BIER-TE TEAS framework

IETF101

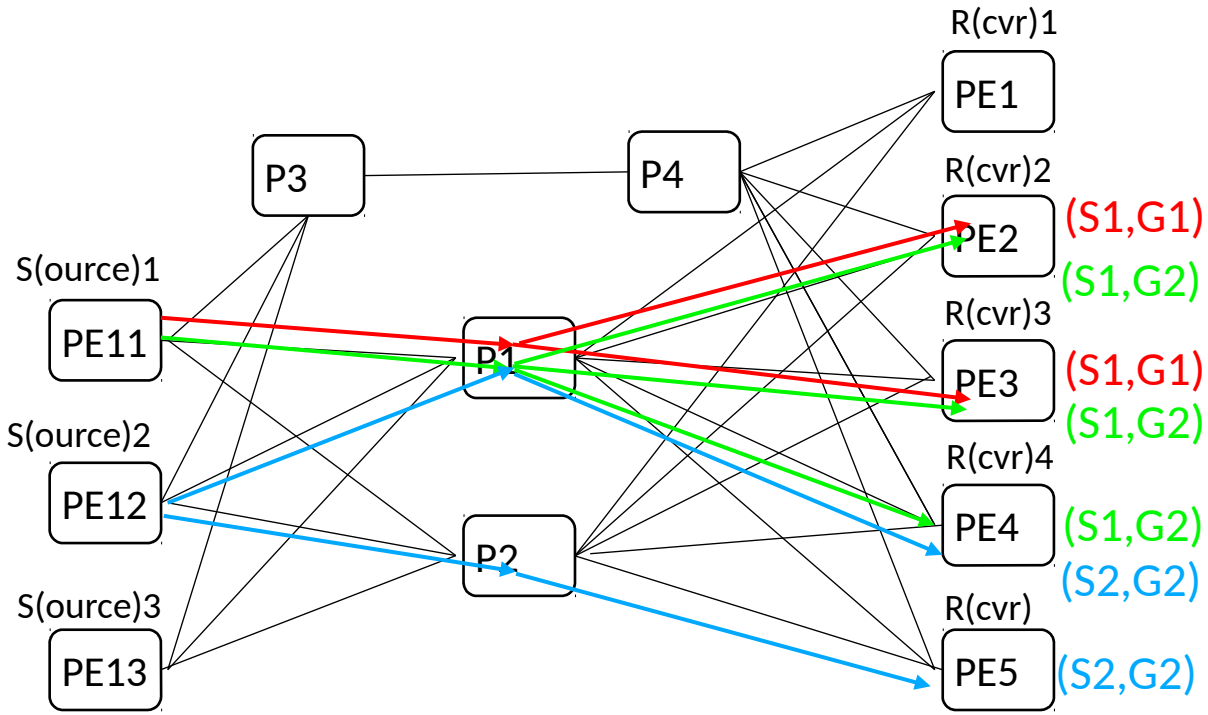
draft-eckert-teas-bier-te-framework-00

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

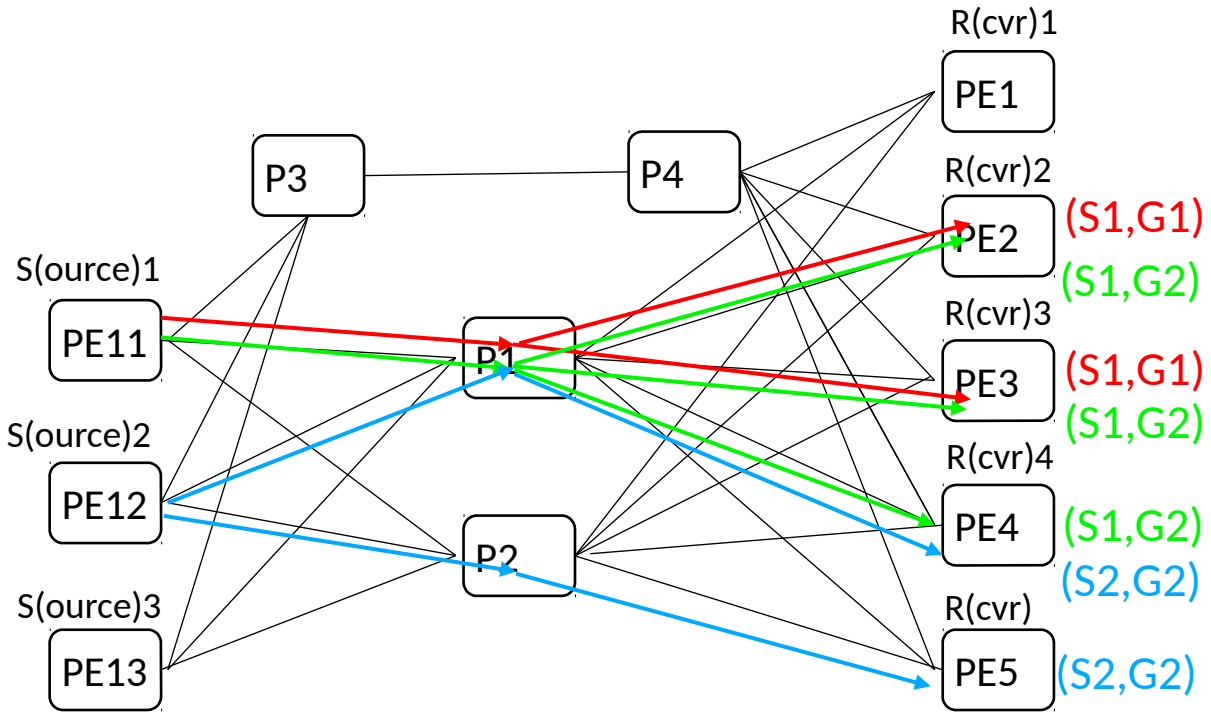
Background
Multicast, BIER, BIER-TE

Slides with text only for reference after IETF101 presentation:

Traditional IP multicast problems

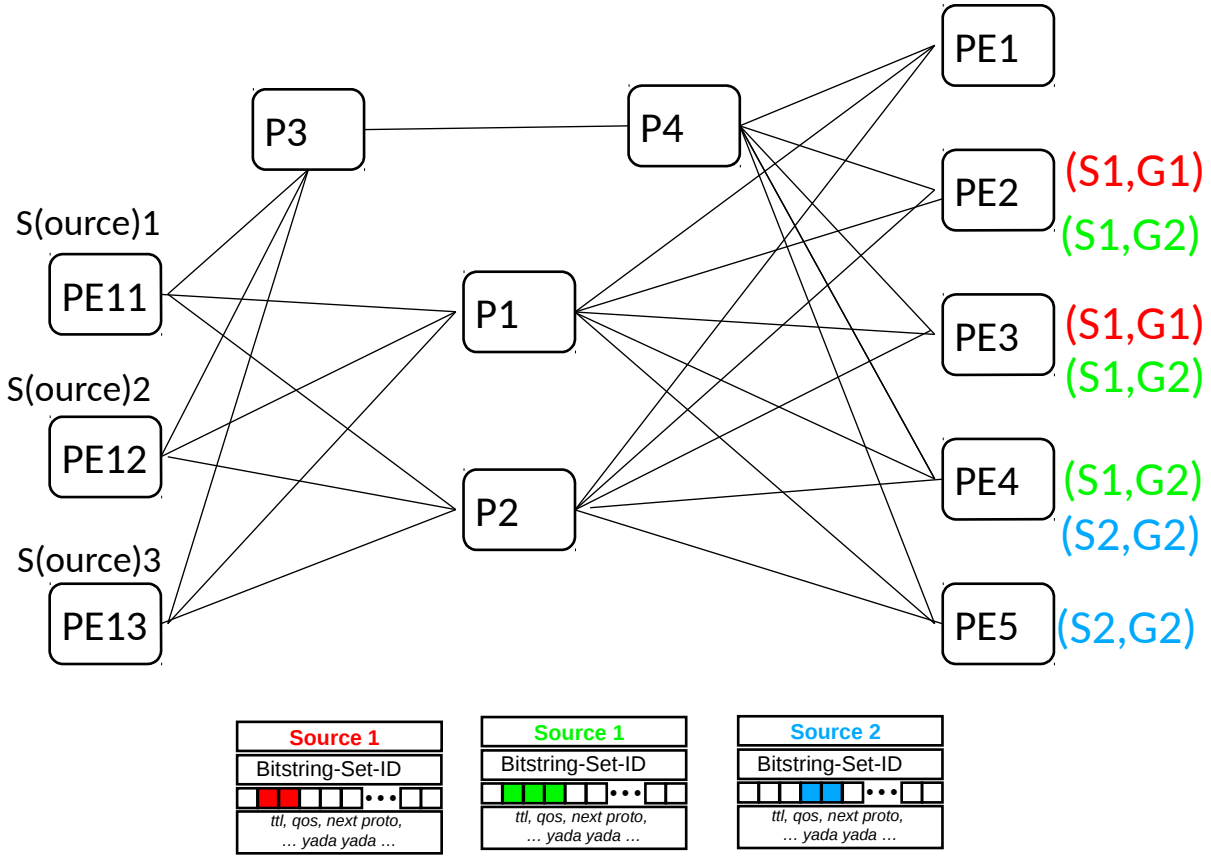


Traditional IP multicast problems

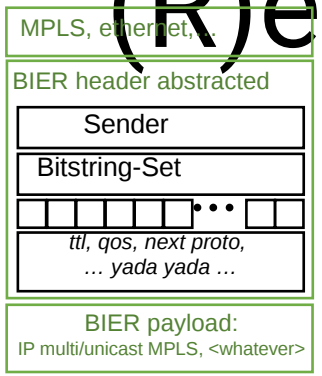


- Tree state on P nodes
  - (S,G) – per source S, per receiver group G
  - 3 sender, 5 receiver: up to 2^3*2^5 trees
  - Real networks (src,group large) -> impossible
    - Aggregation == wasted traffic
  - Forwarding, control plane state, signaling
    - Performance operations problem long before limits
- PIM, mLDP
  - No non-shortest path tree support native (use MT-IGP)
  - No cost reduced tree (eg: (S2,G2) – better both via P2)
  - “randomized” ECMP control
  - mLDP somewhat better than PIM (later design)
- RSVP-TE P2MP
  - Most expensive state (control, signaling)
  - But allows to path engineer trees arbitrarily
  - No support for (*,G) trees (as in PIM, mLDP)

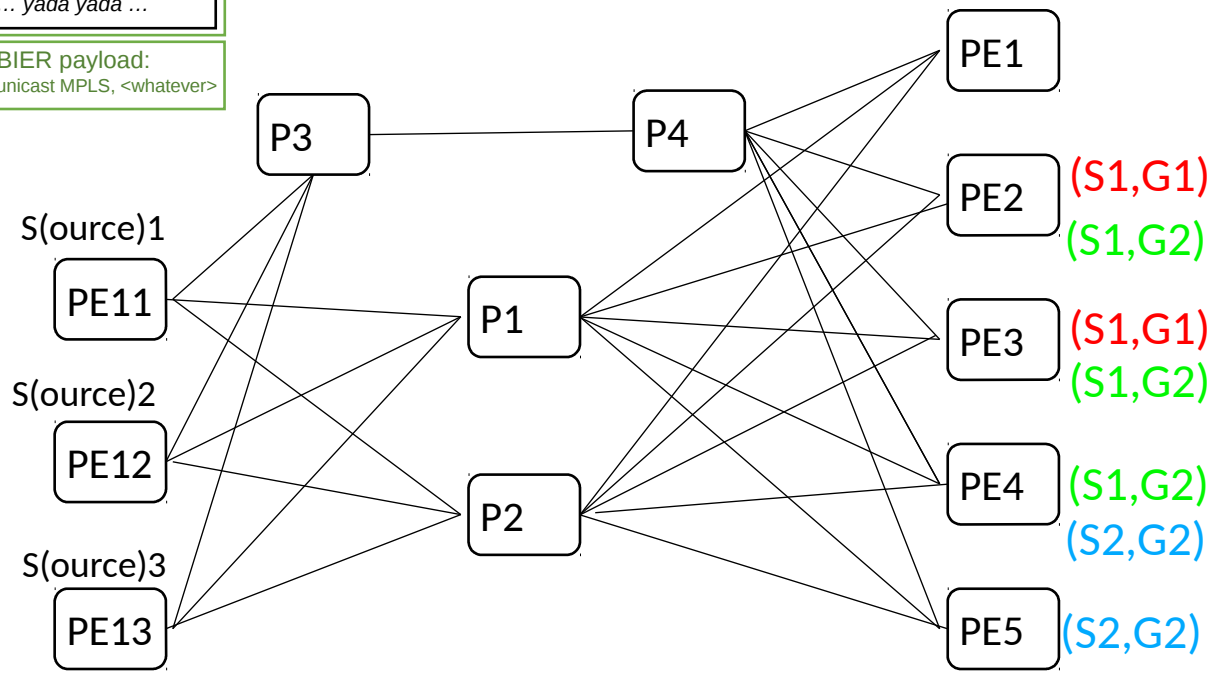
BIER – (B)IT (I)ndexed (E)xplicit (R)eplication

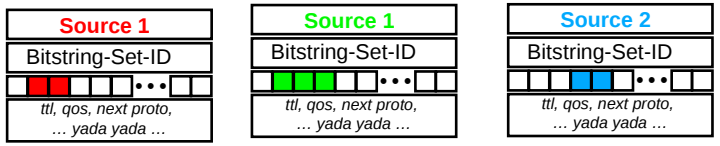


BIER – (B)IT (I)ndexed (E)xplicit (R)eplication



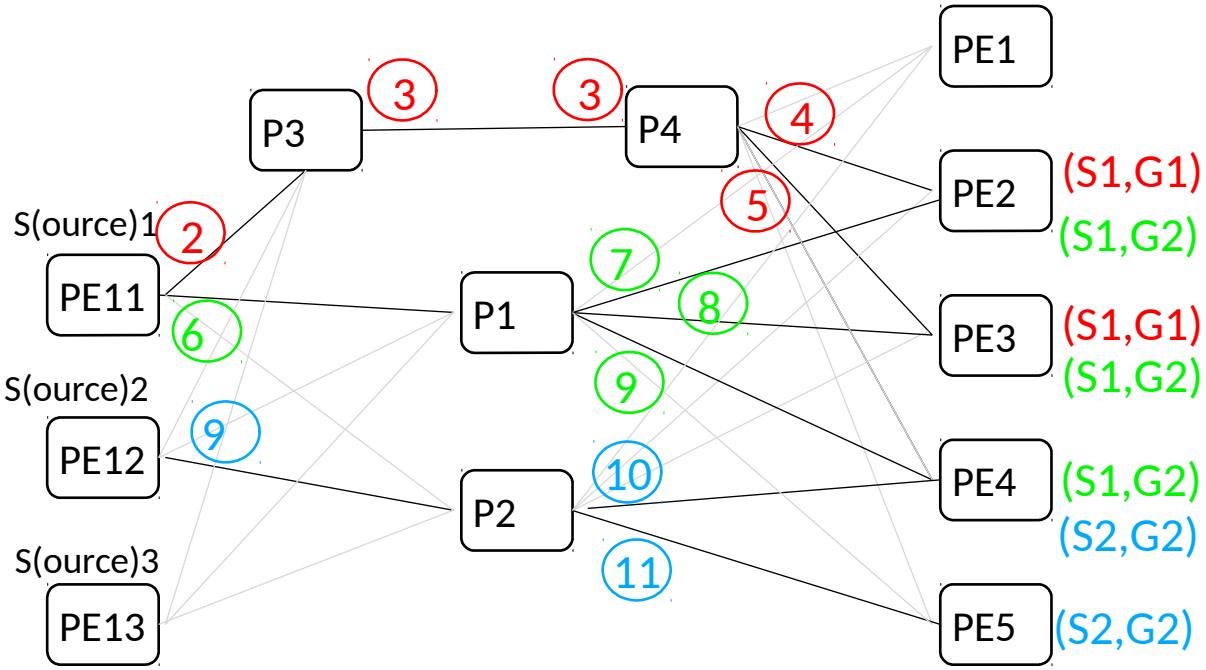
Actual name: BFIR-id
Actual name: BIFT-id
The BitString





- **STATELESS**: No tree state on P nodes
  - No tree signaling/control either !
- BIER 'for SR ~~dummies~~ experts'
  - 'BIER packet header indicates a SET OF egres-PE node-SIDs'
- Up to 256 egres PE, each one encoded as 1 bit in 256 bit "bitstring" in the bier packet header
- BIER-IGP extensions:
  SPF routes for these ~~SIDs~~ bits
- PE/P node forwards/replicates BIER packet:
  - One copy sent to each interface that is (according to IGP) leading to one or more bits set in packets BitString.
  - (also reset on each copy bits not reachable according to SPF route via that interface)
- Many sets of 256 possible BitStrings:
  - Bit set identifier in BIER header (BIFT-id)
  - Source needs to send one packet for each set of up to 256 receivers
- Nice ECMP and MT-IGP support, but
  - But no generic path engineering

BIER-TE – BIER with traffic engineering (1)



Unused links/adjacencies greyed out for clarity

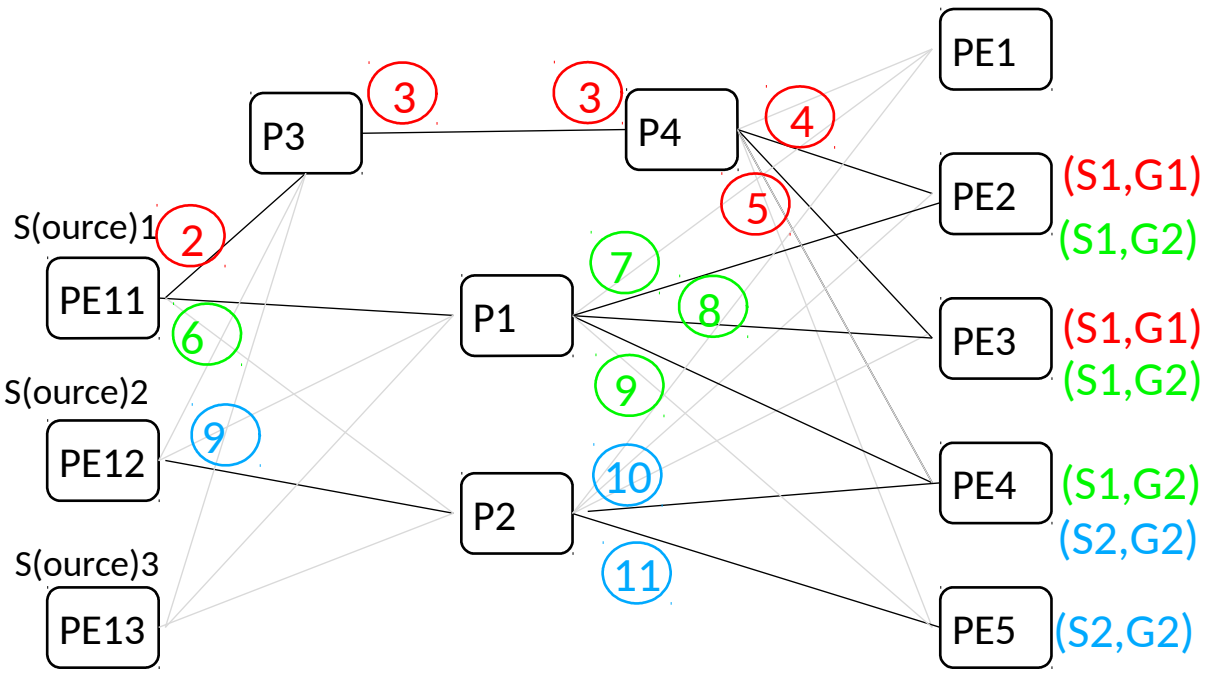
Bitstrings:

(S1,G1) = 2 3 4 5

(S1,G2) = 6 7 8 9

(S2,G2) = 9 10 11

BIER-TE – BIER with traffic engineering (1)



Unused links/adjacencies greyed out for clarity

Bitstrings:

(S1,G1) = 2 3 4 5

(S1,G2) = 6 7 8 9

(S2,G2) = 9 10 11

- BIER BitString indicate BFER-id
  - Aka: Receiver PE (or wherever BIER domain ends)
- BIER-TE BitStrings indicate transit adjacencies
  - Most simple: every interface in topology is a bit
- Forwarding rule: every node (BFR = P/PE):
  - Replicate based on only on direct adjacency bits
  - Resets bit when using its adjacency
  - Eg: P1 - looks only at bits 7, 8, 9 in example & resets them
- Optimizations to reduce “bit-waste”
  - Bit semantics:
  - P2p link bit (e.g.: bit 3 on both adjacencies of interface)
  - Lan, stub, flood, punt, … bits
- Any traffic engineering
  - NO STATE –
    Engineer path (graph!) of every packet individually through bitstring fom sender (BFIR) in BIER(-TE) header.
- Bit waste… ?
  - BIER: 1 packet ~ 256 receivers
  - BIER-TE 1 packet ~ 100 receivers ?
    - See further slides

BIER-TE – BIER with traffic engineering (2)

- Routed adjacencies (*save the bits*):
  - Tunnel adjacency (GRE/MPLS/SR label stack/…) to desired next-hop
  - Replication may only be required on limited number of nodes in (larger) topologies
  - Tunnel through non BIER-TE capable nodes

- DetNet (or similar)
  - PREF – Packet Replication and Elimination Function (DetNet)
    - Transmit packets twice with flow-ID and sequence number – across disjoint paths
    - Remove duplicate copies via sequence number “deduplication” on destination
  - BIER-TE header proposed to include sequence number (and ‘existing’ flow-id)
  - BIER-TE can be interesting not only for multicast but also unicast
    - Replication e.g.: only/primarily for PREF. not for ‘multicasting’
  - PREF suggested to be part of the BIER-TE TEAS framework
    - Can maybe also be defined to be independent of BIER-TE
    - But some BIER-TE specific OAM aspects.

Pathsets: Determine BIER-TE Bitstrings

- Pathset: result of (controller/BFIR) calculations of paths
  - PathSet-i(bfir-j) = (bfer-k | {bitstring-i-j-k})

- Configure traffic classes to use a BIER-TE Pathset:
  - E.g.: BFIR-10: VPN-foobar traffic should use Pathset-7(10)

- BIER: BitString(set of BFER-k) = OR (BFER-k-id bits)
- BIER-TE: BitString(set of BFER-k) = OR (bitstring-i-j-k)

- Bitstring-i-j-k can be redundant (e.g.: for PREF)

- More complex with minimum cost (“steiner”) trees
  - Adding/removing destination requires recalculation
  - Still much faster/easier than recalculation plus re-signaling (RSVP-TE/P2MP)

BIER-TE TEAS framework (proposed / incomplete)

BIER-TE signaling architecture (proposed)

Configuration

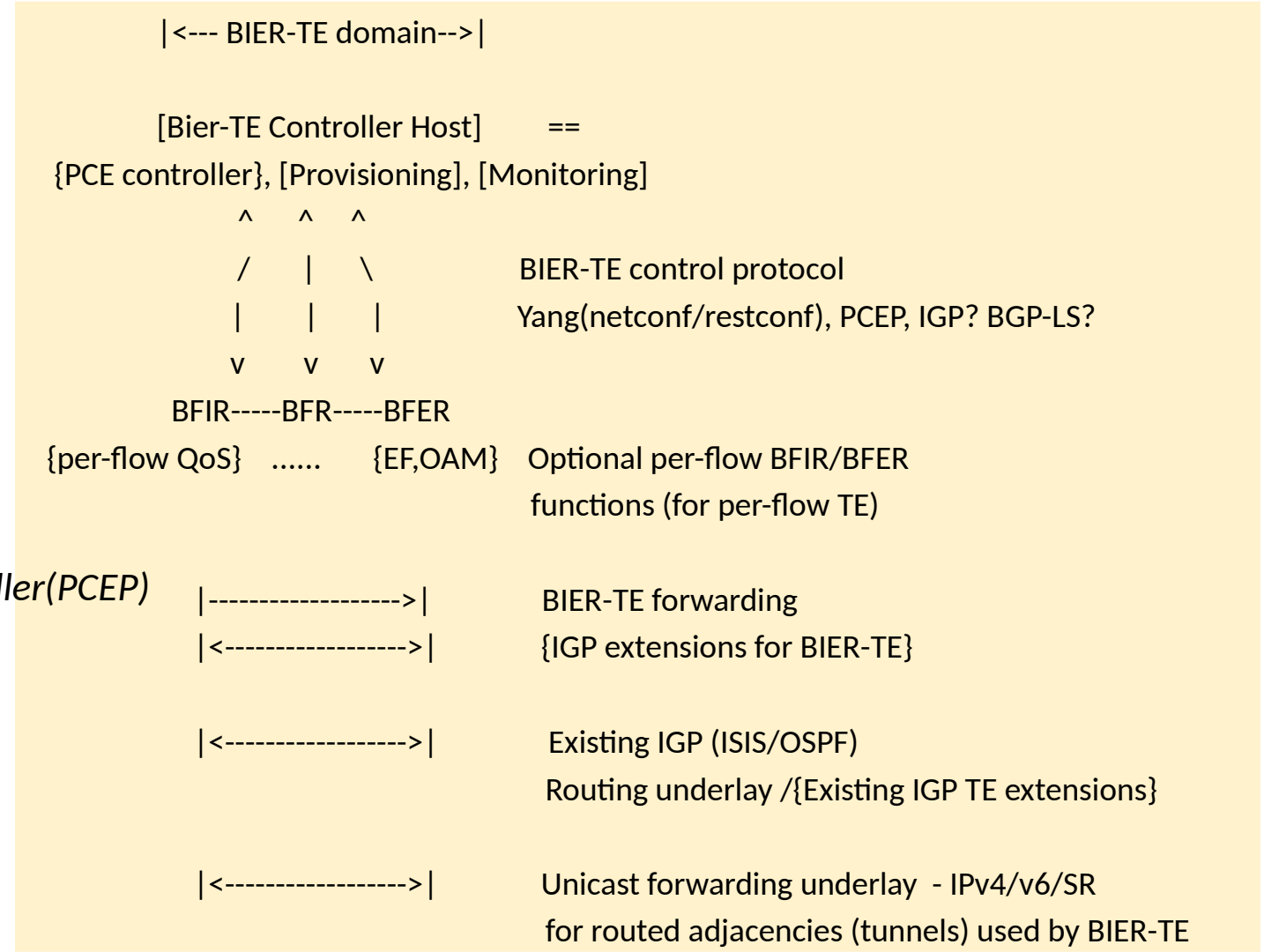
- "BIER-TE topology"
  - *When BIER-TE service added/changed*
  - *When network topology changes*

Traffic: Bitstrings/PathSets

- *Precalculate on controller/PCEP*
  - *Send to BFIR (and BFER for PREF/OAM)*
- *Allow BFIR to calculate itself*
- *Allow BFIR to dynamically request from Controller(PCEP)*

PREF, flow QoS *(optional, e.g: DetNet)*

- BFIR
  - *Insert PREF sequence number, flow-id*
- BFER (receiver)
  - *Elimination function, OAM / Sequence number, flow-id*

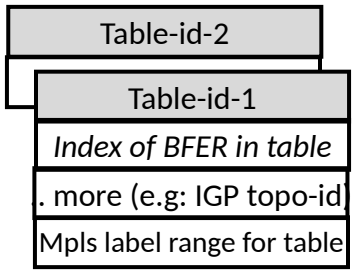


BIER-TE data model (topology)

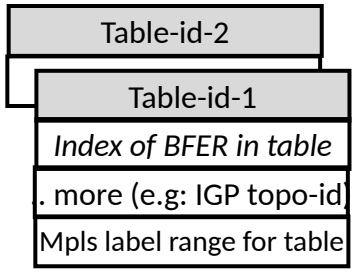
BIER - Expressing Topology

- BIER Topology
  - Flooded information by BFR about themselves
  - BFER include their BFR-ID
  - MPLS: All BFR include label ranges (similar to SR)
    Each table identified by a label from the range.
- BIER Routing Table
  - Constructed from received IGP announcements
  - List of bit (indices) for BFER
  - Next-hop – from path calculation
  - BFER IP identifier (“BFR-Prefix”)
    - Just tying BFER bitindex (BFER-id) to IP routing
      Not needed by BIER forwarding
- BIER Forwarding Table
  - BitIndex and Next-hop copied from BIER Routing Table
  - F-Bitmask: mask of all bits to the same neighbor
    - Used during forwarding when creating copy to neighbor
      reset all other bits for copy to this neighbor

BFER-1 IGP
“topology” announcement



BFER-n IGP
“topology” announcement



Flooded via IGP

Path selection – e.g.: SPF
for each received topology Announcement

Routing Table-id-2

Routing Table-id-1

BitIndex	BFER IP identifier	Next-hop
1	…	R1
…		
256	…	R5

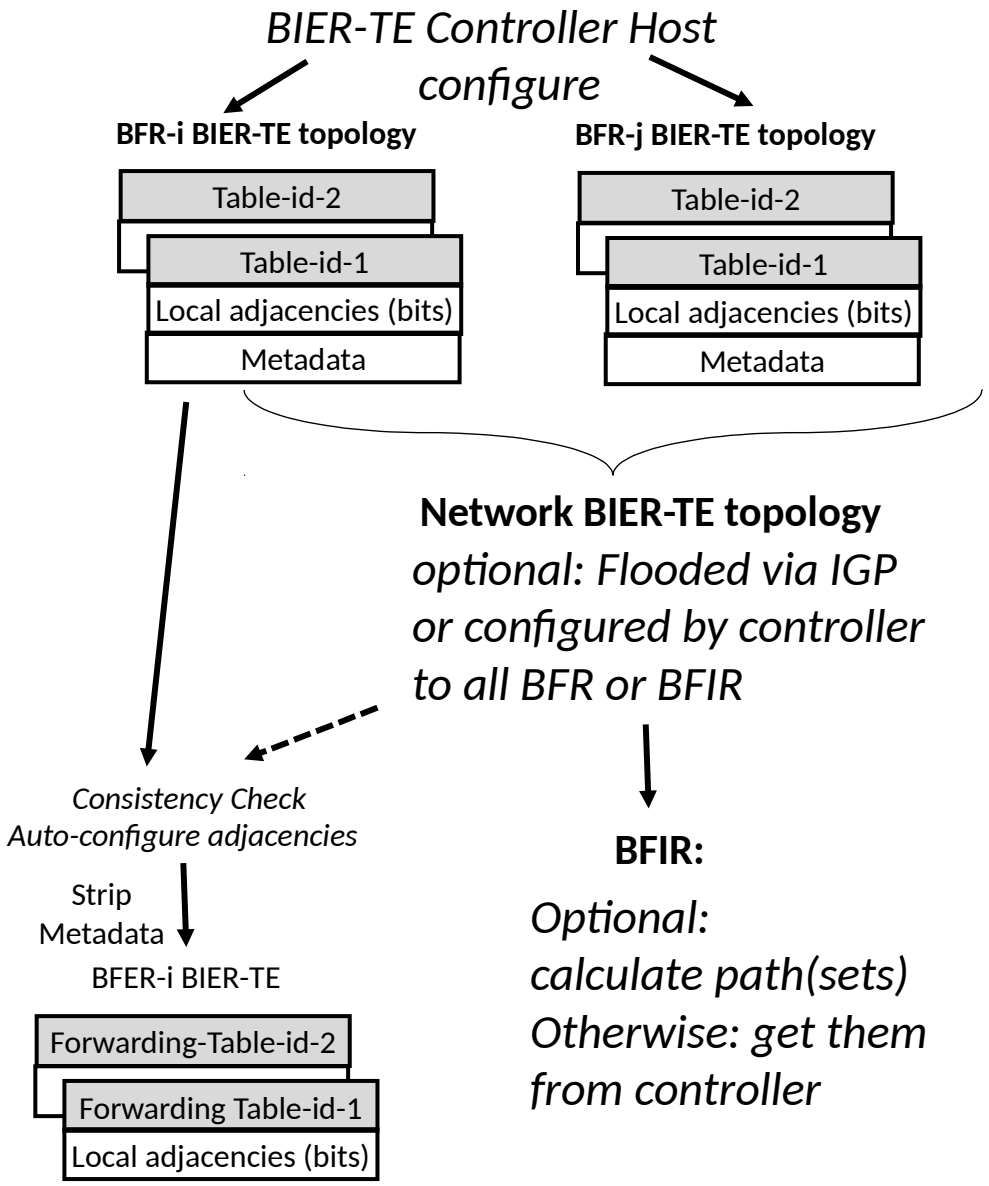
Forwarding Table-id-2

Forwarding Table-id-1

BitIndex	F-Bitmask	Next-hop
1	0111	R1
…		
256	11000	R5

BIER-TE - Expressing Topology (proposal) (1)

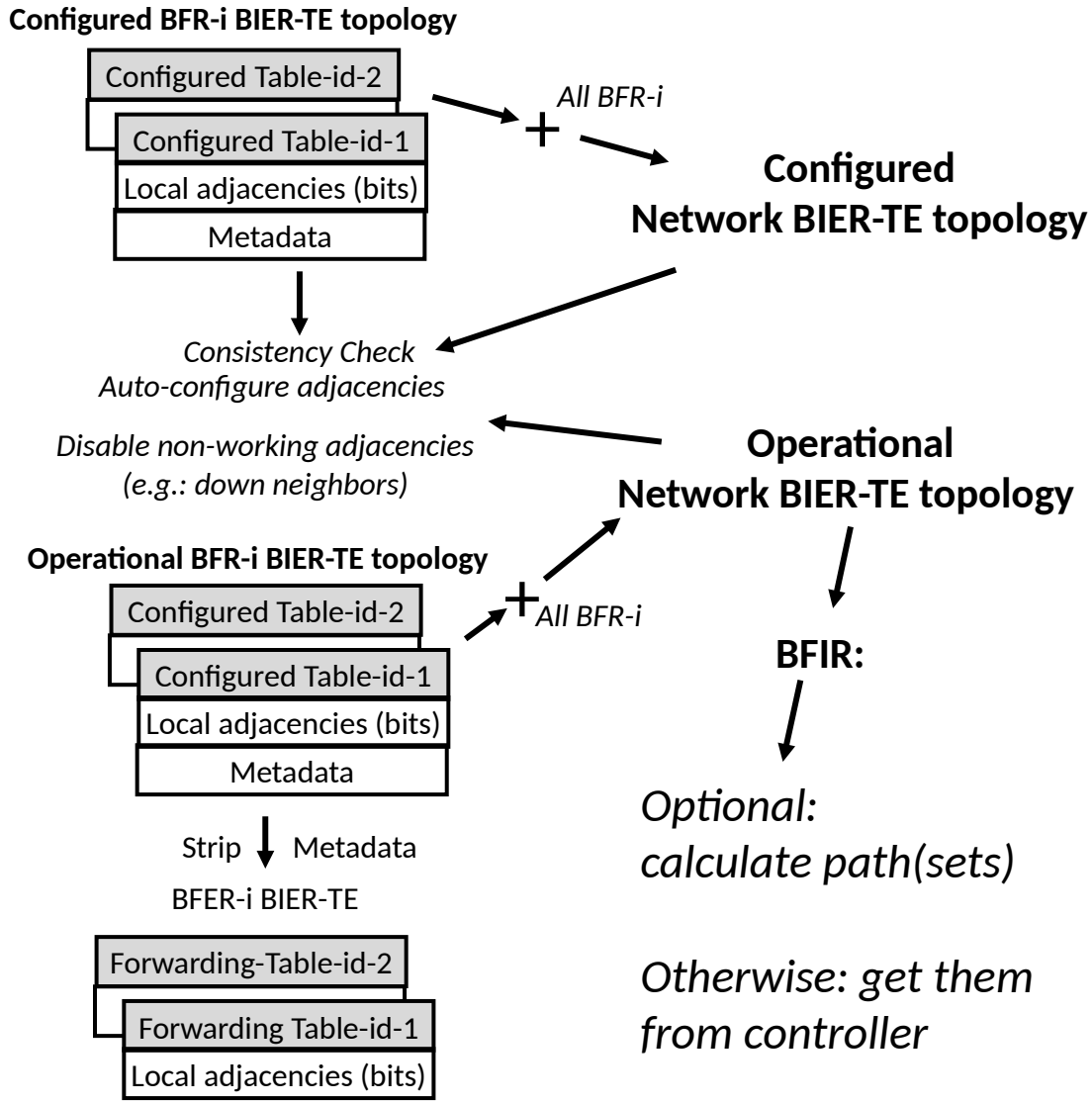
- BIER-TE BFR-i Topology
  - Local adjacencies (bits used by BFR), metadata
  - Configured by controller to each BFR-I
- BIER-TE BFR-i Forwarding Table
  - Almost the same as BIER-TE BFR-i Topology without metadata
  - Plus auto configured bits/adjacencies
  - Minus inconsistent/inoperable bits
- BIER-TE Network Topology
  - Set of all BIER-TE BFR-i Topologies
  - Needed on other BFR only for consistency check or adjacency auto-configuration
  - Needed on other BFIR for local path calculation
- No equivalent of BIER Routing Table
  - But table of path(sets)/bitstrings required on BFIR



BIER-TE Topology: configured / operational

- Distinguish “configured” and“operational”
  - Path calculation (controller, BFIR) depends on actual operational BIER-TE network topology
    - Because configured topology does not include auto- configured bits/adjacencies. But does include adjacencies that may not be operational.
  - Inconsistency discovery / auto-configuration depends on configured consistency
    - Because operational topology will not show inconsistency when remode node already disabled bits due to inconsistency discovered.

- BIER-TE Forwarding table same as configured topology table
  - Except no need for metadata in forwarding table
  - Operational topology table stands in for forwarding table externally
  - No need to export forwarding table (device internal) ?!



BIER-TE Topology: Adjacency types

local_decap:
    VRF / context: (TBD)

forward_connected: (send to interface)
    dest: link (ifIndex)
           [, addr (nexthop)]
    DNR: boolean (Do Not Reset)

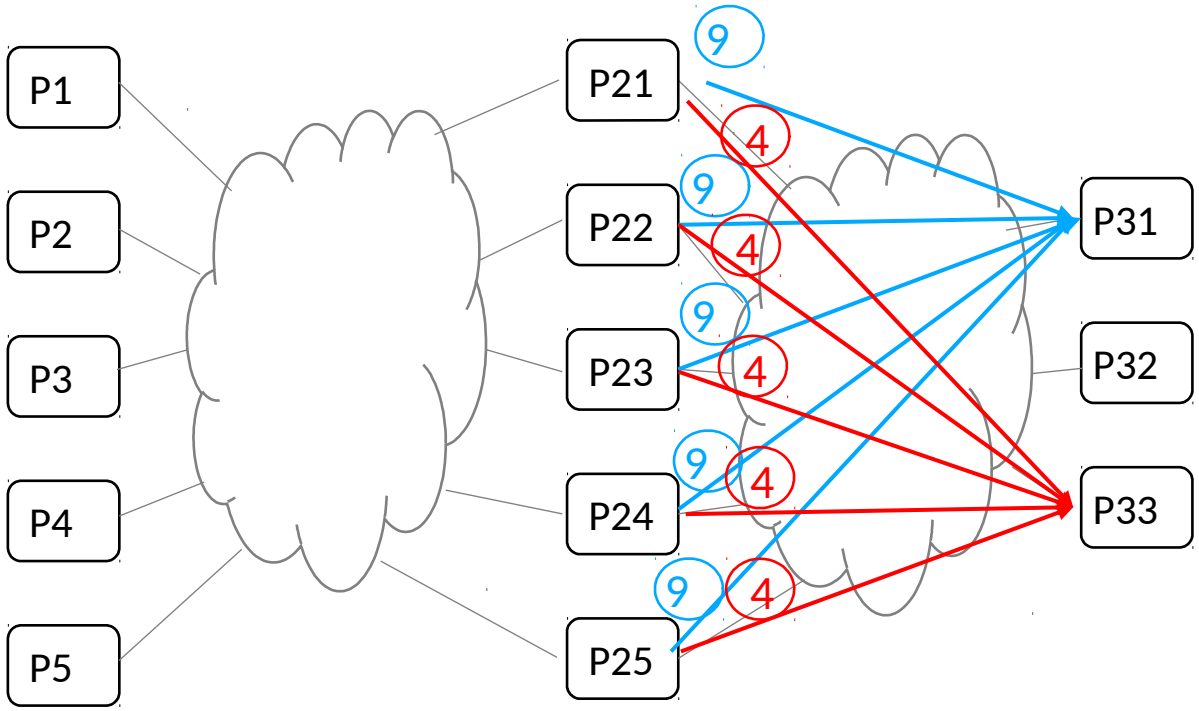
forward_routed:
    destination: ... (router-id, SID
    TBD: path/encap info (e.g: SR SID stack)

ECMP:
    list of 2 or more adjacencies,
    forward_connect and/or forward_routed

BIER-TE Topology

BFR: <bfr> (eg: BFR-prefix of BFR)
Instance: "configured", "operational", (of this BFR itself)
"learned-configured", "learned-operational" (from another BFR)
BIFT-ID: <SD subdomain,BSL bitstring length,SI Set Identifier>
BIFT-Name: string (optional)
BFR-id: 16 bit (BIER-TE ID of the <bfr> in this BIFT or undefined if not BFER)
Ingres-groups: (list of) string (1..16 bytes) (group that <bfr> is a member of)
EF: <TBD> (optional, parameters for EF Function on this BIFT)
OAM: <TBD> (optional, parameter for OAM Function on this BIFT)
Bits: #BSL (List of bits – BitStringLength, e.g.: 265)
BitIndex: 1...BSL
BitType(/Tag): "unassigned", “down”, (no adjacencies – *maybe compress data struct*)
"unique", "p2p", "lan", "leaf", "node", "flood", "group”
(**Names**: (list of 0 or more) string (1..16 bytes) (for BitTypes that require it)
List of 0 or more **adjacencies**:
as on previous slide (most bits have 1 adjacency, but could be list)

BIER-TE – (partial) auto configuration (proposal)



Ingres-group:
midpoint2

- Avoid configuring bits 4, 9 each on P21,…P25
- Configure P21,…P25:
  - member of ingres-group: midpoint2
- Configure for P31
  - bit 9 type “group”, name “midpoint2”
- Configure for P33
  - bit 4 type “group”, name “midpoint2”
- “configured” instance of topology shows above config
  - Not operational – no adjacencies for bits 4, 9!
- “operations” instance of topology shows
  - P21,…P25:
    - Bit 4 type “p2p_unidirectional”, routed_adjacency to P33
    - Bit 9 type “p2p_unidirectional”, routed_adjacency to P31

BIER-TE path selection

TBD: Path selection

- Fist model to define ?
- Yang model for PathSet
  - Configuration/Provisioning from controller/operator
  - Map to traffic classes

- Request/Reply model via PCEC ?

- Hopefully guidance from TEAS
  - Would like reuse of existing solutions, adopt to BIER-TE

BIER-TE bandwidth management

TBD: Bandwidth/QoS management

- Bandwidth allocation / bandwidth aware path selection

  - Local decision on controller
    -> Requires dynamic request of Bitstrings/Pathsets by BFIR from controller
    -> Preferred initial option

  - Local decision on BFIR
    -> Not currently considered, but possible:
    -> Keep midpoint BFR free of traffic related state (BIER principle)
      -> RSVP-TE/IGP bandwidth extensions inappropriate
    -> BFIR could signal path resources it has allocated to other BFIR
      -> Signaling could use BIER/BIER-TE – only BFIR need to be receivers

Next steps ?!

- Discuss / determine order of next steps
  - Yang/PCEP configuration model first ?
- Improve framework according to TEAS guidance
- Finalize topology model
  - Discuss in LSR acceptable topology information
- PREF, OAM,…