

**Framework for Traffic Engineering with BIER-TE forwarding (Bit Index  
Explicit Replication with Traffic Engineering)  
draft-eckert-teas-bier-te-framework-00**

Abstract

BIER-TE is an application-state free, (loose) source routed multicast forwarding method where every hop and destination is identified via bits in a bitstring of the data packets. It is described in [[I-D.ietf-bier-te-arch](#)]. BIER-TE is a variant of [[RFC8279](#)] in support of such explicit path engineering.

This document described the traffic engineering control framework for use with the BIER-TE forwarding plane: How to enable the ability to calculate paths and integrate this forwarding plane into an overall TE solution.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 6, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	Introduction and Overview . . . . .	<a href="#">2</a>
<a href="#">2.</a>	BIER-TE Topology management . . . . .	<a href="#">6</a>
<a href="#">2.1.</a>	Operational model . . . . .	<a href="#">6</a>
<a href="#">2.2.</a>	BIER-TE topology model . . . . .	<a href="#">7</a>
<a href="#">2.3.</a>	Consistency checking . . . . .	<a href="#">10</a>
<a href="#">2.4.</a>	Auto-configuration . . . . .	<a href="#">11</a>
<a href="#">3.</a>	Flow Management . . . . .	<a href="#">12</a>
<a href="#">3.1.</a>	Operational / Architectural Models . . . . .	<a href="#">12</a>
<a href="#">3.1.1.</a>	Overprovisioning . . . . .	<a href="#">13</a>
<a href="#">3.1.2.</a>	PCEC . . . . .	<a href="#">13</a>
<a href="#">3.1.2.1.</a>	per-flow QoS - policer/shaper/EF . . . . .	<a href="#">14</a>
<a href="#">3.1.2.2.</a>	DiffServ QoS . . . . .	<a href="#">15</a>
<a href="#">3.2.</a>	BIER-TE flow model . . . . .	<a href="#">15</a>
<a href="#">4.</a>	Security Considerations . . . . .	<a href="#">17</a>
<a href="#">5.</a>	IANA Considerations . . . . .	<a href="#">17</a>
<a href="#">6.</a>	Acknowledgements . . . . .	<a href="#">17</a>
<a href="#">7.</a>	Change log [RFC Editor: Please remove] . . . . .	<a href="#">17</a>
<a href="#">8.</a>	References . . . . .	<a href="#">18</a>
	Author's Address . . . . .	<a href="#">19</a>

## [1.](#) Introduction and Overview

This document proposes a framework and abstract data model for the control plane of BIER-TE as defined in [[I-D.ietf-bier-te-arch](#)] (BIER-TE-ARCH). That document primarily defines the forwarding plane and provides some example scenarios how to use it.

BIER-TE is a forwarding plane derived from BIER ([\[RFC8279\]](#)) in which the destinations of packets are bits in a bitstring. Every bit indicates a destination (BFER - BIER Forwarding Exit Router) and an IGP is used to flood those "bit addresses" so hops along the path from sender (BFIR - BIER Forwarding Ingress Router) through intermediate nodes (BFR) can calculate the shortest path for each destination (bit) and simply copy the received packet to every interface to one or more bits set in the packet.

In BIER-TE, shortest path calculation is replaced by bits of the bitstring indicating intermediate hops and pre-populated forwarding tables (BIFT - Bit Index Forwarding Tables) on every BFR indicating



those bits. In the simplest case, every interface on a BFR has a unique bit assigned to it, and the BIFT of only that BFR will have in its BIFT for this bit an adjacency entry indicating that interface. This ultimately allows to indicate any sub-graph of the network topology as a bitstring and hop-by-hop perform the necessary forwarding/replication for a packet with such a bitstring. More complex semantics of bits are used to help saving bits. A typical bitstring size supportable is 256 bits, the original BIER specification allows up to 1024 bits. BIER-TE may be specifically interesting for typically smaller topologies such as often encountered in DetNet scenarios, or else through intelligent allocating and saving of bits for larger topologies, some of which is exemplified in BIER-TE-ARCH.

One can compare BIER-TE in function to Segment Routing in so far that it attempts to be as much as possible a per-packet "source-routed" (for lack of better term) forwarding paradigm without per-application/flow state in the network. Whereas SR primarily supports simple sequential paths indicated as a sequence of SIDs, in BIER-TE, the bitstring indicate a directed and acyclic graphs (DAG) - with replications. BIER-TE can also be combined with SR and then bits in the bitstring are only required for the nodes (BFR) where replication is desired, and the paths between any two such replication nodes could be SIDs or stack of SIDs that are selected by assigning bits to them (routed adjacencies in the BIER-TE terminology).

In BIER-TE-ARCH, the control plane is not considered. In its place, a theoretical BIER-TE Controller Host uses unspecified signaling to control the setup of the BIER-TE forwarding-plane end to end (all bits/adjacencies in all BFR BIFTs) and during the lifecycle of network device install through the determination of paths for specific traffic and changes to the topology. This document expands and refines this simplistic model and intends to serve as the framework for follow-up protocol and data model specification work.

The core forwarding documents relevant to this document are as follows:

- o [[RFC8279](#)] (BIER-ARCH): as summarized above.
- o [[RFC8296](#)] (BIER-ENCAP): The encapsulation for BIER packets using MPLS or non-MPLS networks underneath.
- o [[I-D.ietf-bier-te-arch](#)] (BIER-TE-ARCH): as summarized above.
- o [[I-D.thubert-bier-replication-elimination](#)] (BIER-EF-OAM): Extends the BIER-TE forwarding from BIER-TE-ARCH to support the Elimination Function (EF) and an OAM function. The Elimination



Function is a term from DetNets resilience architecture: Multiple copies of traffic flows are carried across disjoint path, merged in a BFR running the EF and duplicates are eliminated on that BFR based on recognizing duplicate sequence numbers. Engineered multiple transmission paths are a key reason to leverage BIER-TE.

- o [[I-D.huang-bier-te-encapsulation](#)] (BIER-TE-ENCAP): Proposed encapsulation based on an extension of BIER-ENCAP. Identifies whether the packet expects to use a BIER or BIER-TE BIFT. Also adds a control-word in support of (optional) elimination function (EF) and interprets the pre-existing BFIR-ID and entropy fields as a flow-id.
- o [[I-D.eckert-bier-te-frr](#)] (BIER-TE-FRR): This document describes protections methods applicable to BIER-TE. 1:1 / end-to-end path protection is referenced in this document in the context of DetNet style PREF path protection. The options not discussed yet (TBD) in this document are link protection tunnels (such as used in RSVP-TE as well) and the novel BIER-TE specific protection method, in which nodes modify the bitstring upon local discovery of a failure.

The relevant routing underlay documents are as follows:

- o [[I-D.ietf-bier-isis-extensions](#)] (BIER-ISIS), [[I-D.ietf-bier-ospf-bier-extensions](#)] (BIER-OSPF): The BIER-ISIS and BIER-OSPF documents describe extensions to those two IGP in support of BIER. Effectively, every BFR announces the <SD,SI-range> BIFTs it is configured for, the MT-ID (IGP Multitopology-ID) they are using, and the BFR-ID it has in each SD (none if it does not need to operate as a BFER). For MPLS encapsulation, the base label for every SD is announced as well as the SI-range (one label per <SD,SI> is used).
- o There is currently no document describing IGP extensions for BIER-TE, but the goal is to define those based, using the proposals made in this framework, and as feasible re-using and/or amending those existing BIER IGP extensions.
- o [[I-D.ietf-bier-bier-yang](#)] (BIER-YANG): This document describes the YANG data model to provision on every BFR BIER. It also provides OAM functions. There is currently no model expanding this to support BIER-TE. This framework document defines elements that should be included in a BIER-TE YANG model.
- o TBD: incomplete list ?.



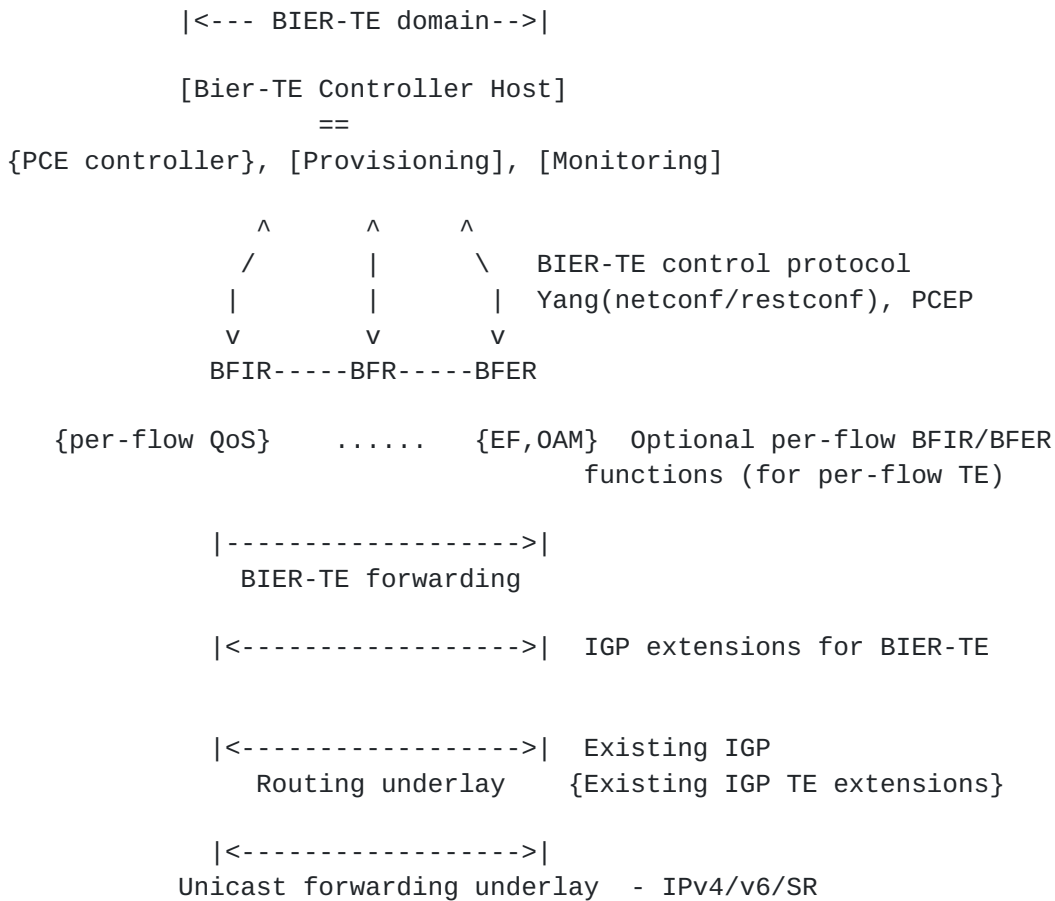


Figure 1: BIER-TE signaling architecture

The above picture is a modified version of Picture 2 from BIER-TE-ARCH reduced by the elements not considered in this document, and refined with those that are intended to be described by this document.

In comparison with BIER-TE-ARCH, Picture 2, this picture and this document do not include considerations for specific multicast flow overlay elements. Instead, it adds description of optional BFIR/BFER elements for per-flow QoS/EF (Elimination Function) and OAM, which are optional parts of an overall BIER-TE traffic engineering architecture. See BIER-EF-OAM for more background.

The routing underlay is refined in this document to consider a unicast forwarding underlay of IPv4/IPv6 and/or unicast SR (Segment Routing) for BIER-TE "forward\_routed" adjacencies. It also assumes an existing IGP, such as ISIS or OSPF as the routing underlay. This may include (TBD) extensions already supporting TE aspects (like those IGP extensions done for RSVP-TE).





This framework intends to support a wide range of options to instantiate it:

In one extreme (PCEC only), there is no IGP in the network that BIER-TE depends on, but all BIER-TE operations is managed in an SDN-style fashion from centralized components called "BIER-TE Controller Host" in BIER-TE-ARCH. This central packend can be further subdivided into a Configuration/Provisioning component to install the BIER-TE topology into the network and a PCEC (Pat Computation Engine Controller) and (TBD) monitoring components. After BIER-TE is operational, the PCEC calculates BIER-TE bitstrings for BFIR when they need to send traffic flow to

In the other extreme (IGP only), there is no need for a PCEC or NMS. The initial setup of the BIER-TE topology can be performed manually, using configuration options to support automatic consistency checking and partial auto-configuration to simplify this work. BIER-TE extensions of the IGP are used for consistency checking and autoconfiguration and finally to provide the whole BIER-TE topology to BFIR that can then autonomously calculate BIER-TE bitstrings without the help of a PCEC.

## **2. BIER-TE Topology management**

### **2.1. Operational model**

When a network is installed, BIER-TE is added as a service or later when it is meant to change, BFR need to be (re)provisioned. This involves a planning phase which physical adjacencies (links) should be used in the BIER-TE topology, and which virtual adjacencies (routed adjacencies) should be created and assigned bits. Ultimately this means the definition of the BIER-TE topology.

When the physical topology if the network is smaller than the possible bitstring size (e.g.: 256 bits), then this can be a simple, fully automated process. Likewise, if multiple disjointed services for BIER-TE each require active subsets of the network topology smaller than the network topology, it likewise can be simple to create a different SD (subdomain) BIER-TE topologies for each such service.

When the required network topology for a BIER-TE service exceeds the supportable bitstring size, bit-saving mechanisms can be employed as described in BIER-ARCH. Some of them such as p2p link bits or lan-bits are easily automatically calculated. Creation of virtual adjacencies (routed adjacencies) may likely best be done with operator defined policies applied to a system a system calculating the bits for the BIER-TE topology.



Ultimately, if the set of required destinations plus transit hops exceeds the size of available bitstrings after optimization, multiple BIFT == bitstrings need to be allocated to support this case. These multiple BIFT will likely need to be engineered to minimize duplicate traffic load on the network and minimize bit use. One example shown in BIER-TE-ARCH is to allocate different <SD,SI> BIFT to different areas of a network, therefore having to create one BIER-TE packet copy per required destination region, but in result having only one packet copy in each of those regions.

Provisioning / initial setup can be done manually in simpler networks or through a provisioning system. A PCEP may equally perform this function. If a PCEP is not used to perform this function, but a PCEP is used later for Flow Management, then the PCEP does of course need to also learn the BIER-TE topologies created by the provisioning system.

Unless a PCEC is used for provisioning/initial setup, YANG is likely the preferred model to install the BIER-TE topology information into the BFR. If a PCEC is used, YANG or PCEC seem to be valid choices.

When the network topology expands, bit assignments for the new parts of the topology need to be made. If expansion was not factored into the initial bit assignment plans, this can lead to the need to reassign bits for existing parts of the topology. Support for such processes could be simplified through additional topology information, for example to enable seamless switching of traffic flows from bits in one SD over to bits in another SD. This is currently not considered in this document.

## **2.2. BIER-TE topology model**



```

<BFR> BIFT information:
  Instance: "configured", "operational",
           "learned-configured", "learned-operational" (pce, igp)
  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 in this BIFT)
  Ingres-groups: (list of) string (1..16 bytes)
                (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 - BitStringLength)
    BitIndex: 1...BSL
    BitType(/Tag): "unassigned",
                  (if unassigned, must have no adjacencies)
                  "unique", "p2p", "lan", "leaf", "node", "flood",
                  "group"
                  (more BitTypes defined in text below)
  Names: (list of 0 or more) string (1..16 bytes)
        (for BitTypes that require it)
  List of 0 or more adjacencies:
    (The following is the list of possible types of adjacencies,
     as defined in BIER-TE-ARCH with parameters)
    local_decap:
      VRFcontext: string (TBD)
    forward_connected:
      destination-id: ip-addr (4/16 bytes, router-id/link-local)
      link-id: ifIndex Value (connecting to destination)
      boolean: DNR (Do Not Reset)
    forward_routed:
      destination-id: 20 bit (SID), 4 or 16 bytes (router-id)
      TBD: path/encap information (e.g: SR SID stack)
  ECMP:
    list of 2 or more forward_connect and/or
    forward_routed adjacencies

```

Figure 2: BIER-TE topology information

The above picture shows informally the data model for BIER-TE topology information. <BFR> is a domain-wide unique identifier of a BFR, for example the router-id of the IGP (if an IGP is used). Every <BFR> has a "configured" instance of the BIFT information for every BIFT configured on it. This configuration could be created from legacy models, a YANG model, PCEP, or other means.

Every <BFR> also has an "operational" instance of the BIFT information. If the BFR has nor "learned-configured" / "learned-operational" information, then the "operational" instance is just a



copy of the "configuration" instance, but would take additional local information into account. For example, if resource limits do not allow to activate configured BIFT. Or when bits in the BIFT point to interfaces/adjacencies that are down, this could potentially also be reflected in the operational instance. While the "configuration" instance is read/write, the operational instance is read-only (from NMS or PCEC).

To calculate paths/bitstrings through the topology without the help of a PCEC, a BIFT would need to know the network wide BIER-TE topology. This topology consists of the "operational" BIFT informations of the BFR itself plus the "learned-operational" BIFT information from all other BIER-TE nodes in the network plus the underlay routing topology information, for example from an IGP. When an IGP is used, the "learned-operational" information of another BFR is simply learned because the BFRs are flooding this information as IGP information.

In the absence of any IGP, or the desire not to use it to distribute BIER-TE topology information, an NMS or PCEC could collect the "operational" BIER-TE topology information from BFRs and distribute it to BFIR to enable them to calculate BIER-TE bitstrings autonomously.

The operational instance of the topology information can depend on the presence of an IGP. If the adjacency of a bit in the BIFT is configured to use a nexthop identifier that has to be learned from an IGP, such as a Segment Routing SID or a router-ID, then the operational instance (as well as distributed learned-operational ones) would indicate that such an adjacency is non-operational if the BFR could not resolve this nexthop information. Forward\_connected adjacencies do not require a routing underlay, but just link-local connectivity.

Some information elements in the BIER-TE topology information is metadata to support automatic consistency checking of learned topology information which permit to prohibit use of adjacencies that would not lead to working paths or worst case could create loops. The same information can also be used to auto-configure some adjacencies, specifically routed adjacencies, allowing to minimize operator work in case BIFT topology information is not auto-created from an NMS/PCEP but through manual mechanisms, but also to automatically discover mis-wirings and avoid them to be used.

The semantic of BitType and Names are described in conjunction with consistency checking and autoconfiguration in the following sections.





### **2.3. Consistency checking**

The BitType and associated Name or Names for the bit are intended to support automated consistency checking and different reactions. an NMS can for example discover misconfiguration or miscablings and alert the operator. BFIR can likewise discover misconfiguration when the "configured" and "operational" instances of BFR are distributed via the IGP and are therefore available as "learned-configured" and "learned-operational" on the BFIR. The BFIR can then fr example stop using those misconfigured bits in any bitstrings it calculates and further escalate (e.g.: overlay signaling) unreachability of any BFER (or inability to calculate paths supporting required TE features).

"Unique" bits doe not require a name, but the <SD,SI> bit in question must only have an adjacency on one BFR. If it shows up with adjacencies on more than one BFR, this is an inconsistency.

"p2p" bits need to be the same bit on both BFR connected to each other via a subnet, and must be pointing to each other via "forward\_connected" adjacencies. A "p2p" bit needs to have one Name parameter unique in the domain - for example constructed from concatenating the IfIndex of both sides. Note that the actual subnet does not need to be p2p, a BFR can have multiple bits across a multiaccess subnet, one for each neighbor.

Not listed in the above picture, but a "remote-p2p" could be a BitType when a bidirectional adjacency between two remode BFR using forward\_routed adjacencies.

A "leaf" bit is the one shared bit in a <SD,SI> bitstring assigned to the "local\_decap" adjacency on all leaf BFER. Leaf BFER do not need a separate bit. See BIER-TE-ARCH. If more then one "lead" bits are used in an <SD,SI> across the domain that is an inconsistency - waste of bits.

A "node" bit is associated with a Name that follows a standardized form to identify a node - e.g.: its router-id. On a non-leaf BFER, this bit can only have one local\_decap adjacency on the node indicated itself. On a leaf BFER, the "node" bit must be assigned to adjacencies on one or BFR that connect to the indicated BFER. Other configurations (or wirings) are a misconfiguration.

A "lan" bit indicates a bit for a LAN, as discussed in BIER-TE-ARCH. It must have one domain wide unique name. It must only be used by BFR connecting to the same subnet with a set of forward\_connected adjacencies pointing to the other BFR on that subnet. Disabling the use of a "lan" bit either on a BFIR when sending packets, or even more son on the actual BFR connecting to a subnet and recognizing



inconsistent BIER-TE topology configuration for it - is the most important automatic function to avoid mis-routing of BIER-TE packets. The looping will be also stopped because bits are reset when packets traverse the paths, or ultimately by TTL, but neither mechanism can provide as specific OAM information about what went wrong than recognizing inconsistencies via the IGP.

TBD: flood bit, DNR (like lan bit, but more complex).

Consistency checking may happen directly during configuration as well as later during rewiring/remot changes of topology.

In general, the operational instance of the BIER-TE topology are relevant to topology consistency checking (as they are for path calculations). For example, future extensions may actually introduce some form of node/BFR redundancy where different BFR are configured for the same bits, but only one at a time is actively using a bit, and therefore announcing it in the operational instance of the BIER-TE topology.

#### **2.4. Auto-configuration**

For subnets, the actual adjacency to the neighbor on a link may not actually be configured explicitly, but only the interface. Discovery of the neighbor via the IGP would result in a complete working adjacency for a bit, and that adjacency would show then in the operational instance - while the configured instance would only show an incomplete adjacency and the bit that was configured for the adjacency. The Name parameter can be used in configuration to lock in the BFR that is expected to be on the other side of a subnet interface. If that node is not the one actually connected, the adjacency in the operational instance would not be completed.

When a "p2p" BitType is used, but the bit is configured inconsistently on both sides of a p2p link, an autoconfiguration mechanism may be specified to select which of the two bits should be used (e.g.: bit number configured on the higher router-id peer). This could help to auto-correct a configuration mistake, but it does of course not recover the inconsistently configured bit directly, it just ignores it.

When a "lan" or "flood" BitType is configured, likewise auto-configuration can be done to overcome misconfigurations. TBD: more details.

Most importantly, configuration of routed adjacencies can create most need for network-wide consistent configuration. This can be automated with the proposed "group" bittype.



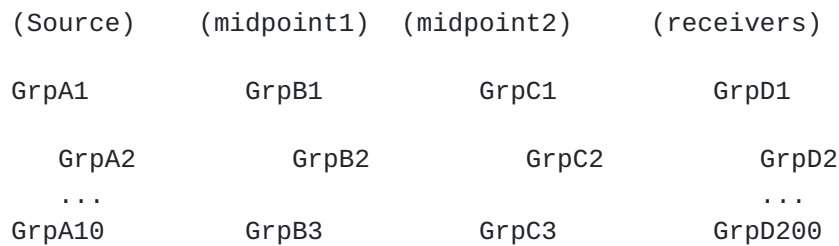


Figure 3: Group BitType use

The typical set of forward\_routed adjacency is to allow steering of BIER-TE packets through a sequence of one or more members of a hop-group, load-balancing across them for TE reasons. In the above picture, those paths would start from a BFIR in GrpA and go via one (or more) nodes in GrpB, then GrpC and then BFER (GrpD).

To half-automate the setup of such loose hops, each member of GrpC would for example be configured with one unique bit of BitType "group" and the Name parameter would be set to "GrpB". Each midpoint1 BFR would "GrpB" in the list of strings for the BIFT Ingres-Group parameter. When such a BFR discovers (e.g.: via the IGP) a BFR "learned-operational" bit of BitType group with a name "GrpB" (and no adjacency!), then that midpoint1 BFR would create an adjacency in its "operational" instance, pointing to the announcing BFR with a "forward\_routed" adjacency.

The saving through such group BitTypes is therefore that the bit had only to be configured on one node (the receiver side of the forward\_routed adjacency), but would be configured on any number of ingres BFR for the adjacency. In the above picture, the benefit would be biggest if forward\_routed adjacencies were used from Source to midpoint1, because the number of Sources is potentially largest (e.g: as shown in the picture 10 BFIR in Source group).

### 3. Flow Management

#### 3.1. Operational / Architectural Models

Once a BIER-topology is active in a network, it can be used to pass BIER-TE packets. Typically this also requires the provisioning of some routing overlay because today, all applications defined for BIER today are classical SP PE-PE application where some customer traffic is mapped to SP traffic via PE-PE "overlay" signaling.

Applications in future environments such as industrial control or IoT may result in different overlay signaling. Even native end-to-end BIER-TE from application stacks is possible, but has so far not been defined.



Overlay signaling is currently out of scope of this document.

### **3.1.1. Overprovisioning**

In the "overprovisioning flow management" model, the network operator is responsible to engineer the available network resources, BIER-TE Topology and applications generating BIER-TE flows such that the required resources can be guaranteed without contention - and potentially without the help of either PCEP or IGP, but simply using provisioning to configure BFIR and overlay signaling to determine active destinations.

Overprovisioning is the most control/signaling lightweight approach and currently the standard approach in most enterprises and service provider for IP multicast traffic.

For example: An ISP with a ++40Gbps network and a comparable small amount of high-value muticast traffic requiring in aggregate less than 5 Gbps can easily carry all of that multicast traffic across any available path. This is especially easy when the majority of traffic is best effort traffic (such as Internet traffic). In that case, the multicast traffic would be carried in a traffic class that is overprovisioned, for example with 6 Gbps guaranteed on every link. Calculated BIER-TE bitstrings would for example be used to reduce cost of multicast distribution (e.g.: steiner tree calculation), use disjoint paths (in conjunction with EF), or simply load-balance across all available non-ECMP paths. Overprovisioning flow management is traditional in most SP networks (core/edge/access) for IP multicast traffic and requires no additional signaling.

The overprovisioning flow management model is one that likely would request for (only) a YANG model to provision the BIER-TE topology.

### **3.1.2. PCEC**

In the PCEC based flow management model, a PCEP determines (calculates) the (flow-id,<SD,SI>,bitstring) for a traffic flows and signals this to the BFIR sourcing the flow (its BFR-ID is part of the flow-id). If the flow was not statically defined, then this step would be preceeded with the BFIR requesting the resources for the, indicating the requested resources as well as the set of destinations. The destinations could be indicated as BFR-ID or (likely easier for the BFIR) by their unique identifiers in unicast routing (e.g.: router-ID). The bitstring returned by the PCEP would include not only engineered paths to all these destinations, but those paths could also be disjoint paths, carrying the traffic twice towards each destination and merging them via the EF function. The BFIR could be fully agnostic to these PCEP choices.





One of the core benefits of using BIER-TE forwarding is the ability to change the bitstring on a per-packet basis to re-route traffic by setting different transit bits, or to quickly add/delete destinations. When the BFIR should be empowered to perform any of these functions without the need for help by the PCEP, then the PCEP needs to provide additional information back to the BFIR.

If a BFIR has for example an OAM capability to determine without the help of a controller that a path has failed (too much packet loss on destination, signalled back to BFIR), and dual-transmission is not desired (due to double resource usage), then the PCP and BFIR could co-operate on a path-protection scheme in which the PCEP provides for flows not one, but two bitstrings, one being the backup path which is used by the BFIR when it discovers via OAM loss on the currently used path. This approach can extremely reduce the need to rely on controller help during failures.

When the destinations for a particular flow can potentially change over time, this can often be faster and more efficiently signalled directly via the overlay signaling to the BFIR instead of going through the PCEP. To support this mode of operations, the BFIR could request from the PCEP not simply the current set of destinations for a flow, but instead the maximum superset of receivers and request per-destination information. The PCEP would then return not just one bitstring, but one bitstring per destination (BFER). The BFIR would simply OR the bitstrings for all required destinations for each packet to create the final bitstring for that packet. Note that this description is of course on a per- $\langle SD, SI \rangle$  (aka: per BIFT) basis. Destinations using different BIFTs require always different BIER-TE packets to be sent by the BFIR.

#### **3.1.2.1. per-flow QoS - policer/shaper/EF**

In the PCEP based resource management model, it is up to the PCEP to determine how explicit resource reservations should be managed, e.g.: whether or how it tracks resource consumption. The BIER-TE forwarding plane itself does not support per-flow state with the exception of EF, which would usually be a function enabled on BFER.

Likewise, per-flow policer and/or shaper state may be a useful optional feature that the PCEP should be able to request to be enabled on a BFIR to ensure that the traffic passed by the BFIR into the BIER-TE domain does not overrun resources available. In the simplest case, such a shaper/policer could simply reflect the resources indicated by the BFIR in its request to the PCEP.

Per-flow policer/shaper or EF may need to be explicitly instantiated by BFIR/BFER. Instantiation of the Policar/Shaper on the BFIR can



happen as a function of the PCEP signaling to the BFIR, but instantiation of the EF would also require signaling of the PCEP to the BFER(s) for flows. Note that EF could also be instantiated on any midpoint BFR, so the PCEP would need to know the BIER-TE topology including where EF is considered and manage it through appropriate signaling.

Note that it is unclear yet, if EF implementations could or should be implemented with or without the need for explicit instantiation, the BIER-TE-EF-OAM document allows both options. Even in the absence of explicit signaling, per-flow Policer/Shaper and EF are limited resources and PCEP should keep track of how much of these resources are allocated and available for future flows. Like other path resources, exhaustion may require PCEP failure to allocate responses or other mitigating options.

#### **3.1.2.2. DiffServ QoS**

The only resource management that could be expected to exist in the BIER-TE domain hop-by-hop would be DiffServ QoS. As outlined in the above overprovisioning resource management model, it can serve as an easy method for lightweight resource management, and as soon as the network intends to use more than one such DiffServ codepoint across different BIER-TE flows, the PCEP should likely be able to understand and manage the DiffServ assignments of BIER-TE flows and signal the selected codepoint back to the BFIR.

#### **3.2. BIER-TE flow model**



```

BIER-TE traffic flow (change) request (from BFIR):
  Flow-control-ID: <identifier>
  Ingres BFIR of flow: (IGP router-id ?!)
  Destination-ID: set of BFER identifiers (IGP router-id ?!)
  extended-reply-required (boolean)
  Requirements:
    TSPEC (bandwidth, burst size,...)
    resilience: dual-transmission with EF
    shared-group: name

BIER-TE traffic flow reply/command (to BFIR):
  Flow-control-ID: <identifier>
  Ingres Policer/Shaper parameters (applies to each BIFT)
  Set of 1 or more BIFT:
    <SD, SI, BSL>
    BFIR-ID, entropy  (form together flow-ID)
    Bitstring
    QoS, TTL,

BIER-TE traffic flow extended reply/command (to BFIR):
  Flow-control-ID: <identifier>
  Ingres Policer/Shaper parameters (applies to each BIFT)
  Set of 1 or more BIFT:
    <SD, SI, BSL>
    BFIR-ID, entropy  (form together flow-ID)
    QoS, TTL
    List of 1 or more destinations
      Destination-ID, Bitstring

BIER-TE traffic flow command (to BFER):
  Flow-control-ID: <identifier>
  Ingres BFIR of flow: BFIR-ID (in BIER-TE packet header)
  Set of 1 or more BIFT:
    <SD, SI, BSL>
    BFIR-ID, entropy  (form together flow-ID)
    EF parameter (window size etc..)

```

Figure 4: Flow request/reply/commands

The above picture shows an initial abstract representation of the data models for the different type of request/replies discussed in the previous section between PCEC and BFIR (and in one case BFER).

The Flow-control-ID identifies the managed object itself: a flow to be sent from one BFIR to a set of BFER with some TE requirements, which ultimately may require BIER-TE packets for one or more BIFT.



BFIR and BFER need to be identified in the request in a form not specific to the bits of BIFT, so the PCEP can select the appropriate BIFT(s) to use. The above picture assumes the router-id of BFIR and BFER are appropriate.

The request includes TE requirements, including (something like a) TSPEC for bandwidth, burst-size or the like, whether or not dual-transmission via PREF is required, and if the resource used are to be shared across multiple flows, then the name of a shared group. One example of sharing would for example be a video-conference where the speaker transmits video, every speaker requests/allocates a BIER-TE flow from the PCEP, but the resources for those flows are of course shared (only one flow active at a time).

The reply from the PCEP lists the BIFTS/packets that must be sent by the BFIR to reach the desired destinations as well as any other BIER-TE packet header fields relevant <SD,SI,BSL>, BFIR-ID, entropy, QoS, TTL. Beside the BIER-TE packet header, the parameters for the policer and/or shaper to be used by the BFIR are signalled back.

The extended reply does not provide simply the bitstring to use for each BIFT, but instead lists the bitstrings required for each destination so that (as described above), the BFIR can simply add/delete destinations on a packet-by-packet basis OR'ing those bitstrings.

Finally, a command to BFER is required to instruct the creation of EF state in case this can not be done automatically.

#### **4. Security Considerations**

TBD.

#### **5. IANA Considerations**

This document requests no action by IANA.

#### **6. Acknowledgements**

TBD.

#### **7. Change log [RFC Editor: Please remove]**

00: Initial version.





## 8. References

- [I-D.eckert-bier-te-frr]  
Eckert, T., Cauchie, G., Braun, W., and M. Menth,  
"Protection Methods for BIER-TE", [draft-eckert-bier-te-frr-02](#) (work in progress), June 2017.
- [I-D.huang-bier-te-encapsulation]  
Huang, R., Eckert, T., Wei, N., and P. Thubert,  
"Encapsulation for BIER-TE", [draft-huang-bier-te-encapsulation-00](#) (work in progress), March 2018.
- [I-D.ietf-bier-bier-yang]  
Chen, R., hu, f., Zhang, Z., dai.xianxian@zte.com.cn, d.,  
and M. Sivakumar, "YANG Data Model for BIER Protocol",  
[draft-ietf-bier-bier-yang-03](#) (work in progress), February 2018.
- [I-D.ietf-bier-isis-extensions]  
Ginsberg, L., Przygienda, T., Aldrin, S., and Z. Zhang,  
"BIER support via ISIS", [draft-ietf-bier-isis-extensions-09](#) (work in progress), February 2018.
- [I-D.ietf-bier-ospf-bier-extensions]  
Psenak, P., Kumar, N., Wijnands, I., Dolganow, A.,  
Przygienda, T., Zhang, Z., and S. Aldrin, "OSPF Extensions  
for BIER", [draft-ietf-bier-ospf-bier-extensions-15](#) (work  
in progress), February 2018.
- [I-D.ietf-bier-te-arch]  
Eckert, T., Cauchie, G., Braun, W., and M. Menth, "Traffic  
Engineering for Bit Index Explicit Replication (BIER-TE)",  
[draft-ietf-bier-te-arch-00](#) (work in progress), January  
2018.
- [I-D.thubert-bier-replication-elimination]  
Thubert, P., Eckert, T., Brodard, Z., and H. Jiang, "BIER-  
TE extensions for Packet Replication and Elimination  
Function (PREF) and OAM", [draft-thubert-bier-replication-elimination-03](#) (work in progress), March 2018.
- [RFC8279] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A.,  
Przygienda, T., and S. Aldrin, "Multicast Using Bit Index  
Explicit Replication (BIER)", [RFC 8279](#),  
DOI 10.17487/RFC8279, November 2017,  
<<https://www.rfc-editor.org/info/rfc8279>>.



[RFC8296] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Tantsura, J., Aldrin, S., and I. Meilik, "Encapsulation for Bit Index Explicit Replication (BIER) in MPLS and Non-MPLS Networks", [RFC 8296](https://www.rfc-editor.org/info/rfc8296), DOI 10.17487/RFC8296, January 2018, <<https://www.rfc-editor.org/info/rfc8296>>.

Author's Address

Toerless Eckert  
Futurewei Technologies Inc.  
2330 Central Expy  
Santa Clara 95050  
USA

Email: [tte+ietf@cs.fau.de](mailto:tte+ietf@cs.fau.de)

