ASSEMBLER
A BGP-COMPATIBLE MULTIPATH INTER-DOMAIN ROUTING PROTOCOL

Universidad Carlos III de Madrid/University of Twente
June 2011

José Manuel Camacho Camacho

Supervisor: Francisco Valera Pintor (UC3M)
Co-Supervisor: Geert Heijenk (UT)

Contents

Abstract

Multipath routing offers several potential advantages compared to unipath in terms of resources usage, reliability and security. The idea of using several paths concurrently to send traffic towards a destination has already been explored and deployed for cost-based routing solutions, like those typically found in intra-domain routing. Nevertheless, in policy-based routing scenarios, like inter-domain routing, existing multipath solutions have not been embraced yet, mainly because of the backwards compatibility requirements with BGP and the impossibility of performing a global coordinated upgrade of the whole Internet.

This work presents the design and implementation of a multipath inter-domain routing protocol that is backwards compatible with BGP and does not require any kind of inter-AS coordinated deployment. The protocol supports the current policies of ASes and defines a more flexible set of path selection rules to fully exploit the multipath infrastructure of an AS.

The protocol is shown to advertise multipath information consistently in regular unipath BGP updates. In addition, the protocol stability analysis is provided to characterize its behavior and which policies are supported without creating oscillations.

The second part of the work presents an implementation of the protocol in a real software router using XORP. The implementation of the protocol is combined with a multipath FIB designed using CLICK in a testbed to carry out performance measurements of the protocol.

Chapter 1

Introduction

The provision of multiple paths between two nodes has been envisioned for many years as a natural way to enhance communication networks. Once multiple paths are in place, nodes can divert traffic from failed links or split load among them, achieving fast recovery [37] and load balancing [19, 15] respectively. Those techniques should improve the reliability and the performance of the network.

Recent contributions [36, 25, 27] point out that the usage of multipath routing can be advantageous in inter-domain scenarios. Since most of the ASes through the Internet already have redundant connections with their neighbors [24], by embracing multipath inter-domain routing they could benefit from a more flexible use of their resources [35]. The reachability information advertised through these redundant connections should provide ASes with multiple alternatives to route the traffic towards a destination. Those alternative paths could be used simultaneously and enable the aforementioned recovery and balance techniques. Unfortunately, in most cases the unipath nature of the *Border Gateway Protocol* [28] impedes making use of those multiple paths concurrently.

Most ASes have no choice but to rely on techniques such as prefix deaggregation [28] or load sharing [8] to relax the constrains of BGP. Nevertheless, those techniques present their own limitations. By deaggregating prefixes, and autonomous system can handle the traffic corresponding to each sub-prefix differently and forward each split traffic flow through different ASes. In the case of load balancing, the balancing is widely used in intra-domain among equal-cost paths. Each packet, or a flow of packets (e.g. packets sharing the same origin and destination transport addresses) are routed through the available paths. However, with the current load sharing approach [7, 13, 15] used in BGP, the egress point for a certain prefix can be changed periodically in terms of minutes, but it cannot be changed for each packet or flow. The control plane of the network cannot keep up with the necessary changes since every time a packet follows an alternative path, the control plane of BGP generates a new BGP advertisement to avoid routing inconsistencies and loops. The generated churn in the network makes load balancing unfeasible. In practice, only stub ASes exploit their multihoming connections to perform load balancing among different egress ASes, given that they do not have to re-advertise BGP information.

The previous example shows that ASes are keen on more flexible routing configurations. However, in spite of the potential benefits that using multipath routing can bring about in inter-domain scenarios, so far, the lack of economic incentives to replace BGP has hindered

Internet-wide multipath deployments. Moreover, the latter imposes that any approach to deploy multipath inter-domain must be BGP-compatible.

Aimed at hastening large-scale deployments, some backwards compatible solutions have appeared in the literature in the latest years. BGP extensions such as [18, 6] provide multipath capabilities by taking advantage of the multiple interconnections between two ASes. Those paths have the same BGP attributes, such that every selected path can be advertised with the same BGP update. Whereas the latter ensures backwards compatibility with BGP, the multipath set yielded by these solutions is rather limited, e.g. traffic cannot be forwarded across different egress ASes simultaneously even though available paths exist.

An alternative to use richer sets of multiple paths (i.e. multipath sets) is forwarding packets among all available paths and advertise only one. That would require additional mechanisms to detect traffic loops [37, 25] or advertise paths that may be less attractive to legacy routers (e.g. routers advertise the longest received path [31]). Other solutions rely on a separate protocol to incrementally request or advertise additional paths [36, 32] and they can provide more flexible multipath configurations. Yet, they require that at least two neighbor ASes must coordinate to deploy that type of solutions, which represents a main drawback for those approaches.

In this work, a novel protocol for multipath inter-domain routing, ASSEMBLER, is presented. ASSEMBLER stands for *AS-SEt-based Multipath BLending Routing* since the protocol operation resembles a mixing of paths. It is the first inter-domain routing protocol that features both, flexible multipath routing and backwards compatibility with BGP, without any kind of coordination between ASes or additional protocols.

Furthermore, not only is ASSEMBLER backwards compatible with BGP, but also it adheres to its philosophy. It is able to support and map to routing policies the existing business relationships among ASes. Current routing policies, path import and export rules, and traffic engineering techniques are supported and in some cases extended. ASSEMBLER advertisements do not incur in any penalization when compared to BGP thanks to its *path assembling* technique and the selection process (so-called *K-Best Routing Optimizer*) can be locally tuned to cover a myriad of multipath configurations ranging from unequal AS path length multipath through different egress ASes to a *fallback* configuration that mimics exactly the BGP behaviour.

This work is an original unpublished contribution that began with the early idea pointed out by Dr. Alberto Garcia-Martinez suggested in [27] of exploiting prefix aggregation to deploy multipath solutions compatible with BGP. The contribution of this work is the result of a series of discussions among the main author, the supervisor and Dr. Garcia-Martinez. The enumeration of requirements for a backwards compatible multipath inter-domain routing protocol, the evolution of the original idea to the current definition of the assembling technique, the analysis of the implications of adding the assembling technique to BGP, analysis of interoperability in mixed environments, the definition of a proof-of-concept multipath decision process, implementation of the protocol in a state-of-the-art software router and the stability analysis and resulting stability guidelines can be fully attributed to the main author.

The structure of the work is as follows, after reviewing briefly BGP in Chapter 2, Chapter 3 introduces the requirements that are aimed for the protocol design. The protocol itself in presented in Chapter 4 along with an example to show the flexibility supported by ASSEMBLER in its configurations. A group of important deployment considerations are detailed in Chapter 5. The stability of the protocol is proven and configuration guidelines to guarantee stability are given in Chapter 6. Chapter 7 presents the implementation of the protocol and

the validation using a virtual testbed. The work is completed with a comparison between ASSEMBLER and the existing multipath inter-domain proposals in the related work in Chapter 8 along with the conclusions and future work.

Chapter 2

The Border Gateway Protocol

This chapter is aimed at introducing the basics of the Border Gateway Protocol used in inter-domain routing. The terminology and the concepts presented in this chapter are used throughout the work to describe the multipath extensions for BGP. The Border Gateway Protocol (hereafter BGP [28]) is the de-facto standard for advertising reachability information in the Internet, where several independent organizations interconnect to create a large scale network and profit from the exchanged traffic between end-hosts. Those organizations are the so-called Internet Service Providers (i.e. ISPs). Each ISP runs one or more Autonomous Systems (i.e. AS) or *domains*, which are networks that hauls traffic according to an economic-driven policy. The AS networks interconnect among them and exchange traffic. When the exchanged traffic between two ASes is uneven or one of them has a better location in the network (e.g. a main provider or *tier-1*), it is said that they keep a *transit relation*, one AS plays the role of the *provider*, offering hauling service towards a destination to the other AS, its *customer*. The provider charges a per-bit rate to the customer for the coursed traffic from and to the customer network. On the other hand, when the exchanged traffic is roughly the same or both ASes are of similar importance, the two ASes have a *peering relation*, they both act as peers without charging each other.

Hence, the fact that some paths may provide larger profit than others makes that cost-based protocols such as OSPF cannot be used in this context, since the path with lowest sum of weights is not necessarily the most profitable. In inter-domain scenarios ISPs must define routing policies according to their business model, such that routers select the most profitable path for the ISP. Moreover, the advertisement of some paths may cause the ISP to incur in extra losses for carrying undesired traffic, therefore in addition to the import policy, ISPs must also define an export policy that states to which ASes a path must not be announced. The BGP standard provides the necessary mechanisms to disseminate the reachability information, techniques to implement routing policies and path attributes to enforce them. To that extent, BGP defines for each path which attributes may be used to describe the path characteristics. The attributes are used in a decision process to select the bests path according to the policy.

The dissemination of reachability information happens in three different steps. Firstly, one or more neighbor ASes advertise reachability information to different border routers in the AS through external BGP (i.e. eBGP) sessions. Secondly, after the eBGP dissemination happens between ASes, the internal BGP (i.e. iBGP) redistribution takes place, such that every BGP router inside the AS is aware of the available paths learnt at different border

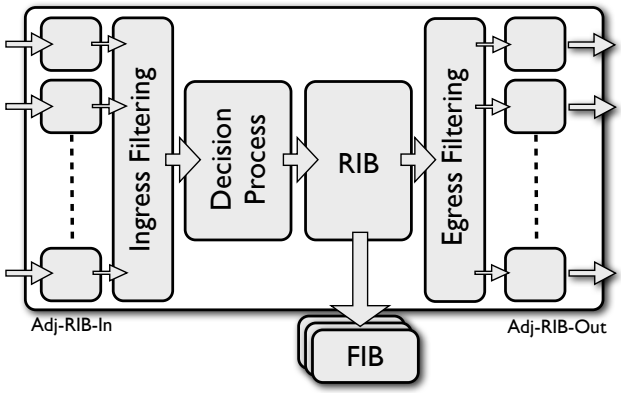


Figure 2.1: BGP Process Architecture

routers and the path selection becomes a distributed process, with every BGP router taking a consistent decision according to all the received announcements. The decision process carried out at each router selects the best path for that router. In some cases, a router will play the role of egress traffic point (i.e. it has learnt its most suitable paths through eBGP) and in others the routers will play the role of an intermediate node or an ingress point (i.e. their best path comes from an iBGP session). The third step consists in advertising further the decision made by each router to neighbor ASes.

The regular operation of each BGP router in the AS consists in establishing and maintaining a session with other BGP routers and exchanging BGP updates with them for different prefixes. Upon the reception of an advertisement for a prefix, the BGP router receives the path to that prefix along with a set of values for the different BGP attributes such as preference values, the neighbor advertising the path and the ASes that the traffic will cross. BGP attributes are rewritten by the router depending on their scope, e.g. an attribute can be meaningful to the border router, to the entire AS, to the neighbor ASes or end-to-end.

The received paths are passed from left to right through the blocks depicted in Fig.2.1 starting at the import filter, which checks that the paths are compliant to the routing policy and tags some of their attributes, such as the local preference. Afterwards, if multiple paths are available for the same prefix, the *decision process* is carried out to select which is the most suitable given the routing policy. Every path received from any BGP session towards the same prefix is compared with other paths for that prefix. The selection of one path or another depends on the attribute values assigned to each path. The paths are compared on different attributes following the rules defined in Table 2.1, which represents the BGP decision process [28]. The process executes rules sequentially until only one path is left within the candidate set, i.e. the *winner*. Then, the winner is passed onto the *Routing Information Base* RIB in order to be deployed in the FIB.

The first decision rule (the WEIGHT attribute is typically not used) is based on the LOCAL_PREF attribute value. The latter reflects the preference of the network administrator for a certain path or set of paths and overrides the values of other attributes since it is the first decision rule. According to the typical business relations between ASes mentioned above, the paths coming from customer ASes are preferred over paths coming from peers, since the AS relaying the traffic earns money using them. Paths from peer ASes are preferred over paths from providers since no money is either paid or received per coursed traffic. Finally, paths coming from providers are economically less attractive since that implies that the AS using them pays for the coursed traffic. These preferences are mapped to numeric integer values of

Table 2.1: BGP Decision Process

1.-	Keep paths with highest LOCAL_PREF value
2.-	Keep paths with shortest AS path
3.-	Keep paths with lowest ORIGIN value
4.-	For each advertising AS, select the path with lowest MED value
5.-	If there is a remaining path with session TYPE eASSM, delete paths with TYPE iASSM
6.-	Keep paths with lowest IGP cost
7.-	Keep paths with lowest BGP_ID
8.-	Select the path advertised from the lowest network address

the LOCAL_PREF (e.g. 60 to a provider paths and 100 to customer paths).

Since two paths may have the same LOCAL_PREF value, e.g. two paths coming from two different customer ASes or two path coming from the same AS but received at two different border routers, additional rules are required to decide on one path. Before analyzing the next rule, the AS_PATH concept must be introduced. The necessity of hiding connectivity information consistently to avoid the economic losses as pointed out above and the scale of the network conditioned the design of BGP to be an extension of *distance vector* protocols called *path vector* protocols. In particular for BGP, the path vector is an AS-level representation. Each AS is assigned with a unique identifier called *AS Number* such that each time a border router of an AS advertises a path outside its own AS, it appends the local AS Number to the path. The collection of AS Numbers of ASes crossed along the path is the value of the AS_PATH attribute. The AS_PATH is in turn formed by a collection of segments. Each segment can be either an ordered sequence of AS Numbers, called AS_SEQUENCE or an unordered set of AS Numbers delimited by braces, called AS_SETs. The AS_PATH length is computed by counting the length of each AS_SEQUENCE as the number of ASes within the sequence and the length of AS_SETs as length one.

The third rule is related to the way the reachability information is generated by the first AS. If the advertisement was dynamically generated by redistributing intra-domain routing information into BGP, the ORIGIN attribute gets a lower value. Otherwise (e.g. statically configured) the ORIGIN is higher. The reason for this comes from the idea that in case something fails, a dynamic configuration will advertised that the reachability information formerly propagated is not valid anymore, whereas static configurations are not responsive to network failures.

If at this point of the decision process there is more than one path available, either they come from the same AS but from different border routers or from different ASes but having the same AS_PATH length. In the former case, if the AS receives the same path through different routers (*exit points*), it may have to satisfy the preferences of the neighbor AS. Think for instance in the case of a customer AS advertising one path through two different BGP sessions with the same provider, the provider should respect the preferences of its customer.

To that extent, some BGP attributes are used to influence the treatment received by a path in a neighbor AS, like the *multi-exit discriminator*, i.e. MED. The paths coming from the same AS and not removed until here have the same LOCAL_PREF and the same AS_PATH length. Then, the advertising AS can suggest that it prefers to receive traffic over one path or another by properly setting the MED value. Rule 4 removes paths coming from the same AS that are not of minimum MED value.

Another way of influencing the decision of a neighbor AS is the use of *BGP Communities*, which are designed to give an homogeneous treatment to the paths containing them (e.g. assign a certain LOCAL_PREF value). BGP Communities are optional attributes and not every AS support them. Typically, the actions taken over a path with a certain community are posted publicly by provider ASes such that its customer can use them. Communities used between peers are typically negotiated privately between the peers. BGP Communities are processed before the decision process is executed and they do not have an specific rule in the process, although they can influence the outcome of a certain rule by modifying the BGP attributes of a paths.

After the preferences of the customers are processed taking into account the MED values, the BGP router by means of Rule 5 gives preferences to paths learned from a BGP session with a router in an external AS to paths received from an iBGP session. Otherwise, all the routers may end up selecting an internal advertisement and loops and oscillations may occur.

Rule 6 perform what is known as *hot-potato* routing, this is if there are several alternatives compliant with the routing policy of the AS, try to course the traffic towards the closest egress point of the AS, since the traffic will stay as less time as possible and the operational costs per bit will be the lowest. Therefore, among the multiple possibilities those with the lowest internal cost are chosen.

The remaining decision rules do not enforce a given preference or an optimization over the selected paths but they perform a tie-break, such that only one path is left after the decision process. Rule 7 selects the routes coming from the BGP neighbor router with the lowest BGP_ID exchanged in the BGP session establishment. If there is more than one path coming from the same neighbor router (e.g. two routers have two connections in parallel) then the path advertised from the network interface with the lowest network address is chosen, as stated in Rule 8.

After the decision process, exporting rules are configured per BGP session. When the router selects a certain path, there are some BGP session through which the path must not be advertised. Regarding external connections with other BGP border routers, the router configuration usually follows export rules as defined in [10]. For instance, if a router selects and eBGP path coming from a provider AS, it must advertise that path only through customers, since the AS will pay the provider for the traffic towards that destination and the customers are paying to the AS. Otherwise, if the AS announces to the rest of its providers it pays for relaying the traffic to the advertising provider and it is charged by the rest of the providers for sending traffic to it. The case in which the AS announces the path to its peers is similar except for the fact that the AS is not charged by the peers. Paths coming from peers can be only advertised to customers for the same reason. Only paths coming from customers can be advertised to providers and peers.

For iBGP, the export rules are typically simpler and the most relevant is the one that states that paths learned by means of an iBGP session must not be re-advertised through another iBGP session. Notice that paths advertised through iBGP does not add any kind of information about the hops that would be performed inside the AS, therefore there is no way

to detect internal loops among iBGP speakers.

If the selected rule does not match any of the discarding egress filters, it can be advertised through that BGP session in a BGP update message. BGP updates contains typically multiple entries, each of them regarding a different prefix. Since BGP is designed as a unipath routing protocol, each router is expected to select and use only one path per prefix. Therefore, two advertisements regarding the same prefix are included in the same update (this should not happen in practice since the routers update the information of a prefix if it has not been advertised yet) or an update is received after another, the last information received overwrites the previous information announced by that peer.

Finally, to conclude with this brief description of BGP, although internal scalability techniques such as route reflectors, route servers and confederations are not covered in this work and no multipath extensions are proposed to them, the multipath protocol proposed in this work is able to co-exists (under certain conditions) with legacy BGP routers within the same AS and interoperate with already existing multipath intra-AS solutions like BGP-AddPaths [32].

Chapter 3

Protocol Requirements

The main goal of ASSEMBLER is to provide ASes with a backwards compatible solution for inter-domain routing that enables multipath routing. In this chapter, the defining requirements for ASSEMBLER are introduced prior to describing the relevant parts of the protocol in depth. Those requirements motivate the design choices that are presented in the next chapter and provide a clear view of the main features of the protocol. The discussion addresses the issues of target multipath configurations, backwards compatible updates, stability and data plane growth.

3.1 Flexible Multipath Routing

ASSEMBLER must flexibly let administrators choose the characteristics and the amount of paths used in the routing system. The multipath protocol must feature enough flexibility to concurrently select paths that, (1) have different next AS, (2) have different AS path length and (3) have different internal cost. Moreover, the protocol must be able to select a subset of the paths matching the previous conditions using a deterministic tie-break. ASSEMBLER must empower administrators with the tools to implement such a broad range of routing policies. In some cases, the administrator would like to provide a router with the whole set of paths and in other cases, administrators may prefer keeping only those with certain attributes, (e.g. shortest AS path length).

The first requirement that we impose on ASSEMBLER is that regardless how the selection process of multiple paths is tuned, the BGP *winner* path is always included in the multipath set. In addition to the BGP winner, according to criterium (1), a router can either include paths through different egress ASes or limit the multipath set to go through the same next AS. The criterium (2) defines if *equal AS length multipath* (i.e. ELMP) is enabled or additional paths, longer than the shortest one, can be selected. Criterium (3) implies that internal *equal cost multipath routing* (i.e. ECMP) is supported and additionally, the administrator can tune how much the internal paths deviates from the hot-potato routing behaviour [30]. A strong requirement is that ASSEMBLER must allow every AS to select the type of multipath that they need independently from other ASes, i.e. without any type of coordination.

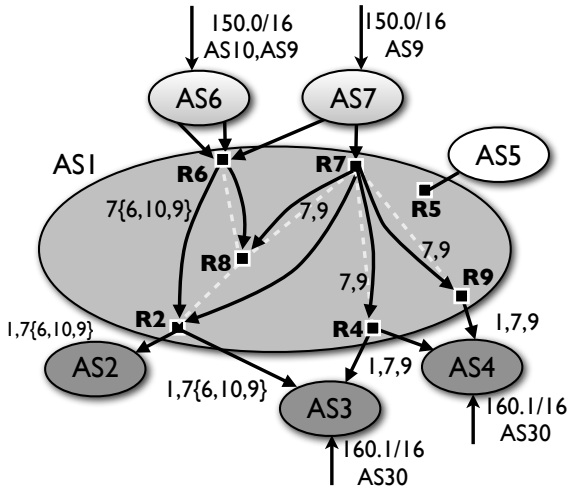


Figure 3.1: Model of a transit AS with Path ASSEMBLER Routers

3.2 BGP-Compatible Advertising Scheme

ASes advertise each other reachability information by means of BGP updates. Whereas processing regular BGP updates should not present any shortcoming for multipath routers, advertising multiple paths per network prefix in a single BGP update is not a trifle. Multipath routers should respect the structure and the semantic of the attributes included in the updates, such that legacy routers can keep on processing them. Concatenating multiple paths to the BGP message is not enough, provided that the update of a path has implicit the withdrawn of previous updates.

Therefore, ASSEMBLER must carry out some additional processing to merge information from multiple paths and accommodate them into regular BGP updates. To that extent, path *assembling* (Section 4.2), a particular case of prefix aggregation [28] seems an outstanding candidate. It is a crucial requirement that generated advertisements must be representative of the aggregated paths, such that a router (legacy or not) can perform any regular BGP processing over the advertisement, as if the paths were announced separately. For instance, when a router receives an announcement containing an aggregate of paths, it must be able to derive the local preference for the aggregate or apply MED values comparison consistently. The protocol must identify those cases in which the advertisements are not representative and do not perform aggregation. Even for BGP, RFC4271 [28] identifies different situations where it is not consistent to aggregate multiple prefixes due to conflicting attributes. Thus, the protocol must avoid those situations in which inconsistent network advertisements may be created as a consequence of the aggregation process.

3.3 Controlled Routing Table Growth

The design must address the well known problem of the inter-domain routing table growth. Whereas routers feature more processing and memory capacity at the control plane, the situation at the data-forwarding plane is completely different. The hardware that forwards packets at wire-speed is expensive and its storage space constrained. The adoption of multipath does

nothing but worsening the problem as multiple next-hops are stored per prefix. A growth in the amount of paths selected can potentially rise issues with the limitations of the data plane.

Therefore, the protocol must be aware of those constrains and must limit the amount of paths relayed to the data plane. The requirement for the protocol is to be able to select a subset of *k-best* paths per prefix, such that the size of each routing table entry is limited. Every path in the subset must be compliant with the routing policy and the *k-best tie-break* must be deterministic.

3.4 Stable under Common Configurations

BGP has been proven to be unstable under conflicting routing policies. The existing relation among those routing policies that cannot be fulfilled simultaneously is called*dispute-wheel* [12]. In presence of dispute-wheels BGP is not guaranteed to converge and the network may end up in a permanent oscillation. Permanent oscillations do not happen often in practice since routing policies are typically overruled by the business relationships among ASes. It can be proven that when routing policies align with those relationships dispute-wheels cannot be created.

The work in [5] presents a more abstract framework for the analysis of the stability in policy-based routing protocols. The framework extends the concept of dispute-wheels to *reflexive* policy relations. The concept of reflexive relations is more powerful in the sense that it covers the BGP dispute-wheels and allows to extent stability results to multipath policy-based routing protocols. ASSEMBLER must be able to converge in absence of reflexive relations among policies. Relaying on the abstract framework in [5], the protocol must be proven stable in those situations in which conflicting policies do not exists, specially in those that align with business relations among ASes.

Chapter 4

Path ASSEMBLER

ASSEMBLER is a novel multipath inter-domain routing protocol inspired in the BGP prefix aggregation to compact the multipath information. ASSEMBLER stands for AS-Set-based Multipath BLEnding Routing, since the protocol *blends* the additional AS_PATHs and stores the result in AS_SETs. ASSEMBLER keeps backwards compatibility and allows for a progressive deployment of multipath-capable routers. The specification of ASSEMBLER relies on two main cornerstones: a multipath selection process and a BGP-compatible multipath advertising scheme.

Fig.4.1 shows the block diagram of an ASSEMBLER process running in the control plane of a router. There are some differences between Fig.4.1 and a BGP process diagram (see Fig.2.1). The import policy (*ingress filtering*in Fig.4.1) is applied first, like in BGP. The BGP decision process has been replaced by the multipath selection algorithm K-BESTRO (pronounced *cabestro*) that stands for *K-Best Routes Optimizer*. K-BESTRO is presented in detail in Section 4.1. The output of the K-BESTRO block is a set of K paths instead of a single winner path. K-BESTRO features three parameters to tune the characteristics of the multipath set. The parameter ELMP defines the maximum difference in AS path length between the shortest and the longest AS path. ECMP defines the difference in internal cost among selected paths. Finally, the KBEST parameter limits the maximum size of the multipath set, which should be set depending on the capacity of the routing table to store prefixes with multiple next-hops.

The paths in the multipath set are passed to the RIB in order to be installed in the data plane (through the FIB). Afterwards, they undergo the export policy. The export policy (*egress filtering*) generates the same advertisement for all the peering sessions that the router maintains. Therefore, a neighbor is either advertised or the export policy discards the whole multipath set as soon as one path matches a filter. Otherwise, different paths could be discarded for each peering session, generating different advertisements. Adding *neighbor-specific* announcements [34] is out of the scope of this paper.

Next to the egress filtering block, there is the new block called *Assembling*, which is responsible for generating the advertisements. The assembling algorithm ensures backwards compatibility, creating special BGP announcements that can be processed by legacy routers, do not incur in penalization when competing with regular unipath BGP announcements in the selection process and allow multipath capable ASes to use several paths concurrently. The algorithm takes its name from the way of constructing those announcements that resembles an

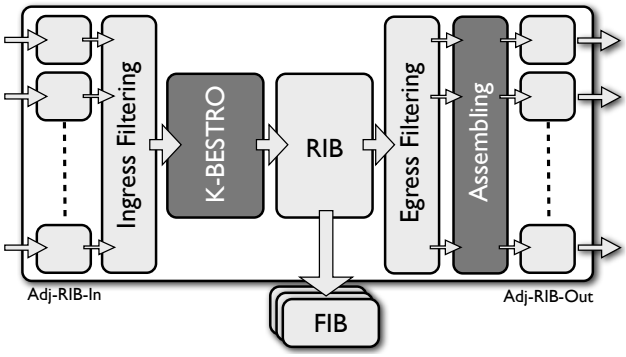


Figure 4.1: Path ASSEMBLER Process Architecture

assembling of pieces (e.g. AS_NUMBERs in this case). The announcement is an aggregated version of the multipath set that cannot be distinguished from the outcome of a regular prefix aggregation. See Section 4.2 for details about the assembling procedure for external (i.e. eASSM) and internal (i.e. iASSM) ASSEMBLER peering sessions (eASSM/iASSM are also used to refer BGP peering sessions, unless stated otherwise). Finally, the advertisement containing the *assembled* path is propagated to the neighbor routers.

4.1 Decision Process: The K-BESTRO Algorithm

The decision process of ASSEMBLER is carried out over the set of advertisements for a given prefix that are not discarded by the import policy. The decision rules resemble those for BGP. Meanwhile the BGP decision process clearly has a tie-breaking character and paths are *trimmed* from the set of candidates on the look out for the most suitable path. The requirements regarding flexibility of K-BESTRO completely redefine its philosophy and it is inspired in the decision process of Morpheus [33]. In the design of Morpheus, the decision process creates a *ranking* of the candidate paths according to some configurable criteria rather than discarding them. Afterwards, the set of best paths is selected, possibly according to different criteria, this time applied over the paths already sorted (e.g. select the first k paths in the ranking with MED value equal to 10). Therefore, K-BESTRO can be seen as a particular instance of a Morpheus ranking with the criteria presented in the next paragraphs. Each of them is mapped into a phase of the algorithm depicted in Table 4.1,

The ranking criteria of K-BESTRO rank the BGP winner in first position and the rules respect the semantic of the BGP attributes Rules 1 to 5 discard paths like a regular BGP decision process. The BGP winner is never discarded by those rules and rules 6.a-d give higher ranks following the order used by BGP to tie-break the paths. Therefore the BGP winner is always ranked first. Generally speaking, the algorithm must always advertise the winner and propagate other aggregatable paths whenever possible.

In addition, in order to keep the semantic of BGP attributes and make the paths sortable, some of them must be discarded before the remaining paths are sorted in a rank. Otherwise, inconsistent multipath decisions can be made with respect to the semantic of the attributes. For instance, it is not consistent that two paths with different LOCAL_PREF appear in the final ranking or in the selected multipath set. It is not sounded either that two paths coming

from the same AS and with different MED values are simultaneously used, since the customer is explicitly stating that it prefers one path to the other to receive the traffic.

The first phase starts with rule 1, which keeps just the paths with highest local preference. The next step in BGP is keeping paths with shortest AS path length. Instead, K-BESTRO considers paths that satisfy the relation AS_PATH_LENGTH $<=$ *shortest-l*+ELMP, being *shortest-l* the shortest AS path length value found in the candidate set and ELMP is the parameter introduced earlier. The latter is implemented in rule 2.

The ranking criteria of K-BESTRO keep the order of the BGP rules For example, the latter implies that the BGP AS path length rule must not be overridden by the MED rule, i.e. a path with lower MED but longer AS path must not cause any path with shorter AS path length to be overlooked by the algorithm. K-BESTRO ensures that every path taken into account in the ranking honours the highest LOCAL_PREF, highest ORIGIN, lowest MED per AS and session TYPE (i.e. eASSM or iASSM) criteria exactly as BGP does.

The second phase applies these criteria (rules 4-5). For each AS advertising a path for the prefix, the algorithm looks for the paths of shortest length through that AS and the lowest MED value of that path. Every path from that AS and different MED value is removed at rule 4.c. The phase is completed by leaving paths only from one session TYPE. If there is a path from an external session, i.e. eASSM, paths with session TYPE iASSM are removed (rule 5). This is needed to avoid that two border routers try to course traffic through the external path of the other (see [28] for further details).

The final ranking of paths must be performed over monotonically increasing and bounded attribute values Applying rules 1-5 leads to consistent results regarding the selected multipath set. Once the considered paths are compliant with those rules, the ranking can be performed upon the remaining attributes without violating the specified routing policy and the order of BGP rules. For example, two paths with equal preference, origin and coming from different ASes, can be ranked according to their AS path length without creating any inconsistency.

The algorithm executes the third phase (rule 6) and ranks the paths according to the criterium of shortest AS_PATH_LENGTH first. If a several paths in a subset draw in AS path length, it sorts the subset from lower internal cost to higher. Within the subset, if the first ranked path has a cost of *lowest-c* it removes the paths with internal cost higher than *lowest-c*+ECMP, where ECMP is a parameter that can be tuned by the administrator. While either tunnelling or IGP *equal cost multipath* are used inside the AS, ECMP should be equal to 0. If at this point some paths have the same AS path length and interior cost towards the next-hop, the paths with lower BGP_ID are ranked first. If several paths have the same BGP_ID attribute, then the ones advertised from the interface with the lowest address are ranked first.

The K-BESTRO algorithm selects paths in order of appearance in the ranking and selected paths are aggregatable As described at the beginning of the chapter, the selected multipath set ends up aggregated into a single BGP advertisement. Therefore, the selected paths must be aggregatable, otherwise the generated announcement is not representative of the multipath set and that may lead to routing inconsistencies. RFC4271 [28] defines that two path with different MED values should not be aggregated. This restriction only applies to paths advertised through iBGP sessions. Similarly, only paths with the same NO_EXPORT:X Community (i.e. do not export this path to a specific peer X) can be aggregated, since as

mentioned above, neighbor-specific configurations are not supported by ASSEMBLER. If the first ranked path does not include the NO_EXPORT:X community for any peer, the algorithm should overlook other paths in the ranking including any NO_EXPORT:X community when selecting the multipath set.

This last criteria is implemented in the fourth phase (rules 7-10) is executed. The parameter KBEST is defined by the administrator and limits the maximum size of the multipath set. The fourth phase takes care of selecting a maximum of KBEST paths that can be aggregated and advertised together. According to the previous paragraph, if the first ranked path is tagged with the BGP Community NO_EXPORT:X, then the multipath set contains only the first KBEST ranked paths with the same community. Otherwise, paths with any NO_EXPORT:X community are deleted from the rank to avoid aggregation conflicts. Thereafter, if the ranked paths come from an iASSM session, then select the first KBEST paths in the ranking. Else if they come from eASSM sessions, select the first ranked KBEST paths coming from the same AS and with the same MED value as the first ranked path, as stated in RFC4271.

The algorithm finishes relaying the set of KBEST paths to the egress filtering block that implements the export policy.

4.2 Route Dissemination: Path Assembling

The decision process constructs a multipath set compliant with the preferences of the administrator. The multipath set must be advertised to every neighboring AS with an established peering session. Advertising an array of paths for each network prefix is not supported by BGP. The algorithm presented in this section is applied to the set of paths to embed the multipath information into a single BGP advertisement.

The algorithm follows the philosophy used in prefix aggregation to compact the multipath information. Prefix aggregation defined in [28] defines how the attributes of two advertisements can be combined under some conditions, such that two contiguous prefixes propagated within each advertisement can be combined into a larger prefix and advertised in a single BGP update message. The attributes of the new message are the result of aggregating those in the two advertisements. Our *path assembling* can be understood as the aggregation of several advertisements carrying the same prefix.

Besides other attributes like the NEXT-HOP or the ORIGIN, of special interest is the AS_PATH attribute aggregation. When two contiguous prefixes are aggregated, it is necessary to keep the AS_PATH information to maintain path loop-freeness. In [28] the minimum requirements for the path aggregation algorithm are specified. Any algorithm compliant with those minimum specifications can safely combine the AS_PATH information from several announcements. The algorithm used by ASSEMBLER meets the minimum requirements of [28] and creates an AS_PATH following the most commonly found format in current routing tables. Data sets collected at some Internet vantage points [23, 24] brings out that recorded aggregations construct always an AS_PATH with an AS_SEQUENCE followed by an AS_SET. For example, the aggregate of paths $P = 1, 2, 3, 5$ and $Q = 2, 3, 4, 5$ should look like $A = 2, 3, \{1, 4, 5\}$.

In addition, the algorithm tries to be consistent in the assembling and keep meaningful information. For example, it creates AS_PATHs whose length is equal to the path length

Table 4.1: K-BESTRO Algorithm

1.-	Keep paths with highest LOCAL_PREF value
2.-	Look for the shortest AS path, store the length in *shortest-l* and keep paths with AS_PATH_LENGTH<=*shortest-l*+ELMP.
3.-	Keep paths with lowest ORIGIN value
4.-	For each advertising AS,
	4.a.-Look for the subset of paths with lowest AS path length
	4.b.-Select the lowest MED value in that subset
	4.c.-Delete the paths from that AS with different MED value
5.-	If there is a remaining path with session TYPE eASSM, delete paths with TYPE iASSM
6.-	Rank the paths according to,
	6.a.-Paths with shortest AS_PATH_LENGTH go first
	6.b.-If a subset paths have the same length, paths with lowest internal cost goes first Discard paths within the subset with internal cost>lowest-cost+ECMP (Default ECMP=0)
	6.c.-If equal cost, lowest BGP_ID goes first
	6.d.-If same BGP_ID, lowest peer address goes first
7.-	If the first ranked path has the NO_EXPORT:X Community,
	7.a.-Then select only the first KBEST ranked path with the same NO_EXPORT:X Community
	7.b.-Else, delete paths with any NO_EXPORT:X Community
8.-	If the ranked paths have session TYPE iASSM, select the first $KBEST$ paths
9.-	Else if the ranked paths have session TYPE eASSM, select the first $KBEST$ paths from the same AS and MED value as the first ranked path
10.-	Return the selection. K-BESTRO ENDs

BGP would advertise. Thus, using assembling neither represents a penalty nor an advantage to multipath nodes, what we believe is fair. Moreover, the algorithm also preserves the last AS added to the AS_PATH as it is consider meaningful in some policies (e.g. neighboring AS filtering). The algorithm does not preserve the position of the origin AS, given that typically ASes rely on RIRs to check the origin and ASes included in an advertisement [22].

Table 4.2: Assembling Algorithm

1.-	Create an empty AS_SEQUENCE and AS_SET.
2.-	Pick up the shortest path from the multipath set and initialize *shortest* to its AS_PATH_LENGTH.
3.-	Copy the most to the left AS_NUMBER of the shortest path into the AS_SEQUENCE.
4.-	Keep parsing the AS_NUMBERs in the shortest path (if repeated, process it only once): Check its presence in other paths in the set.
4.a.-	If present in all paths and after AS_NUMBERs already in the AS_SEQUENCE, concatenate it to the AS_SEQUENCE.
4.b.-	Else add it to the AS_SET
5.-	Append the AS_SET at the end of the AS_SEQUENCE to create the AS_PATH.
6.-	For each remaining path, parse every AS_NUMBER. If the number has not been previously added to the AS_PATH, add it to the AS_SET.
7.-	Compare the length of the resulting path, if longer than *shortest* run 7.a, otherwise run 7.b,
7.a.-	Starting from the most to the right AS_NUMBER in the AS_SEQUENCE, move as many AS_NUMBERs into the AS_SET until the path length is equal to *shortest*.
7.b.-	Append at the beginning of the AS_SEQUENCE the most to the left AS_NUMBER as many times as needed until the path length is equal to *shortest*.
8.-	If the assembled path is advertised through an iASSM sessions run 8.a, run 8.b otherwise,
8.a.-	Return the resulting AS_PATH
8.b.-	Append the local AS_NUMBER at the beginning of the path and return the resulting AS_PATH

The algorithm is displayed in Table 4.2. The AS_SEQUENCE is constructed with the AS_NUMBERs common to all the paths in the multipath set as suggested in RFC4271. The order between AS_NUMBERs is kept. If two AS_NUMBERs X and Y appear always one after the other in every path (even though some AS_NUMBERs may appear in the middle), they are said to be in order (rule 4.a). If two ASes, common to all paths, are not in order, then the second in appearance within the shortest path is put in the AS_SET (rule 4.b). The *assembled* AS_PATH is the result of concatenating the AS_SET at the end of the AS_SEQUENCE (rule 5). The remaining AS_NUMBERs not common to every path are added to the AS_SET (rule 6). Afterwards, rules 7.a-b check that the AS_PATH_LENGTH of the resulting AS_PATH is exactly the same as the shortest path in the multipath set. Rules 8.a-b deal with the fact that the local AS_NUMBER is not added to the advertisements until it is propagated to a peering router outside the AS.

4.3 Example: An ASSEMBLER-Capable Autonomous System

This example refers always to the AS depicted in Fig.3.1. The figure represents a transit AS (AS1) with three customers (AS2,AS3,AS4), one peer (AS5) and two providers (AS6 and AS7) connected to AS1 by means of ASSEMBLER-capable routers. Router (R5,R7,R9) set (ELMP=0,ECMP=0,KBEST=1) and (R2,R4,R6,R8) aggregate the maximum number. Routers establish a full-mesh of intra-AS peering sessions to redistribute routing information. The example present two cases, a prefix propagated downstream from providers to customers and another prefix propagated upstream from the customers.

4.3.1 Downstream Advertisement

In the first case, two paths are advertised to AS6 and AS7 towards 150.0/16. The paths are propagated further and four paths reach AS1 and all of them are assigned with the same local preference. The egress routers for the paths in AS1 are R6 and R7. The router R6 can aggregate up to three paths depending on the ELMP parameter. If ELMP=0, then only the path from AS7 is selected according to rule 3 in Table 4.1. Otherwise, if ELMP=1 or higher the three paths can be aggregated as depicted in the figure. The aggregated path from R6 is constructed using the algorithm in Table 4.2. The first AS_NUMBER of the shortest path, AS7 leads the AS_SEQUENCE. Only AS9 is common to all paths and it is aggregated to the AS_SEQUENCE, as well. The remaining ASes are added to the AS_SET, AS6 and AS10. The length of the aggregated path is checked and it is 3 where it should be 2, therefore AS9 is moved into the AS_SET to compensate the length. Finally, the update is re-advertised through iASSM. Router R7 is unipath and selects only the path through AS7. Routers R2 and R8 receives both announcements from R6 and R7. The announcements have equal AS path length and equal IGP cost, therefore they are aggregated by the routers, although in this case the resulting aggregated paths are the same as for R6. Routers R4 and R9 are advertised as well. Both paths from R6 and R7 have the same AS_PATH_LENGTH therefore the ranking is done according to the IGP cost. Routers R4 and R9 has its ECMP parameter equal to 0 and consider only paths with lowest internal cost. Both, R4 and R9 select in this case the path through R7. Routers R2,R4 and R9 propagate the paths towards AS1 clients adding the AS_NUMBER 1 to the advertised AS_PATH.

4.3.2 Upstream Advertisement

In this second case, a couple of customers of AS1, AS3 and AS4 advertise a path towards their customer AS30. The effect of the MED values on the ranking function is shown in this second case. AS3 advertises two paths towards AS30, one to R2 with MED=20 and another one to R4 with MED=10. AS4 does not use MED values. Every path is assigned with the same local preference. Hence, three routers end up with a path from eBGP sessions and redistribute them through iASSM. The router R4 discards the internal path through R2-AS3 with highest MED and selects the eASSM path from AS3 and from AS4. Nevertheless, the two paths cannot be aggregated since the path from AS3 has MED value, therefore only the path through AS3, which has a lower BGP_ID, is ranked first and selected. R2 discards its own eASSM path an selects the internal path through R4 with the lowest MED for AS3. Assuming there is no restriction for R2 on the IGP cost to R9, it can aggregate the internal path through R9 as well, since the aggregated path is advertised only through eASSM sessions and the MED values are not taken into account.

Chapter 5

Deployment Considerations

When it comes to the deployment of ASSEMBLER in a real AS there are some considerations to be taken into account beforehand. This chapter outlines them and points out how the main shortcomings that may appear during the deployment can be solved using the appropriated settings. As mentioned during the introduction and shown in the previous chapter, ASSEMBLER does not required of any kind of coordination between ASes to take advantage of multiple paths with high flexibility. Therefore, the deployment issues may rise while deploying it inside an AS. The issues are collected in three categories, problems related to mixed configurations, inconsistent multipath routing policies and traffic engineering techniques.

5.1 Deployments with Legacy Routers

The incremental deployment when legacy routers are present depends on the intra-AS technique used to forward the traffic. Two different types of intra-AS techniques to forward the traffic are considered: internal redistribution and tunnelling. Internal redistribution implies that every router inside the AS understands reachability information and fills in the routing tables accordingly. An internal protocol such as iBGP/iASSM is required to perform the redistribution. Tunnelling techniques rely on the encapsulation of packets. Only the ingress and the egress routers need to be aware of the paths advertised to the AS. In this discussion we consider IP over IP tunnelling and MPLS tunnelling [29] as representative techniques.

If the AS performs a full deployment, such that every legacy router is replaced inside the AS, both internal redistribution and tunnelling can be used without any kind of limitation. The only potential advantage of tunnelling is the addition of eiBGP configurations [6] but that topic is out of the scope of this work. On the other hand, if the AS is not planning a full upgrade of the network, ASSEMBLER can still be deployed progressively. The difference in this case is that any router in the network can be randomly replaced if tunnelling is used, whereas special attention must be paid for internal redistribution. Using redistribution and legacy routers, ASSEMBLER routers must not aggregate paths received from internal peering sessions. The reason is that that internal aggregation may lead to routing inconsistencies. For instance in Fig.3.1, assuming that R2, R6 and R7 are unipath routers, if R2 receives the paths from R6 and R7 through the iASSM sessions and chooses the one from R6. R8 is multipath and aggregates the paths through R6 and R7, however it does not advertise the aggregate through iASSM to R2 to avoid internal loops RFC4271. If the IGP path between R2 and R6

passes through R8. The router R2 announces to the AS2 and AS3 its choice through AS6, however when packets get to R8, the multipath router can forward some of them towards R3 which is inconsistent with the network view that R2 is advertising.

5.2 Multipath Routing Policies

In addition to the considerations regarding the deployment of ASSEMBLER along with legacy routers, some other issues may arise related to the policy configurations of the ASSEMBLER routers. Defining simultaneously import and export policies for several paths that match a given criteria is supported nowadays in regular BGP routers. For instance, Cisco IOS uses the *route-maps* to define policies for several paths at once. ASSEMBLER-capable routers should support the same definition of policies. A BGP router can be transparently replaced and provide the same functionality configuring the same *route-maps* and setting K-BESTRO with the combination (ELMP=0,ECMP=0,KBEST=1).

However, the combination of policies for multiple paths with more lax K-BESTRO configurations may lead to inconsistent states regarding the export of paths. In contrast to BGP, the ASSEMBLER decision process yields a set of KBEST paths instead of a *winner* path. If several paths are assigned with the same LOCAL_PREF, and the import policy is not designed appropriately, a path coming from a provider may end up in the selected multipath set with a path coming from a customer, which cannot be exported together. No ASSEMBLER advertisement is generated towards the providers, although BGP would advertise the path from the customer. Therefore, LOCAL_PREF assigned by the import policy of a router should be the same only for paths coming from the same type of neighbor AS.

5.3 Enhanced Traffic Engineering

Once several paths are selected and advertised by an ASSEMBLER router they can be used simultaneously to forward packets. Outgoing traffic engineering (hereafter TE) is usually based on the local preference defined in the import policy. Nevertheless, in BGP only one path at a time is selected and the most flexible outgoing TE techniques are load-sharing [8] and tuning of IGP costs. Multipath BGP defined in [6, 18] allows for load-balancing among multiple parallel connections between two ASes but they only support load sharing to split the traffic among multiple ASes. ASSEMBLER allows to perform load-balancing across multiple ASes and in contrast to the proposals in [6, 18], the traffic can be switched from one egress AS to another without re-advertising additional routing information. In addition, how much traffic balance over each path becomes a new TE parameter.

Regarding TE using IGP costs, currently ASes send the traffic to the closest egress point in the network, which can be used as a form of performing TE. ASSEMBLER extends this TE technique and administrators can modify the ECMP parameter of K-BESTRO to define how close to this hot-potato routing they want to stick to.

On the other hand, the most widespread incoming TE techniques acccording to [4, 26] are prefix deaggregation, path prepending and TE with BGP Communities. Prefix deaggregation and BGP Communities for TE are supported. Yet, in order to respect the TE performed by neighbor ASes using path prepending, a maximum value must be setup for the ELMP parameter. That maximum value does not have to be public since in practice downstream

ASes tune the amount of AS numbers to prepend on a trial and error basis [26].

Chapter 6

Stability Analysis

Even though several equally preferred paths are available, BGP routers selects just one path. According to the results in [5], the fact that BGP converges to a stable solution does not guarantee that the network is stable if routers select several paths instead. The relaxation of the selection process may trigger oscillations that did not happen before in the unipath case. Those results are the main motivation of this section, since ASSEMBLER selects multiple paths and additional information is advertised between routers. The goal of our stability analysis is to show that the deployment of ASSEMBLER in a network does not affect the stability.

The section starts with a discussion to motivate the stability analysis. Afterwards, the stability of ASSEMBLER is studied. In order to prove the stability of ASSEMBLER, we extend the network model in [5] to include *assembled* paths. Afterwards, the stability condition of ASSEMBLER is derived. Once that condition is stated, the guidelines for routing policies proposed in [10] are reformulated for multipath scenarios. The condition is used to prove their stability.

6.0.1 On Dispute Wheels in Unipath and Multipath Scenarios

The goal of this section is to introduce a discussion about the impact of multipath in the stability of policy-based routing protocols. Some notation is introduced before the discussion. The notation comes from the framework defined in [5].

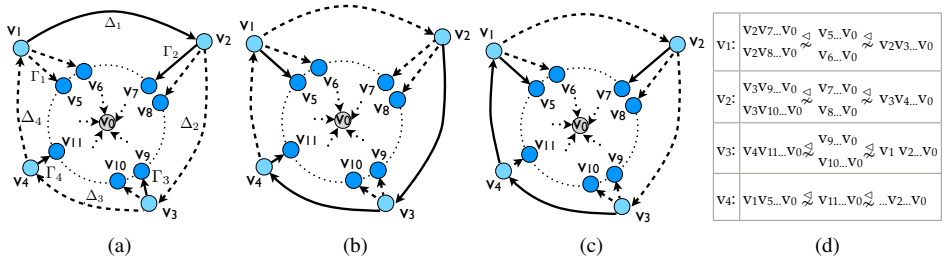


Figure 6.1: Unipath Dispute Wheel

In policy-based scenarios where a path vector protocol is running, paths propagate from one node to another as they are chosen in the ranking procedure as most preferred and announced to the routers with peering sessions. When a path P is preferred over a path Q according to the preferences of a node, that relationship of *preference* can be denoted like,

$$P \overset{\triangleleft}{\precsim} Q \tag{6.1}$$

If a path P is announced and it is chosen as most preferred by an arbitrary number of nodes $v_{i+n}, \ldots, v_{i+1}$ in the network, the assigned path to v_{i+n} is a propagated version of P, i.e. $P' = v_{i+n}, \ldots, v_{i+1}, v_i, \ldots, v_0 = (v_{i+n}, \ldots, v_{i+1})P$ and its relationship of *composition* with P can be expressed like,

$$P \overset{\blacktriangleleft}{\precsim} P' \tag{6.2}$$

Using these two simple concepts different relations among the policies of the different nodes and the paths announced by them can be denoted. A particular type of relations between paths caused by routing policies are the *reflexive* relations. Hereby, just a fair definition is introduced for the shake of clarity. For a formal and rigorous definition of *anti-reflexive* and *reflexive* relations see [5]. If there is an alternating sequence of preference ($\overset{\triangleleft}{\precsim}$) and composition ($\overset{\blacktriangleleft}{\precsim}$) relations created among a set of paths $P_1, P_2, \ldots, P_n, Q_1, Q_2, \ldots, Q_n$, it is said that the relation is a reflexive relation if there is a path for which the sequence is cyclic, e.g.

$$P_1 \overset{\blacktriangleleft}{\precsim} Q_n \overset{\triangleleft}{\precsim} P_n \overset{\blacktriangleleft}{\precsim} \ldots \overset{\triangleleft}{\precsim} \ldots \overset{\blacktriangleleft}{\precsim} Q_2 \overset{\triangleleft}{\precsim} P_2 \overset{\blacktriangleleft}{\precsim} Q_1 \overset{\triangleleft}{\precsim} P_1 \tag{6.3}$$

Reflexive relations cannot be fulfilled simultaneously. For instance the path Q_n composed by an arbitrary path and P_1 is more preferred than P_1, which is a contradiction since when Q_n selected P_1 is not selected and Q_n is feasible only if P_1 is selected. Hence, the protocol cannot find a solution and it is likely to oscillate.

The first reflexive relations depicted in the literature for policy-based routing are the BGP *dispute wheels* shown in [12]. Although the analysis in [12] shows the stability results using an abstraction of the ranking function, which covers all the different rankings that can be configured for ASSEMBLER, the analysis does not cover the case in which multiple paths are ranked as most preferred instead of just one. The framework in [5] considers multiple equally preferred paths and provided that our ultimate goal is to analyze the stability in mixed environments in which multipath nodes coexist with unipath nodes, it is interesting to study the stability of the protocol under that framework.

The condition used in the framework to guarantee existence of multipath solution is that the preferences assigned by the ranking functions do not create a reflexive relation among the paths propagated throughout the network [5]. Since the generation of a dispute wheel involves a specific relation among the ranking functions, by changing the ranking functions and announcing additional paths, it is expected that the relation among them changes as well. An stable state may no longer be reachable, or it is under a different solution. Before addressing the formal analysis of the problem, two examples are provided to show that the propagation of additional paths can provide a network running ASSEMBLER with an stable solution in unstable unipath scenarios, whereas in stable cases it can activate a dispute wheel.

Example 1 In Fig.6.1 a network is depicted in which every node selects only one path like BGP. The path ranked in first position by each node is displayed with a solid arrow. Feasible paths lower rank are displayed in dashed arrows. The preferences of each node are shown

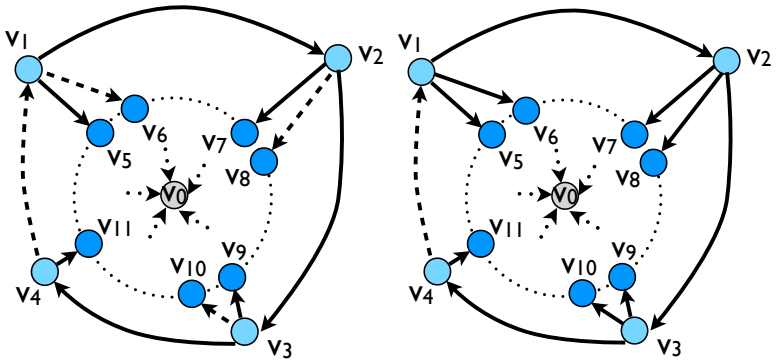
in the table at Fig.6.1d. In Fig.6.1a the node v_1 receives three paths through v_2, v_5 and v_6 respectively. The three paths are of the same AS length and v_1 chooses the path $v_2v_7 \dots v_0$ with the lowest BGP_ID. The node v_2 has three paths of equal path length through v_7, v_8 and v_3. It is not aware yet of the path $v_3v_9 \dots v_0$ and chooses to go through v_7 using the lowest BGP_ID criteria. Node v_3 is configured in a similar way, it chooses v_9 as next-hop since it is not aware of the path $v_4v_{11} \dots v_0$. The node v_4 receives a path through v_{11} but it is not aware of the path $v_1v_5 \dots v_0$, therefore it chooses v_{11} even though its highest preference is to use $v_1v_5 \dots v_0$. In addition, v_4 filters any AS path containing v_2.

In Fig.6.1b, v_2 becomes aware of the path through v_3 and changes its assignment, forcing v_1 to change its path as well. Node v_1 does not select the new path of v_2 because is longer than the path through v_5. However, v_3 becomes aware of the path through v_4 and changes as well. The path assignment is again modified in Fig.6.1c. Node v_2 looses its path $v_2v_3v_9 \dots v_0$ as it is longer than $v_2v_7 \dots v_0$ and v_4 prefers the path announced by v_1. In the next step, the nodes go back to the initial assignment shown in Fig.6.1a completing a cycle in the oscillation. The reflexive relation can be expressed in this case as follows,

$$\begin{aligned} v_1v_5 \dots v_0 \overset{\blacktriangleleft}{\precsim} v_4v_1v_5 \dots v_0 \overset{\triangleleft}{\precsim} v_4v_{11} \dots v_0 \overset{\blacktriangleleft}{\precsim} \\ \overset{\blacktriangleleft}{\precsim} v_3v_4v_{11} \dots v_0 \overset{\triangleleft}{\precsim} v_3v_9 \dots v_0 \overset{\blacktriangleleft}{\precsim} v_2v_3v_9 \dots v_0 \overset{\triangleleft}{\precsim} \\ \overset{\triangleleft}{\precsim} v_2v_7 \dots v_0 \overset{\blacktriangleleft}{\precsim} v_1v_2v_7 \dots v_0 \overset{\triangleleft}{\precsim} v_1v_5 \dots v_0 \end{aligned} \tag{6.3}$$

If the K-BESTRO selection algorithm is tuned to select 2 equal-length AS path, the scenario becomes stable. Every node chooses the path through one neighbor node on the dashed circle and a path through an outer neighbor except for v_4. For instance, v_1 selects the paths through v_5 and v_2. Fig.6.2a shows the final path assignment. Node v_4 ranks only the path through v_{11} since the advertisement from v_1 contains v_2, so that v_4 assigns a lower preference.

Finally, if ASSEMBLER neither constrains the path length nor the amount of paths, the stable path assignment displayed in Fig.6.2b can be achieved.



(a) 2-Best Equal Path Length (b) Multiple Unipath Solutions

Figure 6.2: Examples of the effect of relaxing the maximum number of paths

Example 2 In this second example we want to show that a particular configuration for which there is unipath solution but no multipath. Nodes v_1 and v_2 are running ASSEMBLER

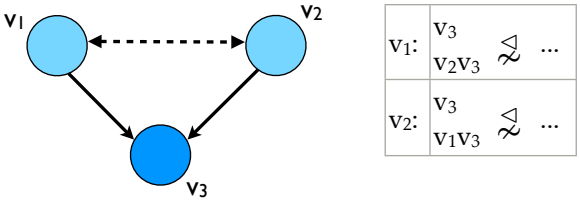


Figure 6.3: Scenario with unipath but not multipath solution.

to select maximum 2 paths with a AS path length difference of one. Assume that, v_1 and v_2 give the same preference to their peering connection than to the connection to the customer that is multihomed to them. The policies in that case can be expressed as in the table on the right side of Fig.6.3. The reflexive relation is in this case,

$$v_1 v_0 \overset{\blacktriangleleft}{\precsim} v_2\{v_1, v_0\} \overset{\triangleleft}{\precsim} v_2 v_0 \overset{\blacktriangleleft}{\precsim} v_1\{v_2, v_0\} \overset{\triangleleft}{\precsim} v_1 v_0 \tag{6.4}$$

The reflexive relation is created due to the fact that both nodes prefer the aggregated of the direct and indirect paths, to only the direct path. Using equal-length multipath there is solution, in which ASSEMBLER assign only direct paths to each node.

Those two examples lead to the conclusion that the propagation of additional paths can either stabilize a network running ASSEMBLER or activate a reflexive relation. Therefore, it is necessary to analyze the stability of ASSEMBLER, since the stability results inferred for BGP do not apply.

6.0.2 Synchronous Model of Path ASSEMBLER

Let $G = \langle V, E \rangle$ be a topology graph, where V is the set of vertex and E is the set of edges of the graph and $v_0 \in V$ denotes the origin of a prefix advertisement. Let $\mathcal{P}(v_i, v_0)$ be the set of *reachable* paths between v_i and v_0 in G, i.e. any path that can be physically constructed from v_i to v_0. Now $\mathbb{P}(\mathcal{P}(v_i, v_0))$ is defined as the super-set of the subsets in $\mathcal{P}(v_i, v_0)$, so to speak, every element $\Phi_{v_i} \in \mathbb{P}(\mathcal{P}(v_i, v_0))$ is an arbitrary collection of elements in $\mathcal{P}(v_i, v_0)$. A multipath assignment over G is defined as,

$$\mathbf{\Phi} = \{\Phi_{v_i} \in \mathbb{P}(\mathcal{P}(v_i, v_0)), \forall v_i \in V\} \tag{6.5}$$

Additionally, a partial multipath assignment is defined as,

$$\mathbf{\Phi} = \{\Phi_{v_i} \in \mathbb{P}(\mathcal{P}(v_i, v_0) \cup \emptyset), \forall v_i \in V\} \tag{6.6}$$

that is, for some nodes the assignment may be empty. The protocol is modelled as a fixed-point iteration of a distributed synchronous Bellman-Ford mapping $\boldsymbol{\mathcal{F}}(\mathbf{\Phi})$ over the multipath assignment $\mathbf{\Phi}$.

The protocol starts growing from the initial iteration, in which the origin v_0 announces the path containing itself to its neighbors, and it increases the path assignment until reaching an assignment $\mathbf{\Phi}$ that verifies the fixed-point equation,

$$\mathbf{\Phi} = \boldsymbol{\mathcal{F}}(\mathbf{\Phi}) \tag{6.7}$$

In the ASSEMBLER mapping, nodes exchange announcements constructed as depicted in Section 4.2. The most recent advertisement received by node v_i from node v_j at iteration k is denoted as

$$adv(v_j \rightarrow v_i)_{[k]} \tag{6.8}$$

Hereafter, we use indistinctly the terms path and assembled path, (a path is a particular case of an assembled path in which only one element is assembled). The set of available paths for a node v_i at iteration k is the set,

$$\Phi_{v_i[k]} = \{adv(v_j \rightarrow v_i)_{[k-1]}, \forall v_j \in peers(v_i)\} \tag{6.9}$$

The ASSEMBLER mapping is defined locally at node v_i as the operation of selecting the K best assemblable paths in $\Phi_{v_i[k]} \in \mathbb{P}(\mathcal{P}(v_i, v_0))$ according to the K-BESTRO algorithm (Section 4.1) and propagate the ASSEMBLER advertisement corresponding to them. The analysis assumes that each node defines a different K-BESTRO configurations and the following notation is used to differentiate among K-BESTRO procedures, so that $\mathcal{F}_{v_i}(\Phi_{v_i[k]})$ denotes the procedure at node v_i. The mapping can be also defined as the set of local operations at each vertex during the k^{th} iteration like,

$$\boldsymbol{\mathcal{F}}(\boldsymbol{\Phi}_{[k]}) = \left\{\mathcal{F}_{v_i}(\Phi_{v_i[k]}), \quad v_i \in V\right\} \tag{6.10}$$

Before advertisement the candidate paths are ranked. The rank value of a path is denoted like $\lambda(\theta)$ and all the paths in the selected set has a ranking value of $\lambda_{max}(\Phi_{v_i[k]})$.

Given the set of advertisements in Φ_{v_i} and the ranking procedure at v_i, $\mathcal{F}_{v_i}$ (which establishes ranking values λ, its maximum for a given set of announcements λ_{max}), the set of most preferred paths at iteration k can be defined as,

$$\beta_{v_i[k]} = \mathcal{F}_{v_i}(\Phi_{v_i[k]}) \tag{6.11}$$

where,

$$\beta_{v_i[k]} = \{\theta \in \Phi_{v_i[k]} \ / \ \lambda(\theta) = \lambda_{max}(\Phi_{v_i[k]})\} \tag{6.12}$$

Each node v_i, its candidate set Φ_{v_i} is updated with the advertisements $adv(v_j \rightarrow v_i)_{[k]} \equiv \beta_{v_i,[k]}$ from the peering routers, such that the updated overall path assignment is defined like

$$\boldsymbol{\Phi}_{[k+1]} = \{\Phi_{v_i,[k+1]}, \forall v_i \in V\} \tag{6.13}$$

and according to Eq.6.10 we get to the iterative equation

$$\boldsymbol{\Phi}_{[k+1]} = \boldsymbol{\mathcal{F}}(\boldsymbol{\Phi}_{[k]}) \tag{6.14}$$

As the synchronous execution of the mapping goes on, the paths in $\Phi_{v_i[k]}$ change dynamically. Before expressing those dynamic changes and the evolution of the mapping in a formal way, first we have to define the concepts of *feasible* and *stabilized* set of paths.

The concept of feasible multipath assignment is rather intuitive if we define the set of all possible multipath assignments as the following Cartesian product (including partial assignments),

$$\chi = \prod_{v_i \in V} \mathbb{P}(\mathcal{P}(v_i, v_0) \cup \emptyset) \tag{6.15}$$

Therefore a multipath assignment $\mathbf{\Phi} = (\Phi_{v_0}, \Phi_{v_1}, \ldots, \Phi_{v_n}) \in \chi$ provides to each vertex v_i a set of paths Φ_{v_i} to reach the origin. In addition, for each vertex in V, we define the following,

Definition 1 Given two vertex $v_i, v_j \in V$ such that $(v_i\ v_j) \in E$, the assignment Φ_{v_i} is said to be *consistent* with Φ_{v_j} if $\forall \rho \in \Phi_{v_i}$ of the form $\rho = (v_i\ v_j)\theta$, it holds that $\theta \in \Phi_{v_j}$ and $v_i \notin \theta$ (i.e. to ensure loop-freeness).

It seems clear from the definition of χ that not all the components of $\mathbf{\Phi} \in \chi$ are necessarily consistent with each other. Since our protocol handles only local information, the paths that it is able to construct must be consistent for all the vertex in the path. Hence, the definition of a feasible multipath assignment for our protocol can be expressed like,

Definition 2 A multipath assignment $\mathbf{\Phi} \in \chi$ is said to be *feasible* if $\forall v_i, v_j \in V$ and $(v_i\ v_j) \in E$ then Φ_{v_i} is consistent with Φ_{v_j}.

Before defining the concept of *stabilized* multipath assignment, the following relations between multipath feasible assignments must be defined,

Definition 3 Let $\mathbf{\Phi}, \mathbf{\Phi}' \in \chi$ be two feasible partial multipath assignments then, $\mathbf{\Phi}'$ contains $\mathbf{\Phi}$, i.e. $\mathbf{\Phi} \subseteq \mathbf{\Phi}'$, if $\Phi_{v_i} \subseteq \Phi'_{v_i} \quad \forall v_i \in V$ and $\mathbf{\Phi} \subsetneq \mathbf{\Phi}'$, if $\mathbf{\Phi} \subseteq \mathbf{\Phi}'$ and $\Phi_{v_i} \subsetneq \Phi'_{v_i}$ for some $v_i \in V$.

Definition 4 Given a partial feasible multipath assignment $\mathbf{\Phi}$, the set $\mathbf{\Psi}(\mathbf{\Phi})$ defined as,

$$\mathbf{\Psi}(\mathbf{\Phi}) = \{\mathbf{\Phi}' \in \chi \ / \ \mathbf{\Phi} \subseteq \mathbf{\Phi}'\} \tag{6.16}$$

is the set of feasible assignments which contain the path assignment $\mathbf{\Phi}$.

Definition 5 An assignment $\mathbf{\Theta}_{[k]}$ is said to be *stabilized* if for all the sets of feasible sets containing $\mathbf{\Theta}_{[k]}$, i.e. $\forall \mathbf{\Phi} \in \mathbf{\Psi}(\mathbf{\Theta}_{[k]})$, it holds that

$$\mathbf{\Phi} \supseteq \mathbf{\Theta}_{[k]} \quad \text{implies} \quad \boldsymbol{\mathcal{F}}(\mathbf{\Phi}) \supseteq \mathbf{\Theta}_{[k]} \tag{6.17}$$

The latter means that for any feasible assignment $\mathbf{\Phi}$ containing $\mathbf{\Theta}_{[k]}$, an iteration of ASSEMBLER over the assignment $\mathbf{\Phi}$ does not remove any path in $\mathbf{\Theta}_{[k]}$. Therefore, any path $\theta \in \mathbf{\Theta}_{[k]}$ is part of the fixed-point solution of Eq.6.7 for the function $\boldsymbol{\mathcal{F}}$ and its ranking value verifies that

$$\lambda(\theta) = \lambda_{max}(\mathbf{\Psi}(\mathbf{\Theta}_{[\mathbf{m}]})) \quad \forall m \geq k \tag{6.18}$$

Definition 6 Let $C_{[k]}$ be the set of *converged* nodes at the k^{th} iteration. Any node $v_i \in C_{[k]}$ verifies that the set $\beta_{v_i[k]}$ belongs to the stabilized assignment at iteration k, what means that v_i converged at iteration k or before. Being $m \leq k$ the iteration at which v_i converged, then $\forall n > m$, $\beta_{v_i[n]} = \beta_{v_i[m]}$ and, therefore $adv(v_i \rightarrow w)_{[n]} = adv(v_i \rightarrow w)_{[m]}$.

Definition 7 Let $D_{[k]} \subseteq V - C_{[k]}$ be the set of nodes which are *direct peers* of converged nodes, then, $\forall v \in D_{[k]}$, $\exists u \in C_{[k]}$ and $e = (v\ u) \in E$.

6.0.3 Path ASSEMBLER Convergence

In this section, the anti-reflexive property of routing policies, the notation and the protocol model depicted in the previous section are combined to asses the stability of ASSEMBLER. The analysis begins with the progress condition of the protocol. We show that if the progress condition does not hold for some iteration of the protocol, then the policies create a reflexive relation among the advertised paths. Afterwards, we prove that the progress condition implies that at each iteration the protocol gets closer to the fixed-point solution. Finally, the safety of the protocol is shown at the final theorem stated in this section.

Lemma 1 (Progress Condition) Let S be the set of routing policies of nodes in G. If any relation among policies in S is anti-reflexive and the current overall stabilized assignment $\boldsymbol{\Theta}_{[k]}$ is not a fixed-point of the mapping, then there is an assignment $\boldsymbol{\Theta}_{[k+1]}$ such that,

1. $\boldsymbol{\Theta}_{[k+1]} \supsetneq \boldsymbol{\Theta}_{[k]}$
2. $\boldsymbol{\Theta}_{[k+1]}$ is also stabilized
3. $\forall \boldsymbol{\Phi} \in \boldsymbol{\Psi}(\boldsymbol{\Theta}_{[k]})$, then $\boldsymbol{\mathcal{F}}(\boldsymbol{\Phi}) \supseteq \boldsymbol{\Theta}_{[k+1]}$

Proof: If $\boldsymbol{\Theta}_{[k]}$ is stabilized it means that $\boldsymbol{\mathcal{F}}(\boldsymbol{\Theta}_{[k]}) \supseteq \boldsymbol{\Theta}_{[k]}$ and $\mathcal{F}_{v_0}(\boldsymbol{\Theta}_{[k]}) = \{v_0\}$, $k \geq 0$. In order to increase the multipath stabilized assignment there must be at least one node $v \in D_{[k]}$ peer of $u \in C_{[k]}$ such that,

1. By definition 6 u has a stabilized set $\Theta_{u[k]}$, then $\forall \theta \in \Theta_{u[k]}$ it holds $\theta \in \boldsymbol{\Theta}_{[k]}$
2. Given $\rho = (v\ u) \in E$, $\alpha = adv(u \rightarrow v)_{[k]}$, it holds that

$$\lambda(\rho\alpha) = \lambda_{max}(\boldsymbol{\Psi}(\Theta_{v[m]})) \quad \forall m \geq k \tag{6.19}$$

   hence $\rho\alpha \in \Theta_{v_i[k+1]}$ (i.e. $\rho\alpha$ is stabilized since no path with higher rank will replace it in later iterations).
3. At iteration k, $\rho\alpha \in \boldsymbol{\Theta}_{[k+1]} = \boldsymbol{\mathcal{F}}(\boldsymbol{\Theta}_{[k]})$ and $\rho\alpha \notin \boldsymbol{\Theta}_{[k]}$

If such a node v exists then the proof of the lemma is completed since by construction $\boldsymbol{\mathcal{F}}(\boldsymbol{\Theta}_{[k]}) \supsetneq \boldsymbol{\Theta}_{[k]}$. By definition 5 and Eq.6.19, $\rho\alpha$ will not be removed in further iterations, therefore $\boldsymbol{\mathcal{F}}(\boldsymbol{\Theta}_{[k]})$ is stabilized.

Now we show that if that node $v \in D_{[k]}$ does not exist then the anti-reflexivity property does not hold over the policies in S. If v does not exists then no node $v_1 \in D_{[k]}$ is be able to find a *direct* path Γ_1 constructed like above $\Gamma_1 = \rho\alpha$ for any peer node $u \in C_{[k]}$ and being $\lambda(\Gamma_1) = \lambda_{max}(\boldsymbol{\Psi}(\Theta_{v[m]}))\ \ \forall m \geq k$. Therefore, v_1 prefers more a path Δ_1 that is not through a converged peer. Using the *preference* and *composition* relations, we can express the policy relation between Γ_1 and Δ_1 like,

$$\Delta_1 \precnsim \Gamma_1 \tag{6.20}$$

Then if Δ_1 is not at one hop to a converged vertex, then Δ_1 must come from a propagated version of a direct path of some node $v_2 \in D_{[m]}$, therefore it can be constructed like $\Delta_1 = \Pi_2\Gamma_2$. Path Π_2 is an arbitrary path passing through nodes in $V - C_{[m]}$ and $\Gamma_2 = \rho'\alpha'$, with $\rho' = (v_2\ u') \in E$ and $\alpha' = adv(u' \rightarrow v_2)_{[m]}$, is a direct path of v_2. In terms of policy relations the latter can be expressed like,

$$\Gamma_2 \overset{\blacktriangleleft}{\precsim} \Delta_1 \overset{\lhd}{\precsim} \Gamma_1 \tag{6.21}$$

Using the same reasoning, v_2 is not choosing any direct path Γ_2, otherwise the path Δ_1 would become stabilized and the stabilized paths assignment would grow. Therefore the set $\beta_{v_2[m]}$ is formed by at least one path Δ_2 which is not direct and goes through a direct path announced by some node $v_3 \in D_{[m]}$. The same procedures repeats for v_3 and we get to the relation,

$$\Gamma_3 \overset{\blacktriangleleft}{\precsim} \Delta_2 \overset{\lhd}{\precsim} \Gamma_2 \overset{\blacktriangleleft}{\precsim} \Delta_1 \overset{\lhd}{\precsim} \Gamma_1 \tag{6.22}$$

The relation keeps repeating for every element in $D_{[m]}$ until it hits v_1 again, producing a circular relationship of policies that cannot be fulfilled simultaneously,

$$\Gamma_1 \overset{\blacktriangleleft}{\precsim} \Delta_n \overset{\lhd}{\precsim} \ldots \overset{\lhd}{\precsim} \Gamma_3 \overset{\blacktriangleleft}{\precsim} \Delta_2 \overset{\lhd}{\precsim} \Gamma_2 \overset{\blacktriangleleft}{\precsim} \Delta_1 \overset{\lhd}{\precsim} \Gamma_1 \tag{6.23}$$

The latter relation implies that Γ_1 is less preferred than a path which is composed by an arbitrary path and Γ_1, which is a contradiction. The latter completes the proof by showing that if the protocol gets stuck, then a reflexive relation exists among the policy relations.

Lemma 2 If a path $\theta = v_i v_{i-1} \ldots v_0$ does not appear infinitely often in the multipath set β_{v_i} of v_i, then there is an iteration k after which any path of the form $\rho\theta$ disappears from the network.

Proof: Given a vertex v_i, $\theta = \rho'\theta'$ with $\rho' = v_i v_{i-1} \ldots v_{j+1} v_j$ and $\theta' = v_j v_{j-1} \ldots v_1 v_0$, if $\theta = \rho'\theta'$ does not appear in $\beta_{v_i[m]} \quad \forall m \geq k$ it means that there is at least one node $v_j \quad 0 \leq j \leq i$, for which there is a path $\theta'' \in \mathcal{P}(v_j, v_0)$ such that $\lambda(\theta'') > \lambda(\theta')$, therefore θ' is not part of $adv(v_j \rightarrow v_{j+1})_{[k]}$ after iteration k. At iteration $k+1$ the nodes $w \in peers(v_j)$ cannot use the path θ' any longer. The process repeats at each iteration along the next-hop in the path ρ' until $\rho'\theta'$ disappears. Thus, v_i cannot announce θ any longer and eventually $\rho\theta$ also disappears.

Lemma 3 The successive iterations of the ASSEMBLER mapping $\boldsymbol{\mathcal{F}}(\boldsymbol{\Theta}_{[0]})$, $\boldsymbol{\mathcal{F}}(\boldsymbol{\Theta}_{[1]})$, ..., $\boldsymbol{\mathcal{F}}(\boldsymbol{\Theta}_{[k]})$, over the stabilized partial assignments reduce at each step the set of feasible path assignments $\boldsymbol{\Psi}$, i.e. $\boldsymbol{\Psi}(\boldsymbol{\Theta}_{[0]}) \supsetneq \boldsymbol{\Psi}(\boldsymbol{\Theta}_{[1]}) \supsetneq \cdots \supsetneq \boldsymbol{\Psi}(\boldsymbol{\Theta}_{[k]})$.

Proof: Since we are using a synchronous model, we can assume that changes made by the mapping at v_i are propagated to the peers of v_i in the next iteration. At iteration zero, the set of feasible paths is equal to the super-set $\boldsymbol{\Psi}(\boldsymbol{\Theta}_{[0]})$ whose elements are any feasible set $\boldsymbol{\Phi}$ defined by Eq. 6.16. Since the mapping evolves by repeatedly applying Lemma 1, then all those paths with lower rank than stabilized paths in the current iteration are not announced anymore. Then, by Lemma 2 lower ranked paths and those constructed upon them eventually disappear. In other words, following iterations of the mapping will not propagate them throughout the network and they are not feasible paths anymore. Those paths are removed from the set of feasible sets at that iteration, proving Lemma 3.

Theorem 1 (Safety) Given a network graph $G = \langle V, E \rangle$, given the set of policies S defined by each vertex in V, a synchronous distributed Bellman-Ford mapping $\boldsymbol{\mathcal{F}}(\boldsymbol{\Theta}_{[k]})$ iterating over the path assignment $\boldsymbol{\Theta}_{[k]}$ which is initially defined as,

$$\Theta_{v_i[0]} = \begin{cases} \{v_0\}, & i = 0 \\ \emptyset, & i \neq 0 \end{cases} \tag{6.24}$$

If every policy relation over S is *anti-reflexive* then the mapping is able to grow the path assignment at each iteration until the fixed-point of the following equation is reached at some iteration m,

$$\boldsymbol{\Theta}_{[m]} = \boldsymbol{\mathcal{F}}(\boldsymbol{\Theta}_{[m]}) \tag{6.25}$$

Thus, it can be stated that in absence of reflexive policies the protocol is able to synchronously converge.

Proof: By applying Lemma 1 at each iteration, in absence of conflicting policy relations, the mapping is always able to increase the path assignment with at least one path such that the new assignment $\boldsymbol{\mathcal{F}}(\boldsymbol{\Theta}_{[k]}) \supsetneq \boldsymbol{\Theta}_{[k]}$ is also stabilized. By Lemma 3, as the mapping is consolidating stabilized paths at each vertex, the set of feasible paths $\boldsymbol{\Psi}$ is decreasing, until the highest ranked paths feasible at each node are announced. Hence, there is one iteration k at which the only feasible set of paths at a certain node v_i is the set $\Phi_{v_i[k]} \in \mathbb{P}(\mathcal{P}(v_i, v_0))$ formed by elements that verify the equation $\lambda(\theta) = \lambda_{max}(\boldsymbol{\Psi}(\Phi_{v_i[m]}))$, $\forall \theta \in \Phi_{v_i[k]}$ and $\forall m \geq k$. Since the mapping does not remove paths from a stabilized assignment and it cannot find higher ranked paths at any node, the next iteration $k+1$ will have as outcome the same path assignment. Therefore, we can say that the fixed-point has been hit at iteration k.

6.0.4 Asynchronous Convergence

Despite using a reliable transport protocol, which guarantees ordered message delivery among nodes, the execution of our protocol is not free from communication delays since data synchronization is not enforced between peers. Therefore it may happen that the set of paths used by a node v_i to compute its multipath set at time t (i.e. $\Phi_{v_i[t]}$),

$$\Phi_{v_i[t+1]} = \mathcal{F}_{v_i}(\Phi_{v_i})_{[t]} \tag{6.26}$$

it is a distorted version due to delay in the propagation of some of the announcements coming from peers. A distortion due to a delay of $t - \tau_{v_i,v_j}(t)$ between peers v_i and v_j, with $0 \leq \tau_{v_i,v_j}(t) \leq t$, makes v_i perceive,

$$\Phi_{v_i[t+1]} = \mathcal{F}_{v_i}(\Phi_{v_i})_{[\tau_{v_i,v_j}(t)]} \tag{6.27}$$

Since the iteration over the fixed point is now distorted by $t - \tau_{v_i,v_j}(t)$ we cannot ensure convergence of the fixed-point iteration anymore. According to the general results in [2], it is possible to ensure convergence for a totally asynchronous distributed fixed-point iteration if,

1. The propagation of information happens infinitely often. In other words, it can be assumed that after a certain time $t' > t$ all the announcements $adv(v_j \rightarrow v_i)_{[t]}$ have been propagated and renewed at peer nodes.

2. *Synchronous Condition*: The protocol creates at each iteration $k = 0, 1, \ldots, m$, a sequence of sets,
$$X_{[0]} \supsetneq X_{[1]} \supsetneq \cdots \supsetneq X_{[n-1]} \supsetneq X_{[n]} \supsetneq \ldots \tag{6.28}$$
and it holds
$$\boldsymbol{\mathcal{F}}(x) \in X_{[k+1]}, \forall x \in X_{[k]} \tag{6.29}$$

3. *Box Condition*: For each iteration $k = 0, 1, \ldots, m$ and each node v_i, $i = 0, 1, \ldots, n$, there exist sets of elements $X_{v_i[k]}$ such that the set of elements $X_{[k]}$ can be expressed as the Cartesian product,
$$X_{[k]} = \prod_i X_{v_i[k]} \tag{6.30}$$
The box condition implies that different elements in $X_{v_i[k]}$ can be exchanged without affecting the final result of the iteration, so to speak, the order in which the Bellman-Ford mapping allocates paths does not affect to the evolution of the mapping.

If those three condition hold, the following theorem can be proven,

Theorem 2 (Asynchronous Convergence) Given a network graph $G = \langle V, E \rangle$ with n nodes, the set of policies S defined by each node and a distributed ASSEMBLER mapping $\boldsymbol{\mathcal{F}}(\boldsymbol{\Theta}_{[k]})$ iterating over the path assignment $\boldsymbol{\Theta}_{[k]}$, if every policy relation over S is *anti-reflexive* then the mapping is able to asynchronously converge to a multipath assignment of paths over G.

Proof The condition (1) is guaranteed since ASSEMBLER uses a reliable transport protocol to exchange the information and every node advertises its neighbors with every change in the selected multipath set. Condition (2) is guaranteed by Theorem 1. In addition, replacing $X_{[k]}$ by $\boldsymbol{\Psi}(\boldsymbol{\Theta}_{[\mathbf{k}]})$, the set of feasible sets containing $\boldsymbol{\Theta}_{[\mathbf{k}]}$, both Eq.6.28 and 6.29 can be rewritten as follows. Lemma 3 proves that the protocol creates the sequence,

$$\boldsymbol{\Psi}(\boldsymbol{\Theta}_{[0]}) \supsetneq \boldsymbol{\Psi}(\boldsymbol{\Theta}_{[1]}) \supsetneq \cdots \supsetneq \boldsymbol{\Psi}(\boldsymbol{\Theta}_{[k]}) \ldots \tag{6.31}$$

which can be easily identified with the sequence in Eq.6.28. Moreover, it can be stated by definition 4,

$$\boldsymbol{\Theta}_{[k]} \subseteq \boldsymbol{\Phi}, \forall \boldsymbol{\Phi} \in \boldsymbol{\Psi}(\boldsymbol{\Theta}_{[k]}) \tag{6.32}$$

and by definition 5, applying an iteration of the algorithm on both sides, if $\boldsymbol{\Theta}_{[\mathbf{k}]}$ is stabilized then,

$$\boldsymbol{\mathcal{F}}(\boldsymbol{\Theta}_{[k]}) \subseteq \boldsymbol{\mathcal{F}}(\boldsymbol{\Phi}) \Rightarrow \boldsymbol{\Theta}_{[k+1]} \subseteq \boldsymbol{\mathcal{F}}(\boldsymbol{\Phi}) \tag{6.33}$$

again, by definition 4,

$$\boldsymbol{\mathcal{F}}(\boldsymbol{\Phi}) \in \boldsymbol{\Psi}(\boldsymbol{\Theta}_{[\mathbf{k+1}]}), \forall \boldsymbol{\Phi} \in \boldsymbol{\Psi}(\boldsymbol{\Theta}_{[k]}) \tag{6.34}$$

so that Eq.6.29 can be rewritten as well identifying $X_{[k]} \equiv \boldsymbol{\Psi}(\boldsymbol{\Theta}_{[\mathbf{k}]})$ and $\boldsymbol{\Phi} \equiv x$.

Finally, in order to complete the proof of Theorem 2, the box condition must be verified. The stabilized assignment can be also expressed like $\boldsymbol{\Theta}_{[k]} = (\Theta_{v_0[k]}, \Theta_{v_1[k]}, \ldots, \Theta_{v_n[k]})$. Lemma 1 guarantees that at least one node v_i that increases its feasible assignment, so that there is a set $\Phi'_{v_i[k]}$ such that $\Theta_{v_i[k]} \subsetneq \Phi'_{v_i[k]}$. As there can be more than one, the following super-set $\boldsymbol{\Psi}_{v_i[k]}$ can be defined as the set of feasible assignments that contain the stabilized

assignment $\Theta_{v_i[k]} \subsetneq \Phi^{(p)}_{v_i[k]}$, i.e. $\mathbf{\Psi}_{v_i[k]} = \{\Phi^{(p)}_{v_i[k]}, \forall p\}$. Provided that all the assignments within the super-sets, $\mathbf{\Psi}_{v_i[k]}, \forall i$, are feasible, therefore the set of feasible sets containing $\mathbf{\Theta}_{[k]}$, can be rewritten as the following cartesian product,

$$\mathbf{\Psi}(\mathbf{\Theta_{[k]}}) = \mathbf{\Psi}_{v_0[k]} \times \mathbf{\Psi}_{v_1[k]} \times \cdots \times \mathbf{\Psi}_{v_n[k]} \tag{6.35}$$

which can be arranged to resemble Eq.6.30 as follows,

$$\mathbf{\Psi}(\mathbf{\Theta_{[k]}}) = \prod_{0 \leq i \leq n} \mathbf{\Psi}_{v_i[k]} \tag{6.36}$$

and the box condition is proven. Provided that the three conditions are verified for the protocol, by the *General Asynchronous Convergence Theorem* (Proposition 2.1 in [2]) it can be stated that the protocol is able to converge asynchronously.

6.0.5 Stable Multipath Policy Guidelines

The work in [10] studies how to define routing policies in the Internet to avoid instabilities inGBP. The resulting guidelines are based on the business relations between ASes. Using the stability condition derived above, the guidelines in [10] can be reformulated for multipath. They can be proven stable if no reflexive relation can be constructed in the network.

Two guidelines are presented. We do not claim that the resulting routing policies are the only policies ASes can follow to achieve stability. Instead, we choose them because an AS needs to know its commercial relation with each neighbors to define its policy and the guidelines cover the most common business relations found on the Internet. The following two assumptions must hold for both guidelines,

Assumption 1 An AS advertises paths coming from its provider only to its customers. Paths coming from peers only to its customers and finally, paths coming from its customers to other customers, peers and providers.

Assumption 2 A customer AS cannot be an indirect provider of one of its direct providers.

Now, we present the guidelines. Basically the proofs fail to construct reflexive relations when the policies are defined according to the guidelines.

Guideline 1 If every AS assigns a higher local preference to the paths received from its customers than to paths received from its peers and they assign higher preference to paths coming from its peers than to paths coming from its providers, then ASSEMBLER is able to converge.

In addition, convergence is possible regardless of the size and characteristics of the paths in the multipath set.

Proof The proof fails to construct the reflexive relation in Eq.6.3. Without loss of generality the proof refer to the scenario in Fig.6.1. First we assume that Γ_1 comes from a customer of

v_1, then according to Guideline 1, Δ_1 must come from another customer, otherwise it cannot have higher preference. Then Δ_1 is of the form $\Delta_1 = \Pi_2\Gamma_2$ (in Fig.6.1 Π_2 is just a link between v_1 and v_2 but in general is an arbitrary path). Since v_1 is a provider of v_2, according to Assumption 1, the latter can only advertise paths from its customers to v_1 (notice that if intermediate nodes between v_1 and v_2 exists the situation is the same). If Δ_2 has higher preference than Γ_2 and v_2 is following the Guideline 1, then it must come from another customer of v_2. The same reasoning apply for v_3. Now, at v_4, the path Δ_4 through v_1 should be preferred over the path Γ_4. That can only happen if Δ_4 comes from a customer of v_4, however if v_1 is a customer of v_4 and v_4 is in the chain of customers from v_1 it means that Assumption 2 is broken.

In the second case, we assume that Γ_1 comes from a peer of v_1. Therefore, Δ_1 must come from either a peer or a customer of v_1 (Assumption 1). In both cases, it means that Γ_2 and Δ_2 come from customers of v_2, otherwise they cannot be advertised to a peer or a provider. The chain of customers continue until v_4. A reflexive relation would be constructed if v_1 is a customer of v_4. If v_2 is a peer of v_1, then Δ_1 cannot be advertise to v_4 as it is a v_1 provider. If v_2 is a customer of v_1 we are in the previous case.

In the last case, v_1 learns Γ_1 from a provider, then v_2 can by a customer, peer or provider of v_1. If v_2 is a provider of v_1 and Γ_2 and Δ_2 are advertised to v_1 since they come from customers or peers of v_2, the chain continues like in the two previous cases. Otherwise, if Γ_2 and Δ_2 come from providers of v_2, then the chain of providers continue to v_4. If Γ_4 comes from a customer of v_4 then according to Guideline 1 Δ_4 must come from a customer, however v_1 does not advertise its provider v_4 with paths from other providers, therefore Δ_4 is not announced and the reflexive relation is broken. If Γ_4 comes from a peer the situation is the same. Only if Γ_4 comes from a provider, Δ_4 can be more preferred, therefore if v_1 is the provider of v_4 then Δ_4 is advertised, however in that case v_4 becomes the indirect provider of one of its direct providers (through v_3 and v_2) and Assumption 2 is broken.

Guideline 2 An AS can assign the same local preference to paths coming from a customer or a peer ASes. Yet, stability can be only guaranteed if paths from peers and customers are not selected together in the multipath set.

Proof Again we try to construct a reflexive relation among the propagated paths. The scenario in Fig.6.1 is used again. First we assume that Γ_1 comes from a customer of v_1, then according to Guideline 2, Δ_1 can now come either from another customer or a peer, otherwise it cannot have higher preference. Therefore, under Guideline 2, the first and second cases of the proof of Guideline 1 can be combined. Due to Assumption 1, the most complex case is an alternation of peers and customers links, since two consecutive peering links cannot happen. If we follow this alternation from v_1 till v_4, a reflexive relation can happen only in two cases. First, if v_3 and v_4 are peers, then Γ_4 and Δ_4 must come from customers of v_4 which implies that Assumption 2 must be broken to create the reflexive relation. Second, if v_3 is a provider of v_4 then path Γ_4 must be then from a customer of v_4. Now v_1 and v_4 can be peers and Δ_4 can be still be more preferred than Γ_4. However, in that case v_4 is again breaking the hierarchy of the network, acting as a particular case of provider defined as *peer-provider* [10].

So far the proof states the stability under equal preference for peers and customers, however nothing is said about the mixture. The proof is completed with the counter-example. In Fig.6.3 ASSEMBLER oscillates if the policy allows to select a path from a customer and another from a provider at the same time. That is, an AS can choose to send the traffic among

paths through customers or a peers, but a router must not mix paths from both types. Notice that the corresponding guideline for BGP (Guideline 5.2, [10]) does not make this distinction because BGP selects only one path.

Chapter 7

Implementation of an ASSEMBLER-Capable Router

In this chapter the implementation of a multipath software router running ASSEMBLER as multipath inter-domain protocol is presented. Given that our protocol inherits a large functionality from BGP, the implementation sets off from an implementation of BGP. The implementation is intended as a proof of concept to show that multipath capabilities can be added to border routers introducing little modifications in the software. Furthermore, the implementation is part of a testbed developed within the scope of the Trilogy Project [1] and the integration of multipath routing with the implementation of MPTCP [9].

There are already some available open source implementations of BGP. Some of them are stand-alone BGP daemons like OpenBGPd [3]. Other BGP implementations are part of complete routing suites. Those packages offer the possibility of analyzing the interactions between different routing protocols, which can be a highly interesting characterization. The two most widely referenced open source routing suites (that include BGP) are Quagga [17] (formerly Zebra) and XORP [14]. The routing software package Quagga encloses several routing modules, which can be launched simultaneously. One of the Quagga best strengths is its implementation efficiency (memory consumption of the Quagga BGP daemon is about 20MB at initialization time). Among its drawbacks to be used as base for multipath extensions, there are its scarce documentation for developers, the BGP implementation is not very modular and it uses relies directly in the OS kernel to install and remove routes in the data-forwarding plane, thus in case a special forwarding plane architecture is needed, kernel libraries must be modified directly.

An alternative to Quagga is XORP. The routing project XORP is a truly modular routing package to create software routers. In contrast to Quagga, XORP offers certain flexibility to choose which forwarding plane management interface to use, either the kernel forwarding plane, CLICK [20] and supports distributed forwarding planes. Among XORP disadvantages, it can be found that in exchange for modularity, its code tends to be more inefficient (for instance the BGP daemon has a memory footprint of about 100MB at initialization time). Despite Quagga's efficiency, XORP has been chosen as the preferred routing suite to introduce the multipath extensions thanks to its modularity, flexibility, integrated support for CLICK and extensive documentation.

The implementation of XORP is running in the control plane of the router and a multi-

path data-fowarding plane has been implemented using CLICK. The data plane is using the CLICK setting presented in [20] and the routing table module has been modified to add multiple next-hops per prefix and use them to balance traffic flows through different next-hops preserving the packet ordering.

The extensions for XORP and CLICK are integrated in virtualized appliances and a testbed is settled to run the experiments. The details of the testbed are presented in the next section. The extensions of XORP and CLICK are presented afterwards. Finally, the section concludes with the results obtained over a small sample topology.

7.1 The Evaluation Testbed

The architecture of the testbed is illustrated in Fig.7.1. The testbed is deployed among several physical PCs, which provide the hardware layer. Each physical host can be devoted to either router emulation or network emulation tasks. A routers emulator is able to multiplex several virtual routers in a physical host. The network emulation is performed in different hosts. All the PCs are connected through Gigabit Ethernet LAN.

Fig.7.1 also depicts the different layers involved. On top of the hardware layer there is the virtualization layer, which has in turn two sublayers, the router virtualization sublayer and the network virtualization sublayer. One physical host provides the functionality of the network virtualization sublayer. That host is running a network emulation software. Each virtual router sends and receives their traffic over a VLAN towards the network emulator. When a packet gets into the emulator, the latter creates the illusion that packets are forwarded across a real network, delaying, dropping, mixing them in queues, etc. Afterwards, the emulator sends the packets back to the destination host using another VLAN.

On the other hand, the router virtualization layer handles the virtual machines that emulate routers and the virtual network interfaces of those routers, which are *bridged* to the physical network interfaces.

In each virtual machine emulating a router, there is a Linux Debian appliance running CLICK and XORP. CLICK processes and forwards packets at the kernel level and it has been used to implement the multipath Forwarding Information Base (FIB). XORP is used the control plane of the routers and CLICK in the data-forwarding plane. In the control plane the routing protocols are executed. In this testbed only BGP and ASSMBLER are running, although all the protocols in the XORP bundle are supported. The rest of the section describes the multipath extensions for XORP and CLICK. The modifications to the RIB process are described first. Afterwards, multipath extensions to the unipath BGP daemon to implement ASSEMBLER are shown and finally the implementation of a multipath FIB using CLICK is detailed.

7.2 The Control Plane

Four processes from the XORP suite are running at each router. Depending on whether a router is multipath capable or not, either the ASSEMBLER process or the BGP daemon is running respectively. The Static Routes module is running such that the prefixes to be announced to other routers can be configured. In addition, the RIB process merges the infor-

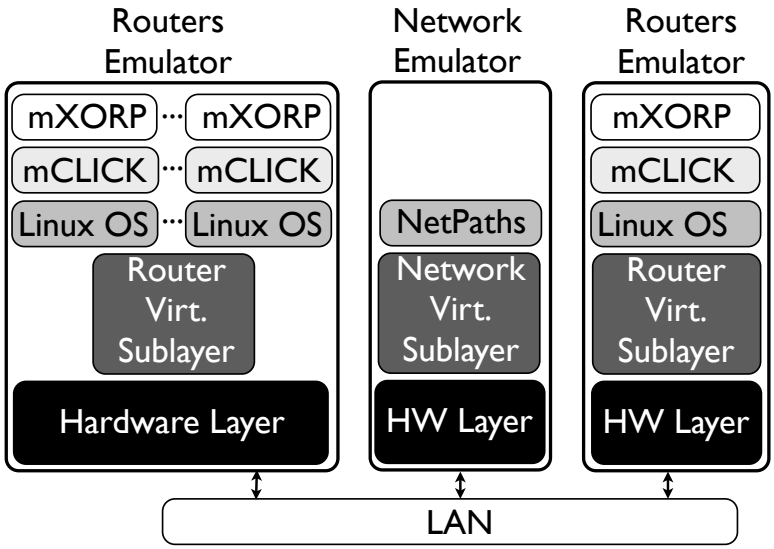


Figure 7.1: Different layers involved in the virtual testbed.

mation from the different routing protocols and update each routing process when changes, such as connectivity, failure occurs. The RIB process does not directly install the most optimum paths in the FIB, instead it relays those paths to a forth process, the *forwarding engine abstraction*(i.e. FEA). The FEA process is the process which acts as interface to the data-forwarding plane for the RIB.

All those processes are intercommunicated using a RPC-like (i.e. remote procedure call) mechanism. The methods that can be called upon one process are defined by an interface in a language called XRL (i.e. eXtensible Router Language). Automatic tools generate the client object and the stub code to achieve the inter-process communication. In XORP the modifications are carried out at three different levels: the BGP daemon, the RIB process and at the FEA process. The multipath extensions for the BGP daemon to obtain an implementation for ASSEMBLER are described first. The redefinition of the XRL for the RIB and the multipath extensions for its modules are defined afterwards. Finally, in order to complete the top-down analysis, the implementation of a plug-in to let the FEA process handle a CLICK-based data-forwarding plane is detailed. Before getting into the details of the ASSEMBLER implementation, we briefly review the normal operation of the XORP BGP Daemon which serves as a base for the ASSEMBLER implementation.

7.2.1 The Standard BGP Daemon

This section describes the architecture and modules of the unipath BGP process. In order to have a general overview and identify the different modules, we always refer to Fig.7.2. Every module labelled as PeerHandler handles a BGP session with a peer router. Every time a BGP Update message is received, the PeerHandler creates one or several internal messages of type ADD ROUTE, DELETE ROUTE or REPLACE ROUTE. For instance, when a peer router announces to the router in Fig.7.2 that a new destination is reachable through it for the first time, an ADD ROUTE message crosses the chain of modules from the PeerHandler until the module DecisionTable is reached. Otherwise, it gets filtered at some module in between.

Once in the DecisionTable, the route contained within the ADD ROUTE message under-

goes the BGP tie-breaking process along with other routes announced by other peer routers for the same destination. The DecisionTable queries the different RIBInTable on each branch to obtain those alternative routes. If the route becomes the new winner then two new internal messages are propagated downstream through every output branch thanks to the FanOutTable module, which replicates the ADD ROUTE. A DELETE ROUTE is propagated to delete any stored/cached state of the previous winner.

Afterwards an ADD ROUTE message establishes state for the new winning route, which in turn upon reaching the PeerHandler module is included into an external BGP Update message. The latter will be sent to each peer router for which a branch exists. Nevertheless, there is an additional branch, which is not connected to any peer, it ends in a IpcRIBHandler module, instead. That special module acts as the interface with the control process of the RIB.

In addition to the announcement or withdrawn of routes towards a given prefix, peering session establishment is another common event in BGP. No sooner than the TCP connection is established, the router starts dumping its current routing information to its new (or recovered) peer. To that extent, the BGP daemon of XORP defines and additional module called DumpTable. The DumpTable is in charge of feeding the new router in background while consistency is kept as new routes arrive in the meantime to the different RIBInTables.

7.2.2 Path ASSEMBLER Extensions in XORP

In this section, the ASSEMBLER modifications carried out over the XORP BGP daemon modules are detailed. From the analysis of the unipath XORP BGP process, several key modules can be identified in order to accomplish the ASSEMBLER extensions.

Before addressing the specific modifications of each module, there is a modification required at every module (with the exception of the *PeerHandler* module). The modules in Fig.7.2 implement an interface called *RouteTableBase*. The *RouteTableBase* interface defines the operations that one module can call upon another. For instance, the interface defines operation to relay paths between connected modules. The current definition of the interface allows to call an add, delete or replace operation over just one path at a time. Operations called afterwards regarding the same network prefix override the state generated in the module by previous operations. Therefore, the first modification is the addition of a new operation on the interface, so-called *ROUTE_MADD*, such that multiple paths can be relayed between modules. Now, we present the modifications module by module as they appear *downstream* in Fig.7.2 (i.e. from left to right).

PeerHandler The first module that we come across in the picture is the *PeerHandler* module. One of the most remarkable features of the design of ASSEMBLER is the backwards compatibility with BGP. As a direct consequence, the messages received or generated by a multipath capable router must allow unipath routers to process them. The only subtle difference in a message generated by a multipath capable router running ASSEMBLER is that the AS_PATH attribute is generated following particular rules, but that does not modify the way it is processed by the router receiving the announcement. Therefore, the module interacting with routers for which a BGP session is established does not require any modification.

RIBInTable, CacheTable, In/OutFilterTable and NextHopLookupTable Since a router receives one announcement/path per network prefix from each neighbor, the *RIBInTable* does

not require any further modifications to implement ASSEMBLER. Nevertheless, the module has been extended to return all the most recent paths received from the same peer upon quering from the *DecisionTable*. The *CacheTable* functionality remains the same and the definition of new filtering policies for multipath scenarios is out of the scope of this work, thus the *FilterTable* implementation is not extended. The next-hop resolution is exactly as in the unipath case. Again, in order to make the modifications of the XORP BGP process as general as possible, if another protocol propagates several paths for the same network prefix, these modules run through each each path in the multipath set executing their functionality over each path independently. For instance, if one of the paths in the multipath set is not resolvable by the *NextHopLookupTable*, then only that path is removed from the set propagated downstream.

DecisionTable The *DecisionTable* carries out the ASSEMBLER path selection process. Every time a new ROUTE_ADD, ROUTE_DELETE or ROUTE_REPLACE operation is going on the decision process is carried out in order to check whether the new information discloses a more optimum set of paths to forward traffic. The *DecisionTable* module retrieves all the available alternative paths querying the different *RibInTable* modules at each branch.

In contrast to the unipath case, the rules in the decision process are those depicted in Section 4.1. Therefore the outcome of the decision process is not a single best path, but the set of most preferred paths. Nevertheless, the paths in the selected set are not announced independently to router peers. An special announcement is constructed as depicted in Section 4.2. The AS_PATH attribute is constructed by taking the path with the shortest AS_PATH_LENGTH and adding an AS_SET. The AS_NUMBERs included in the AS_SET are those in the remaining paths in the multipath set which are not already in the shortest path.

In addition to removing some tie-break rules and adding the functionality to construct the multipath set, now the *DecisionTable* makes two calls upon the next module, the *FanOutTable*. The first call the default ROUTE_ADD operation, which supports only one path with the constructed aggregated AS_PATH and a flag in the attributes of the internal message indicating that the path is intended to be announced to the peers. The second call contains the whole multipath set with each path passed independently in the call. This time a ROUTE_MADD operation is called. In that way, the *DecisionTable* indicates the *FanOutTable* that the set of paths relayed to it in the call should be relayed to the branch that connects the ASSEMBLER process to the RIB process. The paths are passed separately, such that several next-hops will be installed in the data-forwarding plane, enabling multipath forwarding of traffic.

FanOutTable The path selection procedure is carried out once per operation. The result of the selection procedures must be announced to the peers in order to guarantee path consistency and loop-freeness. Every established session with another peer creates a branch of modules like in Fig.7.2. The *FanOutTable* module is responsible for duplicating the output of the selection procedure to every branch so that all the peers are eventually updated with the new state of the router.

In addition, given the peculiarities of ASSEMBLER the *FanOutTable* distinguishes between the information to create the announcements to other peers and the information to create the new state in the FIB (which has to be relayed to the RIB process first). Therefore, the *FanOutTable* receives a ROUTE_ADD operation carrying the aggregated AS_PATH created by the *DecisionTable* module and a ROUTE_MADD operation containing the expanded version of the previous AS_PATH, i.e. the set of paths aggregated in the AS_PATH of the first

call. The ROUTE_ADD operation duplicates the internal message to every branch connecting to a peer router whereas the ROUTE_MADD operation forwards the internal message through the branch connecting to the RIB process.

RIBOutTable The RIBOutTable is responsible for coordinate the inner operations and the announcements towards peer routers. The RIBOutTable receives operation and schedules them in a queue. Each element in the queue comprises an operation object, which in turn contains a path attribute. When the *PeerHandler* module notifies the RIBOutTable that it is idle, the *RIBOutTable* pops messages from the queue and the *PeerHandler* creates the announcement. This module has been extended to support the multiple paths passing in the branch connecting to the RIB. Each element in the waiting queue is an operation object which is able to store now a linked list of paths instead of a path element.

IPCRIBHandler The RIB process works based on transactions. The next transaction does not start until the ongoing is committed. The operations meant for the following transaction are withheld in the *RIBOutTable*. The main modification in this part of the module chain is not the modification in the module itself, since it consist in changing the type of object called to relay routes to the RIB process, from the RPC (i.e. remote procedure call) *client* class `XrlRibV0p1Client` to `XrlRibV0p2Client`. RPCs are called XRLs in XORP. The key modification here is the definition of a new interface to allow other processes to interact with the RIB process.

DumpTable As soon as a new peer session is established the router dumps all its most preferred paths for each destination to the peer so it can get a quick vision of the network. In XORP BGP, the *DumpTable* is in charge of dumping the paths. The *DumpTable* polls every *RIBInTable* looking for the paths marked as the most preferred.

Nevertheless, ASSEMBLER creates artificial AS_PATHs. Those paths are not in the *RIBInTable* modules since they are not paths announced by a peer, but internally created. To that extent, the *AuxiliarTable* path storage module is added to the architecture in Figure 4. The *AuxiliarTable* simplifies the implementation of the *DumpTable*. Now the *DumpTable* does not need to synchronize with all the *RIBInTables*, synchronization with the *AuxiliarTable* is enough to dump the current routing information to new peers.

7.2.3 Modifying the RIB and FEA Processes

In addition to the routing protocols, XORP runs the RIB and FEA processes. The RIB merges the routing information acquired by each routing protocol. The outcome of the merging process is passed to a third process, the FEA. The FEA allows XORP to decouple from a specific data-forwarding plane. Thanks to that abstraction XORP can use different data-forwarding planes, such as the Linux routing module, CLICK or even a distributed data-forwarding plane among several PCs running XORP.

The routing protocols can add, delete or replace routes in the forwarding plane executing a transaction against the RIB process interface. Paths are added or removed from the data-plane by means corresponding transactions and no changes are made effective in the routing table until the transaction has been committed. A new transaction has been added to the RIB interface. The new transaction allows a routing protocol to add or delete several paths simultaneously.

The FEA has an interface with common abstract operations over the data-forwarding plane. That interface is used by the RIB process to execute the changes in the data-plane. Once the operation is called from the RIB, the FEA translates the abstract operations into the specific set of operations to be executed by the data-forwarding plane used by XORP.

In order to know which are the operations to be carried out over the data-plane, for each type of data-plane there is a plug-in that allows the FEA module of XORP to interact with the FIB. We have extended the current plug-ins for CLICK to support multi-path addition and removal of paths. The details of this new plug-in are explained later on along with the implementation of a multi-path FIB using CLICK.

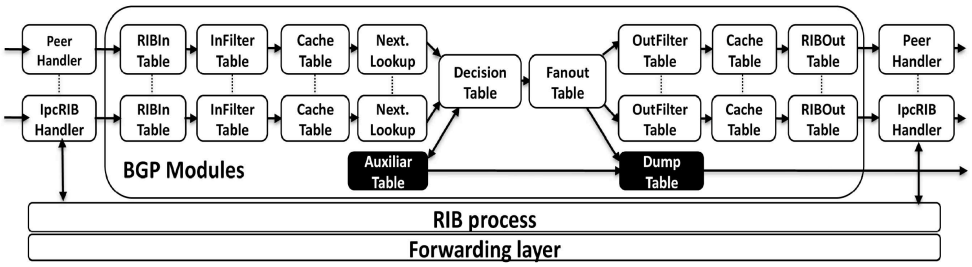


Figure 7.2: Module diagram of XORP BGP daemon.

7.3 The Data-Forwarding Plane

The data-forwarding plane of one of the ASSEMBLER-capable virutal routers is implemented using CLICK. CLICK comes with a default data-forwarding plane implementation that can be found in [20]. CLICK allows its modules to have *hooks* to interact with external processes. Hooks are nothing but inter-process communication pipes as those commonly used in Linux-based operating systems.

Depending on the type of hook, external processes can read, write or both, information that CLICK modules can parse/publish. In our case, we set off from the default implementation of the routing table (so-called *LookupTable*) and a new hook has been added. The new hook allows to progressively add entries to the routing table for the same network prefix without overwriting previously added entries (the default add operation replace the current entry by the new one). The new hook appears in the folder corresponding to the routing table module in the CLICK file system as a file called *madd* (for further details about the CLICK file system check [20]).

The FEA plug-in handling CLICK dumps to the CLICK routing table the routing information extracted from the multipath set chosen by the ASSEMBLER process. The plug-in transform the routing information from the data format used by XORP into a format that can be parsed by the CLICK module. Afterwards, the plug-in initiates a transaction. The transaction consist in two different steps. First, the FEA flushes the previous entries for a given network prefix. Second, it iteratively writes the information of each of the paths into the *madd* file introduced earlier. After, the FEA closes and commits the transaction. The deletion step is used simply because, it simplifies the installation of paths.

The multipath addition creates a routing table like the one in Fig.7.3, where for each network prefix we can find several $< nexthop, interface, metric >$ tuples. The difference

between the appearance of multiple entries per prefix in the routing table in the unipath and the multipath case, is that the additional entries in the unipath are used as backup, such that only the first entry matching the destination network prefix of the packet and the others are used in case the first path becomes unavailable. In the multipath case, the multiple entries can be used to forward packets towards the same destination concurrently.

Many forwarding policies can be defined, however some of them like random forwarding or round-robin increase packet reordering which can be harmful, specially using a transport protocol such as TCP or MPTCP, which guarantee ordered delivery of packets. Therefore, it is desirable to have a forwarding algorithm, which guarantees that at least certain sequences of packets follow the same path, minimizing reordering.

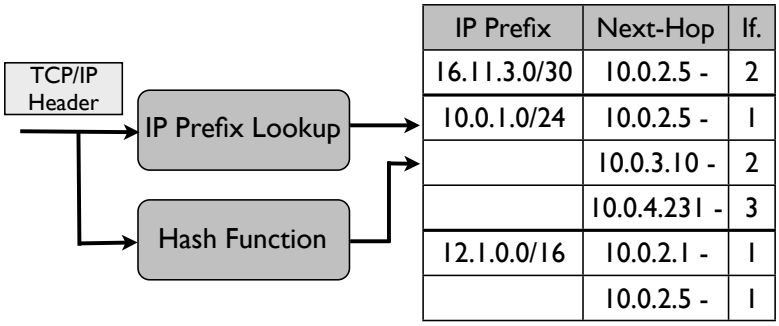


Figure 7.3: Multipath FIB.

Therefore, we propose the following forwarding algorithm, a hash function is computed out of the so-called 5-Tuple of TCP/IP fields (see RFC [16]), which comprises the source IP address, destination IP address, source TCP port, destination TCP port and TCP sub-flow identifier (only if MPTCP is enabled). The 5-Tuple uniquely identifies each flow . The outcome of the hash function is then mapped to an output interface such that the same TCP flow is always forwarded to the same next-hop.

There are different ways to map flows with the interfaces. The simplest is to compute the result of the hash function and compute that number modulo the number of available paths in the routing table for a certain prefix. Another solution is to assign equally distributed identifiers to each path in the output range of the hash function and forward to the closest higher identifier to the hash function value over the TCP 5-Tuple. Figure 5 shows a diagram of the forwarding mechanism. The packet header information is passed to both the entry selector module (HashFunction in the figure) and to the lookup algorithm. When the lookup algorithm finds and entry in the routing table, it retrieves the number of alternatives and along with the result of the hash function computes the next-hop that will be used to forward the packet.

7.4 Disclosed Path-Diversity

The network depicted in Fig.7.4 is running ASSEMBLER. The testbed introduced in previous sections emulates the network. Even in this small topology is interesting to see how much path diversity ASSEMBLER can disclose. As discussed during the introduction, as more path-diversity is introduced and operated, the adaptation and responsiveness of multipath inter-domain protocols to network changes should improve.

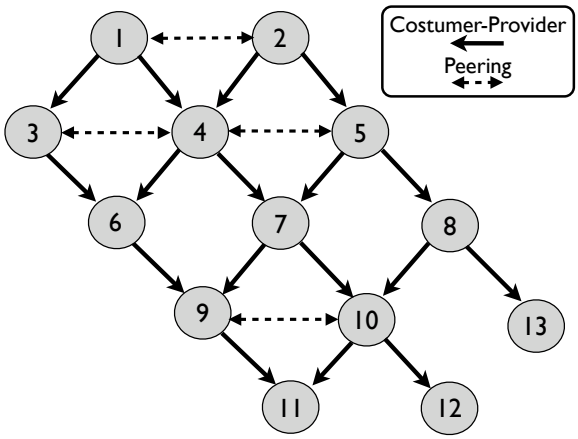


Figure 7.4: Sample Topology

The setup of the experiment is as follows. Every node depicted in Fig.7.4 is an ASSEMBLER router. Each router belongs to one AS and it announces one network prefix of the format 150.X.0.0/16 where X is the AS number the router belongs to. Once the advertisement is propagated from the router, the network converges and for the amount of paths from each of the remaining nodes to the advertising node is calculated. Each pair origin-destination is called a *case*. The process is repeated for each router such that all the prefixes are propagated and the amount of paths for each case measured.

The results in Fig.7.5 show the path diversity that ASSEMBLER is able to expose in the sample topology in Fig.7.4. The figure is the cumulative distribution function of the number of different paths for each case. Two paths are considered different if they differ at least in an intermediate node or link. The chart shows that in about 60% of the cases, a node is left only with one path towards the destination AS. That percentage should decrease if the connectivity between ASes is not represented as a single link and each AS is a single router. On the other hand, about 40% of the cases a node is using at least an additional end-to-end path towards a destination, which is an acceptable results for such a small topology. Moreover, roughly in 20% of those cases nodes get 2 or 3 additional paths. Larger amount of additional paths are rare and only happen in a reduced number of cases.

Dealing with the improved reliability introduced by the network diversity, Fig.7.6 shows the probability distribution function of having an additional path towards a destination after one and two node in the shortest path fail simultaneously. The experiment setup is similar to the previous one. The difference is that whereas in the previous case every pair origin-destination was considered a case, now for every pair there are several cases, one per each node on the shortest path that is assumed to be down to compute the addition paths to overcome the failure. Cases in which there is one path between the origin and destination are not considered.

The results show that even for a small topology a certain fast recovery can be achieved compared to the unipath case. In 65% of the cases, in which a node has at least one alternative path to another node, connectivity between the two ASes is possible without waiting for the ASSEMBLER process to reconverge. As expected, as the number of simultaneous failures increases the connectivity drops dramatically, even if multi-path is enabled. In the sample topology, if two nodes in the shortest path fail simultaneously, in less than 10% of the cases nodes have an available end-to-end path without executing the BGP selection process.

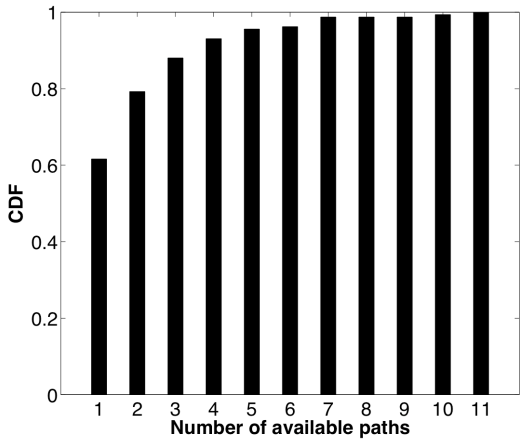


Figure 7.5: CDF of alternative paths available at each node.

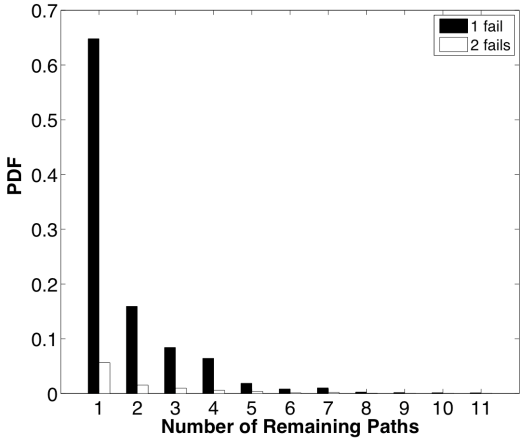


Figure 7.6: PDF of alternative paths after 1 and 2 nodes have failed.

Chapter 8

Related Work and Conclusions

This chapter compares the most relevant BGP-compatible multipath inter-domain routing proposals with ASSEMBLER and presents the conclusions of this work. Alternative protocols that require a global upgrade of the network (see for instance [11]) are not considered in this discussion. The first set of solutions comparable to ASSEMBLER achieve backwards compatibility using BGP to exchange the primary path (ensuring backwards compatibility) and they use a parallel protocol or BGP extension to advertise additional paths. This is the case of R-BGP, which advertise failover paths [21] to achieve fast recovery. BGP Add-Paths [32] is another solution in which routers add a new BGP capability to incrementally advertise extra paths. Finally, MIRO [36] relies also on an additional negotiation of paths. Although, they are compatible with BGP, these solutions require that two or more neighbor ASes coordinate to deploy multipath border routers. ASSEMBLER does not require of such an incremental/additional negotiation of paths and a coordinated deployment between neighbor ASes is not required.

Another set of multipath inter-domain protocols compatible with BGP do not modify the BGP protocol at all and no additional/incremental advertisement of paths is performed. The first solution is the Multipath-BGP proposed by the manufacturers Cisco and Juniper [6, 18], in which all the considered paths must share the same attributes except for the BGP_ID and interface address of the announcing border router. This type of multipath is intended for a set of particular settings in which several physical connections exists between two ASes. Hence, the multipath set yielded is constrained to have the same AS_PATH attribute in all the routes. As shown in Section 4.1, the K-BESTRO path selection algorithm and the assembling technique used by ASSEMBLER does not constrain the path diversity and any set of aggregatable paths can be used concurrently, even if they have different egress ASes.

Some other BGP-compatible protocols propose to advertise one path and use the different alternatives received through BGP to forward the traffic without advertise them. For instance, the inter-domain flavours of Routing Deflections [37] and Path Splicing [25] forward traffic among the available alternative BGP paths according to a *tag* in the packet header. Thus, since BGP advertises only one of those paths, the loop-freeness of the multiple BGP routes cannot be guaranteed, since routes that are not propagated to further ASes are used in practice to forwards the packets. Therefore the control plane information in neighbor ASes is inconsistent with how the traffic is forwarded. The authors in [25] argue that if the common routing policies presented in [10] are followed, no routing loops are possible. In addition, they propose some additional mechanisms to overcome that limitation like deflection coun-

ters or include the AS number of the ASes crossed before in the packet header. An alternative that solves the loop-freeness problem is to propagate the longest available path like in LP-BGP [31], however longer paths are likely to suffer a penalization when compared with other paths at a legacy router.

Thanks to the advertisement scheme presented in Section 4.2, ASSEMBLER is able to advertise information that is consistent with the forwarding of traffic currently used. Using ASSEMBLER, the forwarding mechanisms of Routing Deflections and Path Splicing can be used while preserving loop-freeness and without propagating paths that are likely to be considered worse by legacy routers like in [31], since the AS path length attribute in the advertisement is equal to the shortest path within the multipath set.

8.1 Conclusions

In this work, ASSEMBLER a novel protocol for multipath inter-domain routing has been presented. It is the first inter-domain routing protocol that features both, flexible multipath routing and backwards compatibility with BGP, without limiting the path diversity or using another protocol in parallel. Thanks to its design, ASes can benefit from multipath capabilities upgrading progressively their network equipment inside the AS and no coordination or global upgrade is required to take advantage of multiple inter-domain paths.

The characteristics of the multipath set provided by ASSEMBLER can be flexibly tuned using a few parameters to fully exploit the available path diversity or constrain the amount of paths installed in the data-plane (avoiding an exponential growth of the routing tables).

The ASSEMBLER announcements are regular BGP updates generated with an special algorithm such that advertised updates gather information from multiple paths in one message. Those updates can be processed by legacy routers, they are not penalized when compared to regular BGP paths and loop-freeness is maintained. The deployment in a real AS can be carried out progressively and current routing policies and traffic engineering techniques are supported by ASSEMBLER. It can be combined with multipath forwarding techniques to split the traffic amount those installed paths.

The stability of the protocol has been proven in absence of conflicting configurations. Whenever the ASes can simultaneously fulfil their preferences with the available paths, the protocol is able to converge. Two guidelines have been introduced in order to guarantee global stability for every possible configuration of ASSEMBLER.

The adoption of ASSEMBLER as multipath inter-domain routing should provide ISPs with more flexible routing configurations, simplified and dynamic traffic engineering techniques and decrease inter-domain churn.

8.2 Future Work

In order to complete and extent the research work initiated by this thesis, there are some open questions that can serve as the base for future research work. Of special interest are the interoperability aspects. For instance, large ISP do not have a planar architecture for their border routers. The reason behind this is the scalability of the system, since a planar architecture im-

plies that the full-mesh of iBGP between border routers grows exponentially with the size of the network and the system may get to a saturation point in terms of connections and internal churn. ISPs avoid that situation by introducing special nodes called *route reflectors* which help to reduce the amount of existing connections creating a hierarchy. Typically the portion of border routers per route reflectors is 10 to 1, the border routers keep only one connection to the route reflector and the full-mesh is only created among the reflectors, which reduces the scalability problem. It could be interesting to perform an analysis of the possible effects that introducing ASSEMBLER inside and AS may have if legacy route reflectors are present.

Another interesting interoperability analysis could be the interactions with BGP Add-Paths, such that the architecture of the AS would be inter-domain routers, internally communicated using Add-Paths and using ASSEMBLER to communicate with external ASes that do not support multipath. A transition analysis between these technologies could be interesting as well.

The design of traffic engineering techniques that exploit the advantages of using multipath routing is also an open question. It is an intuition of the author that multipath should provide more flexible and finer granularity in the handling of traffic, reducing in some cases the inter-domain churn and achieving fast convergence when local failures occur.

Finally, another interesting future research line could be the design of new network services using the additional paths. Since it was not possible before, there is not applications of multipath inter-domain routing appearing in the literature or in hands-on practical manuals. Actually the basis for this research line has been already initiated by the author and the supervisor of the thesis, getting promising positive feedback from several ad-hoc conversations with former partners of the FP7 Trilogy Project and in the Routing Research Group (RRG) of the IETF. Therefore the real potential of multipath routing in inter-domain scenarios remains as an open question.

This research work is planned to be presented as a paper publication along with more empirical results, specially from simulations. This prospective publication is work in progress by the time writing.

Bibliography

[1] 7th Frame Programme. Trilogy project: Architecting the future internet. `http://trilogy-project.org/`.

[2] D. Bertsekas and J. Tsitsiklis. Parallel and distributed computation. 1989.

[3] H. Brauer and C. Jeker. OpenBGPd.

[4] M. Caesar and J. Rexford. BGP routing policies in ISP networks. *Network, IEEE*, 19(6):5–11, 2005.

[5] C. Chau. Policy-based routing with non-strict preferences. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 387–398. ACM, 2006.

[6] Cisco. Bgp best path selection algorithm. `http://www.cisco.com/en/US/tech/tk365/technologies_tech_note09186a0080094431.shtml`.

[7] Cisco. How does load balancing work? `http://www.cisco.com/en/US/tech/tk365/technologies_tech_note09186a0080094820.shtml`.

[8] Cisco. Load sharing with bgp in single and multihomed environments: Sample configurations. `http://www.cisco.com/en/US/tech/tk365/technologies_configuration_example09186a00800945bf.shtml`.

[9] A. Ford and C. Raiciu. M. Handley," TCP Extensions for Multipath Operation with Multiple Addresses", draft-ietf-mptcp-multiaddressed-03 (work in progress), 2010.

[10] L. Gao and J. Rexford. Stable Internet routing without global coordination. *IEEE/ACM Transactions on Networking (TON)*, 9(6):681–692, 2001.

[11] P. Godfrey, I. Ganichev, S. Shenker, and I. Stoica. Pathlet routing. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pages 111–122. ACM, 2009.

[12] T. Griffin, F. Shepherd, and G. Wilfong. The stable paths problem and interdomain routing. *IEEE/ACM Transactions on Networking (TON)*, 10(2):232–243, 2002.

[13] F. Guo, J. Chen, W. Li, and T. Chiueh. Experiences in building a multihoming load balancing system. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 1241–1251. IEEE, 2004.

[14] M. Handley, O. Hodson, and E. Kohler. XORP: An open platform for network research. *ACM SIGCOMM Computer Communication Review*, 33(1):53–57, 2003.

[15] J. He and J. Rexford. Toward internet-wide multipath routing. *Network, IEEE*, 22(2):16–21, 2008.

[16] C. Hopps. Analysis of an Equal-Cost Multi-Path Algorithm, IETF. Technical report, Internet RFC 2992, Novembre 2000.

[17] K. Ishiguro, T. Takada, Y. Ohara, A. Zinin, G. Natapov, and A. Mizutani. Quagga routing suite.

[18] Juniper. Configure bgp to select multiple bgp paths. `http://www.juniper.net/techpubs/software/junos/junos53/swconfig53-ipv6/html/ipv6-bgp-config29.html`.

[19] S. Kandula, D. Katabi, S. Sinha, and A. Berger. Dynamic load balancing without packet reordering. *ACM SIGCOMM Computer Communication Review*, 37(2):51–62, 2007.

[20] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. Kaashoek. The Click modular router. *ACM Transactions on Computer Systems (TOCS)*, 18(3):263–297, 2000.

[21] N. Kushman, S. Kandula, D. Katabi, and B. Maggs. R-BGP: Staying connected in a connected world. In *Proc. NSDI*, pages 341–354, 2007.

[22] T. S. A. Labs. Best practices for network interconnection. NANOG 43.

[23] T. McGregor, S. Alcock, and D. Karrenberg. The RIPE NCC internet measurement data repository. In *Passive and Active Measurement*, pages 111–120. Springer, 2010.

[24] D. Meyer. University of oregon route views archive project. *at http://archive. route-views. org*.

[25] M. Motiwala, N. Feamster, and S. Vempala. Better interdomain path diversity with BGP path splicing, 2007.

[26] B. Quoitin, C. Pelsser, L. Swinnen, O. Bonaventure, and S. Uhlig. Interdomain traffic engineering with BGP. *Communications Magazine, IEEE*, 41(5):122–128, 2003.

[27] B. Ramamurthy, G. Rouskas, and K. Sivalingam. *Next-Generation Internet: Architectures and Protocols*. Cambridge Univ Pr, 2011.

[28] Y. Rekhter, T. Li, and S. Hares. RFC 4271: a Border Gateway Protocol 4 (BGP-4). *Internet Engineering Task Force, Tech. Rep*, 2006.

[29] E. Rosen, Y. Rekhter, et al. Bgp/mpls vpns, 1999.

[30] R. Teixeira, A. Shaikh, T. Griffin, and G. Voelker. Network sensitivity to hot-potato disruptions. In *ACM SIGCOMM Computer Communication Review*, volume 34, pages 231–244. ACM, 2004.

[31] I. van Beijnum, J. Crowcroft, F. Valera, and M. Bagnulo. Loop-freeness in multipath BGP through propagating the longest path. In *Communications Workshops, 2009. ICC Workshops 2009. IEEE International Conference on*, pages 1–6. IEEE, 2009.

[32] V. Van den Schrieck, P. Francois, and O. Bonaventure. BGP add-paths: the scaling/performance tradeoffs. *Selected Areas in Communications, IEEE Journal on*, 28(8):1299–1307, 2010.

[33] Y. Wang, I. Avramopoulos, and J. Rexford. Design for configurability: rethinking interdomain routing policies from the ground up. *Selected Areas in Communications, IEEE Journal on*, 27(3):336–348, 2009.

[34] Y. Wang, M. Schapira, and J. Rexford. Neighbor-specific BGP: more flexible routing policies while improving global stability. In *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*, pages 217–228. ACM, 2009.

[35] D. Wischik, M. Handley, and M. Braun. The resource pooling principle. *ACM SIGCOMM Computer Communication Review*, 38(5):47–52, 2008.

[36] W. Xu and J. Rexford. MIRO: multi-path interdomain routing. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 171–182. ACM, 2006.

[37] X. Yang and D. Wetherall. Source selectable path diversity via routing deflections. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 159–170. ACM, 2006.