Towards a modular architecture for future Internet control plane protocols

Olivier Bonaventure, Simon van der Linden
and Laurent Vanbever[*]
ICTEAM, Université catholique de Louvain, Louvain-la-Neuve, Belgium

ABSTRACT

Today, the Internet depends on the interactions of dozens of control planes protocols (such as routing and signaling protocols) in order to provide reachability to hundreds of millions of hosts. Yet, despite their critical importance, control plane protocols have been developed organically without any reference architecture to structure their organization and interactions. As a consequence, control plane protocols end up implementing the same basic services (e.g., failure detection, neighbor discovery) in different ways, thus increasing the overall complexity and severely impacting manageability. Moreover, this lack of organization is also the root cause of many networking problems such as forwarding loops or traffic loss.

By analyzing several existing protocols, we identified a set of ground principles that every control plane protocol should respect and designed a first modular architecture around them. We describe seven reusable modules and report our experience with a prototype implementation. We believe that protocol designers using such architecture will be able to develop new control plane protocols efficiently reusing the existing modules in novel ways.

1. INTRODUCTION

The network layer plays a key role in today's Internet. This layer can be conceptually divided in two different planes. The data plane includes all the functions that are necessary to process and forward data packets. These functions include header processing, checksum verification, packet forwarding, generation of error messages, etc. The data plane functions are well defined on both endsystems and routers : endsystems generate and receive packets while routers forward packets towards their destination based on information stored in their routing and forwarding tables.The control plane includes all the routing, signalling and management protocols that are responsible for the maintenance of all the data structures that enable routers to forward packets towards their final destination. The Internet control plane protocols, including BGP, OSPF, IS-IS, PIM, RSVP-TE or LDP among many others, play a key role in the operation of the global Internet.

During the 1970s and early 1980s, there were many architectural debates on how to best organise a protocol stack. This debate lead to the development of the layered reference models such as the OSI 7 layers model [36] or the reference model used by the TCP/IP protocol suite [6] (left part of figure 1). These reference models serve as a reference to protocol designers. They define the services provided by each layer and the interactions among the different layers. Although there are many interactions between control plane protocols, there is no reference model or architecture that defines the services provided by each control plane protocol nor the services that it can use from other protocols. The lack of a control plane architecture is best illustrated in [16] and reproduced in figure 1.

The absence of a reference architecture for control plane protocols has several implications. First, there are many complex interactions among different protocols of the control plane. Some of these interactions can result in packet losses or forwarding loops while others increase the complexity of configuring a network. Second, the designer of a new control plane protocol is often forced to include in his/her protocols mechanisms that already exist in other control plane protocols (e.g. neighbour discovery, failure detection, reliable delivery). This is inefficient and increases the complexity of the entire control plane.

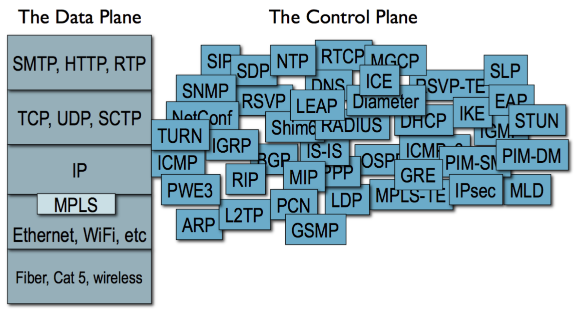

Figure 1: The layered data plane and the unorganised control plane in the Internet (from [16])

Our contribution in this paper is twofold. First, we anal-

[*]Supported by a FRIA grant

yse several of the problems that are due to the absence of a control plane architecture and identify several guiding principles for a control plane architecture (section 2). Second, we describe the core modules that should be included in a control plane architecture (section 3) and report our experience in developing a first prototype implementation of several of these modules to support three unicast routing protocols (section 4).

2. ISSUES WITH CONTROL PLANE PROTOCOLS

As illustrated in figure 1, the Internet control plane contains several dozens of different protocols. In this section, we mainly discuss unicast routing protocols. These protocols were the first protocols of the control plane and they are still among the most important ones in today's Internet. We consider both intradomain routing protocols (RIP, OSPF and IS-IS) and the Border Gateway Protocol (BGP). We sometimes mention issues with multicast routing (PIM) and MPLS protocols (LDP and RSVP-TE).

In the remainder of this section, we provide examples of four classes of problems that arise in the control plane due to the lack of a reference architecture. Based on these problems, we identify four principles that should guide the development of a control plane architecture.

2.1 Starting and stopping control plane protocols

Although a router typical runs several control plane protocols, each protocol defines its own mechanism to detect neighbouring routers and create adjacencies with them. OSPF, IS-IS use special hello packets, LDP sends UDP segments while BGP relies on manual configuration. Each protocol must include a discovery mechanism since there is no control plane protocol that allows a router to discover the control plane protocols that are supported by each of its neighbours.

When a router boots, there are several transient problems that may occur while its control protocols start. For example consider a router on which a BGP session and an IS-IS adjacency start. When the IS-IS adjacency has been brought up, the router is part of the IGP and it can receive packets from other routers. However, its FIB might not yet be complete and it therefore will blackhole packets towards the destination that it does not already know [22]. This problem can be solved by setting the overload bit IS-IS until the router has learned all BGP routes [22]. Similar issues arise with PIM and LDP that also suffer from black-hole problems [23, 10].

Another example is when a router needs to be upgraded, e.g. to install a new version of the router's operating system or to change some of its interfaces. Most operators perform such upgrades by shutting down the router during a maintenance window. The router is then quickly upgraded and restarted. Unfortunately, control plane protocols have not been designed to allow a router to gracefully stop their operation. There are some protocol specific techniques that can be used to gracefully shutdown some control plane protocols [12, 13], but most router reboots cause a network disruption. On some platforms, it is possible to perform some types of upgrades while continuing to forward packets. This requires special *restart* extensions for each control plane protocol used by the router [29, 4, 25, 30]. However, each of these restart mechanisms has been defined independently and restarting several control planes at the same time could create the same issues as when a router boots.

The examples above show that the transient problems that arise when a control plane protocol starts or stops are important.

Principle 1 : A control plane architecture should include mechanisms to start, stop and restart control plane protocols without causing unnecessary packet losses.

2.2 Securing control plane messages

Security and authentication of control plane messages are an important concern for network operators. Over the years, control plane protocols have evolved to include authentication information to allow routers to authenticate received control plane messages. Some mechanisms such as setting the TTL to 255 for packets exchanged between directly connected neighbours [1] are applicable to several protocols. However, this solution does not solve all authentication issues. Each protocol has defined its own authentication mechanism and there are subtle differences between the different mechanisms [34]. This forces network operators to define authentication keys in the configuration of each control plane protocol.

Securing each control plane protocol independently is far from satisfying for several reasons. First, this adds complexity to the configuration of each control plane protocol [20]. Second, the security extensions have several weaknesses from a security viewpoint. Manral et al. list in [20] administrative and technical issues that affect the existing security extensions to control plane protocols. The main issues are the ability to use different cryptographic algorithms and to regularly exchange keys. As of this writing, the syntax of control plane messages allows keys to roll, but no rollover mechanism has been defined besides out-of-band techniques.

Principle 2 : A control plane architecture should include mechanisms that enable to secure all control plane protocols.

2.3 Failure detection

Most control plane protocols need to detect when a link or adjacency has failed. Although this problem is common, each protocol includes its own failure detection mechanism. RIP detects that a neighbour is not reachable anymore after some period without receiving messages from it. OSPF [24], PIM [7] and IS-IS [35] use regular exchanges of HELLO messages to detect neighbours and failures. BGP [27] runs above TCP and considers the session to be down after some period without receiving any Keepalive message.

Using different mechanisms to detect link failures creates several problems. First, each router must implement the failure detection technique that is specific to each protocol, even if it can quickly detect link failures thanks to physical or datalink layers triggers [11]. Second, different protocols running on the same router may not detect a failure at the same time. One protocol may consider that a link has failed while another protocol assumes that the link is still up and continues to advertise it. Such inconsistencies may cause transient problems including packet losses, deflections or forwarding loops.

Fortunately, during the last years, router vendors have developed the BFD protocol [19] as a generic mechanism to monitor links and detect failures. Existing control plane protocols can be configured to use BFD to quickly detect failures instead of using their own failure detection mechanism. Future control plane protocols should rely on the services provided by BFD instead of defining their own failure detection mechanism.

Principle 3 : Failure detection should be one of the services provided by a control plane architecture to the control plane protocols.

2.4 Interactions among control plane protocols

The Internet control plane protocols interact sometimes in unintended ways. A first problematic interaction is between intradomain and interdomain routing. BGP relies on the intradomain routing protocol to detect when a nexthop becomes unreachable. In the past, routers performed this verification by scanning the intradomain routing table every *bgp-scan-time* seconds [31]. More recent implementations receive notifications from the intradomain routing protocol when the reachability of a nexthop changes [21]. Similar issues arise with multicast routing protocols such as PIM that depend on the intradomain routing protocol to select the shortest path to build their multicast trees.

Another source of problems is iBGP. The first deployments of iBGP assumed that transit Autonomous Systems would use a full-mesh of iBGP sessions. This ensures that all the BGP routers in the AS receive all the interdomain routes that border routers use to forward packets. In this case, interdomain packets are correctly forwarded by the transit AS. Large transit ASes often use route reflection [3] to scale their iBGP. Unfortunately, in this case the iBGP signalling graph is not necessarily congruent with the forwarding graph built by the IGP and deflections, forwarding loops or oscillations can happen [15]. These problems can occur either because the iBGP organisation is incorrect or after a link or router failure has jeopardised a correct iBGP organisation.

Besides these explicit interactions, there are also some unintended interactions. Consider a router that runs two routing protocols, say BGP and OSPF. If this router learns a route towards prefix P from both BGP and OSPF, e.g. due to misconfiguration, it will not be able to install both routes in its FIB. The BGP and OSPF specifications do not define how a router should behave in this case. Current router implementations use an administrative distance to prefer one route over the other. However, two major router operating systems disagree on whether a control plane protocol should advertise a route that has not been installed inside its FIB. By default, the BGP implementation of one vendor [18] does not advertise these inactive routes while another vendor advertises them [5]. This can lead to problems that are difficult to debug.

The examples above illustrate some of the problems that occur when the interactions between different control plane protocols have not been carefully studied and specified.

Principle 4 : The interactions among the control plane protocols must be explicitly defined in the control plane architecture.

3. MODULAR CONTROL PLANE

Based on the problems identified above, and as a first step towards an architecture for the Internet control plane protocols, we propose to organise the control plane around seven core modules. We identify the key functions provided by each module and explain their roles.

1. The **RIB** module is an optimised data-structure that stores all the information of the control plane protocols. It performs all interactions with the FIB on behalf of the control plane protocols.

2. The **Information** module converts the objects that are handled by the control plane protocols to and from their binary representation on the wire.

3. The **Neighbour Discovery** module is a specialised protocol that allows a router to detect the control plane protocols that are supported by each of its neighbours.

4. The **Failure Detection** module monitors all the links attached to the router (and the reachability of some remote routers) to inform the control plane protocols of link and router failures.

5. The **Security** module is responsible for securing the messages that are exchanged.

6. The **Transport** module contains the protocols of the data plane that are use to exchange control plane messages between routers.

7. The **Management** module provides all the mechanisms that are required to monitor and configure the control plane protocols.

Figure 2 illustrates the organisation of the different modules and their interactions.

The **RIB** module contains the data-structures that allow to store efficiently the routes that are learned by the control plane protocols. Such data-structures are already found in

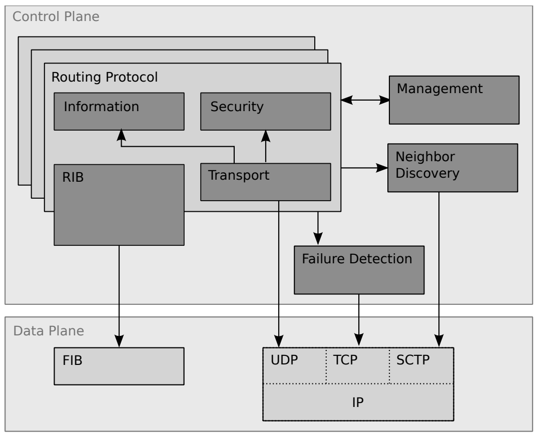

Figure 2: A modular architecture for the control plane

existing router implementations [37, 17]. A **RIB** module should not only allow the control plane protocols to store and retrieve information from its data-structures, but also register notifications to track modifications to some specific routes. The **RIB** module manages all the interactions with the FIB, including the utilisation of multiple nexthops to support load balancing or fast-reroute.

The **Information** module is a presentation layer that has been optimised to support control plane protocols. It receives objects from the control plane protocols and converts them in a binary representation before sending them through the network. We expect that providing a standard and efficient mechanism to encode control plane messages will ease the development of new control plane protocols. This is in contrast with today's control plane protocols, except SNMP, that define their own encoding. This forces protocol implementers to write a new parser for each protocol. Unfortunately, writing safe parsers is not simple and several implementation have suffered from parsing attacks in the past [1]. At this stage, we have not yet chosen one technique to encode control plane messages.

The **Neighbour Discovery** module discovers the neighbouring routers and the control plane protocols that they use. It includes a neighbour discovery protocol that can be reused by any control plane protocol. This protocol can also be used to negotiate options for each control plane protocol. Each router maintains a neighbour discovery session with each of its direct neighbours. This session is used when a new control plane protocols is activated on a router and also when a control plane protocol needs to be restarted.

The **Failure Detection** module contains all the mechanisms that are necessary to detect link and router failures. It provides notifications to control plane protocols when a link or router failure is detected. This module combines two types of mechanisms. First, a link failure can be detected locally by the physical or datalink layers. Second, BFD [19] can be used to monitor all the links attached to neighbouring routers. BFD multihop [19] can also be used to monitor the reachability of remote routers, e.g. those that terminate BGP sessions.

The **Security** module contains all the cryptographic mechanisms that are necessary to secure control plane messages. We expect that each router will be configured with a public-key pair and a certificate as proposed for the 4D architecture [14]. This certificate will allow a router to sign the neighbour discovery messages that it sends. Once the control plane session has been established, cryptographic keys can be exchanged and a key rollover mechanism can be used. This module could be built above the IPSec protocols or include specific security protocols. We intend to leverage on the work performed with the KARP working of the IETF [2] to improve the authentication module.

The **Transport** module manages all the interactions between the control plane protocols and the transport and network layers of the data plane. This module provides abstractions that allow control plane protocols to efficiently exchange information without having to handle all the lower layer details. For example, on a point-to-point link, there is no benefit in using explicit acknowledgements inside each control plane protocol instead of relying on TCP or SCTP. On Local Area Networks, the **Transport** module should leverage multicast to avoid wasting bandwidth. The **Neighbour Discovery** module should allow to negotiate which transport protocol will be used to exchange the control plane messages between two routers. We expect that several control plane protocols will use the same transport layer and possibly the same connection to exchange control plane messages. In case of congestion, the **Transport** module should be able to prioritise some control plane messages over others. These priorities should be explicitly indicated by the control plane protocols (e.g. a BGP update is urgent while a link-state that refresh is not urgent). This contrasts with the current Internet protocols where control plane messages of different importance compete for both bandwidth and CPU resources.

Finally, the **Management** module should allow the operator to configure the control plane protocols and retrieve statistics about their operation.

4. PROTOTYPE IMPLEMENTATION

To evaluate the feasibility of such a modular architecture, we have developed a first prototype in python. Our prototype includes three control plane protocols: a distance-vector protocol similar to RIP, a link-state routing protocol similar to OSPF and a path-vector protocol similar to BGP. The complete implementation contains about 5000 lines of code and is described in more details in [8].

Our implementation follows the event-driven programming paradigm, supported by the Twisted framework [33]. It contains many objects that are observable. Objects called observers may be registered to observable objects. This allows us to trigger a callback function when an even occurs.

[1]See e.g. the CERT vulnerabilities 929656 989406 or 936177 at `http://www.kb.cert.org/vuls/id/`.

Our **RIB** module aggregates the routing tables of our three routing protocols. Each protocol maintains one or more routing tables and registers each of them in the RIB, associated with an administrative distance. Other protocols can access the content of the RIB or any routing table and register observers to be notified of route addition and removal. Each route contains a destination network, a metric, and either an interface or a next hop. The **RIB** module contains the logic to select the best route to be installed in the FIB when there are several routes towards the same destination. The basic decision process is to select the least-cost route, which is sufficient for our distance vector and link-state protocols. For our path-vector protocol, it uses a subclass of the routing table which stores routes with additional attributes (e.g. AS path) and implements a decision process similar to BGP's.

In our **Information** module, we have explored different techniques to marshall and unmarshall objects. Our module currently uses XDR [9] for elementary types such as integers, sets and sequences and the Python `struct` module [26] for IP addresses and network prefixes. Each message sent on the wire includes the type of the marshalled object. Each control plane protocol must register functions to pack and unpack their objects. Filters can be installed to filter out and operate on incoming and outgoing messages. For example, We implemented a compression filter, using `zlib`. Using compression level 6, we observed an average 80% reduction in message size for our path vector protocol. We intend to explore other marshalling techniques such as a marshaller that takes a description of the message format as input, written in an XML language similar to NetPDL [28] for example or using compressed XML.

Our **Transport** module manages sockets and connections. It allows protocols to send objects to neighbours by using unicast or multicast transmission. Sockets are integrated within Twisted's event loop. TCP listening sockets are associated with a protocol factory, while established TCP sockets and UDP sockets are associated with an object whose methods get called upon data arrival. To send and receive messages, routing protocols use and observe objects representing IP interfaces that manage the aforementioned protocols and protocol factories.

Our **Security** module supports message authentication. It is implemented as a filter that appends a MAC to every outgoing message and checks the authenticity of each incoming message. If the MAC is not valid, the message is discarded. A Security Association (SA), provided by the control plane protocol, specifies the hash function to use with HMAC and the key. In order to allow routers to progressively update their SAs, multiple SAs can be used at the same time for incoming messages, thanks to a SA identifier carried along the MAC.

Our **Neighbour Discovery** module contains a table with pieces of information about the neighbours that are associated to each interface. Each neighbour has a unique identifier in the routing domain, e.g. the address of its loopback interface, and an address in the local network. Protocols may register observers to be notified of neighbour discovery and failures. Our discovery protocol allows each control plane protocol to advertise arbitrary data (key-value pairs) in the neighbour discovery messages. Protocols may use this to advertise and negotiate options. As soon as a piece of data change, it is advertised again to the neighbours.

Our prototype does not contain a **Failure Detection** module yet. For now, control plane protocols can rely on the **Neighbour Discovery** module to be notified when a neighbour disappear. Our further work will be to integrate a BFD implementation such as [32].

5. CONCLUSION AND FURTHER WORK

Control plane protocols, such as routing or signalling, play a key role in the Internet. Despite of their importance, these protocols have been developed organically without a reference architecture to structure them and their interactions.

In this paper, we have listed some of the problems that are caused by the lack of organisation of the control plane. Based on an analysis of existing protocols, we have identified seven core modules. Each of these modules can provide services to several control plane protocols. With such an architecture in place, protocol designers will be able to develop new control plane protocols by reusing the services provided by existing modules like Internet applications reuse the transport and network layers of the data plane.

Realising this architecture requires further work in several directions. First, other protocols should be analysed to evaluate how they fit in this architecture. Second, each module should be specified in details. Third, a fully functional implementation should be written to evaluate the feasibility of using such a modular architecture to support new control plane protocols.

6. REFERENCES

[1] J. Abley, B. Black, and V. Gill. Goals for IPv6 Site-Multihoming Architectures. RFC 3582 (Informational), August 2003.

[2] W. Atwood and G. Lebovitz. Framework for cryptographic authentication of routing protocol packets on the wire, February 2010.

[3] T. Bates, E. Chen, and R. Chandra. BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP). RFC 4456 (Draft Standard), April 2006.

[4] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. RFC 2205 (Proposed Standard), September 1997. Updated by RFCs 2750, 3936, 4495.

[5] Cisco. Suppress bgp advertisement for inactive routes. `http://www.cisco.com/en/US/docs/ios/12_2s/feature/guide/fs_sbair.html`.

[6] D. Clark. The design philosophy of the darpa internet protocols. In *Proc. SIGCOMM'88*, August 1988.

[7] Stephen Deering, Deborah L. Estrin, Dino Farinacci, Van Jacobson, Ching-Gung Liu, and Liming Wei. The pim architecture for wide-area multicast routing. *IEEE/ACM Trans. Netw.*, 4(2):153–162, 1996.

[8] Simon Van der Linden. Factorizing ip unicast routing protocols. Master's thesis, Université catholique de Louvain, 2010. Available from `http://inl.info.ucl.ac.be`.

[9] M. Eisler. XDR: External Data Representation Standard. RFC 4506 (Standard), May 2006.

[10] Luyuan Fang, A. Atlas, F. Chiussi, K. Kompella, and G. Swallow. Ldp failure detection and recovery. *IEEE Communications Magazine*, 42(10):117–123, 2004.

[11] P. Francois, C. Filsfils, J. Evans, and O. Bonaventure. Achieving sub-second IGP convergence in large IP networks. *SIGCOMM Comput. Commun. Rev.*, 35(3):35–44, 2005.

[12] Pierre Francois, Pierre-Alain Coste, Bruno Decraene, and Olivier Bonaventure. Avoiding disruptions during maintenance operations on bgp sessions. *IEEE Transactions on Network and Service Management*, 2007.

[13] Pierre François, Mike Shand, and Olivier Bonaventure. Disruption-free topology reconfiguration in ospf networks. In *IEEE INFOCOM*, Anchorage, USA, May 2007. INFOCOM 2007 Best Paper Award.

[14] Albert Greenberg, Gisli Hjalmtysson, David A. Maltz, Andy Myers, Jennifer Rexford, Geoffrey Xie, Hong Yan, Jibin Zhan, and Hui Zhang. A clean slate 4d approach to network control and management. *SIGCOMM Comput. Commun. Rev.*, 35(5):41–54, 2005.

[15] Timothy G. Griffin and Gordon Wilfong. On the correctness of ibgp configuration. *SIGCOMM Comput. Commun. Rev.*, 32(4):17–29, 2002.

[16] Mark Handley. Re-thinking the control architecture of the internet. Keynote talk, Rearch 2009, December 2009.

[17] Mark Handley, Orion Hodson, and Eddie Kohler. Xorp: an open platform for network research. *SIGCOMM Comput. Commun. Rev.*, 33(1):53–57, 2003.

[18] Juniper. Interactions between bgp and igps. `http://www.juniper.net/techpubs/software/erx/erx41x/swconfig-routing-vol2/html/bgp-config11.html`.

[19] D. Katz and D. Ward. Bidirectional Forwarding Detection (BFD). RFC 5880 (Proposed Standard), June 2010.

[20] V. Manral, M. Bhatia, J. Jaeggli, and R. White. Issues with existing cryptographic protection methods for routing protocols. Internet draft, draft-ietf-opsec-routing-protocols-crypto-issues-06.txt, work in progress, June 2010.

[21] P. Marques. Junos bgp. Presented at RIPE 45, 2003.

[22] D. McPherson. Intermediate System to Intermediate System (IS-IS) Transient Blackhole Avoidance. RFC 3277 (Informational), April 2002.

[23] T. Morin and G. Cauchie. Multicast blackhole mitigation with pim adjacency conditions on routing announcements. Internet draft, draft-morin-mboned-mcast-blackhole-mitigation-01, work in progress, February 2008.

[24] J. Moy. *OSPF : anatomy of an Internet routing protocol*. Addison-Wesley, 1998.

[25] J. Moy, P. Pillay-Esnault, and A. Lindem. Graceful OSPF Restart. RFC 3623 (Proposed Standard), November 2003.

[26] Python documentation -struct- interpret strings as packed binary data. `http://docs.python.org/library/struct.html`.

[27] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Draft Standard), January 2006.

[28] F. Risso and M. Baldi. Netpdl: An extensible xml-based language for packet header description. *Computer Networks*, 50(5):688–706, April 2006.

[29] A. Satyanarayana and R. Rahman. Extensions to GMPLS Resource Reservation Protocol (RSVP) Graceful Restart. RFC 5063 (Proposed Standard), October 2007.

[30] Aman Shaikh, Rohit Dube, and Anujan Varma. Avoiding instability during graceful shutdown of multiple ospf routers. *IEEE/ACM Trans. Netw.*, 14(3):532–542, 2006.

[31] P. Smith. Day in the life of a bgp update in cisco ios. Presented at RIPE 45, 2003.

[32] H. Tazaki. kbfd. `http://kbfd.sourceforge.net/`.

[33] Twisted. `http://twistedmatrix.com/`.

[34] Y. Wei, H. Wang, and X. Liang. Analysis of security association for current routing protocol. Internet draft, draft-wei-karp-analysis-rp-sa-00, Work in progress, July 2010.

[35] R. White and A. Retana. *IS-IS: Deploygment in IP networks*. Addison-Wesley, 2003.

[36] H. Zimmerman. Osi reference model - the iso model of architecture for open systems interconnection. *IEEE Transactions on Communications*, 28(4):425–432, April 1980.

[37] Alex Zinin. *Cisco IP Routing: Packet Forwarding and Intra-domain Routing Protocols*. Addison Wesley Professional, 2002.