Reliable BIER with Peer Caching

Yoann Desmouceaux, Juan Antonio Cordero Fuertes, Thomas Heide Clausen

Reliable BIER with Peer Caching

Yoann Desmouceaux, Juan Antonio Cordero Fuertes, Thomas Heide Clausen

***Abstract*—BIER (Bit-Indexed Explicit Replication) alleviates the operational complexities of multicast protocols (associated to the multicast tree and the incurred state in intermediate routers), by allowing for source-driven, per-packet destination selection, efficient encoding thereof in packet headers, and stateless forwarding along shortest-path multicast trees. BIER per-packet destination selection enables efficient reliable multicast delivery: packets not received by a subset of intended destinations can be efficiently BIER-retransmitted to only that subset. While BIER-based reliable multicast exhibits attractive performance attributes, relying on source retransmissions for packet recovery may be costly – even unnecessary, if topologically close peers are able to provide a copy of the packet.**

Thus, this paper extends the use of reliable BIER multicast to allow recovery also from peers, using Segment Routing (SR) to steer retransmission requests through a set of potential (local) candidates, before requesting retransmissions from the source as a last resort only. A general framework is introduced, which can accommodate different policies for the selection of candidate peers for retransmissions. Simple (both static and adaptive) policies are introduced and analyzed, both (i) theoretically and (ii) by way of simulations in data-center-like and real-world topologies. Results indicate that local peer recovery is able to substantially reduce the overall retransmission traffic, and that this can be achieved through simple policies, where no signalling is required to build a set of candidate peers.

***Index Terms*—Multicast, Reliable multicast, Bit-Indexed Explicit Replication (BIER), Segment Routing (SR), Policies, Performance evaluation.**

I. Introduction

As the size and complexity of Data-Center Networks (DCNs) [1] and Content Distribution Networks (CDNs) [2] grow, efficient multicast distribution of content becomes increasingly desirable [3], [4], [5]. Multicast protocols were never widely deployed in the Internet, due to their intrinsic complexity and their requiring state in intermediate routers [6]. Protocols such as PIM (Protocol Independent Multicast) [7], which operate by clients "subscribing" to multicast traffic flows by sending *join* messages, offer a best-effort data delivery service. Several protocols have been proposed, which offer different data delivery services – notably, reliability. NORM (Negative-acknowledgement-Oriented Reliable Multicast) [8], among others, uses sequence numbers in data packets to detect packet losses, and negative acknowledgements (NACKs) which trigger a multicast transmission of the missing packet to the multicast group. Other protocols [5], [9] use unicast retransmissions to those destinations having missed a packet.

Bit-Indexed Explicit Replication (BIER) [10] is a multicast protocol that removes the need for flow-state in intermediate routers. Intermediate routers replicate and forward packets over the interfaces providing shortest paths (according to unicast routes) to the specified destinations, as illustrated in figure 1. BIER-based reliable multicast [11] enables the source to detect and efficiently retransmit (with BIER), missing packets to their requesting destinations.

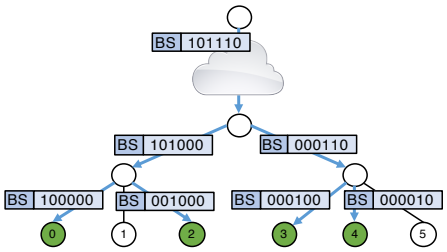


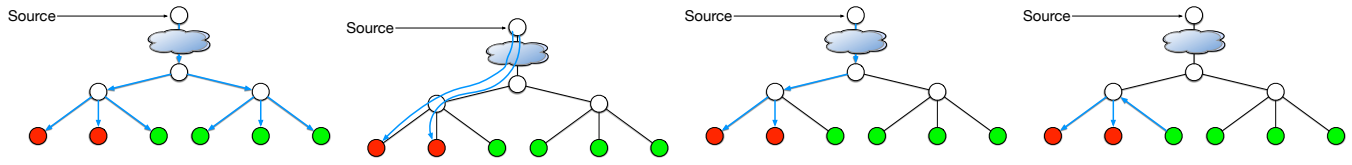
Figure 1. Example of BIER packet forwarding

"Local recovery" may, if available, be preferable to retransmissions from the source, and numerous reliable multicast protocols have proposed such an approach [5], [12], [13]. This has benefits in terms of Quality-of-Experience (*e.g.,* when the source and the destination having lost a packet experience a substantial round-trip delay), or when there are links towards the source whose usage should be limited (*e.g.,* so as to not overload the source, or for traffic engineering or economic reasons). Several usage scenarios can be listed, in which it is interesting to provide local recovery rather than source-based retransmissions:

- live [14], [15] or linear [16] multicast video delivery, for which it is crucial to provide low-latency services to the end users, and where retransmissions from sources behind costly (for instance, transcontinental) links can be detrimental to the packet reception delay;
- multicast pre-placement of content in CDNs [3], [4], [17], for which it can be costly (in terms of link usage, and in terms of global transmission speed) to rely on remotely-located sources to perform retransmissions;
- pushing of software updates to multiple machines in data-center networks [18] (or more generally, of arbitrary files to distributed storage systems relying on replication [19]) from exterior sources, for the same reasons as above – and especially because data centers feature sub-millisecond machine-to-machine latencies, making them interesting candidates for local retransmissions.

Therefore, a desirable reliable multicast protocol would be able to minimize the amount of state in the routers and retransmission traffic, while allowing for locally-emitted retransmissions.

The authors are with École Polytechnique, 91128 Palaiseau, France, emails: {yoann.desmouceaux, juan-antonio.cordero-fuertes, thomas.clausen}@polytechnique.edu. Y. Desmouceaux is also with Cisco Systems Paris Innovation and Research Laboratory (PIRL), 92782 Issy-les-Moulineaux, France, email: ydesmouc@cisco.com.



(a) Multicast retransmission by source. (b) Unicast retransmissions by source. (c) BIER retransmission by source. (d) BIER retransmission by peer.

Figure 2. Comparison of different reliability mechanisms. In this example, red clients are assumed to have missed reception of a packet and sent a NACK. With "standard" multicast retransmissions [8], the source re-floods the whole tree as a result (a). With BIER retransmissions [11], the source re-floods only the subset of failing destinations (b). With peer-based BIER retransmissions introduced in this paper, the NACKs are sent to a peer that had cached the packet, which then floods the failing destinations, spanning a smaller tree (c).

A. Statement of Purpose

The purpose of this paper is to extend *reliable BIER* [11] to allow a destination, having lost a multicast packet, to request local retransmission thereof from local *peers* (*i.e.,* which are topologically close, and part of the destination set for the multicast flow) which may have successfully received a copy thereof – before requesting a retransmission from the source. This is achieved by (i) each destination caching successfully received packets for a small amount of time, and (ii) destinations detecting a packet loss sending a NACK through an ordered set of peers (that might offer a retransmission) followed by the source, by way of Segment Routing (SR) [20].

The advantages of this approach are threefold: (i) the use of peer-based recovery reduces the number of retransmissions from the source, (ii) the use of SR allows a destination to generate a single NACK for a lost packet, which ultimately and automatically will be forwarded to the source if no local retransmissions are possible, and (iii) BIER-based retransmissions (from the source or from any peer) reduce the overall traffic by avoiding both unnecessary duplicate unicast retransmissions across a link close to the source, and multicast floods across the entire multicast tree.

B. Related Work

Different approaches to reliable multicast exist, most of which are *not* based on BIER, and will be reviewed in section I-B1. The proposal in this paper is built on edge-caching and cooperative content management – which has some notions in common with Information Centric Networking (ICN), discussed in section I-B2. BIER, and *reliable BIER*, as well as Segment Routing, will be explored in further details in section I-B3 and in section I-B4.

1) Reliable Multicast: Reliable multicast protocols assume the existence of a multicast tree (*e.g.,* built using PIM, or similar), to which are added (i) detection, (ii) reporting, and (iii) loss recovery via packet retransmission. Depending on the notion of *reliability*, the mechanisms differ on the intended set of receivers, on transmission requirements, and on available trade-offs for each purpose [21]. Since strict reliability is hard to achieve and does not scale, a weaker notion of *semi-reliability*, associated to time-limited efforts (retransmissions, NACKs) before dropping a packet, is sometimes used in multicast systems with large number of destinations.

Loss estimation and recovery can be handled exclusively at the source, as in XTP [22], or through receiver detection and source retransmission, as in RMP [23], SRM [24] or NORM [8], [25]. Local recovery has been explored by several approaches, either from designated intermediate devices, as in PGM [26], from designated receives, as in RMTP [27], or from a backup overlay of destinations, as in RDCM [5].

2) Information-Centric Networking (ICN): ICN [28] is a paradigm where content is stored as named *data packets*, and where users send *interest packets* requesting named data packets. As data packets are only identified by names, caching by intermediate routers is possible, and a packet needs only be delivered once per interface, corresponding to a previously-emitted interest. ICN thus shares properties with reliable multicast protocols, by enabling recovery of data from nearby routers.

What is proposed in this paper differs from ICN in that it (i) assumes push-based multicast applications and (ii) does not rely on routers performing caching, by offloading this task exclusively to the set of destinations – *i.e.,* requires no extensive modifications to routers nor to applications.

3) Bit-Indexed Explicit Replication (BIER): BIER [10] is a multicast protocol allowing for delivery of a packet to a group, with each destination explicitly indicated by the source. Each possible destination is assigned an index; a *bitstring*, where the i-th bit is set if and only if the i-th possible destination is intended to receive the packet, is included in each packed header.

On receiving a packet, a BIER router maps intended destinations to IP addresses, consults its routing table to identify over which interfaces the packet should be forwarded to reach the destinations indicated in the bitstring, and makes one replica of the packet for each of these interfaces. Finally, just before transmitting the (replica of the) packet over an interface, the included bitstring is updated, leaving only those bits corresponding to destinations reachable via this interface, set (figure 1). No per-source or per-flow state is required in any router – only a (standard unicast) routing table and the static mapping between bit index numbers and IP addresses must be present.

BIER can be used to provide *reliable BIER*, a reliable multicast service with per-packet (rather than per-flow) granularity [11]. Schematically, when a destination detects a packet to be lost, it sends a NACK towards the source – which collects NACKs for this packet for a small amount of time, recording the destinations requesting retransmission. When that time expires, it uses BIER to send the retransmission to exactly the set of destinations which sent a NACK (figure 2c). This both

avoids flooding the whole original multicast tree (figure 2a), and prevents duplicate retransmissions over the same link (as would be the case, for unicast retransmissions to each destination, figure 2b).

4) Segment Routing (SR): SR [20] is an architecture allowing packets to traverse a source-specified, explicit, ordered set of interconnections (called *"segments"*), before reaching their final destination. A segment can be associated with different functions, from the simple (forwarding a packet to the next segment) to arbitrarily complex (*e.g.,* handing over a packet to a Virtual Network Function for processing). In IPv6 Segment Routing [29], segments are identified by IPv6 addresses, and carried in an IPv6 Extension Header [30]. After a segment is processed, the processing router will replace the current IP destination address of the packet with the available next segment – and forward it using regular unicast IP forwarding. This enables transparent delivery of segment routed packets across non-SR-capable routers.

C. Paper Outline

The remainder of this paper is organized as follows. Section II gives a birds-eye view of the proposed extension to *reliable BIER*, and section III a detailed specification of how BIER is used to construct a reliable multicast framework which can accommodate diverse policies for selecting peers for local retransmissions. Section IV introduces basic taxonomy and example peer selection policies. Section V provides a theoretical analysis and discussion of performance and cost trade-offs for each of these policies, which are experimentally evaluated by way of network simulations in sections VI-VIII: section VI introduces the characteristics of the simulation environment, and sections VII and VIII describe and discuss the main performance results over both a data center topology and a real ISP topology. Finally, section IX concludes this paper.

II. Overview: Reliable BIER with Peer Caching

In this paper, the *reliable BIER* mechanism introduced in [11] is extended to support recovery from *peers*. Rather than sending a NACK directly to the source to request retransmission of a lost packet, this paper proposes that a NACK be first sent through an ordered set of peer(s), each of which might be able to provide a retransmission if they have a cached copy of the lost packet (figure 2d). Retransmission from the source is solicited as a "last resort". As with *reliable BIER*, peers can, of course, aggregate NACKs, before performing a BIER-based retransmission. This locality in retransmissions is expected to reduce delays, as well as to reduce the load on the source, and on its egress links [13], [21].

A. Segment Routing Recovery

Requesting retransmission from an ordered list of peers, followed by the source, is done by sending a NACK using Segment Routing (an "SR-NACK"). Each segment will trigger an action which is: (i) if the peer is unable to perform the retransmission, forward the packet to the next segment; (ii) if the peer is able to perform the retransmission, stop forwarding the segment and perform a BIER retransmission (figure 2d).

This is illustrated in figure 3: in (a) an SR-NACK is received by a peer, which is able to satisfy the retransmission; in (b) an SR-NACK is received by a peer, which is not able to satisfy the retransmission request, and where the next segment is another peer – which, then, is able to satisfy the retransmission request; in (c) neither of the two peers receiving the SR-NACK is able to satisfy the retransmission request, and the SR-NACK is therefore forwarded to the source of the multicast packet.

The combined approach thus consists of the source performing an initial BIER transmission of a multicast packet. Destinations receiving the packet may (in addition to processing it) cache it for a short amount of time for possible peer-retransmission. A destination detecting a packet loss (*e.g.,* by receiving a subsequent packet belonging to the same multicast flow) will construct an SR-NACK, containing a number of peers followed by the source. A peer receiving an SR-NACK for which it is able to offer a retransmission will behave as if it was the source in *reliable BIER*: collect NACKs for this packet for a small amount of time, record the destinations requesting retransmission, and use BIER for retransmitting the packet to exactly the set of destinations from which an SR-NACK was received, when the timer expires.

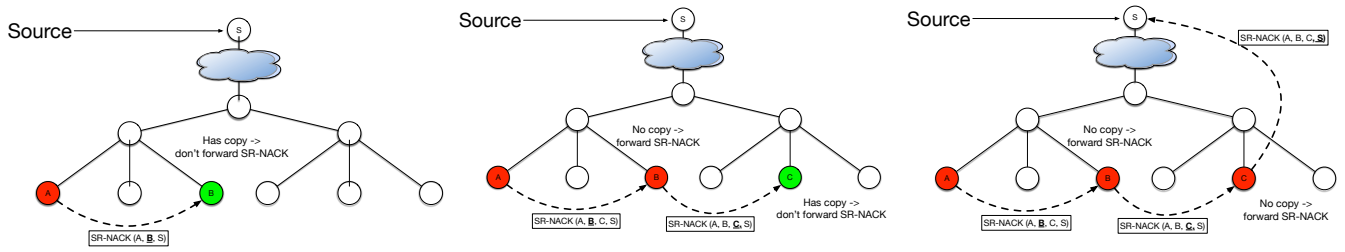
B. Peer Caching with Peerstrings

This recovery mechanism is, of course, agnostic to the manner in which the set of candidate peers is chosen. If the network operator has instrumented the network in such a way that some peers are "better" candidates for retransmissions (*e.g.,* they are more likely to have cached packets, they are behind less costly or less lossy links, etc.), destinations can be administratively configured to send NACKs to those – with the drawback of requiring instrumentation and configuration. Thus, this paper introduces a modification to the BIER forwarding plane, allowing destinations to learn about candidate peers.

To this end, an additional bitstring is introduced in BIER headers, henceforth denoted a *peerstring*. This peerstring is set so as to allow a destination, detecting a packet loss, to identify potential peers from which a retransmission can be requested. A-minima, the peerstring is empty, which defaults to requesting retransmission from the source, as in *reliable BIER*. A-maxima, the peerstring contains all destinations (*i.e.,* is a copy of the bitstring as inserted by the source) — and any peerstring in between these two extremes is valid.

With the goal of encouraging locality in retransmissions, one simple policy is that, for a given destination, the peerstring contains the set of destinations that share the same parent as itself. This is illustrated in figure 4: when destination 0 receives a packet, the peerstring has a bit set for all destinations, which have the same parent as itself (*i.e.,* destination 2). When a router forwards a *reliable BIER* packet over an interface i, it must, in addition to updating the bitstring for that interface, update the peerstring – essentially setting the peerstring to the union of the bitstrings for all other outgoing interfaces.

An extension to this principle is to include two peerstrings in a data packet received by a destination d: one peerstring



(a) SR-NACK to peer+source, peer retransmission. (b) SR-NACK to two peers+source, second peer retransmission. (c) SR-NACK to two peers+source, source retransmission.

Figure 3. SR-based recovery scenarios: SR-NACK sent by A. A red destination indicates that it is not able to satisfy the retransmission request, whereas a green destination indicates that it is.

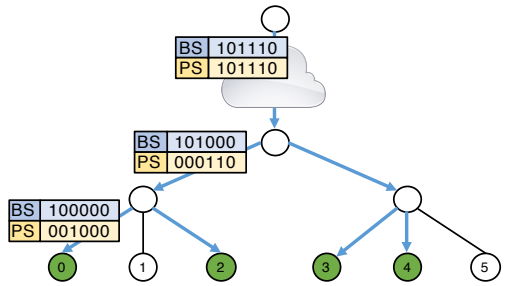


Figure 4. Example of peerstring operation

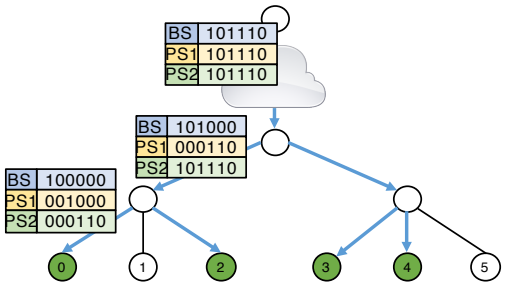


Figure 5. Example of double-peerstring operation

indicating destinations with the same parent as d, and a second peerstring indicating destinations with the same grandparent (but, not the same parent) as d – as illustrated in figure 5. Thus, this affords more flexibility, but at the expense of more per-packet overhead. This mode of operation will be referred to as the *two-peerstrings mode*.

III. Specification

In the proposed BIER extension, the BIER header defined in [31] is extended to include an additional bitstring, the *peerstring*, denoted $PS1$ and described in section II-B. When BIER operates in *two-peerstrings mode*, the header will also include a second peerstring, $PS2$. This header is included in all multicast data packets, and is processed by each intermediate router. Each multicast data packet also includes a reliable BIER *header*, defined in [11], which conveys flow identifiers and sequence numbers, and which allows detecting lost packets in a flow. This header only carries end-to-end semantics, and is not processed by intermediate BIER routers.

A. Source Operation

Packet transmission: The source adds a reliable BIER header to each multicast data packet, containing a flow identifier and a sequence number, as well as a standard BIER header [31] extended with peerstrings, as described above. The bitstring BS in the BIER header contains the set of destinations receiving packets from within the flow. Also, the peerstring $PS1$ is set to BS and, if included, $PS2$ is also set to BS, as illustrated in figures 4 and 5. The multicast source also caches a copy of each sent packet during a time interval Δt^s_{cache}.

Packet retransmission: When receiving an SR-NACK for a given packet, a source starts a timer Δt^s_{agg}, during which it collects (potential) further SR-NACKs for the same packet from other destinations. Upon expiration of this timer, a retransmission of the packet is performed, with the set of destinations that have sent a NACK as the BIER bitstring.

B. Intermediate Router Operation

BIER bitstring processing: BIER packets are processed according to the BIER specification [10]. Only bits corresponding to destinations for which the shortest-path is via interface i are preserved in the bitstring contained in multicast data packets transmitted over that interface i.

Peerstring processing: Upon receipt of a *reliable BIER* packet with a bitstring BS^{in}, and before forwarding it over interface i (with outgoing bitstring BS^{out}_i), a router must update $PS1$ and, if included, also $PS2$. In *two-peerstrings* mode, first $PS2^{out}_i$ is set to $PS1^{in}$. Then, the peerstring $PS1^{out}$ is set to the OR of the bitstrings sent over all other interfaces (formally, $PS1^{out}_i \leftarrow \cup_{j \neq i} BS^{out}_j$)[1]. This way, the $PS1$ sent over each interface contains the set of those other destinations, to which this router has sent a copy of the packet. Note that the use of bitwise operators to compute peerstrings makes it a simple operation to be implemented in hardware.

C. Destination Operation

Packet reception: Upon receipt of a packet by a destination, the packet can be cached for a duration of Δt^p_{cache} (for potential retransmission to a peer). The peerstring(s) of the packet are inspected. The included $PS1$ is used for updating

[1] $PS1^{out}_i \leftarrow BS^{in} \setminus BS^{out}_i$ is an equivalent way of proceeding, as $\cup_{j \neq i} BS^{out}_j = BS^{in} \setminus BS^{out}_i$ holds as an invariant.

the set $\mathcal{P}_1$ of "local" peers. If included, $PS2$ is used for updating the set $\mathcal{P}_2$, of second-most local peers.

Packet loss: Upon detection of loss of a packet in a given flow (*e.g.,* by receiving a packet in the same flow, whose *reliable BIER* header indicates a higher sequence number), a destination builds an SR-NACK packet. The SR-NACK contains a *reliable BIER* header with the flow identifier and the sequence number of the requested packet, and its SR header segment list is set to $(p_1, \ldots, p_n, s)$, where the p_i's are peers selected from $\mathcal{P}_1$ and $\mathcal{P}_2$, and where s is the source. Different policies can be used for deciding which peers to include from $\mathcal{P}_1$ and $\mathcal{P}_2$, and in which order, as described in section IV.

While sending the SR-NACK, the destination starts a Δt^d_{retry} timer. When this timer expires, if the no retransmission is received, the same SR-NACK is retransmitted – until either the missing packet is received, or a retransmission request limit R_{lim} is reached – after which recovery is aborted. (If semi-reliability is unacceptable, R^d_{lim} must be set to ∞.)

Packet retransmission: Upon receipt of an SR-NACK, a peer inspects the reliable BIER header of the SR-NACK and extracts the flow identifier and sequence number. If a cached copy of the requested packet is available, it is scheduled for retransmission; otherwise, the SR-NACK is forwarded to the next entry in the segment list (*i.e.,* to the next peer or, ultimately, to the source).

Retransmissions from peers use the same mechanism as those from the source: a timer Δt^p_{agg} is used to collect other NACKs before sending the copy to the set of destinations which have NACKed the packet.

IV. Peer Selection Policies

As introduced in section III-C, upon missing a packet, a destination will build an SR-NACK with peer(s) extracted from $\mathcal{P}_1$ and $\mathcal{P}_2$. The framework introduced in this paper is agnostic to the policy used for selecting those peers. To illustrate this, the remainder of this section suggests *examples* of *simple* policies for selection of peers to be included in SR-NACKs: random selection of peers, clustered selection of peers, and a simple adaptive (statistically-driven) policy.

A. Random Peer Selection

This policy builds an SR segment list (p, s) where s is the source, and p is a peer randomly selected from $\mathcal{P}_1$. Randomly selecting peers from $\mathcal{P}_1$ may increase locality of retransmissions, but rarely allows aggregation of multiple retransmissions into a single BIER packet. As an example, if ten destinations $d_1, \ldots, d_{10}$ have the same parent (thus, for each d_i, $\mathcal{P}_1 = \{d_1, \ldots, d_{10}\} \setminus \{d_i\}$), and d_1, d_2 both detect loss of the same packet, the probability that d_1 and d_2 send an SR-NACK for this packet to the same peer in $\{d_3, \ldots, d_{10}\}$ is $1/8$.

B. Deterministically Clustered Peer Selection

This policy builds an SR segment list (p, s) such that all destinations with the same parent router (*i.e.,* all destinations with the same $\mathcal{P}_1$) select the same p. As a convenient convention, for this paper, all d will select as p the element in $\mathcal{P}_1 \setminus \{d\}$ with the highest index.

This policy generalises for *two-peerstring mode* by building a SR segment list (p_1, p_2, s). All d will select as p_1 the element in $\mathcal{P}_1 \setminus \{d\}$ with the highest index, and as p_2 the element in $\mathcal{P}_2 \setminus \{d\}$ with the highest index.

While this policy favours aggregation of local retransmissions into a single BIER packet, it does not guard against selecting an unsuitable peer, *e.g.,* a peer located behind a particularly lossy link.

C. Adaptive Statistically-driven Peer Selection

As long as the constraint that the last segment in the SR segment list for a SR-NACK must be the source is satisfied, any adaptive policy – allowing a destination to observe and "learn" which peers are good candidates from whom to request retransmissions – can be be used for selecting additional peers for inclusion.

Formally, this is an instance of the Multi-Armed Bandit problem: an agent (a destination needing a retransmission) repeatedly activates one of several casino arms (in this case: sends an SR-NACK to a peer) and collects a reward (obtains, or not, a retransmission). The goal is a policy for maximizing the expected reward [32], [33], [34].

As an illustration, a simple ϵ-greedy peer selection policy is employed: with probability ϵ ($\epsilon \ll 1$), a destination detecting a packet loss sends an SR-NACK to a random peer among the set of available peers. With probability $(1-\epsilon)$, it sends an SR-NACK to the peer from which it so far has received the highest number of successful retransmissions. The value of ϵ reflects the trade-off between *exploration* (contacting random peers and gathering statistics) and *exploitation* (requesting retransmission from the best known candidate) – between reactivity to changes and performance after convergence.

V. Policy Analysis

The impact of using peer caching and peer retransmissions, and of each of the policies introduced in section IV, can be quantified analytically. Section V-A provides a basic analysis of the benefits of local retransmissions, and section V-B derives an analytical model for the *random* and *clustered* policies. Section V-C then explores the benefits from a simple, adaptive policy. For tractability, as well as for ease of interpretation of the results, the models formulated in this section focus on a single multicast group, in a regular tree. Formulating and solving a model to derive policies applicable in the case of arbitrary topologies and multicast groups would be an interesting extension that is out of the scope of this paper. Proofs of the theoretical results presented in this section are available in the appendix.

A. Recovery Locality

The assumption in this paper is, that sending local recovery requests to a local peer is likely to be both "cheaper" than sending the request to the multicast source, and successful –

i.e., a "close" neighbour is likely to have successfully cached the requested packet. This section explores the latter of these two assumptions, by quantifying the probability distribution of the distance from a destination having not received a given packet, to the closest peer that has (and, therefore, is able to perform retransmission).

A regular tree topology is assumed, wherein inner nodes are intermediate BIER routers, and leaves are destinations. Nodes at a given depth are assumed to have the same number of children, and links at a given depth have the same loss probability. As corresponds to the operation of BIER, the root node of the tree is assumed to be the multicast source.

The tree is of height h, with node ranks indexed by their depth in the tree, from 0 (the source) to h. Similarly, links ranks are indexed from 0 (links from source to first descendants) to $h-1$. Each node at rank $i \in \{0,\dots,h\}$ has c_i children (with $c_h = 0$) and α_i is the loss probability of links at rank $i \in \{0,\dots,h-1\}$. Multicast transmission of a single packet to all leaves is considered.

Lemma 1 gives the probability that no nodes in the subtree, rooted at a given node, do not receive the multicast transmission.

Lemma 1 *The probability b_i that a multicast transmission from a node at rank $i \in \{0,\dots,h\}$ does not reach any destination (within its subtree), is:*

$$b_i = [\alpha_i + (1-\alpha_i)b_{i+1}]^{c_i} \tag{1}$$

Proposition 1 *The distribution of D,* i.e., *the shortest distance from an arbitrary destination to a destination having successfully received the multicast transmission can for $k \in \{1,\dots,h\}$ be derived from* (1)*:*

$$f_D(2k) = \left[\prod_{i=0}^{h-k-1}(1-\alpha_i)\right](\alpha_{h-k} + (1-\alpha_{h-k})b_{h-k+1})\times \times[1-[(1-\alpha_{h-k})b_{h-k+1} + \alpha_{h-k}]^{c_{h-k}-1}] \tag{2}$$

with:

$$\begin{cases} f_D(0) &= \prod_{i=0}^{h-1}(1-\alpha_i) \\ f_D(\infty) &= b_0 \end{cases}$$

In proposition 1, $f_D(0)$ corresponds to the probability that a destination has successfully received the multicast transmission.

$\bar{D}$ denotes the random variable of distance towards closest successful destination for destinations having not received the packet. The probability distribution of $\bar{D}$ can be computed as the conditional distribution of D (see proposition 1) given that the packet is not received (which has probability $1 - f_D(0)$), as shown in corollary 1.

Corollary 1 *The distribution of $\bar{D}$, the minimum distance from a destination that missed a packet and to a destination which successfully received the packet is, for $k \in \{1,\dots,h\}$:*

$$f_{\bar{D}}(2k) = \frac{f_D(2k)}{1-f_D(0)} = = \frac{\prod_{i=0}^{h-k-1}(1-\alpha_i)}{1-\prod_{i=0}^{h-1}(1-\alpha_i)}(\alpha_{h-k} + (1-\alpha_{h-k})b_{h-k+1})\times \times[1-[(1-\alpha_{h-k})b_{h-k+1} + \alpha_{h-k}]^{c_{h-k}-1}] \tag{3}$$

with:

$$\begin{cases} f_{\bar{D}}(0) &= 0 \\ f_{\bar{D}}(\infty) &= \frac{b_0}{1-f_D(0)} = \frac{b_0}{1-\prod_{i=0}^{h-1}(1-\alpha_i)} \end{cases}$$

The probabilities α_i of loss at each link appear in equations (2) and (3), allowing to reflect different topology assumptions. In some topologies, loss probabilities may, *e.g.,* be assumed negligible for links close to the source, and more significant for links close to destinations (*i.e.,* $\alpha_i < \alpha_j$ if $i < j$). Two models for rank-dependent link loss probabilities are considered:

- Linear increase (`lin`): $\alpha_i = \alpha_{\max}\frac{i}{h-1}$.
- Exponential increase (`exp`): $\alpha_i = \alpha_{\max}\frac{e^i-1}{e^{h-1}-1}$.

Figure 6 depicts the distribution of the minimum distance from a destination that missed a multicast transmission, and to a destination successfully receiving a multicast transmission, $f_{\bar{D}}(2k)$. Two tree topologies with 256 destinations are considered: (i) $h = 5$, $(c_0,\dots,c_5) = (1,2,2,2,32)$; and (ii) $h = 7$, $(c_0,\dots,c_7) = (1,2,2,2,2,2,8)$.

It can be observed that the expected shortest distance between a failing destination and a successful peer is lower for the exponential loss model (in which the loss probability for top links is lower) than for the linear loss model. In other words, as expected, if top links are less lossy, it is more likely for a failing destination to be able to recover from a close destination.

This confirms the intuition that in networks with this type of loss distribution (such as can be envisioned in data-centres, or in networks where the "last hop" is, *e.g.,* a wireless link, or a consumer grade residential xDSL), selection of local recovery peers (*e.g.,* peers that are in the same subtree or in the immediately upper subtree) should be preferred to selection of peers farther away.

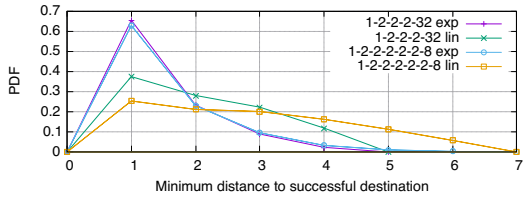


Figure 6. PDF of the recovery at $2k$ hops (conditioning to a loss).

B. Clustered and Random Policies

The properties and performance of the two simplest, static, and (to some degree) most "extreme" peer policies of section IV – random and clustered peer selection – is studied by way of two metrics: the number of *recovery successes*

Figure 7. Example with a policy X over $n = 10$ destinations, with $K = 4$ destinations (2, 5, 6 and 7) having lost a packet. Arrows indicate recovery requests (SR-NACKs): destination 2 requests recovery from peer 1; destination 5 from peer 2; and destinations 6 and 7 from peer 10; $S_X = 3$ recovery requests are successful (*i.e.,* those from 2, 6 and 7), as peers 1 and 10 have correctly received the original packet; $T_X = 2$ retransmissions are performed (from 1 and 10).

(performance) and the number of incurred *recovery retransmissions* (cost). A recovery is *successful* if a destination detecting the loss of a packet sends an SR-NACK to a peer that previously has received and cached a copy thereof. The number of *recovery retransmissions* incurring is the number of *unique* peers, which are selected for retransmission by the set of destinations which have lost that packet. This is because each selected peer will send only one BIER transmission as a response to receiving a set of SR-NACKs for the same multicast packet.

For the remainder of this section, a subtree with n destinations is considered, and recovery of one packet within this subtree is examined.

The following variables are introduced (where X denotes a particular selection policy):

- K, the number of destinations which did *not* receive the packet by way of the original multicast transmission from the source;
- S_X, the number of *recovery successes*, *i.e.,* destinations which did *not* received the packet by way of the original multicast transmission from the source, but which successfully received a retransmission from a peer, in response to an SR-NACK.
- T_X, the number of *recovery retransmissions*, *i.e.,* of unique peers which received an SR-NACK for a given multicast packet.

Figure 7 shows an example with $n = 10$ destinations, $K = 4$ destinations missing a packet, $S_X = 3$ successful requests (out of 4) and $T_X = 2$ retransmissions.

Lemmas 2 and 3 describe the probability density function (PDF) for the number of recovery successes (S_R, S_D) and for the number of retransmissions (T_R, T_D), for the random and clustered policies, respectively. For the number of retransmissions with the clustered policy, it is assumed that a peer, from receiving the first SR-NACK and until retransmission of the packet, waits a sufficiently large amount time so as to maximise the ability of aggregating retransmissions into a single BIER-transmission.

Lemma 2 (Random selection policy) *Given $K = k$ destinations ($0 \leq k \leq n$) that* did not *receive the multicast transmission from the source, the probability that, from among these k destinations, s ($0 \leq s \leq k$) will choose a peer which* did *receive the multicast transmission from the source, $Pr[S_R = s|K = k] \equiv f_{S_R,k}(s)$, is:*

$$f_{S_R,k}(s) = \binom{k}{s}\left(\frac{k-1}{n-1}\right)^{k-s}\left(\frac{n-k}{n-1}\right)^{s} \tag{4}$$

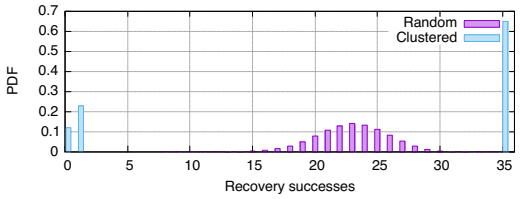


Figure 8. Number of destinations able to obtain a retransmission of a packet from a peer, when $k = 35$ out of $n = 100$ *have not* received the initial multicast transmission.

Given s recovery successes, the probability of t retransmissions ($0 \leq t \leq s$), $Pr[T_R = t|S_R = s, K = k] \equiv f_{T_R,k,s}(t)$, is:

$$f_{T_R,k,s}(t) = \binom{n-k}{t}\sum_{i=0}^{t}(-1)^{t-i}\binom{t}{i}\left(\frac{i}{n-k}\right)^{s} \tag{5}$$

Lemma 3 (Clustered selection policy) *Similar to Lemma 2, the probability that s destinations (out of $k \geq 2$ destinations that* did not *receive the multicast transmission from the source) will choose a peer which* did *receive the multicast transmission from the source, $Pr[S_D = s|K = k] \equiv f_{S_D,k}(s)$, is:*

$$f_{S_D,k}(s) = \begin{cases} \frac{n-k}{n} & \text{if } s = k \\ \frac{k}{n}\left(1 - \frac{k-1}{n-1}\right) & \text{if } s = 1 \\ \frac{k}{n}\frac{k-1}{n-1} & \text{if } s = 0 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

The probability of having t retransmissions, $Pr[T_D = t|K = k] \equiv f_{T_D,k}(t)$, is:

$$f_{T_D,k}(t) = \begin{cases} 1 - \frac{k}{n}\frac{k-1}{n-1} & \text{if } t = 1 \\ \frac{k}{n}\frac{k-1}{n-1} & \text{if } t = 0 \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

As edge cases, when $k = 1$, there is one recovery/retransmission: $S_D = T_D = 1$; and when $k = 0$, there are no recoveries/retransmissions: $S_D = T_D = 0$.

From lemmas 2 and 3 follow two key results, describing the average number of recovery successes (proposition 2) and the average number of retransmissions (proposition 3 and corollary 2).

Proposition 2 *Assuming $K = k$ destinations that* did not *receive the multicast transmission from the source, the expected number of recovery successes for the random and clustered approaches is the same, and has the value:*

$$\mathbb{E}[S_R|K = k] = \mathbb{E}[S_D|K = k] = \frac{k(n-k)}{n-1} \tag{8}$$

Proposition 3 *Assuming $K = k$ destinations that* did not *receive the multicast transmission from the source, the expected number of retransmissions for random (S_R) and clustered (S_D) policies have the following expressions:*

$$\mathbb{E}[T_R|K=k] = (n-k) - (n-k)\left(\frac{n-2}{n-1}\right)^k \quad (9)$$

$$\mathbb{E}[T_D|K=k] = \begin{cases} 1-\frac{k(k-1)}{n(n-1)} & \text{if } k \geq 1 \\ 0 & \text{if } k = 0 \end{cases} \quad (10)$$

The following corollary compares the behavior of both policies when considering that each destination has, independently, the same probability of having not received the multicast transmission from the source.

Corollary 2 *Assuming that each destination has (independently of the others) a probability $\beta \in [0,1]$ of not having received the multicast transmission from the source (*i.e., *the probability of having k destinations, that* did not *receive the multicast transmission from the source, is binomial with parameter β, and thus $Pr[K=k] = \binom{n}{k}\beta^k(1-\beta)^{n-k}$), then the expected number of retransmissions with the random policy grows linearly with the number of destinations, whereas the expected number of transmissions with the clustered policy is bounded:*

$$\begin{cases} \mathbb{E}[T_R] \sim_{n\to\infty} n(1-\beta)(1-e^{-\beta}) & = \Theta(n) \\ \mathbb{E}[T_D] \sim_{n\to\infty} 1-\beta^2 & = \Theta(1) \end{cases} \quad (11)$$

From proposition 2, both policies yield the same performance *on average*, *i.e.,* the expectation of the number of recovery successes is the same for both policies. The clustered policy achieves the same reliability as the random policy by concentrating recovery requests on a single peer, which allows aggregation of retransmissions into a single BIER retransmission. Since aggregation occurs less often in the random policy, in terms of retransmission cost, the random policy is more expensive (linear in the number of destinations vs constant), as shown in corollary 2.

This difference is only possible because random and clustered policies achieve their (equal) average performance through different probability density distributions, as shown in figure 8. In this example, it is assumed that $k = 35$ destinations out of $n = 100$ have not received the initial multicast transmission. While with the random policy there is negligible chance that no less than 15 and no more than 30 destinations are able to obtain a retransmission from a peer, the clustered policy operates on an all-or-nothing fashion: all 35 destinations will obtain a retransmission from a peer with high probability – but recovery may be mostly unsuccessful with non-negligible probability ($\sim 12\%$ for 0 successes, 23% for only one success over 35).

For the purpose of illustrating the relative performance of random and clustered policies, it is possible to consider a Service Level Agreement (SLA) commitment specifying a minimum fraction of destinations $(1-\delta)$ (with $\delta \ll 1$) being served without the need for source retransmission. Source retransmissions are unneeded when destinations receive the packet in the first BIER transmission from the source, and when the first peer recovery request is successful. Since source retransmission may lead to a substantial increase in packet latency, the previously described SLA can be reformulated in terms of reduced latency experienced by a higher fraction of destinations. The probability that a fraction $(1-\delta)$ of destinations is served without resorting to source retransmissions, under policy X, $g_{n,\beta,X}(\delta)$, is:

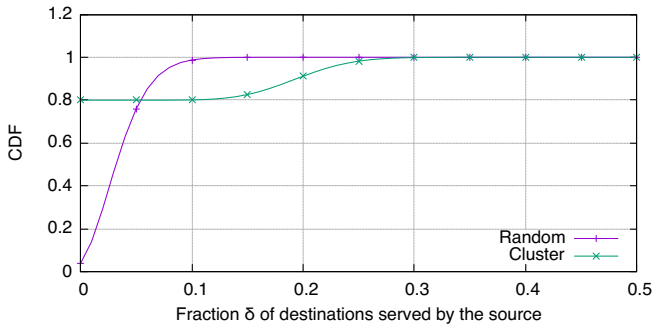


Figure 9. Probability that a fraction of $(1-\delta)$ of destinations successfully receive the packet (1) directly from the source in the first transmission, or (2) from a contacted peer – *i.e.,* excluding source retransmissions, with $n = 100$, $\beta = 0.2$.

$$g_{n,\beta,X}(\delta) = Pr[S_X > K - n\delta] = \\ = \sum_{k=0}^{n} Pr[S_X > K - n\delta | K = k] Pr[K = k] \quad (12)$$

where $Pr[S_X > x|K = k]$ can be computed from equations (4) and (6).

Figure 9 illustrates the value of $g_{n,\beta,X}$ as a function of δ, when $n = 100$, $\beta = 0.2$, and for the random and clustered policies[2]. Figure 9 can be used to help an operator decide which policy to choose. For strict SLAs where a large fractions of the destinations must be served without source retransmissions ($\delta \approx 0$), there is a higher probability that systems are compliant when using the clustered policy ($\delta < 0.05$ in figure 9). Conversely, with looser SLAs, there is a higher probability that systems are compliant when the random policy is used ($\delta > 0.05$ in figure 9), and both policies behave identically for highly-loose SLAs ($\delta > 0.25$ in figure 9). To conclude, when source retransmissions are significantly less preferable than peer retransmissions (*e.g.,* due to a substantially higher delay incurred), the clustered policy might be preferable with respect to the random policy.

C. Going Adaptive: the ϵ-Greedy Policy

For the purpose of this analysis, a binary ($c = 2$) multicast BIER transmission tree, with height $h = 7$, to illustrate the ability of the simple ϵ-greedy adaptive policy and to learn and adapt to changes in networking conditions, under two different scenarios:

- A (*static*) scenario (figures 10(a) and 11(a)), where all links are lossy ($\alpha = 0.1$), except for links between the source and destination 0 – *i.e.,* with a static "best" (ideal) peer from which to request retransmission.

[2]While the gap amplitude is dependent on the value of β (*i.e.,* lower loss probabilities at destination lead to shorter gaps, as it can be expected), the trend shown in figure 9 is invariant for values of n and β.

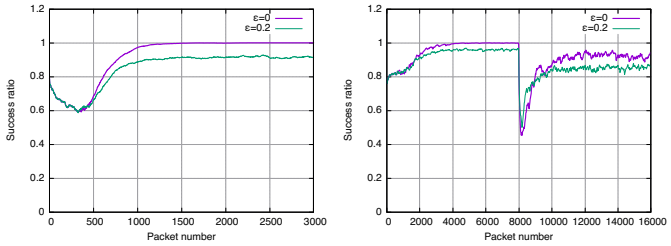


Figure 10. Success ratio (after 1st recovery) for ϵ-greedy policy, for **(a)** static and **(b)** dynamic scenarios.

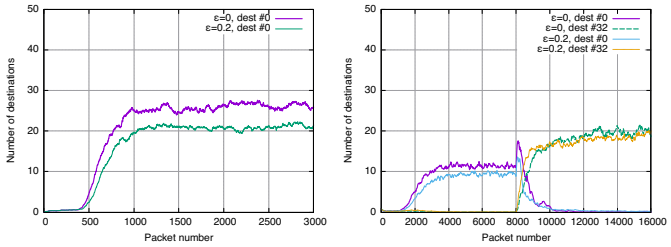


Figure 11. Number of destinations that *did not* receive the multicast transmission from the source, and which are selecting peers 0 and 32 under the ϵ-greedy policy, in **(a)** static and **(b)** dynamic scenarios.

- A (*dynamic*) scenario (figures 10(b) and 11(b)) which reflects a situation with failure of a "good" peer. Specifically destination 0 is, as in the static scenario and for the same reasons, the "best" (ideal) destination ($\alpha = 0$ for links from destination 0 and to the source) up until multicast packet # 8000, after which it becomes a bad destination behind very lossy ($\alpha = 0.4$) links. Concurrently, destination 32 is a relatively good, though not perfect, destination ($\alpha = 0.01$) during the entire duration of the flow.

It is worth to observe that, in these simulations, recoveries are *idealized*: retransmissions from contacted peers are always successful if the contacted peer holds a copy of the requested BIER packet.

Unsurprisingly, $\epsilon = 0$ (*i.e.,* choosing deterministically the destination with highest success record) achieves more steady performance than does $\epsilon = 0.2$ (*i.e.,* choosing random destinations for recovery 20% of the time) in static conditions (figure 10(a)). Using $\epsilon = 0.2$ policy, however, performance is less impacted by the failure of destination 0, and the system adapts faster to the new conditions (figure 10(b)): as shown in figure 11(b), failing destinations when using $\epsilon = 0.2$ switch to destination 32 quicker, after the failure of (former ideal) destination 0.

VI. Simulation Environment

The reliable multicast mechanism, described in this paper, has been implemented in NS-3 [35] as four components: (i) a BIER forwarding plane (as described in section III-B), (ii) a Segment Routing forwarding plane, (iii) a reliable BIER layer for a source (section III-A), and (iv) a reliable BIER layer for destinations (section III-C).

The *reliable BIER* layer at the source interfaces with the UDP socket API, to transform UDP multicast packets into BIER packets, while also caching a copy of sent packets so as to be able to retransmit them on receipt of SR-NACKs. The *reliable BIER* layer at a destination also interfaces with the UDP socket API, collecting received BIER packets before handing them (in-order) over to the UDP socket. This layer also caches a copy of received packets, so as to be able to retransmit them on receipt of a SR-NACK from a peer, and generate SR-NACKs when a packet loss is detected.

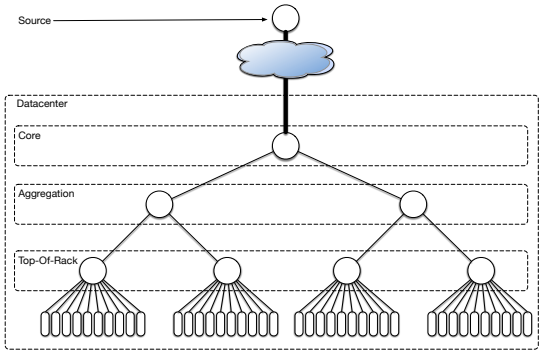


Figure 12. Datacenter topology for simulations of section VII

The parameters as defined in section III are: $\Delta t^s_{agg} = \Delta t^p_{agg} = 7\ ms$, $\Delta t^p_{cache} = 50\ ms$, $\Delta t^s_{cache} = 100\ ms$, $\Delta t^d_{retry} = 15\ ms$, $R^d_{lim} = 3$. Notably, the value of the SR-NACK retry delay is set significantly greater than the NACK aggregation delay, $\Delta t^d_{retry} \gg \Delta t_{agg}$, so that a second NACK is only sent if the source (or peer) has had the chance to send a retransmission and has failed.

A. Links and Link Loss Model

All links are point-to-point, with 1 Gbps throughput and 1 μs propagation delay, have an MTU of 1500 bytes, and are attached to interfaces with drop-tail queues of size 512 packets.

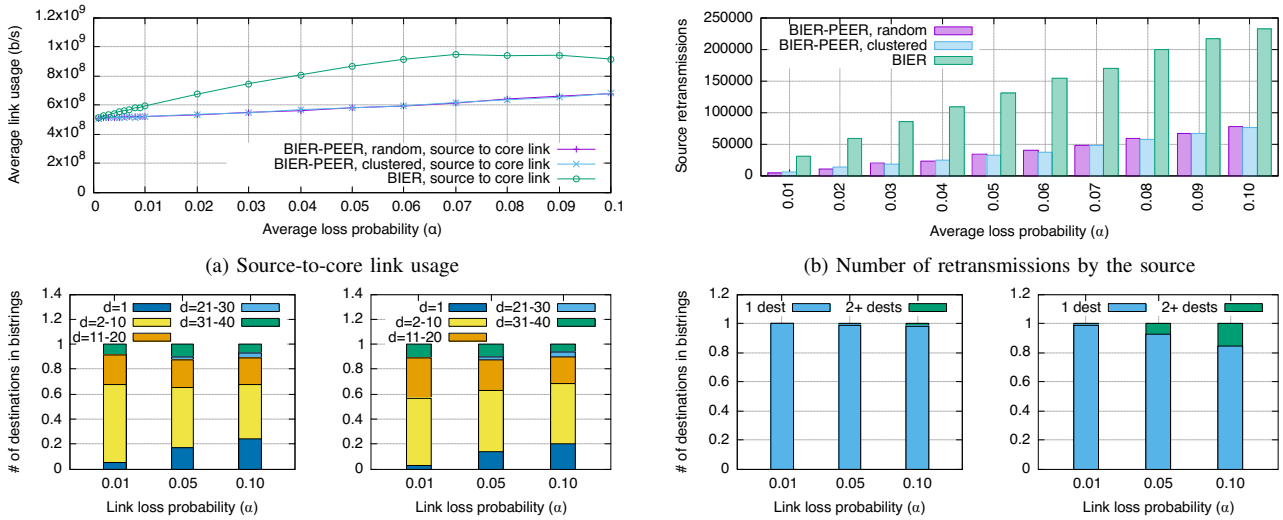
The link loss model used for all links in the simulations is a clock-based Gilbert-Elliott model [36], [37], where the probability of a successful transmission is $k = 1$ in *good* state and $h = 0.5$ in *bad* state. Transitions from *bad* to *good*, and from *good* to *bad*, are triggered with exponential clocks, of mean $1/r = 2.5\ ms$ and $1/p = \frac{(h-\alpha)}{\alpha r} = \frac{(0.5-\alpha)}{\alpha} \times 2.5\ ms$, respectively, where $\alpha \in [0, 1]$ is a parameter representing "packet loss probability".

According to [38], the probability π_B of being in *bad* state is $\pi_B = \frac{p}{p+r} = \frac{1}{1+(h-\alpha)/\alpha} = \frac{\alpha}{0.5}$, yielding an expected link loss rate of $\pi_B(1-h) + (1-\pi_B)(1-k) = 0.5\pi_B = \alpha$. This justifies using α as a parameter to tune the average packet loss probability. This loss model is only applied to multicast data packets – SR-NACKs are not subject to losses, as the path from destinations to the source is supposedly less lossy[3].

VII. Data-Center Simulations

For the purpose of evaluating the mechanism introduced in this paper, a data-center-like topology is first used.

[3]Whereas traffic from the source to the destination is bursty, and therefore likely to cause interface buffers to run full, and drop packets, traffic from destinations to the source is expected to be sparse (essentially, SR-NACKs).



(a) Source-to-core link usage

(b) Number of retransmissions by the source

(c) Distribution of the number d of destinations in bitstrings from source retransmissions: random (left) vs clustered (right)

(d) Distribution of the number d of destinations in bitstrings from peer retransmissions: random (left) vs clustered (right)

Figure 13. Clustered vs random peer selection within same subtree, and for different values of the Gilbert-Elliott link loss probability α.

A. Network Topology

The topology used in this set of simulations is as follows, illustrated in figure 12:

- a source, "outside" the data-center – reachable across a capacity constrained connection (*e.g.,* a connection incurring higher delays, which has limited throughput, or subject to a higher, congestion-induced, loss probability) is attached to a core router;
- the core router is attached to 2 aggregation routers;
- each aggregation router is attached to 2 Top-of-Rack (ToR) routers;
- each ToR router is attached to 10 servers in a rack (destinations).

In other words, the topology is a regular tree, in which successive arities are $(1, 2, 2, 10)$.

This topology has been chosen for the simulations for two reasons. First, it can be used to reason about actual data-center-specific scenarios (as introduced in section I) – *e.g.,* distributed storage, distribution of software upgrades, or pre-placement of content. Second, as BIER packets in any arbitrary topology will effectively be spread along a shortest-path tree, reasoning about a regular tree is a first step towards understanding the properties of the proposed mechanism (simulations on non-regular trees are deferred to section VIII).

It is to be noted that such a topology does not allow for evaluating what happens in multi-path-enabled data-center topologies, such as BCube [39] or DCell [40]. However, such topologies rely on source-routing to effectively transmit packets, and would thus require modifications to the BIER forwarding plane so as to properly encode the multicast tree, which is outside of the scope of this paper. Furthermore, interpreting evaluations based on these topologies would make it difficult to highlight how the mechanism introduced in this paper reacts to congestion on a link, since these topologies would tend to naturally spread congestion across all links.

B. Evaluation Objectives

The objective of using SR-NACKs for requesting retransmissions from peers, rather than directly from the source, is specifically *not* to minimize the number of transmissions, globally – but, rather, to maximize the number of retransmissions that can be satisfied locally, *i.e.,* within the data-center. Consequently, one key metric is the number of retransmissions performed by the source – corresponding to the load of the link between the source and the core router (bold link in figure 12).

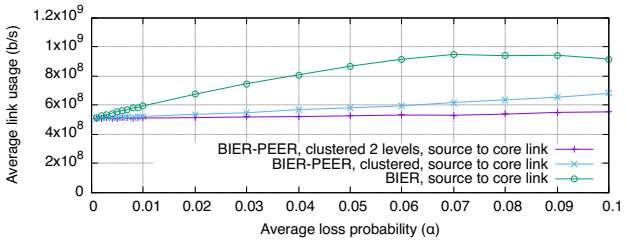
C. Static Peer Policy – One-Peerstring Mode

To baseline the benefits of locality, the two static peer selection policies introduced in section IV-A (random) and in section IV-B (clustered) are tested in one-peerstring mode: a destination, which detects that a packet has been lost, sends an SR-NACK towards first a "local" peer[4] according to the policy, then to the source. These two static peer selection policies are compared with classic *reliable BIER* without peer recovery (*i.e.,* wherein NACKs are sent directly to the source).

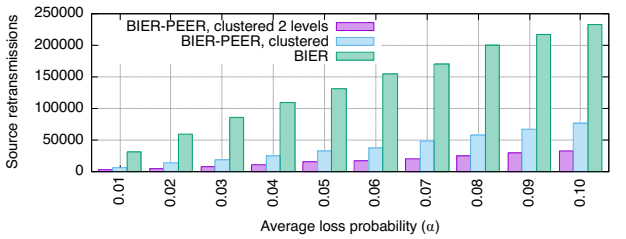
Figure 13 depicts the results of 19 four-second-long simulations using a 500 Mbps multicast flow (*i.e.,* 166673 BIER-packets generated by the source), for different values of α (see section VI) applying uniformly to all links. Figure 13a shows the link usage of the ingress link to the core router (see figure 12), averaged over the duration of the simulation. Using classic *reliable BIER*, the source performs all retransmissions, and the ingress link to the core router saturates for $\alpha \geq 7\%$. With peer-based retransmissions, even with $\alpha = 10\%$, this link remains well below saturation, with a link usage below 680 Mbps. This is detailed in figure 13a, and explained through figure 13b, which depicts the number of times a retransmission has been performed by the source.

On receiving an SR-NACK, a peer waits for Δt^p_{agg}, to allow receiving SR-NACKs from other peers, before a single

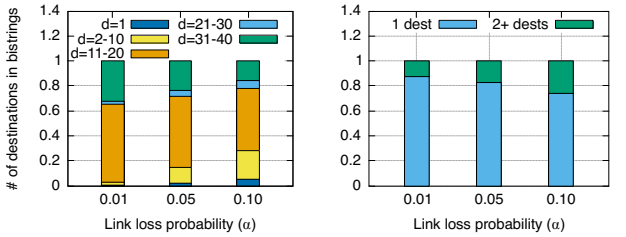
[4]A peer is considered "local" if it has the same parent – in which case, it is indicated in the received $PS1^{in}$.



(a) Source-to-core link usage



(b) Number of retransmissions by the source



(c) Distribution of the number d of destinations in bitstrings from source (left) and peer (right) retransmissions

Figure 14. Data-center topology: clustered peer selection in two subtrees vs in one subtree, for different values of the Gilbert-Elliott link loss probability α.

aggregate BIER retransmission is made. Thus, the number of destinations in the bitstring is an indicator of the ability to reduce the number of retransmissions. This is depicted in figures 13c and 13d, for retransmissions made by the source or by a peer, respectively, and for random and clustered peer selection policies. Retransmissions by the source are required when both the originator of an SR-NACK, and the selected peer, have not received a given packet – *e.g.,* when the packet was lost over the link between an aggregation router and a ToR router (see figure 12), in which case all destinations below that ToR router would need to receive a retransmission. This explains why most source retransmissions are destined to 2 to 10 destinations (figure 13c). Similarly, but to a lesser extent, a consequent number of source retransmissions are destined to between 11 and 20 destinations, which can be explained in most cases by a loss over the link between an aggregation router and the core router. Figure 13d also shows that random peer selection (expectedly) does not facilitate aggregation in peer-issued retransmissions, whereas clustered peer selection allows for some aggregation – which occurs when a packet is lost several time in the same subtree.

D. Static Peer Policy – Two-Peerstrings Mode

A variation of the clustered peer policy (section IV-B) is possible when using two peerstrings. A destination, detecting that a packet has not been received, sends an SR-NACK towards first the leftmost "local" peer, then to the leftmost second-most local[5] peer, and finally to the source. The objective is, again, to reduce the number of retransmissions needed from the source, at the cost of potentially more total (but, local) traffic.

The baseline for this approach is the *clustered peer selection* policy of section VII-C, as well as standard *reliable BIER*, thus the same topology, traffic patterns, and link loss model, are used. The simulation results for different values of α are depicted in figure 14, where figure 14a shows that the two-peerstring approach yields a further reduction in the link usage on the ingress link to the core router, due to fewer retransmissions being required by the source. In figure 14b, this reduction appears to be from $2.2\times$ less (when $\alpha = 0.05$) to $3.3\times$ less (when $\alpha = 0.02$), when comparing to the one-peerstring approach.

To understand the ability of the proposed mechanism to reduce retransmission traffic, figure 14c depicts the number of destinations in the retransmission bitstrings. Using the two-peerstring mode, retransmissions by the source are required only when neither the originator of an SR-NACK, nor its selected local peer, nor its selected second-most local peer, have received a given packet – most usually when the packet was lost over the link between an aggregation router and the core router. This explains why the majority of retransmissions are sent to between 11 and 20 destinations, and confirms a greater degree of aggregation (among 20 destinations, rather than 10) of source retransmissions, as compared to the one-peerstring mode. Furthermore, due to their ability to collect SR-NACKs from an adjacent subtree, designated peers will sometimes perform retransmissions to a whole rack (when a link from an aggregation router to a ToR router has failed), thereby also increasing aggregation as compared to the one-peerstring mode.

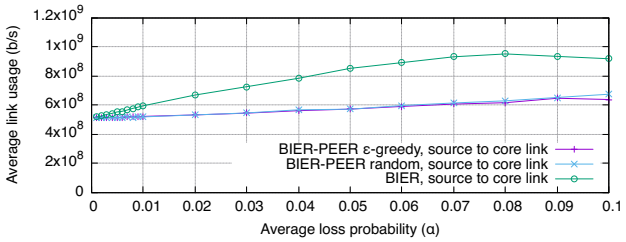
Finally, to understand the impact of the proposed mechanism in terms of incurred signaling traffic, table I reports the signaling overhead caused by SR-NACKs emitted during the simulations, for both the one-peerstring and two-peerstring mode, as well as for standard *reliable BIER*. Signaling overhead is computed in terms of packet-links, *i.e.,* SR-NACKs per traversed links (in the tree). An SR-NACK traversing n links in the tree counts as n packet-links[6]. For each mechanism and each value of α, table I displays the average signaling overhead per multicast data packet, *i.e.,* the number of (SR-NACK) packet-links needed, in average, before all destinations

Table I
SIGNALING OVERHEAD (AVERAGE SR-NACK-TRAFFIC PER DATA PACKET) FOR THE CLUSTERED PEER SELECTION POLICY

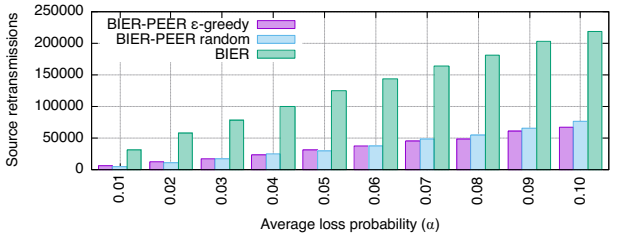
α	Reliable BIER	One-peerstring	Two-peerstring
0.01	2.65	3.23	4.44
0.05	13.0	18.1	26.9
0.10	40.4	34.2	47.7

[5]A peer is considered "second-most local" if it has the same grandparent, but not the same parent – in which case, it is indicated in the received PS_2^{in}.

[6]Given figure 12, an SR-NACK will have traversed two links if reaching the most local peer, six links if then reaching the second-most local, and ten links if then reaching the source.



(a) Source-to-core link usage



(b) Number of retransmissions by the source

Figure 15. Data-center topology with one ideal peer per rack: ϵ-greedy peer selection in subtree vs random, for different values of the Gilbert-Elliott link loss probability α.

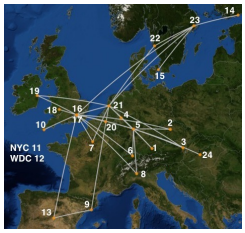
have received a new multicast dat packet from the source. Results show that signaling overhead incurred by the two-peerstring policy is approximately $1.4\times$ the overhead related to the one-peerstring policy, but this extra signaling overhead is what allows for greatly reducing the number of source-induced retransmissions (as depicted in figure 14a), thus providing better quality of service for the clients.

E. Adaptive Peer Policy

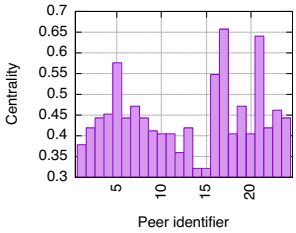
To simulate the existence of some local peers being "more suitable" than others, the link loss model is modified such that each ToR router has exactly one destination with a non-lossy ToR-to-destination link. An adaptive policy should enable all other destinations connected to that ToR router to "learn" that this peer is the "most suitable" peer for retransmissions, and therefore, when detecting that a packet has not been received, send an SR-NACK towards first this "most suitable" peer, and only then to the source. To exemplify this, and to examine the intuitions from section V-C, the ϵ-greedy policy has been implemented and tested against the *random peer selection* policy[7] (section VII-C) and standard *reliable BIER*. In order to allow for sufficient exploration, $\epsilon = 0.2$ is used.

Simulations are run for different values of α, with results depicted in figure 15. As ϵ-greedy allows directing an SR-NACK towards peers that have a greater chance of being able to retransmit a packet, these are less often forwarded to the source, reducing the number of source retransmissions (figure 15b) and the link usage of the ingress link to the core router (figure 15a). Finally, when a SR-NACK is received by a "more suitable" peer, its retransmissions are more often successful (as it is connected to its ToR router over a non-lossy link), reducing further retransmissions of that packet.

[7]The comparison is made to the random peer selection policy, rather than to any of the clustered peer selection policies. Selecting the destination with the non-lossy link for any of the clustering policies would amount to biasing in favour of that policy – selecting any other, would amount to biasing in its disadvantage.



(a) Network topology (picture from [41]).



(b) Closeness centrality

Figure 16. ISP topology for simulations of section VIII

VIII. ISP Simulations

To validate the genericness of the proposed approach with respect to the network topology, another set of simulations has been conducted, this time in a real ISP topology. As compared to section VII, the resulting spanning tree is non-regular. Use cases for reliable multicast in such topologies include video streaming of live events.

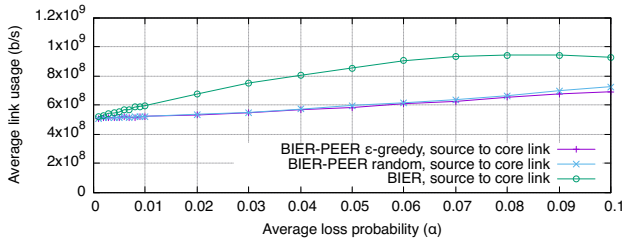
A. Network Topology

The topology used for the purpose of this set of simulations is the *BT Europe, August 2010* topology, extracted from the Internet Topology Zoo [41]. The topology is depicted in figure 16a, and consists of 24 nodes. All nodes are located in Europe, except for nodes 11 (New York) and 12 (Washington DC), which are not visible on the figure, and have each a link to node 17 (London). As this topology features remote (transatlantic) nodes, it is suited to evaluate scenarios wherein it is costly to traverse certain links. As such, the scenario studied in this section will be the transmission of a multicast stream from node 11 (New York) to all other nodes.

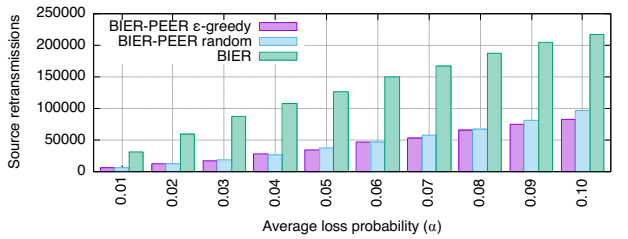
Another point of interest of using such a non-regular topology is, that it provides a more natural framework to test adaptive peer selection policies. Indeed, some destination(s) will automatically be both close to the source (thus, more prone to have received packets because there are less links in the path to the source) and close to an important number of other peers. As an illustration, figure 16b shows the *closeness centrality* of each node in the graph, defined as the inverse of the mean distance to the other nodes. It can be observed that nodes 17 (London), 21 (Amsterdam) and 5 (Frankfurt) exhibit the highest centrality, meaning that they are 'local' to a large part of destinations, and would therefore be natural candidates to act as retransmitting peers.

B. Peer Selection Policies Evaluation

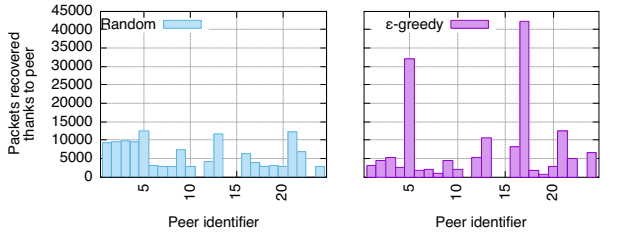
For the purpose of the simulations, a client is attached to each of the nodes, except for node 11 (NYC), to which a source is attached; the Dijkstra algorithm is first run offline to construct routing tables. As in section VII, links have 1 Gbps capacity, and the same simulation parameters are used. A multicast flow of 500 Mbps is sent from the source to each of the clients during four seconds, in 19 different simulations,



(a) Link usage between node 11 (New York) and its router



(b) Number of retransmissions by the source



(c) Histogram of peer contributions to recoveries ($\alpha = 0.10$), for random and ϵ-greedy strategies. Node 11 (NYC) is the source.

Figure 17. ISP topology: ϵ-greedy *vs.* random peer selection in subtree, for different values of the Gilbert-Elliott link loss probability α.

for different values of the average link loss probability α. The *random selection policy* with one peerstring is used to evaluate the core properties of the mechanism introduced in this paper. Additionally, the *ϵ-greedy policy* is evaluated, as a way to examine the convergence of destinations under adaptive policies towards highly-central nodes as retransmitters. Finally, as a baseline, reliable BIER as in [11] is used.

Results are reported in figure 17. Figure 17a depicts the average usage of the link between the source and the router to which it is attached. Whereas with standard reliable BIER the link is saturated for high values of α ($\alpha \geq 6\%$), usage of this link is reduced with peer-based BIER retransmissions (the worst case being 720 Mbps for $\alpha = 10\%$), showing that the proposed mechanism allows protecting the source from having to retransmit too many packets. This can be also observed on figure 17b, depicting the number of individual packet retransmissions performed by the source: the number of source retransmissions falls by a factor of at least 2.2× when using peer-based retransmissions.

With respect to the comparison between static and adaptive policies, figure 17b shows that the number of source-based retransmissions is further reduced when using the adaptive policy. This can be explained by observing, for each peer, the number of packets successfully recovered through this peer, for the static policy and the adaptive policy (figure 17c) – the figures display the distribution for the lossiest tested scenario (corresponding to $\alpha = 0.1$), so as to better visualize the differences. Whereas with the static policy the distribution of retransmitters is relatively uniform, with the adaptive policy it can be clearly observed that peers 17 and 5 (and, to a lesser extend, peer 21) contributes to most of the recoveries. This confirms that retransmissions are handled by peers with high centrality (see figure 16b).

IX. Conclusion

This paper extends the reliable multicast service based on BIER. The proposed extension uses BIER for ensuring that data traffic is forwarded over minimal shortest path trees, and SR-based NACKs for reporting losses and requesting retransmissions. SR-NACKs allow failing destinations to contact, in order, peers for (local) recovery before requesting retransmission from the source. All retransmissions are performed through BIER-enabled shortest paths.

The proposed protocol is compatible with standard BIER operation (RFC 8279 [10]): no caching is made by intermediate routers, retransmissions are regular BIER packets, and the retransmission logic is handled exclusively by sources and destinations. In addition, a lightweight extension to the BIER forwarding plane is proposed (the processing of a peerstring in the BIER header), allowing destinations to automatically learn about potential candidates from which to ask retransmissions. In absence of this extension, peer-based recoveries can still be requested if destinations are manually configured to do so. By allowing for local repair of multicast failures, the proposed mechanism limits the amount of source retransmissions, and thus their impact in terms of network traffic and delay.

The proposed framework is generic enough to accommodate a broad spectrum of policies for selection of recovery peers, including static, adaptive, and operator-defined. Example of such policies are introduced and analyzed, both analytically and by way of network simulations. Evaluation suggests that substantial benefits in terms of increasing locality of recoveries and reducing usage of costly links can be achieved with relatively simple policies.

Acknowledgements

This work was supported in part by the Cisco-Polytechnique Chaire “Internet of Everything” (http://www.internet-of-everything.fr).

Appendix

A. Proof of lemma 2

Proof:

Consider the transmission of a multicast packet to a cluster of n destinations, from among which k do not receive the packet. Each of the k non-successful destinations then selects a peer at random among $(n-1)$ peers (all peers but itself), and there are $(n-k)$ successful peers, thus this selection is successful with probability $\frac{n-k}{n-1}$. Therefore, the probability that s of the k non-successful destinations sends a NACK to a successful peer follows a binomial law with parameter $\frac{n-k}{n-1}$, yielding equation (4).

To derive equation (5), assume that there are s successful recoveries. The number of retransmissions t incurred by these

s recoveries corresponds to the number of unique elements sampled when drawing with replacement s elements from a set of $(n-k)$ elements. From [42], this is:

$$\frac{1}{(n-k)^s} \times \frac{(n-k)!}{(n-k-t)!} \times \left\{ {s \atop t} \right\}$$

where $\left\{ {s \atop t} \right\}$ (Stirling number of second kind) represents the number of ways to partition s elements into t non-empty subsets, and can be expressed as:

$$\left\{ {s \atop t} \right\} = \frac{1}{t!} \sum_{i=0}^{t} (-1)^{t-i} \binom{t}{i} i^s$$

Combining these two expressions yields equation (5), which concludes the proof. ∎

B. Proof of lemma 3

Proof:

Consider the clustered policy, whereby a failing destination sends a NACK to a designated peer p, and p itself if failing sends NACKs to another designated peer p^*. Consider the transmission of a multicast packet to a cluster of n destinations, from among which k do not receive the packet. Each of the k non-successful destinations then sends a recovery request (NACK) to p (or p^* if the non-successful destination is p itself). If the designated peer p is not amongst the k failing destinations (which happens with probability $\frac{n-k}{n}$), all the recoveries are successful, and $S_D = k$. If the designated peer p is among the failing destinations, but not its retransmitter p^* (which happens with probability $\frac{k}{n}\left(1-\frac{k-1}{n-1}\right)$), then only one recovery request is successful (the one from the designated peer p), and $S_D = 1$. Finally, if both the designated peer p and its retransmitter p^* are among the k failing destinations (which happens with probability $\frac{k}{n}\frac{k-1}{n-1}$), no retransmission request is successful, and $S_D = 0$. Combining these three cases yields equation (6).

To derive equation (7), it suffices to note that both cases $S_D = k$ and $S_D = 1$ yield one retransmission, and that $S_D = 0$ yields zero retransmissions. Thus $Pr[T_D = 1|K = k] = Pr[S_D = k|K = k] + Pr[S_D = 1|K = k]$ and $Pr[T_D = 0|K = k] = Pr[S_D = 0|K = k]$, yielding equation (7). ∎

C. Proof of Proposition 2

Proof:

Assuming that k destinations have missed the multicast packet, S_R has a binomial distribution with k samples and with success probability $\frac{n-k}{n-1}$. Thus, $\mathbb{E}[S_R|K = k] = k \times \frac{n-k}{n-1}$ by definition, which proves equation (8) for the random policy.

From equation (6), the expected value of S_D can be computed as:

$$\begin{aligned}\mathbb{E}[S_D|K = k] &= k \times \frac{n-k}{n} + 1 \times \frac{k}{n}\left(1 - \frac{k-1}{n-1}\right) \\ &= \frac{k}{n-1}(n-k)\end{aligned}$$

which proves that $\mathbb{E}[S_R|K = k] = E[S_D|K = k]$. ∎

D. Proof of Proposition 3

Proof: Assuming that k destinations have missed the multicast packet, and that s recoveries are successful, the expected number of incurred recovery retransmissions for the **random policy** is computed by averaging equation (5):

$$\begin{aligned}\mathbb{E}[T_R|S_R = s, K = k] &= \sum_{t=0}^{s} t \times Pr[T_R = t|S_R = s, K = k] \\ &= (n-k)\left[1 - \left(\frac{n-k-1}{n-k}\right)^s\right]\end{aligned}$$

Then, the average number of retransmissions with no assumption on s can be obtained by combining this result with equation (4), yielding (with $p = \frac{k-1}{n-1}$ to ease notation):

$$\begin{aligned}\mathbb{E}[T_R|K = k] &= \sum_{s=0}^{k} \mathbb{E}[T_R|S_R = s, K = k] \times Pr[S_R = s, K = k] \\ &= (n-k) - (n-k)\sum_{s=0}^{k} \binom{k}{s} p^{k-s}(1-p)^s \left(\frac{n-k-1}{n-k}\right)^s \\ &= (n-k) - (n-k)\left(\frac{n-2}{n-1}\right)^k\end{aligned}$$

For the **clustered policy**, deriving the expected number of retransmissions is done by averaging equation (7), and by noting that since the number of retransmissions is either 0 or 1, the expected value is simply the probability that there is one retransmission:

$$\begin{aligned}\mathbb{E}[T_D|K = k] &= Pr[T_D = 1|K = k] \\ &= \begin{cases} 1 - \frac{k(k-1)}{n(n-1)} & k \geq 1 \\ 0 & k = 0 \end{cases}\end{aligned}$$

concluding the proof. ∎

E. Proof of Corollary 2

Proof:

Assuming that each destination has missed the multicast packet independently with probability β, the number K of failing destinations follows a binomial distribution:

$$Pr[K = k] = \binom{n}{k} \beta^k (1-\beta)^{n-k} \tag{13}$$

For the **random policy**, it is possible to deduce the expected number of retransmissions (averaged over the failing destinations k) by combining equations (9) and (13):

$$\begin{aligned}\mathbb{E}[T_R] &= \sum_{k=0}^{n} \mathbb{E}[T_R|K = k] \times Pr[K = k] \\ &= (n - n\beta) - \sum_{k=0}^{n} \binom{n}{k} \beta^k (1-\beta)^{n-k} (n-k) \left(\frac{n-2}{n-1}\right)^k \\ &= n(1-\beta)\left[1 - \left(1 - \frac{\beta}{n-1}\right)^n\right]\end{aligned}$$

which is, if $n \to \infty$ and $\beta \neq 0, \beta \neq 1$, using Taylor's 1st-term approximation of $\left(1 - \frac{\beta}{n-1}\right)^n$:

$$\mathbb{E}[T_R] = n(1-\beta)(1-e^{-\beta}) + \mathcal{O}(1) \tag{14}$$

The same reasoning can be used for the **clustered policy**. The expected number of retransmissions becomes, by using (10) and (13):

$$\begin{aligned}\mathbb{E}[T_D] &= \sum_{k=0}^{n} \mathbb{E}[T_D|K=k] \times Pr[K=k] \\ &= \sum_{k=1}^{n} \binom{n}{k} \beta^k (1-\beta)^{n-k} \left(1 - \frac{k}{n}\frac{k-1}{n-1}\right) \\ &= 1 - (1-\beta)^n - \beta^2\end{aligned}$$

which is, if $n \to \infty$ and $\beta \neq 0$:

$$\mathbb{E}[T_D] = 1 - \beta^2 + o(1) \tag{15}$$

concluding the proof. ■

REFERENCES

[1] K. Bilal, S. U. Khan, L. Zhang, H. Li, K. Hayat, S. A. Madani, N. Min-Allah, L. Wang, D. Chen, M. Iqbal *et al.*, "Quantitative comparisons of the state-of-the-art data center architectures," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 12, pp. 1771–1783, 2013.

[2] M. Pathan and R. Buyya, "A taxonomy of cdns," *Content delivery networks*, pp. 33–77, 2008.

[3] J. Y. Kim, G. M. Lee, and J. K. Choi, "Efficient multicast schemes using in-network caching for optimal content delivery," *IEEE Communications Letters*, vol. 17, no. 5, pp. 1048–1051, 2013.

[4] J. Ni and D. H. Tsang, "Large-scale cooperative caching and application-level multicast in multimedia content delivery networks," *IEEE Communications Magazine*, vol. 43, no. 5, pp. 98–105, 2005.

[5] D. Li, M. Xu, Y. Liu, X. Xie, Y. Cui, J. W. Wang, and G. Chen, "Reliable multicast in data center networks," *IEEE Transactions on Computers*, vol. 63, no. 8, Aug. 2014.

[6] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen, "Deployment issues for the ip multicast service and architecture," *IEEE network*, vol. 14, no. 1, pp. 78–88, 2000.

[7] J. Nicholas, A. Adams, and W. Siadak, "Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)," RFC 3973, 2005. [Online]. Available: https://rfc-editor.org/rfc/rfc3973.txt

[8] B. Adamson and J. P. Macker, "Reliable messaging for tactical group communication," in *Military Communications Conference, 2010-MILCOM 2010*. IEEE, 2010, pp. 1899–1904.

[9] S. Paul, K. K. Sabnani, J.-H. Lin, and S. Bhattacharyya, "Reliable multicast transport protocol (rmtp)," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 3, pp. 407–421, 1997.

[10] I. Wijnands, E. C. Rosen, A. Dolganow, T. Przygienda, and S. Aldrin, "Multicast Using Bit Index Explicit Replication (BIER)," RFC 8279, 2017. [Online]. Available: https://rfc-editor.org/rfc/rfc8279.txt

[11] Y. Desmouceaux, T. Clausen, J.-A. Cordero, and W. M. Townsley, "Reliable Multicast with B.I.E.R." *Journal of Communications and Networks*, vol. 20, no. 2, pp. 182–197, 2018.

[12] R. Yavatkar, J. Griffoen, and M. Sudan, "A reliable dissemination protocol for interactive collaborative applications," in *Proceedings of the third ACM international conference on Multimedia*. ACM, 1995, pp. 333–344.

[13] B. N. Levine and J. J. Garcia-Luna-Aceves, "Improving internet multicast with routing labels," in *Proc. International Conference on Network Protocols (ICNP)*, 1997.

[14] J. Rückert, J. Blendin, and D. Hausheer, "Software-defined multicast for over-the-top and overlay-based live streaming in isp networks," *Journal of Network and Systems Management*, vol. 23, no. 2, pp. 280–308, 2015.

[15] J. Rückert, J. Blendin, R. Hark, and D. Hausheer, "Flexible, efficient, and scalable software-defined over-the-top multicast for isp environments with dynsdm," *IEEE Transactions on Network and Service Management*, vol. 13, no. 4, pp. 754–767, 2016.

[16] J. Choi, A. S. Reaz, and B. Mukherjee, "A survey of user behavior in vod service and bandwidth-saving multicast streaming schemes," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 1, pp. 156–169, 2012.

[17] S. Seyyedi and B. Akbari, "Hybrid cdn-p2p architectures for live video streaming: Comparative study of connected and unconnected meshes," in *2011 International Symposium on Computer Networks and Distributed Systems (CNDS)*. IEEE, 2011, pp. 175–180.

[18] S. James and P. Crowley, "Fast content distribution on datacenter networks," in *2011 ACM/IEEE Seventh Symposium on Architectures for Networking and Communications Systems*. IEEE, 2011, pp. 87–88.

[19] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. Long, and C. Maltzahn, "Ceph: A scalable, high-performance distributed file system," in *Proceedings of the 7th symposium on Operating systems design and implementation*. USENIX Association, 2006, pp. 307–320.

[20] C. Filsfils, N. K. Nainar, C. Pignataro, J. C. Cardona, and P. Francois, "The segment routing architecture," in *2015 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2015, pp. 1–6.

[21] J. W. Atwood, "A classification of reliable multicast protocols," *IEEE network*, vol. 18, no. 3, pp. 24–34, 2004.

[22] W. T. Strayer, B. J. Dempsey, and A. C. Weaver, *XTP: the Xpress Transfer Protocol*. Redwood City, CA: Addison-Wesley, 1992.

[23] B. Whetten, T. Montgomery, and S. M. Kaplan, "A high performance totally ordered multicast protocol," in *Proc. Dagstuhl Seminar on Distributed Systems*, 1994.

[24] S. Floyd, V. Jacobson, S. McCanne, C.-G. Liu, and L. Zhang, "A reliable multicast framework for light-weight sessions and application level framing," *ACM SIGCOMM Computer Communication Review*, vol. 25, no. 4, pp. 342–356, 1995.

[25] C. Bormann, M. J. Handley, and B. Adamson, "NACK-Oriented Reliable Multicast (NORM) Transport Protocol," RFC 5740, 2009. [Online]. Available: https://rfc-editor.org/rfc/rfc5740.txt

[26] J. Gemmell, T. Montgomery, T. Speakman, and J. Crowcroft, "The pgm reliable multicast protocol," *IEEE network*, vol. 17, no. 1, pp. 16–22, 2003.

[27] S. Paul, K. K. Sabnani, J. C.-H. Lin, and S. Bhattacharyya, "Reliable multicast transport protocol (rmtp)," *IEEE Journal on Selected Areas of Communications*, vol. 15, no. 3, pp. 407–421, 1997.

[28] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proc. CoNEXT'09*, 2009.

[29] C. Filsfils *et al.*, "SRv6 Network Programming," Internet Engineering Task Force, Internet-Draft draft-filsfils-spring-srv6-network-programming-03, Dec. 2017, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-filsfils-spring-srv6-network-programming-03

[30] S. Previdi *et al.*, "IPv6 Segment Routing Header (SRH)," Internet Engineering Task Force, Internet-Draft draft-ietf-6man-segment-routing-header-08, Jan. 2018, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-6man-segment-routing-header-08

[31] I. Wijnands, E. C. Rosen, A. Dolganow, J. Tantsura, S. Aldrin, and I. Meilik, "Encapsulation for Bit Index Explicit Replication (BIER) in MPLS and Non-MPLS Networks," RFC 8296, Jan. 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc8296.txt

[32] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, pp. 235–256, 2002.

[33] V. Kuleshov and D. Precup, "Algorithms for the multi-armed bandit problem," *Journal of Machine Learning Research*, vol. 1, pp. 1–48, 2000.

[34] J. Velmorel and M. Mohri, "Multi-armed bandit algorithms and empirical evaluation," in *Proc. ECML'2005*, 2005, pp. 437–448.

[35] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," *Modeling and tools for network simulation*, pp. 15–34, 2010.

[36] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell Labs Technical Journal*, vol. 39, no. 5, pp. 1253–1265, 1960.

[37] E. O. Elliott, "Estimates of error rates for codes on burst-noise channels," *The Bell System Technical Journal*, vol. 42, no. 5, pp. 1977–1997, 1963.

[38] G. Hasslinger and O. Hohlfeld, "The gilbert-elliott model for packet loss in real time services on the internet," in *Proc. 14th GI/ITG Conference on Measurement, Modelling and Evaluation of Computer and Communication Systems (MMB 2008)*, Mar. 2008.

[39] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "BCube: a high performance, server-centric network architecture for modular data centers," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 63–74, 2009.

[40] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "DCell: a scalable and fault-tolerant network structure for data centers," in *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4. ACM, 2008, pp. 75–86.

[41] "Internet Topology Zoo," Feb. 2017. [Online]. Available: http://www.topology-zoo.org

[42] A. F. Mendelson, M. A. Zuluaga, B. F. Hutton, and S. Ourselin, "What is the distribution of the number of unique original items in a bootstrap sample?" *arXiv preprint arXiv:1602.05822*, 2016.