI (bitstring) 10000/256 = 40
- BIER-TE: problem maybe factor 2..4 worse
  - Each 256 bitstring may need 25..75% bits for “steering”, not BFER

Who cares about “small trees in large networks” ?

- Multicast – since introduction of PIM Sparse Mode (PIM-SM, 1994)
  - sparse-mode definition: built for small trees in large networks
- Operators of large networks…

(B) Large(r) header benefits / problems

WARNING
OFFENSIVE
TECHNOLOGY
SUGGESTIONS

10,000 BFER, BitStringLength 10,000

10,000 destinations (BFER) in tree, data traffic size: 1280 bytes
One packet sufficient for all destinations (BFER), header 1250 bytes.
Traffic overhead: 2x, compared to ingress replication overhead of: 10,000x

PIM-SM: 1x – but weigh against all operational and “Traffic Engineering” benefits
Per-packet destination set selection also benefit of BIER over IP multicast!

10,000 BFER, BitStringLength 256, 10,000 destinations:

Traffic overhead: 41x (40x data, ca. 1x bitstring headers)

Why no large headers ?

NOT?!: MTU – IMHO non-issue.
Controlled network MTU often extended beyond 1500 when it was worth it.

NOT?! Hardware accessibility of header
Evolved from 128 byte … 512 byte until today. Easily 1024 bytes in 10 years (if this would become popular)

BUT: Per-packet processing overhead of flat bitstrings
Need to examine all 10,000 bits, even when router can do at most 10 copies (10 interfaces)

Lot of interesting optimization (heurisitc) possible, but still fundamental issue
see e.g.: paper from University Tuebingen (optimized BIER implementation on P4)
https://atlas.informatik.uni-tuebingen.de/~menth/papers/Menth22-Sub-2.pdf

(C) Address autoconfiguration

Assignment of BFER into Bitstrings (aka: bift-id) is the factor impacting replication efficiency

- If all BFER in one edge region are in one SI / bitstring, only one packet needs to go there
- If each BFER in an edge region is in different SI, each BFER requires a separate BIER packet copy

Worse in BIER than BIER-TE

Complex optimization issue

Eliminated in RBS

Benefits review

Scale: No hard division of BFER space across bitstring by SI.

- No “unnecessary many bits” - Arbitrary subset of BFER can go into a single packet -
- As long as the serialization of the RBS-Address for them fits into available header size

Auto-configuration: Every BFR could self-assign bits for its bitstring:

- Number N interfaces/BIER-neighbors with bits 1..N
- Signal bits via IGP. BFIR can now construct RBS-Address structures.

Eliminate dependency against hop-by-hop (IGP) re-routing

- Eliminates IGP Micro-loops
- Eliminates BIFT/FIB updates on topology changes
- Example: Sender / BFIR is VM in DCN getting IGP updates and events on network link/node failure
  - BFIR can quickly re-calculate RBS-Address(es) needed
  - No update of HW-forwarding tables involved – fastest reactive reconvergence ?!

Next Steps

Looking for interest / feedback / collaboration.

Easy enough packet processing ?

+++ Each node only needs to look at its own (local) bitstring

- - - Calculation to determine new RU-Offset/RU-Length for each packet copy

Want to continue evaluating / improving approach

Scalability comparison, implementation feasibility (e.g.: P4)

Evaluate alternative proposals with same / better benefits ?

Q: What are criteria that would make (any) such „non flat-bitstring“ solutions acceptable as BIER-WG work ?

Assuming interest and (as necessarily) rechartering