Preferred Path Routing (PPR) Graphs - Beyond Signaling Of Paths To Networks

Toerless Eckert, Yingzhen Qu, Uma Chunduri
Huawei – Future Networks
Santa Clara, California, USA
{firstname.lastname}@huawei.com

***Abstract*— Preferred Path Routing (PPR) is an innovative architecture to signal explicit paths and per-hop processing including QoS from computation engines to network nodes leveraging distribution mechanisms of existing network routing protocols. PPR supports a wide range of existing forwarding planes including IP and Segment Routing. Through these mechanisms, PPR supports more lightweight, scalable and widely applicable support for high precision network services than prior signaling mechanisms including the ReSource ReserVation Protocol with Traffic Engineering (RSVP-TE).**

This paper describes the next fundamental step for the PPR architecture, with the signaling of PPR Graphs instead of only point to point PPR Paths. This extension provides better scalability and efficiency both for signaling, but even more so for the resulting forwarding and QoS processing entries created in network nodes. For any-to-any connectivity of N nodes, PPR Graphs can achieve the same O(N) scalability of forwarding entries as distributed routing protocols (IGPs), compared to $O(N^2)$ in RSVP-TE or PPR point to point Paths. PPR Graphs therefore enable future networks to support high precision services much more broadly – independent of the chosen forwarding plane.

Keywords—graphs, paths, scalability, routing protocols, preferred path routing, next generation networking architecture.

I. INTRODUCTION

[1] introduces Preferred Path Routing (PPR) paths, a novel signaling architecture for optimized traffic paths in networks with a centralized Path Computation Engine (PCE) [5]. PPR Paths intends to overcome limitations of current solutions:

Segment Routing (SR) [2] does not have the explicit notion of paths on every node along the path, therefore it does not support high-touch per-path hop-by-hop functions such as monitoring, accounting, QoS (Policing, Shaping, Buffering) or other processing. This is the direct result of SR's goal to be lightweight by using only in-packet processing instructions, and primarily source-routing via Segment-Identifiers (SID).

RSVP-TE [7] does establish per-path, per-hop state, but it does so with a high-overhead and therefore slow hop-by-hop per-flow state processing signaling mechanism that is very feature rich, and results in slower than PPR processing, more overhead and lower scalability.

RSVP-TE has also has not been defined to support the variety of forwarding planes used in today's L3 networks but only MPLS.

PPR Paths close these gaps. Through the use of reliable in-network distribution of PPR Paths, first via commonly used routing protocols, it provides a lightweight signaling architecture of hop-by-hop path processing state for a variety of forwarding planes including Segment Routing.

One key limitation of PPR Paths as introduced in [1] is its exclusive reliance on point-to-point paths and the resulting scale limitation: A network with N nodes and full-mesh of PPR Path connectivity between them requires O(N^2) forwarding states.

Due to the PPR signaling scheme, it can easily support more advanced graph types than point-to-point and achieve more scalable forwarding and processing options. Defining these graph types, their encoding and resulting forwarding plane scalability as well as use-case benefits is the contribution to the PPR architecture described in this paper.

II. PPR OVERVIEW, BACKGROUNG AND ARCHITECTURE

This section summarizes the key PPR architecture components laid out in [1]. PPR is intended to support any forwarding plane. So far, it is defined to support the currently most widely deployed forwarding planes of L3 networks: IPv4, IPv6, Segment Routing with MPLS (SR-MPLS) [3] and Segment Routing with IPv6 (SRv6) [4].

As shown in Figure 1, a PCE learns the complete network topology and traffic engineering parameters of links for example via the BGP-LS protocol [11]. From the topology and path requirements (from operator and/or application control), the PCE calculates optimized paths across the network topology, typically from an ingress Provider Edge node (PE) to an egress PE across zero or more Provider (Core) nodes (P).

The PCE encodes this path into a PPR Path object and adds path properties such as QoS parameters (e.g., guaranteed bandwidth, peak bandwidth and latency/jitter). It then sends the object to a (any) node in the network, from where it is distributed to all nodes. Distribution can be via existing Interior Gateway Protocols (IGP) such as Open Shortest Path First (OSPF) or Intermediate System to Intermediate System (IS-IS) or in future potentially faster mechanisms. The network nodes in the IGP

area/domain (both PE and P) examine each object and establish the necessary processing state as described in the PPR Path. At minimum this is a forwarding entry with a next-hop determined from the PPR Path and appropriate QoS for scheduling and/or shaping (buffering).

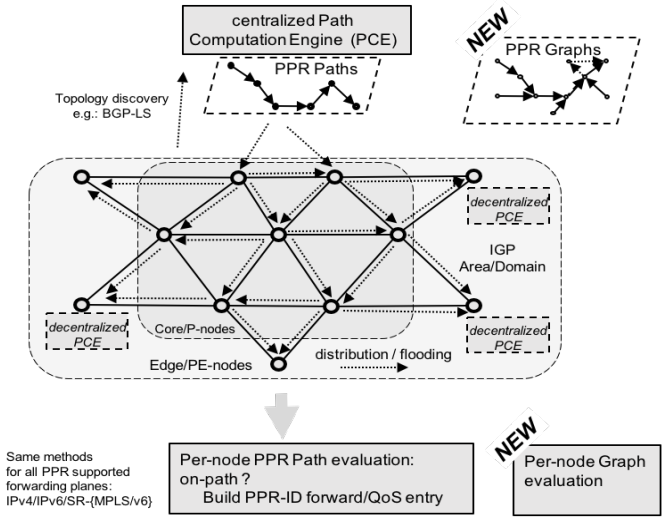

Fig. 1. PPR architecture and new PPR Graphs (decentralized PCE see VI.B)

A PPR Path consists of a sequence of Path Description Elements (PDE), each one identifying a node along the path, starting with the sender PDE and finishing with the destination PDE. Each PDE contains an identifier of the node. These are called PDE-IDs and their format depends on the desired forwarding plane: IP4 or IPv6 address of the node for IPv4/IPv6 forwarding, or a Node-Segment Identifier (Node-SID)/Global-adjacency-SID for SR.

The PPR Path has a PPR Identifier (PPR-ID), which indicates the forwarding entry for the PPR Path on every hop. It needs to be unique per PPR Path and depends also on the forwarding plane chosen: IPv4/IPv6 address or SID assigned to the destination. From the received PPR Path object, the nodes along the path determine that they need to build forwarding state for a particular PPR Path because they recognize their PDE-ID, and then establish an IPv4/IPv6/SID forwarding entry to the next PDE along the path for the PPR-ID.

The core innovative concept of PPR Paths is the architectural approach to let control components create complete path objects that are distributed by the network. This may sound counterintuitive because most nodes receiving a PPR Path object do not need to create any forwarding state for a PPR Path, because they are not PDEs on the path. Practical network design though shows that the parsing of data structures in control plane CPUs of network nodes is orders of magnitude faster than the actual propagation of the information and the establishment of forwarding plane entries.

PPR simply leverages and extends the playbook of Link-State Routing IGPs (OSPF/IS-IS). In prior signaling approaches, such as ReSource Reservation Protocol (RSVP [6]), and its Traffic Engineering variant (RSVP-TE [7]), the signaling method is hop-by-hop with high per-hop propagation delay because each hop needs to operate a per-flow state-machinery before propagating the message. In PPR, the distribution can support a larger number of paths and low propagation delay. In effect, all PDE (hops) on a path can (in optimized implementations) almost simultaneously establish the PPR Path state in accelerated forwarding planes.

This basic PPR architecture is not changed by this work. Instead, this paper extends the architecture by the introduction of more flexible type of graphs than point-to-point paths, as suggested also by Figure 1.

III. From PPR Paths to PPR Graphs

PPR Graphs are designed to scale with O(k*N) or even O(k) PPR-IDs and forwarding entries. Along with PPR Graphs, this paper also provide solutions for other issues, including:

(1) Fragmentation of PPR object encodings (TLVs) to support message limitations in distribution mechanisms, especially in the IS-IS protocol. See IV.A.

(2) A mechanism for grouping of PPR Graphs (policy-groups) to more easily orchestrate PPR Graphs to the traffic on PE nodes that should use them. Ssee IV.C.

(3) More intelligent utilization of the PPR Graph object information than simple nexthop forwarding extraction, as shown via the PPR Graph QoS functions in IV.E.

More than just the sum of these features, PPR Graphs permit a completely new paradigm for the mapping of responsibilities between the distributed nodes of the network and (more intelligent but complex) centralized PCE components: Complex/NP-hard path calculations can be removed from the distributed nodes, while still maintaining the benefits of reliable, scalable distribution of information from the IGPs as well as equally scalable, but more flexible forwarding plane entries as possible with IGPs.

IV. PPR Graphs

A. Encoding of PPR Graphs

As Figure 2 shows, a pre-existing point-to-point PPR Path encodes for a particular PPR-ID a PDE list where the first node is implicitly the sender for all traffic using this PPR-ID and the last node is the destination.

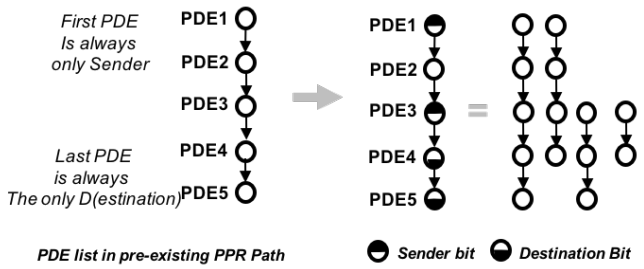

Fig. 2. Sender and Destination bits

In PPR Graphs, PDE include two new bits: sender and destination. In Figure 2 PDE1 and PDE3 are senders and PDE4 and PDE5 are destinations. This is a compressed way to represent the four PDE lists shown to the right. Multiple senders

are used in PPR Trees. Multiple senders and destinations are used in PPR Forests, see IV B.

Figure 3 shows on the left a tree graph. A tree has one destination indicated by the destination bit of its PDE. It is the root of the tree. All leaves must, and any intermediate nodes may be senders on the tree as indicated by the sender bit of their PDE. To represent a graph in a PPR Graph object, it is decomposed into PDE lists called branches. Unless the last PDE of a branch is the destination PDE of the tree itself, it must be a PDE that is also a PDE in another branch. These PDE are where branches connect/merge to form the tree. In Figure 3, these are C1 connecting Branch 3 with Branch 1, C3 connecting Branch 4 with Branch 2 and C2 (the destination) connecting Branch 1 and Branch 2.

When a node receives a PPR Tree, it simply needs to sequentially parse all branches to discover if it is on the tree and determine the forwarding entry it needs to establish from the branch where it is not the last PDE. For example, C1 determines from branch 1 that it needs to set up a forwarding entry towards C2 (and other processing such as QoS). A node determines that it is the destination of the tree from the destination bit and the fact that it is the last PDE in any branch (C2 in the example).

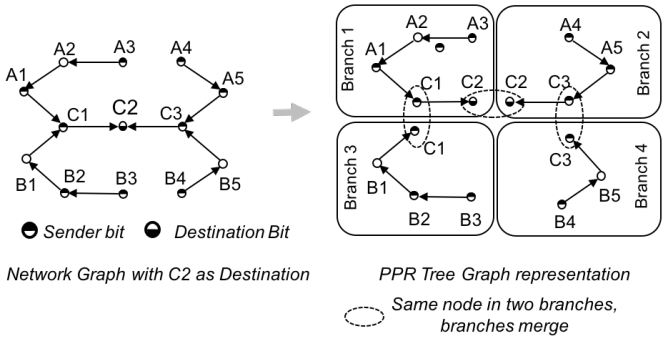

Fig. 3. Composition of Graphs from Branches

Not all transit nodes on a tree need to be senders. In Figure 3, B1 and B5 are not senders. They will still establish the forwarding/processing entries for this trees PPR-ID to C2, but they will not use this tree to reach C2 when sourcing traffic themselves. Instead, they would use another tree for sourcing traffic themselves to C2 as explained below in IV.C.

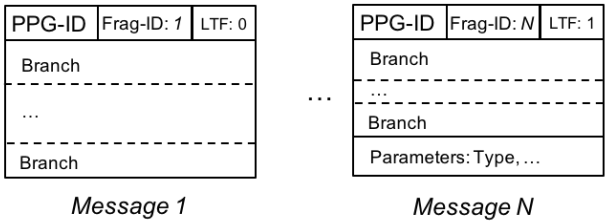

Fig. 4. Fragmentation of PPR Graphs

Figure 4 shows how PPR Graphs are represented as one or more messages, also called fragments to support limited messages sizes. For example, when using the IS-IS protocol to distribute PPR Graph objects, messages need to be encoded as Link State PDUs (LSPs) Type-Length-Value (TLV) objects with a maximum size of 256 bytes. Each message indicates a PPR Graph ID (PPG-ID) and a fragment number 1...N. For PPR Trees, the PPG-ID is the PPR-ID. Each message (fragment) carries as many branches as it can fit. The last message includes all non-branch specific parameters such as the type of the PPR Graph and per-branch or per-PDE (node) default parameters such as default QoS parameters or other PPR attributes. The last message has the LasT-Fragment (LTF) bit set to 1. A node knows that it has received a complete PPR Graph object when it has for a PPG-ID received messages 1...N where N has LTF:1.

There is no separate PPR Graph type for PPR point-to-point graphs. They are encoded as PPR Trees that are just a simple point-to-point path. Typically a path is a single branch, but if the path is so long that it cannot fit into the message size limit, it is fragmented into multiple consecutive branches.

B. PPR Forests and bidirectional PPR Forest

When a PPR Graph has more than one destination, it is called a (unidirectional) PPR Forest, because it can represent multiple PPR Trees - one for each destination. PPR Forests provide another level of compression of the information that needs to be sent from a PCE into the network, and when stored and parsed/examined on the network nodes, forests can further speed up processing because of less data to process. Ideally this can also increase network node cache efficiency due to the smaller amount of memory required to store and parse a single forest vs. multiple trees it represents.

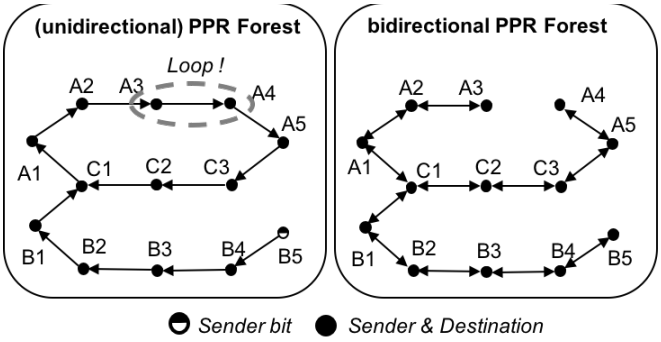

Fig. 5. PPR Forest and Bidir-Forest

The (unidirectional) PPR Forest in Figure 5 is the second PPR Graph type beside PPR Trees. It is indicated in the PPR Type field of the PPR-ID object. A PPR Forest can have (as shown in the figure) at most one loop so that all paths described by this forest are unambiguous. As soon as a graph would have two or more loops, one or more nodes would need to have two or more outgoing adjacencies, and that could introduce undesired ambiguity of path selection. PPR Graphs are meant to be easy to calculate by PCE and not depend on local behavior of nodes such as choosing one out of multiple possible paths. Equal-Cost-Multi-Path (ECMP) adjacencies are therefore also not included.

When processing a forest, each node on the forest creates a separate PPR-ID forwarding entry for every destination that it can reach across the forest. C1 can for example reach A1-A5 and C2-C3 via the forest, but not B1-B5. It therefore establishes no forwarding entries for B1-B5 for this forest. Each destination PDE needs to encode a PPR-ID for itself. These are the PPR-ID installed in the forwarding plane. The PPG-ID of the PPR Forest

itself is a separate unique identifier not used for forwarding entries, but solely to identify the object.

The bidirectional PPR Forest is the third PPR Graph type. This type implies that the adjacencies between nodes in the branches (PDE lists) are bidirectional. These bidirectional forests can have no loops to ensure paths are unambiguous. In the terminology of graph theory, these graphs are therefore bidirectional trees, but in the PPR terminology, they are forests because they have multiple destinations. Note that nodes in such a forest may have different adjacencies towards different destinations in the forest, such as C1, but there is no ambiguity to any individual destination. Specific examples for the use of forests that show their value are described below in V.A.

C. Calculating PPR Graphs

PPR Paths can be calculated with existing algorithms as it would be done for RSVP-TE or SR paths, which can be an NP-hard problem, see [12]. PPR Trees can then be calculated by merging all non-tree-conflicting PPR Paths to the same destination into a PPR Tree. Paths conflict if they have different next-hop adjacencies on the same nodes. PPR Forests can be created by merging non-conflicting PPR-Trees. How an optimized PPR Tree structures are computed at a PCE is subject to future work beyond the scope of this paper. However, Section V.B offers an example with conflicting PPR Trees and a discussion how to optimize the results by taking the goal of non-conflicting paths into account when calculating them.

D. PPR Policy Groups

An individual PPR graph describes a set of unambiguous paths from sources to destinations, but in most cases, more than one graph is needed to define between a set of nodes paths from any node in the set to any other node in the set. A single bidirectional forest can express this, but in many traffic engineering cases, the PCE may not want for all traffic between these nodes to use the same bidirectional forest.

In networks, the need to define any-to-any connectivity between a set of nodes is a common requirement, for example when defining the paths for the endpoints of a particular multipoint service, such as the Virtual Routing and Forwarding (VRF) instances of a Virtual Private Network (VPN) in all the PE that act as senders and destinations.

PPR Graphs can be grouped via an identifier called the PPR Policy group encoded in the parameters of the PPR graph object. A service such as a VPN is then provisioned with that policy group and uses in results only the paths from PPR Graphs tagged with that policy group.

To ensure unambiguous path selection in nodes, the PCE needs to make sure that PPR Graphs in the same policy group do not include multiple paths for the same sender/destination combination. This policy group requirement is the core reason for the sender bit in a PDE. It allows for nodes to be transit nodes in multiple PPR Graphs of a policy group towards the same destination, but to be only a sender in one of them.

E. QoS for PPR Graphs

A PPR Path has exactly one sender and the QoS parameters describe the resources along the path that need to be reserved for exactly the traffic from that sender to that destination.

When using PPR Trees, the goal is to minimize/avoid per-sender state, so not only the packet lookup and forwarding entry are per-PPR-ID (the PPR-ID of the Tree, which is the PPG-ID) but also the QoS is per PPR-ID, resulting in the same scaling benefits for the QoS forwarding elements as PPR Trees do also for next-hop (forwarding) entries (O(N^2) -> O(N)).

The PPR Tree signals the required QoS resources for each sender. Every node determining to be on the tree calculates from the PPR Tree object the aggregate amount of bandwidth required for all upstream tree senders and then it sets up the appropriate QoS for these resources on its outgoing interface and binds it to the PPR-ID, for example as an adjacency/action performed after PPR-ID lookup.

In Figure 6, an example tree with destination A1 is shown on the left side. The tree object indicates that sender A1, A2, A5, and A7 can each send 1 mbps, and A3, A6 and A8 each 2 mbps. A4 is not a sender. A3 for example determines from this tree object, that it will forward traffic for this tree onto link L2 towards A1 and that includes traffic from A5, A6, A7, A8 and itself (A3), for a total of 8 mbps. This QoS parameter is installed for this PPR-ID forwarding entry on A3 together with the forwarding entry A1 via L2.

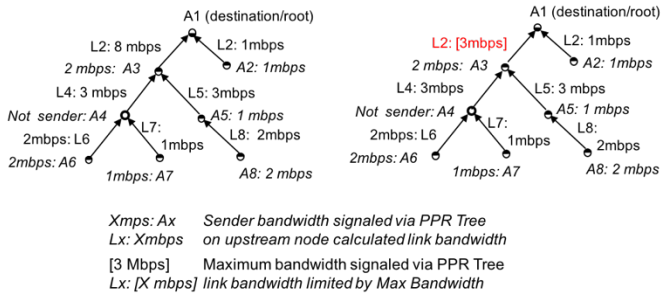

Fig. 6. QoS for PPR Trees

How to determine the aggregate QoS parameters depends on the parameter. Committed Bitrate (CBR) from senders are simply added up and the sum is the bandwidth that needs to be guaranteed for the tree on the outgoing interface as in this example. Likewise, the burst sizes from each sender can be added up and determine the amount of buffers required for the tree on the outgoing interface.

Because only an aggregate QoS is installed in the (transit/egress) forwarding plane of a tree, each sender should also ensure that the traffic it is contributing itself to the tree does not exceed the tree signaled limits for this sender. If the source of the traffic itself is not trusted, this can be done by installing ingress QoS policies for locally sourced traffic on the sender node according to the Qos parameters for this sender in the PPR-Tree.

PPR Trees may be used for traffic from applications where traffic from different senders is coordinated. If A1 for example was connected to a mixer in some Telepresence system, and A2-A8/(A4) are the routers connected to the conference participants, then it could be known that the mixer would always control the participants such that at most two senders are sending simultaneously: The current speaker and the previous speaker in the conference. The maximum aggregate QoS can therefore be

included in the PPR Tree object and limit the per-link resource allocation as shown on the right hand side picture of Figure 6 to 3 mbps on links L4 (performed by A4) and L2 (performed by A3). 3 mbps is the maximum two senders could send at the same time.

For efficient encoding, the QoS parameters used for most senders is encoded as "default QoS parameters" in the PPR parameters (1 mbps in the example), and only senders with different parameters would have those encoded per sender in the senders PDE (2 mbps in the example).

Signaling only the sender contribution in a PPR Tree is not only efficient, but calculating the aggregate on the PCE and signaling it in the tree has also other benefits: the nodes in the network can also dynamically take the IGP signaled physical topology of the network into account. If some link or node in the network fails and sending nodes behind that failure therefore become unreachable, the resource allocation on the local node can accordingly be updated.

When using PPR Forests, the semantic is extended to all the destinations in the forest: A sender may split up the traffic it sends into a forest arbitrarily to the destinations it can reach across the forest, but the aggregate traffic from the sender must never exceed the QoS indicated in the forest for this sender. The calculation for the amount of QoS on every link in a forest stays therefore unchanged from a tree, because every outgoing adjacency on the forest will reach at least one destination. The only difference in forwarding plane setup is the fact that if an adjacency to a next PDE reaches two or more destinations, then this means that all the PPR-ID forwarding entries for those destinations must use the same shared QoS, such as the same per-forest/per-interface queue or shaper.

V. COMPARISONS WITH EXISTING TECHNOLOGIES

A. Dual-Path redundancy engineering with PPR Graphs

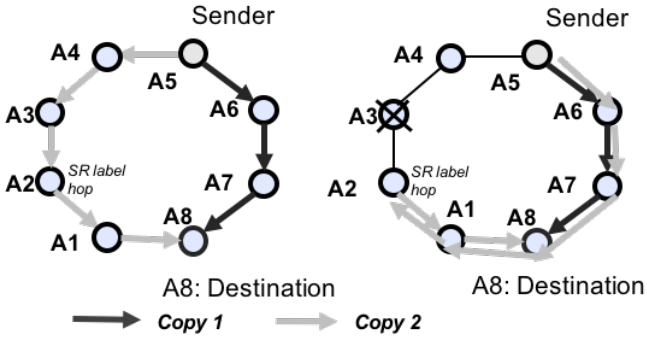

Fig. 7. Segment Routing with dual-path

Even the best resource reservation does not protect against packet loss due to physical layer issues such as bit errors. Proactive protection/Forward-Error protection can and is often used, but it introduces additional delay and/or overhead. High resilient service solutions such as required to transport Time Sensitive Networking (TSN) ethernet packets or other Deterministic Networking (DetNet) services therefore often require the use of two alternative paths from a sender to a destination that are known to not share any single source of failure. The packets are then replicated on entry into the network onto both paths, using sequence numbers to identify packets and remove duplicates on egress from the network. These ingress Packet Replication and egress Elimination Function (PR-EF) are assumed to exist and not something novel in PPR.

Calculating dual-paths can be a complex function on a PCE, depending on the topology and constraints, though distributed algorithms do also exist. See [10] for further references on this topic. The best results are achieved with solutions supporting strict hop-by-hop paths for the forwarding plane, such as possible with RSVP-TE or PPR.

Figure 7 shows an example of the problem in a simple example with Segment Routing (SR). A5 sends copy 1 of the traffic clockwise to A8, and copy 2 counterclockwise. This can be made to work reliable by making SR use strict hop-by-hop label stacks with adjacency SIDs set up to not use fast rerouting. {A6,A7,A8} for copy 1 and {A4,A3,A2,A1,A8} for copy 2. If for example A3 fails then SID A3 becomes unreachable on A4 and copy 2 packets are dropped on A4. This is desired because there is no redundancy benefit of passing copy 2 counterclockwise around the ring and the counter clock path may not even have resources for another copy.

In actual broadband or mobile backhaul access/aggregation networks, rings can have 20 or more nodes based on the geography of deployment, so hop-by-hp explicit label stacks are not feasible. If instead loose label stacks are used, traffic will be undesirably rerouted as shown on the right picture of figure 7. Copy 2 has the minimum label stack to route packets counterclockwise: A2, A8. When A3 fails, traffic is rerouted clockwise, overloading not only the clockwise direction with unnecessary duplicate traffic, but even passing the packet all the way to A2 and back to A8.

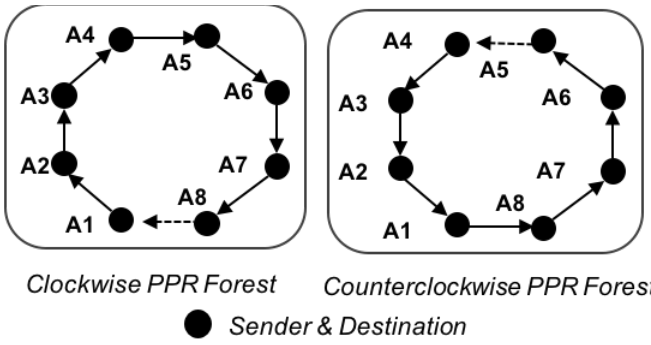

Fig. 8. PPR Forests for a ring

In comparison, consider a setup for the same problem with PPR in Figure 8. Only two PPR Forests are needed to support sending from every node in the ring packets in either direction to any other node. Every node is marked as sender and receiver in both forests, every node uses two PPR-IDs, one used in the clockwise, the other in the counterclockwise ring. When a node fails, there will be no rerouting of PPR traffic because no FRR has been set up explicitly for these forests. The number of forwarding identifier used by PPR in this example is also the same as what would be required in SR: 2 * N: PPR-ID or adjacency SIDs.

B. *Path engineering with PPR Trees*

When engineered / preferred paths are used for a limited set of high value services, explicitly engineered point-to-point paths as supported by RSVP/RSVP-TE or PPR Paths is typically scalable, but the more systematic the engineering requirements become, the more O(k*N^2) becomes an issue. Segment Routing can alleviate some of the scaling issues, but it is limited by the maximum supported label stack size, and it trades P node forwarding state with ingress-PE labels-stack encapsulation forwarding state. Even worse, it cannot create per-hop QoS state for its paths or monitor them independently per-hop, as SR is only a stateless path steering mechanism.

One lightweight form of path engineering that has been deployed in SP networks for more than a decade is the calculation of optimization per-hop metrics of the IGP. Mid to long-term traffic statistics between PEs are taken, and then an exhaustive search for a set of IGP interface metrics is done that results in traffic to be distributed across all paths such that the worst-case maximum load on any link is as low as possible. One of the first PCE products supporting this solution was called "Cariden Mate", and [9] outlines its current version support of this functionality.

Using the IGP to create engineered paths can - as described in the above reference - only achieves limited optimization because every change of an interface metric does affect not only one traffic flow / path, but all traffic flows / paths that potentially would go through that interface. They cannot independently of each other be optimized. But the use of the IGP is very attractive from a scalability perspective: Instead of O(N^2) as with RSVP-TE, the result of using an IGP is instead the establishment of O(N) forwarding entries - one for each destination on every node in the network.

The ideal solution to support O(N) scalability for path engineered forwarding entries is therefore one which does allow to describe similar forwarding as the result of an IGP, but without the limitations incurred by the IGP path calculation: for a particular destination a tree from potentially every node in a network to a particular destination as the root of this tree.

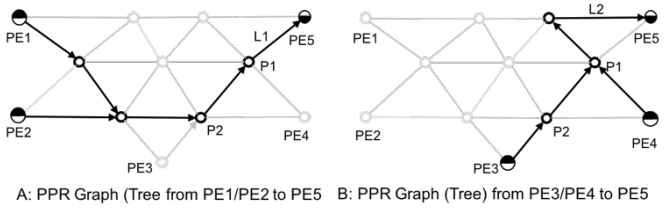

Fig. 9. Two PPR Tree graphs for a single destination

Depending on engineering constraints, a single tree may not suffice though. Figure 9 shows an example of two PPR Tree graphs for path engineered traffic from all other PE to PE5. In this example, link L1 and L2 can only support traffic from half of the sender PEs, so a single tree is not possible because P1 needs to distinguish traffic from PE1/PE2 vs. traffic from PE3/PE4.

When these two PPR Tree graphs are signaled into the network, only the nodes on a tree create forwarding entries for that tree. Only P1, P2 and PE5 need to create two forwarding entries (for both trees), all other nodes only need to create one or no forwarding entry at all.

With PPR graphs, a PCE can calculate exactly the tree(s) that it wants to use for a destination, independent from trees to any other destination (unlike when using engineered IGP link metrics). In a simple option, the PCE simply calculates from every sender of a path to the destination and then it calculates the minimum number of trees representing these paths. In a more advanced calculation, the PCE would take the desire to create a minimum number of trees into account when calculating the paths in the first place.

VI. Conclusions and Next Steps

A. *Summary*

PPR Graphs are the next fundamental extension to the Preferred Path Routing architecture.

PPR Graphs are compact encodings of path and processing information as well as optionally QoS resources, monitoring or other processing instructions from controllers such as PCE(s) to network nodes. PPR Graphs therefore support fast and efficient signaling to network nodes. Per-sender QoS and in network node calculation of aggregate QoS is an example of both that efficient signaling as well as the ability to express aggregated processing state and leveraging the knowledge of the whole graph in each node.

B. *Future work*

The concept introduced with PPR Graphs has a wide range of opportunities for extensions, but its novel functions also require more quantitative assessment (for example through simulations) of achievable performance factors.

Interesting candidate extensions include QoS for deterministic networking, where more than just bandwidth based QoS is required. Interesting routing extensions could and will include Anycasting and signaling of scalable reroute trees as well as multicast for various type of trees - sender specific and shared trees. The PPR Graph concept can especially be attractive for IP multicast when only partial deployment is possible.

In the presented PPR Graph mechanisms, the ability of many forwarding planes to supported aggregation itself is not exploited. Such forwarding plane capabilities can be expressed via structured PPR-IDs and then automatically map to aggregated forwarding entries, such as IPv4/IPv6 prefix lookup forwarding entries. Those can automatically be calculated from information in the PPR Graphs.

Many performance assessments will require prototyping or extrapolation of performance of comparable existing product functions, such as the speed of distribution in IGPs, speed of downloads of forwarding entries and so on.

Assessments that require more fundamental algorithmic work includes calculation mechanisms for engineered paths that leverage the ability of PPR graphs for aggregation: how much can the paths towards each destination be non-tree-conflicting (IV.C) - and therefore provide more scalable, aggregated forwarding state, while still optimizing the overall network throughput ?

REFERENCES

[1] U. Chunduri, A. Clemm, R. Li, "Preferred Path Routing – A Next-Generation Routing Framework Beyond Segment Routing", accepted for 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, UAE, 2018.

[2] C. Filsfils, N. K. Nainar, C. Pignataro, J. C. Cardona, P. Francois, "The Segment Routing Architecture", 2015 IEEE Global Communications Conference (GLOBECOM), San Diego, CA, 2015.

[3] A. Bashandy, C. Filsfils, S. Previdi, B. Decraene, S. Litkowski, R. Shakir, "Segment Routing with MPLS data plane", draft-ietf-spring-segment-routing-mpls-14 (work in progress), IETF, June 2018.

[4] S. Previdi, C. Filsfils, J. Leddy, S. Matsushima, D. Voyer, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-14 (work in progress), IETF, June 2018.

[5] A. Farrel, Q. Zhao, R. Li, C. Zhou, "An Architecture for Use of PCE and the PCE Communication Protocol (PCEP) in a Network with Central Control", IETF RFC 8283, December 2017.

[6] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP)", IETF RFC2205, September 1997.

[7] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", IETF RFC3209, December 2001.

[8] D. Medhi, K. Ramasamy, "Network Routing, Second Edition: Algorithms, Protocols, and Architectures", ISBN-13: 978-0128007372, Morgan Kaufman, September 2017.

[9] "Cisco WAE Design 6.4 User Guide, Metric Optimization", https://www.cisco.com/c/en/us/td/docs/net_mgmt/wae/6-4/design/user/guide/WAE_Design_User_Guide/wd_metric_opt.html, retrieved August 2018.

[10] Enyedi, G., "Novel Algorithms for IP Fast Reroute", Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics Ph.D. Thesis, February 2011. https://repozitorium.omikk.bme.hu/bitstream/handle/10890/1040/ertekezes.pdf.

[11] H. Gredler, J. Medved, S. Previdi, A. Farrel, S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", IETF RFC7752, March 2016.

[12] B. Jozsa, "Traffic Engineering Algorithms for MPLS Networks", Ph.D. Dissertation Summary, https://db.bme.hu/~jozsa/papers/PhD_Booklet_E.pdf , retrieved September 2018.