Internet Engineering Task Force (IETF)
Request for Comments: 7855
Category: Informational
ISSN: 2070-1721
May 2016

S. Previdi, Ed.
C. Filsfils, Ed.
Cisco Systems, Inc.
B. Decraene
S. Litkowski
Orange
M. Horneffer
Deutsche Telekom
R. Shakir
Jive Communications, Inc.

Source Packet Routing in Networking (SPRING) Problem Statement and Requirements

Abstract

The ability for a node to specify a forwarding path, other than the normal shortest path, that a particular packet will traverse, benefits a number of network functions. Source-based routing mechanisms have previously been specified for network protocols but have not seen widespread adoption. In this context, the term "source" means "the point at which the explicit route is imposed"; therefore, it is not limited to the originator of the packet (i.e., the node imposing the explicit route may be the ingress node of an operator's network).

This document outlines various use cases, with their requirements, that need to be taken into account by the Source Packet Routing in Networking (SPRING) architecture for unicast traffic. Multicast use cases and requirements are out of scope for this document.

Status of This Memo

Table of Contents

1. Introduction 3
1.1. Requirements Language 4
2. Data Planes 4
3. SPRING Use Cases 4
3.1. IGP-Based MPLS Tunneling 4
3.1.1. Example of IGP-Based MPLS Tunnels 5
3.2. Fast Reroute (FRR) 5
3.3. Traffic Engineering 6
3.3.1. Examples of Traffic-Engineering Use Cases 7
3.4. Interoperability with Non-SPRING Nodes 13
4. Security Considerations 13
5. Manageability Considerations 15
6. References 16
6.1. Normative References 16
6.2. Informative References 16
Acknowledgements 18
Contributors 18
Authors' Addresses 18

1. Introduction

The ability for a node to specify a unicast forwarding path, other than the normal shortest path, that a particular packet will traverse, benefits a number of network functions, for example:

- Some types of network virtualization, including multi-topology networks and the partitioning of network resources for VPNs
- Network, link, path, and node protection such as fast reroute
- Network programmability
- OAM techniques
- Simplification and reduction of network signaling components
- Load balancing and traffic engineering

Source-based routing mechanisms have previously been specified for network protocols, but have not seen widespread adoption other than in MPLS traffic engineering.

These network functions may require greater flexibility and more source-imposed routing than can be achieved through the use of the previously defined methods. In the context of this document, the term "source" means "the point at which the explicit route is imposed"; therefore, it is not limited to the originator of the packet (i.e., the node imposing the explicit route may be the ingress node of an operator’s network). Throughout this document, we refer to this definition of "source".

In this context, Source Packet Routing in Networking (SPRING) architecture is being defined in order to address the use cases and requirements described in this document.

The SPRING architecture MUST allow incremental and selective deployment without any requirement of a flag day or massive upgrade of all network elements.

The SPRING architecture MUST allow putting the policy state in the packet header and not in the intermediate nodes along the path. Hence, the policy is instantiated in the packet header and does not requires any policy state in midpoints and tail-ends.

The SPRING architecture objective is not to replace existing source-routing and traffic-engineering mechanisms, but rather to complement them and address use cases where removal of signaling and path state in the core is a requirement.

Multicast use cases and requirements are out of scope for this document.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Data Planes

The SPRING architecture SHOULD be general in order to ease its applicability to different data planes.

The SPRING architecture SHOULD leverage the existing MPLS data plane without any modification and leverage the IPv6 data plane with a new IPv6 Routing Header Type (IPv6 Routing Header is defined in [RFC2460]) and a proposal for a new type of routing header is made by [SRH].

The SPRING architecture MUST allow interoperability between SPRING-capable and non-SPRING-capable nodes in both the MPLS and IPv6 data planes.

3. SPRING Use Cases

3.1. IGP-Based MPLS Tunneling

The source-based routing model, applied to the MPLS data plane, offers the ability to tunnel services like VPN ([RFC4364]), Virtual Private LAN Service (VPLS) ([RFC4761], [RFC4762]) and Virtual Private Wire Service (VPWS) ([RFC6624]), from an ingress Provider Edge (PE) to an egress PE, with or without the expression of an explicit path and without requiring forwarding-plane or control-plane state in intermediate nodes. Point-to-multipoint and multipoint-to-multipoint tunnels are outside the scope of this document.

3.1.1. Example of IGP-Based MPLS Tunnels

This section illustrates an example use case.

```
              P1---P2
             /       \
A---CE1---PE1         PE2---CE2---Z
             \       /
              P3---P4
```

Figure 1: IGP-Based MPLS Tunneling

In Figure 1 above, the four nodes A, CE1, CE2, and Z are part of the same VPN. CE2 advertises to PE2 a route to Z. PE2 binds a local label LZ to that route and propagates the route and its label via the Multiprotocol Border Gateway Protocol (MPBGP) to PE1 with next-hop address 192.0.2.2 (i.e., the local address of PE2). PE1 installs the VPN prefix Z in the appropriate VPN Routing and Forwarding table (VRF) and resolves the next hop onto the IGP-based MPLS tunnel to PE2.

To cope with the reality of current deployments, the SPRING architecture MUST allow PE-to-PE forwarding according to the IGP shortest path without the addition of any other signaling protocol. The packet each PE forwards across the network will contain the necessary information derived from the topology database in order to deliver the packet to the remote PE.

3.2. Fast Reroute (FRR)

Fast Reroute (FRR) technologies have been deployed by network operators in order to cope with link or node failures through precomputation of backup paths.

Illustration of the problem statement for FRR and micro-loop avoidance can be found in [SPRING-RESIL].

The SPRING architecture MUST address the following requirements:

- o support of Fast Reroute (FRR) on any topology
- o precomputation and setup of backup path without any additional signaling (other than the regular IGP/BGP protocols)
- o support of shared risk constraints

- support of node and link protection
- support of micro-loop avoidance

3.3. Traffic Engineering

Traffic Engineering (TE) is the term used to refer to techniques that enable operators to control how specific traffic flows are treated within their networks. Different contexts and modes have been defined (single vs. multiple domains, with or without bandwidth admission control, centralized vs. distributed path computation, etc.).

Some deployments have a limited use of TE, such as addressing specific application or customer requirements, or addressing specific bandwidth limitations in the network (tactical TE). In these situations, there is a need to reduce, as much as possible, the cost (such as the number of signaling protocols and the number of nodes requiring specific configurations/features). Some other deployments have a very high-scale use of TE, such as fine tuning flows at the application level. In this situation, there is a need for very high scalability, in particular on midpoints.

The source-based routing model allows traffic engineering to be implemented without the need for a signaling component.

The SPRING architecture MUST support the following traffic-engineering requirements:

- loose or strict options
- bandwidth admission control
- distributed vs. centralized model (e.g., Path Computation Element (PCE) [STATEFUL-PCE], Software-Defined Networking (SDN) Controller)
- disjointness in dual-plane networks
- egress peer engineering
- load balancing among non-parallel links (i.e., links connected to different adjacent neighbors).
- limiting (scalable, preferably zero) per-service state and signaling on midpoint and tail-end routers.
- ECMP-awareness

- o node resiliency property (i.e., the traffic-engineering policy is not anchored to a specific core node whose failure could impact the service).

In most cases, traffic engineering makes use of the "loose" route option where most of the explicit paths can be expressed through a small number of hops. However, there are use cases where the "strict" option may be used and, in such cases, each individual hop in the explicit path is specified. This may result in a long list of hops that is instantiated into a MPLS label stack (in the MPLS data plane) or list of IPv6 addresses (in the IPv6 data plane).

It is obvious that, in the case of long, strict source-routed paths, the deployment is possible if the head-end of the explicit path supports the instantiation of long explicit paths.

Alternatively, a controller could decompose the end-to-end path into a set of sub-paths such as each of these sub-paths is supported by its respective head-end and advertised with a single identifier. Hence, the concatenation (or stitching) of the sub-paths identifiers gives a compression scheme allowing an end-to-end path to be expressed in a smaller number of hops.

3.3.1. Examples of Traffic-Engineering Use Cases

Below are descriptions of two sets of use cases:

- o Traffic Engineering without Admission Control

- o Traffic Engineering with Admission Control

3.3.1.1. Traffic Engineering without Bandwidth Admission Control

In this section, we describe Traffic Engineering use cases without bandwidth admission control.

3.3.1.1.1. Disjointness in Dual-Plane Networks

Many networks are built according to the dual-plane design, as illustrated in Figure 2:

> Each aggregation region k is connected to the core by two C routers C1k and C2k, where k refers to the region.
>
> C1k is part of plane 1 and aggregation region k
>
> C2k is part of plane 2 and aggregation region k

C1k has a link to C2j iff k = j.

The core nodes of a given region are directly connected. Inter-region links only connect core nodes of the same plane.

{C1k has a link to C1j} iff {C2k has a link to C2j}.

The distribution of these links depends on the topological properties of the core of the Autonomous System (AS). The design rule presented above specifies that these links appear in both core planes.

We assume a common design rule found in such deployments: The inter-plane link costs (Cik - Cjk, where i != j) are set such that the route to an edge destination from a given plane stays within the plane unless the plane is partitioned.

```
                             Edge Router A
                                 /  \
                                /    \
                               /      \  Agg Region A
                              /        \
                             /          \
                            C1A----------C2A
                            | \         | \
                            |  \        |  \
                            |   C1B----------C2B
                  Plane1    |    |      |    |     Plane2
                            |    |      |    |
                            C1C--|-----C2C   |
                              \  |        \  |
                               \ |         \ |
                               C1Z----------C2Z
                                  \        /
                                   \      /  Agg Region Z
                                    \    /
                                     \  /
                                 Edge Router Z
```

Figure 2: Dual-Plane Network and Disjointness

In this scenario, the operator requires the ability to deploy different strategies. For example, Edge Router A should be able to use the three following options:

o The traffic is load-balanced across any ECMP path through the network.

- o The traffic is load-balanced across any ECMP path within Plane1 of the network.

- o The traffic is load-balanced across any ECMP path within Plane2 of the network.

Most of the data traffic from A to Z would use the first option, so as to exploit the capacity efficiently. The operator would use the two other choices for specific premium traffic that has requested disjoint transport.

The SPRING architecture MUST support this use case with the following requirements:

- o Zero per-service state and signaling on midpoint and tail-end routers.

- o ECMP-awareness.

- o Node resiliency property: The traffic-engineering policy is not anchored to a specific core node whose failure could impact the service.

3.3.1.1.2. Egress Peering Traffic Engineering

```
                                     +------+
                                     |      |
                                 +---D      F
                    +---------+ /    | AS2  |\ +------+
                    |         |/     +------+ \|   Z  |
                    A         C                |      |
                    |         |\     +------+ /|  AS4 |
                    B   AS1   | \    |      |/ +------+
                    |         |  +---E      G
                    +---------+      | AS3  |
                                     +------+\
```

Figure 3: Egress Peering Traffic Engineering

Let us assume, in the network depicted in Figure 3, that:

- o C in AS1 learns about destination Z of AS4 via two BGP paths (AS2, AS4) and (AS3, AS4).

- o C may or may not be configured to enforce the next-hop-self behavior before propagating the paths within AS1.

- o C may propagate all the paths to Z within AS1 (using BGP ADD-PATH as specified in [ADD-PATH]).

- o C may install in its Forwarding Information Base (FIB) only the route via AS2, or only the route via AS3, or both.

In that context, the SPRING architecture MUST allow the operator of AS1 to apply a traffic-engineering policy such as the following one, regardless of the configured behavior of the next-hop-self:

- o Steer 60% of the Z-destined traffic received at A via AS2 and 40% via AS3.

- o Steer 80% of the Z-destined traffic received at B via AS2 and 20% via AS3.

The SPRING architecture MUST allow an ingress node (i.e., an explicit route source node) to select the exit point of a packet as any combination of an egress node, an egress interface, a peering neighbor, and a peering AS.

The use cases and requirements for egress peer engineering are described in [SR-BGP-EPE].

3.3.1.1.3. Load Balancing among Non-parallel Links

The SPRING architecture MUST allow a given node to load-share traffic across multiple non-parallel links (i.e., links connected to different adjacent routers), even if these lead to different neighbors. This may be useful for supporting traffic-engineering policies.

```
          +---C---D---+
          |           |
PE1---A---B-----F-----E---PE2
```

Figure 4: Multiple (Non-parallel) Adjacencies

In the above example, the operator requires PE1 to load-balance its PE2-destined traffic between the ABCDE and ABFE equal-cost paths in a controlled way where the operator MUST be allowed to distribute traffic unevenly between paths (Weighted Equal-Cost Multipath (WECMP)).

3.3.1.2. Traffic Engineering with Bandwidth Admission Control

The implementation of bandwidth admission control within a network (and its possible routing consequence, which consists in routing along explicit paths where the bandwidth is available) requires a capacity-planning process.

The spreading of load among ECMP paths is a key attribute of the capacity-planning processes applied to packet-based networks.

3.3.1.2.1. Capacity-Planning Process

Capacity planning anticipates the routing of the traffic matrix onto the network topology for a set of expected traffic and topology variations. The heart of the process consists in simulating the placement of the traffic along ECMP-aware shortest paths and accounting for the resulting bandwidth usage.

The bandwidth accounting of a demand along its shortest path is a basic capability of any planning tool or PCE server.

For example, in the network topology described below, and assuming a default IGP metric of 1 and IGP metric of 2 for link GF, a 1600 Mbps A-to-Z flow is accounted as consuming 1600 Mbps on links AB and FZ; 800 Mbps on links BC, BG, and GF; and 400 Mbps on links CD, DF, CE, and EF.

```
       C-----D
      /  \     \
 A---B    +--E--F--Z
      \        /
       G------+
```

Figure 5: Capacity Planning an ECMP-Based Demand

ECMP is extremely frequent in Service Provider (SP), enterprise, and data-center architectures and it is not rare to see as much as 128 different ECMP paths between a source and a destination within a single network domain. It is a key efficiency objective to spread the traffic among as many ECMP paths as possible.

This is illustrated in the network diagram below, which consists of a subset of a network where already 5 ECMP paths are observed from A to M.

```
                C
               / \
              B-D-L--
             / \ /   \
            A   E     \
             \         M
              \   G   /
               \ / \ /
                F   K
                 \ /
                  I
```

Figure 6: ECMP Topology Example

When the capacity-planning process detects that a traffic growth scenario and topology variation would lead to congestion, a capacity increase is triggered, and if it cannot be deployed in due time, a traffic-engineering solution is activated within the network.

A basic traffic-engineering objective consists of finding the smallest set of demands that need to be routed off their shortest path to eliminate the congestion, and then to compute an explicit path for each of them and instantiate these traffic-engineered policies in the network.

The SPRING architecture MUST offer a simple support for ECMP-based shortest-path placement as well as for explicit path policy without incurring additional signaling in the domain. This includes:

- o the ability to steer a packet across a set of ECMP paths

- o the ability to diverge from a set of ECMP shortest paths to one or more paths not in the set of shortest paths

3.3.1.2.2. SDN Use Case

The SDN use case lies in the SDN controller, (e.g., Stateful PCE as described in [STATEFUL-PCE]).

The SDN controller is responsible for controlling the evolution of the traffic matrix and topology. It accepts or denies the addition of new traffic into the network. It decides how to route the accepted traffic. It monitors the topology and, upon topological change, determines the minimum traffic that should be rerouted on an alternate path to alleviate a bandwidth congestion issue.

The algorithms supporting this behavior are a local matter of the SDN controller and are outside the scope of this document.

The means of collecting traffic and topology information are the same as what would be used with other SDN-based traffic-engineering solutions.

The means of instantiating policy information at a traffic-engineering head-end are the same as what would be used with other SDN-based traffic-engineering solutions.

In the context of centralized optimization and the SDN use case, the SPRING architecture MUST have the following attributes:

- Explicit routing capability with or without ECMP-awareness.

- No signaling hop-by-hop through the network.

- The policy state is only maintained at the policy head-end. No policy state is maintained at midpoints and tail-ends.

- Automated guaranteed FRR for any topology.

- The policy state is in the packet header and not in the intermediate nodes along the path. The policy is absent from midpoints and tail-ends.

- Highly responsive to change: The SDN Controller only needs to apply a policy change at the head-end. No delay is introduced due to programming the midpoints and tail-end along the path.

3.4. Interoperability with Non-SPRING Nodes

SPRING nodes MUST interoperate with non-SPRING nodes and in both MPLS and IPv6 data planes in order to allow a gradual deployment of SPRING on existing MPLS and IPv6 networks.

4. Security Considerations

SPRING reuses the concept of source routing by encoding the path in the packet. As with other similar source-routing architecture, an attacker may manipulate the traffic path by modifying the packet header. By manipulating the traffic path, an attacker may be able to cause outages on any part of the network.

SPRING adds some metadata on the packet, with the list of forwarding path elements that the packet must traverse. Depending on the data plane, this list may shrink as the packet traverses the network, by keeping only the next elements and forgetting the past ones.

SPRING architecture MUST provide clear trust domain boundaries so that source-routing information is only usable within the trusted domain and never exposed to the outside world.

From a network protection standpoint, there is an assumed trust model such that any node imposing an explicit route on a packet is assumed to be allowed to do so. This is a significant change compared to plain IP offering the shortest-path routing, but not fundamentally different compared to existing techniques providing explicit routing capability. It is expected that, by default, the explicit routing information is not leaked through the boundaries of the administered domain.

Therefore, the data plane MUST NOT expose any source-routing information when a packet leaves the trusted domain. Special care will be required for the existing data planes like MPLS, especially for the inter-provider scenario where a third-party provider may push MPLS labels corresponding to a SPRING header anywhere in the stack. The architecture document MUST analyze the exact security considerations of such a scenario.

Filtering routing information is typically performed in the control plane, but an additional filtering in the forwarding plane is also required. In SPRING, as there is no control plane (related to source-routed paths) between the source and the midpoints, filtering in the control plane is not possible (or not required, depending on the point of view). Filtering MUST be performed on the forwarding plane on the boundaries and MAY require looking at multiple labels or instructions.

For the MPLS data plane, this is not a new requirement as the existing MPLS architecture already allows such source routing by stacking multiple labels. For security protection, Section 2.4 of [RFC4381] and Section 8.2 of [RFC5920] already call for the filtering of MPLS packets on trust boundaries.

If all MPLS labels are filtered at domain boundaries, then SPRING does not introduce any change. If only a subset of labels are filtered, then SPRING introduces a change since the border router is expected to determine which information (e.g., labels) is filtered, while the border router is not the originator of these label advertisements.

As the SPRING architecture must be based on a clear trust domain, mechanisms allowing the authentication and validation of the source-routing information must be evaluated by the SPRING architecture in order to prevent any form of attack or unwanted source-routing information manipulation.

Data-plane security considerations MUST be addressed in each document related to the SPRING data plane (i.e., MPLS and IPv6).

The IPv6 data plane proposes the use of a cryptographic signature of the source-routed path, which would ease this configuration. This is indeed needed more for the IPv6 data plane, which is end to end in nature, compared to the MPLS data plane, which is typically restricted to a controlled and trusted zone.

In the forwarding plane, data-plane extension documents MUST address the security implications of the required change.

In terms of privacy, SPRING does not propose change in terms of encryption. Each data plane may or may not provide existing or future encryption capability.

To build the source-routing information in the packet, a node in the SPRING architecture will require learning information from a control layer. As this control layer will be in charge of programming forwarding instructions, an attacker taking over this component may also manipulate the traffic path. Any control protocol used in the SPRING architecture SHOULD provide security mechanisms or design to protect against such a control-layer attacker. Control-plane security considerations MUST be addressed in each document related to the SPRING control plane.

5. Manageability Considerations

The SPRING WG MUST define Operations, Administration, and Maintenance (OAM) procedures applicable to SPRING-enabled networks.

In SPRING networks, the path the packet takes is encoded in the header. SPRING architecture MUST include the necessary OAM mechanisms in order for the network operator to validate the effectiveness of a path as well as to check and monitor its liveness and performance. Moreover, in SPRING architecture, a path may be defined in the forwarding layer (in both MPLS and IPv6 data planes) or as a service path (formed by a set of service instances). The network operator MUST be capable to monitor, control, and manage paths (both network and service based) using OAM procedures.

OAM use cases and requirements are detailed in [OAM-USE] and [SR-OAM].

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <http://www.rfc-editor.org/info/rfc4364>.

[RFC4761] Kompella, K., Ed. and Y. Rekhter, Ed., "Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling", RFC 4761, DOI 10.17487/RFC4761, January 2007, <http://www.rfc-editor.org/info/rfc4761>.

[RFC4762] Lasserre, M., Ed. and V. Kompella, Ed., "Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling", RFC 4762, DOI 10.17487/RFC4762, January 2007, <http://www.rfc-editor.org/info/rfc4762>.

[RFC6624] Kompella, K., Kothari, B., and R. Cherukuri, "Layer 2 Virtual Private Networks Using BGP for Auto-Discovery and Signaling", RFC 6624, DOI 10.17487/RFC6624, May 2012, <http://www.rfc-editor.org/info/rfc6624>.

6.2. Informative References

[ADD-PATH] Walton, D., Retana, A., Chen, E., and J. Scudder, "Advertisement of Multiple Paths in BGP", Work in Progress, draft-ietf-idr-add-paths-14, April 2016.

[OAM-USE] Geib, R., Ed., Filsfils, C., Pignataro, C., Ed., and N. Kumar, "A Scalable and Topology-Aware MPLS Dataplane Monitoring System", Work in Progress, draft-ietf-spring-oam-usecase-03, April 2016.

[RFC4381] Behringer, M., "Analysis of the Security of BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4381, DOI 10.17487/RFC4381, February 2006, <http://www.rfc-editor.org/info/rfc4381>.

[RFC5920] Fang, L., Ed., "Security Framework for MPLS and GMPLS Networks", RFC 5920, DOI 10.17487/RFC5920, July 2010, <http://www.rfc-editor.org/info/rfc5920>.

[SPRING-RESIL] Francois, P., Filsfils, C., Decraene, B., and R. Shakir, "Use-cases for Resiliency in SPRING", Work in Progress, draft-ietf-spring-resiliency-use-cases-03, April 2016.

[SR-BGP-EPE] Filsfils, C., Ed., Previdi, S., Ed., Ginsburg, D., and D. Afanasiev, "Segment Routing Centralized BGP Peer Engineering", Work in Progress, draft-ietf-spring-segment-routing-central-epe-01, March 2016.

[SR-OAM] Kumar, N., Pignataro, C., Akiya, N., Geib, R., Mirsky, G., and S. Litkowski, "OAM Requirements for Segment Routing Network", Work in Progress, draft-ietf-spring-sr-oam-requirement-01, December 2015.

[SRH] Previdi, S., Filsfils, C., Field, B., Leung, I., Linkova, J., Kosugi, T., Vyncke, E., and D. Lebrun, "IPv6 Segment Routing Header (SRH)", Work in Progress, draft-ietf-6man-segment-routing-header-01, March 2016.

[STATEFUL-PCE] Crabbe, E., Minei, I., Medved, J., and R. Varga, "PCEP Extensions for Stateful PCE", Work in Progress, draft-ietf-pce-stateful-pce-14, March 2016.

Acknowledgements

The authors would like to thank Yakov Rekhter for his contribution to this document.

Contributors

The following individuals substantially contributed to the content of this document:

Ruediger Geib
Deutsche Telekom
Heinrich Hertz Str. 3-7
Darmstadt 64295
Germany

Email: Ruediger.Geib@telekom.de

Robert Raszuk
Mirantis Inc.
615 National Ave.
94043 Mountain View, CA
United States

Email: robert@raszuk.net

Authors' Addresses

Stefano Previdi (editor)
Cisco Systems, Inc.
Via Del Serafico, 200
Rome 00142
Italy

Email: sprevidi@cisco.com

Clarence Filsfils (editor)
Cisco Systems, Inc.
Brussels
Belgium

Email: cfilsfil@cisco.com

Bruno Decraene
Orange
France

Email: bruno.decraene@orange.com

Stephane Litkowski
Orange
France

Email: stephane.litkowski@orange.com

Martin Horneffer
Deutsche Telekom
Muenster 48153
Germany

Email: Martin.Horneffer@telekom.de

Rob Shakir
Jive Communications, Inc.
1275 West 1600 North, Suite 100
Orem, UT 84057
United States

Email: rjs@rob.sh