# Tiny Internet Project

Bruno Quoitin

January 2, 2008

## 1 Introduction

The objective of this lab session is to model a complex interconnection of networks. For this purpose, you will rely on the C-BGP simulator [QU05, Quo07], a software tool developped at UCL[1]. The model you will build requires the complete configuration of several routers belonging to multiple network domains. This configuration mainly includes the following elements: (1). the physical topology of the network, (2). the intradomain routing protocol configuration as well as static routing setup if needed, (3). the configuration of BGP routers to achieve interdomain connectivity and enforce the business relationships among the participating domains by using routing filters.

In addition to building the network model, you need to provide a detailled report that describes your design choices. For instance, we do not provide the IGP weights of the links. You have to provide them and motivate your choices. The report should also contain a description of how you validated your setup. For instance, how do you ensure that the routing corresponds to your predictions.

This document is organized as follows. Section 2 describes the *Tiny-Internet* topology. Then, Section 3 briefly explains how the modelling tool works. This section also details step-by-step how to describe the *Tiny-Internet* in the modelling tool's language.

## 2 Topology

The topology of the *Tiny-Internet* internetwork[2] is shown in Fig. 1. The topology is composed of 9 different domains. A brief description of the domains is provided in Table 1. This table lists the function of each domain as well as its ASN and the prefixes it owns. Depending on their function, all the domains do not behave equally in the topology and this has an impact on how routing information is exchanged.

First, there are commercial networks such as *BigCarrier*, *Spring* and *iCompany*. Then we have non-profit networks such as *Abilene*, *GEANT*, *BELNET*, *CERN* and university networks: *UCLA*, *UCL* and *ULg*. The commercial company *iCompany*, the university networks and *CERN* are stub networks. They only contain sources and sinks of traffic but they don't provide a transit service to other networks. To the opposite, transit is the core business of the *BigCarrier* and *Spring* networks. *Abilene*, *GEANT* and *BELNET* are special transit networks. They are government-funded networks and they only provide connectivity among non-profit networks (such as universities). We will consider for this lab session that there is one exception: *GEANT* will provide a commercial transit service to *CERN* since it has no other mean to reach the commercial Internet destinations such as *iCompany*.

One **important objective** of this lab session is the configuration of routing policies to constrain the propagation of routing information among domains in conformance with the business relationships described above. You can obtain more information on the link between business

[1] C-BGP is freely available from `http://cbgp.info.ucl.ac.be`.

[2] The *Tiny-Internet* setup is inspired from the real Internet, however, any resemblance to existing network names is purely coincidental.

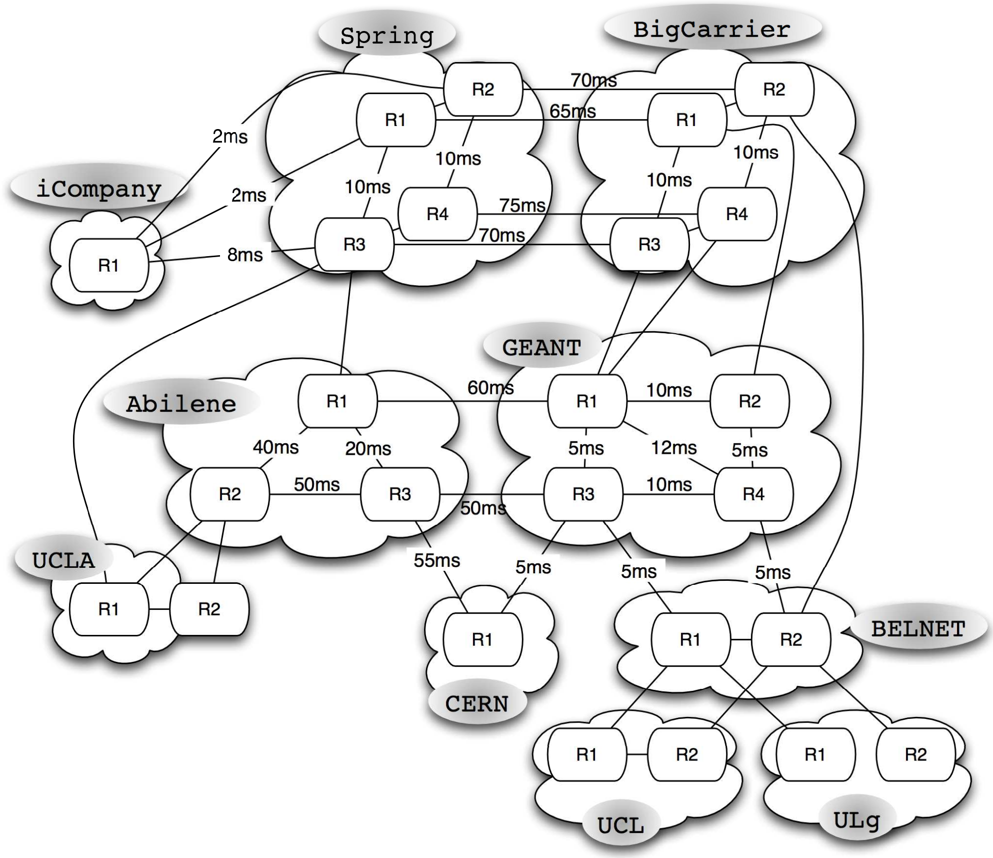


Figure 1: Network topology of the *Tiny-Internet*.

relationships and routing filters in [Gao00, GR00, CR05]. For example, your setup should prevent the *GEANT* network to advertise to *BELNET* routes received from *BigCarrier*, since *GEANT* is not allowed to provide a transit service for commodity Internet traffic to *BELNET*. Another example is the case of *CERN*. Your setup should prevent traffic from transiting through *CERN* as it is a stub network. This must remain true even in case of a link failure between *Abilene* and *GEANT*. We do not list all the business relationships between the different domains but you should be able to infer them from the above paragraph. You are invited to list in your report the business relationships and how your routing policies are defined to enforce them.

Note that in real networks, it is also important to configure routing policies in a way that eases future maintenance. One way to achieve such goal is through the use of BGP Communities. We briefly discuss BGP Communities [CTL96, CB96, BQ03] in Section 3.3.

To build your network model, you will need to make several design decisions just as if you were a network operator. You are of course free to refer to network design and configuration books to learn about best operational practice. Some interesting references are provided at the end of this text [Ste99, HP00, ZB03]. Do not forget to **describe in your report the design decisions you made and why**.

An example of design decision is the assignment of IGP weights to the links. For each router in each domain, you will need to assign an IGP weight. Tuning link weights allows you to influence how the IGP selects shortest paths. You are completely free to select the IGP weights, however, be aware that there are more clever link weight assignments than others. One possibility is to select lowest delay paths. The delay along the links is provided in Fig. 1. We assume that the

| Domain name | ASN | Description | Prefix |
|---|---|---|---|
| BigCarrier | 1 | Commercial Transit Provider | 1.0.0.0/8 |
| Spring | 2 | Commercial Transit Provider | 2.0.0.0/8 |
| iCompany | 200 | Enterprise Network | 2.1/16 |
| Abilene | 11537 | US Research and Education Transit Provider | 3.0.0.0/24 |
| GEANT | 20965 | European Research and Education Transit Provider | 4.0.0.0/24 |
| BELNET | 2611 | Belgian NREN | 4.1.0.0/24 |
| | | | 1.1.0.0/24 |
| CERN | 201 | European Research Institution | 4.200.0.0/16 |
| | | | 3.200.0.0/16 |
| UCLA | 52 | University Network | 3.1.0.0/16 |
| | | | 2.2.1.0/24 |
| UCL | 65535 | University Network | 130.104.0.0/16 |
| ULg | 65534 | University Network | 139.165.0.0/16 |

Table 1:

links without a delay explicitly shown have a 1ms delay.

Note that in this lab session, all the domain networks will be operated by a single person or group: you. However, you need to keep in mind that in the real Internet, these domains are operated autonomously by different entities. In your configuration, you need to take that into account and **maintain a clear distinction** between the configuration of each domain. For example, it is not allowed to configure all the routers of the *Tiny-Internet* in a single IGP domain.

# 3 Modelling tool

In order to better understand the process of building a network model with the C-BGP simulation tool, it is useful to learn some important facts. C-BGP is aimed at computing the outcome of the routing protocols (in particular BGP) in large networks. C-BGP does not model the network at the physical or link levels. For instance, it does not care about Ethernet segments and switches. C-BGP only cares about the layer-3 adjacencies between nodes. Moreover, the model of the intradomain routing protocol in C-BGP is static. That means that the computation of the IGP routes is done in a centralised way based on the knowledge of the topology, without exchanging information between nodes (as in IS-IS or OSPF).

To compute the routes in a network, C-BGP takes into account the network topology and the configuration of all the routers. C-BGP is configurable through a CISCO-like command-line interface. A summary of important C-BGP commands is provided in Section 3.2. When configuring large projects, is is often more convenient to write a script that describes the simulation setup. This setup includes several steps that need to be carried in a well-defined sequence. These parts are described in more details in the following paragraphs.

The first step consists in describing the topology shown in Fig. 1. The topology is composed of nodes (routers) and links. A node is created using the **net add node X** command while a link is added using the **net add link X Y D** command. At the level of the "physical" topology, there is no distinction between domains.

The second step consists in configuring an intradomain routing protocol in each domain. For this, you need to specifiy to which domain each node belongs. In addition, you need to assign an IGP weight to each link in each domain. To create an IGP domain, you need to use the **net add domain D igp** command. Then, each node can be assigned to a domain by using the **net node X domain D** command. You can assign an IGP weight to a link by using the **net link X Y igp-weight W** command. This command assigns the weight in the direction X to Y. You can assign the same weight to both directions by using the **−bidir** option. Finally, the IGP routes are computed by running the command **net domain D compute**. In addition to IGP routing, it is possible to configure static routes. This can be done with the command **net node X route add D G I W**.

The final step consists in configuring BGP routers. To create a BGP router on one existing node, use the command **bgp add router A X**. To add a BGP neighbor to a BGP router, use **bgp router X add A Y**. You need to change the state of the session with each peer in order to activate them. This is done with **bgp router X peer Y up**. Some neighbors will need routing filters to enforce the business relationships among domains. Routing filters can be configured in two directions: for incoming routes (routes received from the neighbor) and for outgoing routes (routes advertised to the neighbor). Adding an inbound route filter can be done with the command **bgp router X peer Y filter in**. The configuration of routing filters is detailled in Section 3.3.

## 3.1 Example

In order to help you write C-BGP scripts, this section provides a small but complete example of configuration. The example topology is shown in Fig. 2. It is composed of two domains, each containing 3 routers. In each domain, the routers are fully meshed. The links are all assigned an IGP weight value of 1. Peering links are not configured in the IGPs; static routes are used instead.

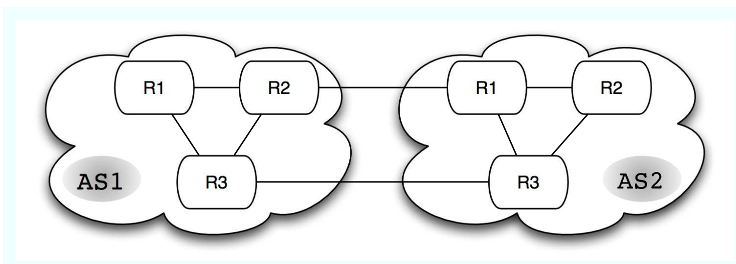


Figure 2: Example topology.

```
# ==========================================================
# AS1
# ==========================================================

# Topology
net add node 1.0.0.1
net add node 1.0.0.2
net add node 1.0.0.3
net add link 1.0.0.1 1.0.0.2 0
net add link 1.0.0.1 1.0.0.3 0
net add link 1.0.0.2 1.0.0.3 0

# IGP
net add domain 1 igp
net node 1.0.0.1 domain 1
net node 1.0.0.2 domain 1
net node 1.0.0.3 domain 1
net link 1.0.0.1 1.0.0.2 igp-weight --bidir 1
net link 1.0.0.1 1.0.0.3 igp-weight --bidir 1
net link 1.0.0.2 1.0.0.3 igp-weight --bidir 1
net domain 1 compute

# BGP
```

```
bgp add router 1 1.0.0.1
bgp router 1.0.0.1
        add peer 1 1.0.0.2
        peer 1.0.0.2 up
        add peer 1 1.0.0.3
        peer 1.0.0.3 up
        exit
bgp add router 1 1.0.0.2
bgp router 1.0.0.2
        add peer 1 1.0.0.1
        peer 1.0.0.1 up
        add peer 1 1.0.0.3
        peer 1.0.0.3 up
        exit
bgp add router 1 1.0.0.3
bgp router 1.0.0.3
        add peer 1 1.0.0.1
        peer 1.0.0.1 up
        add peer 1 1.0.0.2
        peer 1.0.0.2 up
        exit


# =======================================================
# AS2
# =======================================================

# Topology
net add node 2.0.0.1
net add node 2.0.0.2
net add node 2.0.0.3
net add link 2.0.0.1 2.0.0.2 0
net add link 2.0.0.1 2.0.0.3 0
net add link 2.0.0.2 2.0.0.3 0

# IGP
net add domain 2 igp
net node 2.0.0.1 domain 2
net node 2.0.0.2 domain 2
net node 2.0.0.3 domain 2
net link 2.0.0.1 2.0.0.2 igp-weight --bidir 1
net link 2.0.0.1 2.0.0.3 igp-weight --bidir 1
net link 2.0.0.2 2.0.0.3 igp-weight --bidir 1
net domain 2 compute

# BGP
bgp add router 2 2.0.0.1
bgp router 2.0.0.1
        add peer 2 2.0.0.2
        peer 2.0.0.2 up
        add peer 2 2.0.0.3
        peer 2.0.0.3 up
        exit
bgp add router 2 2.0.0.2
```

```
bgp router 2.0.0.2
        add peer 2 2.0.0.1
        peer 2.0.0.1 up
        add peer 2 2.0.0.3
        peer 2.0.0.3 up
        exit
bgp add router 2 2.0.0.3
bgp router 2.0.0.3
        add peer 2 2.0.0.1
        peer 2.0.0.1 up
        add peer 2 2.0.0.2
        peer 2.0.0.2 up
        exit


# ========================================================
# Interdomain peerings
# ========================================================

# Interdomain links
net add link 1.0.0.2 2.0.0.1 0
net add link 1.0.0.3 2.0.0.3 0

# Static routes for peering links
net node 1.0.0.2 route add 2.0.0.1/32 * 2.0.0.1 0
net node 2.0.0.1 route add 1.0.0.2/32 * 1.0.0.2 0
net node 1.0.0.3 route add 2.0.0.3/32 * 2.0.0.3 0
net node 2.0.0.3 route add 1.0.0.3/32 * 1.0.0.3 0

# eBGP sessions in AS1
bgp router 1.0.0.2
        add peer 2 2.0.0.1
        peer 2.0.0.1 next-hop-self
        peer 2.0.0.1 up
        exit
bgp router 1.0.0.3
        add peer 2 2.0.0.3
        peer 2.0.0.3 next-hop-self
        peer 2.0.0.3 up
        exit

# eBGP sessions in AS2
bgp router 2.0.0.1
        add peer 1 1.0.0.2
        peer 1.0.0.2 next-hop-self
        peer 1.0.0.2 up
        exit
bgp router 2.0.0.3
        add peer 1 1.0.0.3
        peer 1.0.0.3 next-hop-self
        peer 1.0.0.3 up
        exit
```

```
# ========================================================
# BGP routes
# ========================================================

# Originate network 1.0.0/24 from AS1
bgp router 1.0.0.1 add network 1.0.0/24

# Originate network 2.0.0/24 from AS2
bgp router 2.0.0.1 add network 2.0.0/24


# ========================================================
# Simulation
# ========================================================

# Run the simulation
sim run

# Perform a traceroute
net node 1.0.0.1 traceroute 2.0.0.2
```

## 3.2 Useful C-BGP commands

This section enumerates a large number of C-BGP commands that your setup will probably need. For each command, a short description is provided along with the list of parameters.

- **include F**
  Load and execute a C-BGP script from file F.
- **net add node X**
  Create a new node with IP address X.
- **net add link X Y D**
  Create a new link between nodes X and Y. An informational delay D is assigned to the link. Note that D is purely informational and it is not taken into account during the simulation. You can assign whatever value you want, it won't affect the outcome of the simulation.
- **net add domain D igp**
  Create a new IGP domain identified by D.
- **net node X domain D**
  Assign node X to domain D.
- **net link X Y igp-weight [–bidir] W**
  Assign weight W to the link X Y in the direction from X to Y. If the option –bidir is specified, the weight is assigned in both directions. Otherwise, it is assigned only in the direction from X to Y.
- **net domain D compute**
  Compute the IGP routes in domain D. Basically, that means that C-BGP will compute shortest path trees rooted at each node in domain D.
- **net node X route add D G I W**
  Add a static route towards destination prefix D in node X. An optional gateway G can be specified (replace by "*" if you don't care). The output interface is I and the route is associated with a weight W.

- **net node X show rt**
  Show the content of node X's routing table.

- **bgp add router A X**
  Create a BGP router on node X. The router is configured as belonging to the BGP domain A.

- **bgp router X**
  Enter the context of router X.

- **bgp router X add network P**
  Originate a prefix P from router X. This will cause the advertisement of BGP routes towards this prefix to all the neighbors of this router.

- **bgp router X add peer A Y**
  Add a peer Y which belongs to the BGP domain A.

- **bgp router X peer Y next-hop-self**
  Configure the router to update the next-hop with its own address for routes received from peer Y.

- **bgp router X peer Y up/down**
  Open/close the session with Y.

- **bgp router X show peers**
  Show the list of peers of a BGP router. This command will also provide the current state of each BGP session (useful for debugging your setup).

- **bgp router X show rib P|***
  Show the content of the BGP routing information base for a single destination prefix (P) or for all the destinations (*).

- **bgp router X peer Y filter in/out**
  Create an input or output BGP filter for peer Y. See Section 3.3 for more details and a complete example.

- **bgp router X peer Y filter in/out add-rule**
  Add a rule to a BGP filter.

- **... match P**
  Define the matching predicate P for the routing filter rule. See Section 3.3.1 below for more details.

- **... action A**
  Define the action A for the routing filter rule. See Section 3.3.2 below for more details.

- **net node X traceroute Y**
  Traces the route from node X to the destination address Y. Note that the traceroute command requires reachability between X and Y in both directions.

- **net node X record-route Y**
  Traces the route from node X to the destination address Y. It is not required that Y be able to reach X in order for the record-route command to succeed.

- **sim run**
  Run the simulation of the BGP convergence.

## 3.3 BGP Routing filters

An important part of the simulation setup concerns the configuration of routing filters. This section explains what is a routing filter and how it is configured. Basically, a routing filter is a sequence of rules. Each rule is composed of two parts: a predicate and an action. The predicate describes to which routes the filter applies. An example of predicate is "*all routes towards prefix P*". The action part describes what must be done with the routes accepted by the predicate. There are three main actions: deny the route, accept the route or change an attribute of the route. An example of action is "*set the value of the route's local-preference to 100*".

In the following example, router 1.0.0.1 is configured to assign the local-preference 100 to all routes received from neighbor 2.0.0.1.

```
bgp router 1.0.0.1
    add peer 2 2.0.0.1
    peer 2.0.0.1
        filter in
            add-rule
                match "any"
                action "local-pref 100"
                exit
            exit
        exit
    exit
```

A common way to configure routing filters in large networks is to rely on a special BGP route attribute named "Communities". Basically, the idea is to define classes of routes that must be treated in the same manner. For example, it is possible to define the class of all the routes received from customer networks. Technically, to do that, all the routes received from customer networks are tagged with a community value. A community value is simply an integer number that is associated with the route. A route can be tagged with multiple communities if it belongs to different route classes.

In the following example, we assume that we have two border routers 1.0.0.1 and 1.0.0.2. Router 1.0.0.1 is connected to provider AS2 through router 2.0.0.1 while router 1.0.0.2 is connected to provider AS3 through router 3.0.0.1. We do not want to provide transit through our providers. Therefore, we need to avoid sending to one provider the routes received from the other one. We show in the example how router 1.0.0.1 is configured to tag all the routes received from 2.0.0.1 with the community 1:1 and how router 1.0.0.2 relies on this community to avoid advertising these routes to provider 3.0.0.1.

```
bgp router 1.0.0.1
    peer 2.0.0.1
        filter in
            add-rule
                match any
                action "community add 1:1"
                exit
            exit
        exit
    exit

bgp router 1.0.0.2
    peer 3.0.0.1
        filter out
            add-rule
                match "community is 1:1"
```

```
                action deny
                exit
            exit
        exit
    exit
```

The following sections list the filter predicates and actions supported by C-BGP. The predicates are shown in Section 3.3.1 while the actions are in Section 3.3.2.

### 3.3.1 Predicates

- **any**
  Match any route.
- **prefix is P**
  Match routes whose destination prefix is exactly P.
- **prefix is P**
  Match routes whose destination prefix is included in P.
- **path RE**
  Match routes whose AS-Path matches the regular expression RE.
- **community is C**
  Match routes whose Communities attribute contain the given community value.

### 3.3.2 Useful C-BGP Filter Actions

- **accept**
  Accept the route.
- **deny**
  Reject the route.
- **local-pref L**
  Set the route's local-pref to L.
- **metric M|internal**
  Set the route's multi-exit-discriminator to M or to the internal IGP weight.
- **as-path prepend N**
  Prepend the route's AS-Path N times.
- **community add C**
  Add the community value C to the route's Communities attribute.
- **community strip**
  Remove the content of the route's Communities attribute.

# 4 Summary

At the end of the project, we expect that you will provide the complete *Tiny-Internet* configuration in the C-BGP scripting language. This configuration can be provided in a single or multiple files (one per domain for example). In addition to this, we expect a detailled report where you describe your design choices. This report must be no longer than 6 pages.

The **deadline** for this project is December, 17th 2007. Please package all your files in a single .tar.gz archive with the following structure. The name of the archive must be `tiny-internet-xxx.tar.gz` where `xxx` must be replaced by your group number. The archive must contain a single directory named `tiny-internet-xxx` (`xxx` is your group number). In this directory, you must place your report in **PDF format**. The name of the report must be `report.pdf`. In the same directory, you must place a sub-directory named "config" that contains all the C-BGP scripts. All the script must have the extension `.cli`. Your project archive must be uploaded on the iCampus website under the section of your group.

If you have questions regarding this lab session, please use the iCampus forum first. If the question remains unanswered, then send me an e-mail at `bruno.quoitin@uclouvain.be`. Make sure you provide me enough details to understand your question (script, example, and so on). If you need to meet me, please first contact me by e-mail in order to make an appointment.

Good luck !

Bruno Quoitin

# 5 Some questions to keep in mind

- Ensure that no traffic is allowed to transit through stub networks and more generally that business relationships are enforced with routing filters. It is especially important to enforce the valley-free property and route ranking (customer > peer > provider). See [Gao00, GR00] for more details. Ensure that no commercial traffic is allowed to transit through the research transit networks.
- Try to select lowest delay routes as much as possible...
- Validate the routing selection obtained. Does it match what you expected ? How would you automate that check ?
- Determine what will happen if the link between X and Y fails ? Are you sure the business agreements are still respected ? Example: links between *Abilene* and *GEANT*.

# References


[BQ03] O. Bonaventure and B. Quoitin. Common utilizations of the BGP community attribute. Internet draft, draft-bonaventure-bgp-communities-00.txt, work in progress, June 2003.

[CB96] E. Chen and T. Bates. An Application of the BGP Community Attribute in Multi-home Routing. Internet Engineering Task Force, RFC1998, August 1996.

[CR05] M. Caesar and J. Rexford. BGP routing policies in ISP networks. *IEEE Network Magazine*, 19(6), November 2005.

[CTL96] R. Chandra, P. Traina, and T. Li. BGP Communities Attribute. Internet Engineering Task Force, RFC1997, August 1996.

[FB05] N. Feamster and H. Balakrishnan. Detecting BGP Configuration Faults with Static Analysis. In *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI)*, May 2005.

[Gao00] L. Gao. On Inferring Autonomous System Relationships in the Internet. *IEEE Global Internet*, November 2000.

[GR00] L. Gao and J. Rexford. Stable internet routing without global coordination. In *SIGMETRICS*, 2000.

[HP00] B. Halabi and D. Mc Pherson. *Internet Routing Architectures (2nd Edition)*. Cisco Press, January 2000.

[MWA02] R. Mahajan, D. Wetherall, and T. Anderson. Understanding BGP misconfigurations. In *Proceedings of ACM SIGCOMM 2002*, August 2002.

[QU05] B. Quoitin and S. Uhlig. Modeling the routing of an Autonomous System with C-BGP. *IEEE Network Magazine*, 19:12–19, November 2005.

[Quo07] B. Quoitin. C-BGP Routing Solver. `http://cbgp.info.ucl.ac.be`, 2007.

[Ste99] J. Stewart. *BGP4 : interdomain routing in the Internet*. Addison Wesley, 1999.

[Sys05] CISCO Systems. BGP Best Path Selection Algorithm. `http://www.cisco.com/warp/public/459/25.shtml`, October 2005.

[ZB03] R. Zhang and M. Bartell. *BGP Design and Implementation: Practical guidelines for designing and deploying a scalable BGP routing architecture*. CISCO Press, 2003.