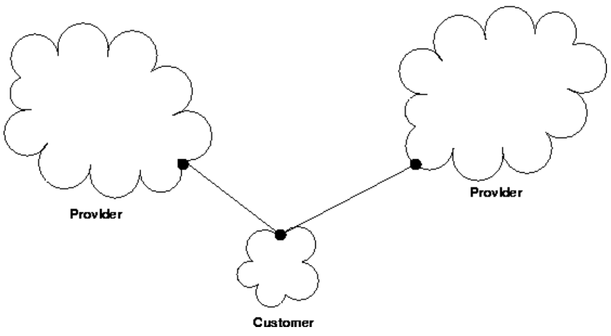
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing Group

Internet Topology

Comparing different approaches of inferring AS relationships



Beatrice Huber

Master Thesis

September 15, 2003 – March 14, 2004

Supervising Professor:	Prof. Dr. Roger Wattenhofer
Supervising Assistant:	Regina O'Dell

Preface

Abstract

The Routing in the Internet is strongly affected by the business relationships between administrative domains (Autonoumous Systems, AS).
This paper compares existing approaches inferring such relationships to each pair of directly connected ASes and proposes modifications. Starting from an algorithm based on the degree of an AS [1] to infer the importance of an AS, we introduce three alternative measures: First, the range of IP address space assigned to the AS; second, the number of important websites (rated by GOOGLE) and third, the number of users having an IP belonging to the AS (approximated by Gnutella log-files).
We also analyze a rank based algorithm [2] that uses routing tables from different vantage points. Additionally, we propose a new approach to infer relationships by parsing the WHOIS Databases. This has the advantage that much more peering relationships are revealed and is the simplest way to a obtain a global view on the AS graph.
Common problems arising are discussed and relationships resulting from the algorithms are compared.

Acknowledgment

I am grateful to all people who supported me in writing this thesis, especially my tutor Simon Leinen who gave lots of helpful ideas. I am much obliged to Regina O'Dell who gave me always the needed support and proof read the paper again and again. I am also grateful to Prof. Roger Wattenhofer and his team.

Contents

Chapter 1

Introduction

1.1 Motivation

The internet is not one big network operated by one large international company, but consists of loose interconnection of networks belonging to different Autonomous Systems. There are more than 16'000 ASes that exchange routing information. An AS is described as "a single administrative domain," although it can span more than one organisation, for example, an ISP (Internet Service Provider) and its (non-BGP speaking) customers. An AS can be seen as one entity from the outside and there is no need to know about the inner details of the network of an AS. The routing inside an AS is controlled by intra-domain routing protocols like OSPF or IS-IS. BGP is used to exchange routing information between Autonomous Systems, that is, how to reach a range of IP addresses, a so called prefix. It is a path vector protocol, which means that the routing information is a sequence of AS numbers between the router and the destination, called AS Path. Only updates and keep-alive messages will be propagated after a session between two BGP speakers is established.

Given these AS Paths one may construct a graph of connections, but this does not tell much about the reachability between the ASes. In contrast, reachability is determined by routing policies which are defined by the business relationships between ASes. As an example, see Figure 1.1, it is not desirable for an AS with several upstreams (multihomed) that its providers exchange traffic between each other through their common customer. So although there is a connection through the customer AS, two providers will send traffic intended for each other using a different path, e.g. via their provider.

Thus, global routing cannot be understood by just looking at the physical links connecting ASes without taking into account the influence of business decisions. Knowing more about the relationships between the ASes is essential in understanding global routing. It could be helpful to know the position of an AS in the global routing graph to find out if this AS is eligible as a peering partner and so alleviates the planning for future peering strategies. It also can aid to select an AS to place replicas of webservers, by knowing how and how redundant by it is connected to the AS where the users of the webserver are located. To reach maximal reliability, the ISP can purchase global access

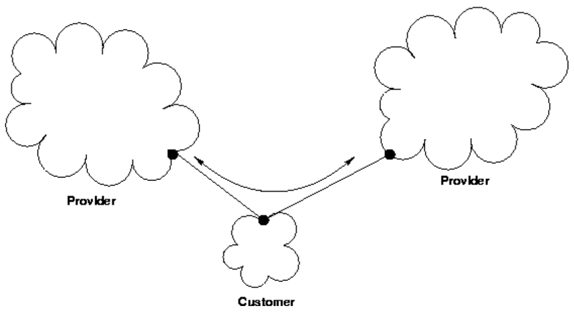


Figure 1.1: Provider should not route over their common customer

from preferably independent providers which can be seen from the relationships between ASes. Knowledge about the relationships also helps to build a hierarchy and classify the ASes into different tiers. BGP misconfigurations could be identified by detecting errors in the exporting rules, see also [12].

1.2 Exporting rules

In this section we want to have a closer look at the rules for exporting route advertisement. They can be derived from the type of relationships between ASes and are assumed by the algorithms analysed. There are four types of relationships: customer-provider, provider-customer, peerings and siblings. Essentially the first three types are discussed here, siblings are treated as peers if nothing else is mentioned. A customer pays the provider for connectivity to the Internet. Peerings decrease the dependency on providers and therefore the cost of purchased Internet transit. Both partners should equally benefit from this rapport and that is why they normally do not pay each other for traffic. Siblings have mutual transit agreements, usually for backup. In this section, we want to have a closer look at the relationships between ASes. The BGP policies are influenced by these relationships. BGP, as an inter-domain routing protocol, is appropriate to support policies between the Autonomous Systems which represent their business relations. Using the community attribute, you can control the behavior of your provider as long as it respects the communities you send. The policies can be expressed by using filters. The implementation of the filters depends on the business relationship you have to your neighbouring ASes. You can define which routes you accept from whom and which routes you announce to whom. The route selection is also driven by financial considerations primarily.

We consider four types of relationships: customer-provider, provider-customer, peerings and siblings. Essentially the first three types are discussed here. Siblings are treated as peers when comparing an approach producing sibling to one without the type sibling. A customer pays the provider for transit access to the Internet. Peerings decrease the reliance on providers and therefore the cost of purchased Internet transit. Both partners should equally benefit from this rapport and that is why they normally do not pay each other for traffic.

Siblings have mutual transit agreements, usually for backup. The following exporting rules are applicable in most of the cases due to the business connection between the parties. It is common to assume these rules when talking about routing policies.(Done so in [1], [2], [7])

Exporting to a provider: An AS can export its internal routes and its customers routes to the provider, but usually does not export its other provider or peer routes.

Exporting to a customer: An AS can export its internal routes and its customers routes and as well as its provider and peer routes.

Exporting to a peer: An AS can export its internal routes and customer routes, but usually does not export its provider or peer routes.

Exporting to a sibling: An AS can exports its internal routes, the routes of its customers and as well its provider or other routes.

Of course, this is a quite simplified view. In reality there are hybrid forms of the four types of relationships. For example, an AS A must pay to get some routes announced by AS B, but for some other routes they act as peers. Partial transit restricted to a geographical region (regional transit), or a combination of being a customer and partial transit provider to an upstream AS are not uncommon. Paid peerings represent such a hybrid form as well.

1.3 Contributions

In this paper, we present a compilation of different methods to infer business relationships between Autonomous Systems. We implemented all discussed approaches and compare the generated results among each other and comment on them. Most other papers only compare their approach with the earliest algorithm due to Lixin Gao.

The algorithm of Lixin Gao, described in Section 2.1, uses the number of neighboring ASes as a weighting mechanism when inferring relationships. We run this algorithm again, but with different weighting mechanisms. We choose the address space assigned (see Section 2.1.2) to an AS on the one hand, on the other hand the distribution of important IP-addresses (see Sections 2.1.3 and 2.1.4) over the ASes to express its size. Additionally, we parse the Internet Routing Registries to infer the relationships.

Chapter 2

Approaches

This section elucidates different approaches on how to infer the relationships between two ASes as classified in Section 1.2. The results of each approach are described in Section 3. In Section 4 you find an overview of the problems and results.

2.1 Valley-free paths

The approach used by Lixin Gao [1] is based on the assumption that every AS-Path is valley-free. This means that after traversing a provider-to-customer or peer-to-peer edge, the AS path can not traverse a customer-to-provider or peer-to-peer edge anymore. In other words: An AS path consists of a maximal uphill path followed by a maximal downhill path. The AS where uphill and downhill path come together is called the top provider of that path. ASes on the uphill path have a customer-provider-relationship while ASes on the downhill path have a provider-customer-relationship.

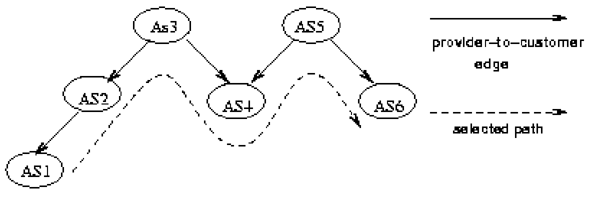


Figure 2.1: Non valley-free path

2.1.1 Degree based

In a first step the algorithm parses the routing tables and calculates the degree of each AS by counting its different neighbors in all of the paths. Next, Gao iterates over all AS paths found in the routing table. The top provider of each AS path is considered to be the AS with the highest degree among all ASes in the current path. This comes from the intuition that a provider typically has

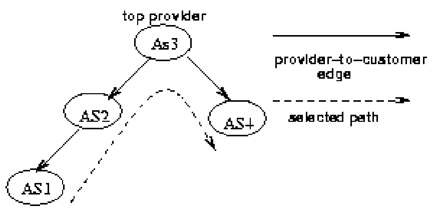


Figure 2.2: Valley-free path

a larger size than its customers and that the size and importance of an AS are typically proportional to its degree in the AS graph.

In a second step, the number of routing tables entries classified as a certain relationship to an AS pair are summed up.

Finally, Gao assigns a relationship to an AS pair according to the entries counted. Peerings may only be located next to the top provider, according to the valley-free path assumption. To be classified as a peer, the ratio of the degrees between the top AS and the potential peer may not be higher than a given factor *R*. Gao classifies siblings, but compared to other algorithms, siblings are considered peerings.

2.1.2 Address space based

Because the results generated by the algorithm in [1] are highly dependent on the degree (see Section 3.1.1), we had the idea to use a different measure to deduce the importance of the AS. An approximation is to choose the size of the IP address range announced by an AS. The routing table available from the data archive of the Route View Project served as a basis. All announced prefixes are assigned to the last AS in the AS path (origin AS). The total address space of an AS is then calculated by summing-up up the number of addresses contained in each prefix.

First of all, the prefixes originating in private AS numbers (64512-65535) have to be replaced by the public AS number from where it originates. We determined this by asking the routing table and by looking at the second-to-last AS in the path where the private AS number appears. Note that private AS numbers should not be announced in the global BGP routing. This mistake happens for about 100 announced prefixes, but most of the prefixes belong to the same AS. Only a few ASes continue announcing private ASes.

Some ASes announce the same network twice, for example once as a /16 and again as a /20, which is completely contained in the /16. (For example AS 271 announces the following prefixes 134.87.0.0/16, 134.87.110.0/23, 134.87.224.0/20 and 134.87.225.0/24. These overlaps are removed when counting addresses.) When looking closer at the prefixes, we see that there are some Multiple Origin AS (MOAS) prefixes, which means that there are two or more (up to six in one case) ASes where the prefix originates. They may be anycast addresses which belong to everyone when comparing. They are counted as belonging to each one of those ASes.

(In)efficient address use It was easy to get a /8 in the early days of the Internet, when the elementary task of maintaining a list of assigned network addresses was carried out voluntarily by Jon Postel, using (according to legend) a paper notebook. Today the Regional Internet Registries (ARIN, APNIC, LACNIC, RIPE) are responsible for the allocation and assignment of Internet Protocol address space. The goal of a controlled assignment are uniqueness, aggregation, conservation, and registration.

A Local Internet Registry must demonstrate an existing, or immediate need for efficient utilisation of at least a /22 of address space. Then it gets a /20, which is the minimum allocation size. Only when about eighty percent of all the address space currently allocated is used in valid assignments or suballocations is it possible to augment the address space. So, we can deduce that a /8 is an old network, but we do not know anything about the density of the addresses used in the network. Maybe it is a huge network with thousands of end users or just a relict. To give less influence to the /8 nets, we weighted them as 4 times a /16 net.

Some of the companies which got a /8 announce the whole /8, others announce just a subnet of it. Some /8 have also been split up and shared between multiple ASes, for example the 12.0.0.0/8 was given in June 1995 to AT&T Bell Laboratories. Today over 600 ASes announce a prefix out of the range originally given to AT&T Bell Laboratories, whereas most of them are in a customer-provider relationship with AT&T.

Some ASes do not have their own address space, because they only provide transit for others. Although they do not have their own address space, they may have several customers and be of some importance from the global point of view. For example AS5673, Pacific Bell Internet Services, provides transit for some ASes, classified as customers. In the routing table, though, AS5673 never appears at the end of an AS path, it only provides transit for customers.

Because of these reasons it is necessary to include the customers' size of an AS to calculate its real size. To do so, there is a need for already knowing customer-provider relationships, in order to calculate the address space recursively. So in the first step, we took the results produced by the original degree based algorithm [1] to determine the importance of an AS based on the address range. When calculating the address space recursively, some cycles become apparent in the data used as input. They had to be analysed and broken to finish the recursive calculation of the address range. (See Section 4.1)

When we now run the algorithm [1] with the address space as a measure instead of the degree, the peer ratio always has to be set to infinity. This comes from the different magnitudes between address space and degree. The degree varies only between 1 and about 2400 whereas the address space counted recursively varies between 254 and some millions.

2.1.3 Important website based

By seeing how difficult it is to find out the actual host density of an address range, we considered a totally different measure to express importance. We collected 65'000 IP addresses by querying google with random words and saving the IP address of the top ranked website. Thus, the more important websites having an IP address belonging to a prefix of your ASes exist, the more important is your AS.

2.1.4 Based on IP addresses of endusers

We were also interested in the ASes at which the endusers are located, and whether there is a correlation between the number of endusers and the number of hosted websites within the same AS. To count users in an AS, we used logs files (dated June 2002) with roughly 500'000 IP addresses of Gnutella users. This comes from the idea that an AS gets more important the more users are connected through it.

Because Gnutella is a very open protocol, IP addresses can be faked or internally-used address may be propagated. The number of addresses in the logs which were not routed was 1%. However, we assume that the percentage of faked addresses is quite low because there is no great motivation to do so.

2.2 Rank based

The rank based algorithm [2] constructs a directed graph out of the views from multiple vantage points. A single view is obtained from a routing table of a telnet or looking glass server. The relationships between the ASes may be interpreted differently from various vantage points. Then every node of every view gets a rank. The rank is assigned by a simple algorithm. Every leaf of the graph gets rank 1 and the leaves are removed. The leaves of the so remaining graph get rank 2 and are then pruned as well. This goes on as long as there are leaves. All nodes of the residual graph get the same rank. In a further step an N dimensional vector, where N is the number of vantage points, $c(i) = (r_{i1}, ...r_{ij})$ is assigned to every AS, where r_{ij} is the rank of AS i in the view j. To decide which type of relationship is applicable for two ASes the following heuristics are used:

Equivalence: Two ASes are said to be equivalent if the number of coordinates which are equal in the two vectors is greater than $N/2$. If they are joined by an edge, they are likely to be peers.
Dominance: Let $l(i, j)$ refer to the number of coordinates where $r_{ij} > r_{jk}$. A vector $c(i)$ is said to dominate $c(j)$, if $l(i, j) > l(j, i)$=0. The fact that $c(i)$ dominates $c(j)$ implies that AS i is a provider of AS j.

All edges not matching these rules are determined by probabilistic rules as follows:

Probabilistic equivalence:

$$\frac{l(i,j)}{l(j,i)} \leq \delta_1, \text{ for } \delta_1 \text{ close to 2.}$$

Probabilistic dominance:

$$\frac{l(i,j)}{l(j,i)} \geq \delta_2, \text{ for } \delta_2 \text{ close to 3.}$$

Its recommend by the authors of [2] to choose the values of δ_1 and δ_2 as above. If none of these rules matches, then the relationship is not decidable.

2.3 Registry

The fourth approach to find out about the relationships between the ASes is to read the information out of the Internet Routing Registries (IRR). About 10'000 of the 16'000 ASes have written some pieces of information about their routing policies. This gives you a much more complete view than you can ever reach with the routing tables from one router, or even several vantage points. However, the main problem is that the entries in the Registries are not always up to date (for some ASes, the last update was made in 1997), and there is no guarantee that these entries are complete or correct. This happens because the registries are mostly maintained manually and there is no regulation to update or enter your routing polices. Some ASes update them very well, because they use router configurations generated from these entries.

The entries in these WHOIS databases are entered in the Routing Policy Specification Language (RPSL) [11]. RPLS provides classes of records to save the policies or maintain information. We are interested in the data about the import and export statements. They can be found in the aut-num class. We used the databases of RIPE and RADB to gather the data.
The data out of the registry entry of AS *A* are interpreted in the following way:

```
import:  from AS B accept ANY
```

is interpreted as AS *B* is a provider, because as the import rule prescribes you only import all routes from your provider. Likewise,

```
export:  to AS B announce ANY
```

means that AS *B* is a customer of AS *A*, according to the exporting rule telling that you only export all your known routes to your customer. Peerings are identified by the following entry

```
export:  to AS B announce AS A
import:  from AS B accept AS B
```

An AS *A* operating partial routing has an entry for its default provider *B*:

```
default:  to AS B networks ANY
```

The statement `default` dominates over other relationships that would be inferred.

According to the exporting rule, you announce all your customer routes to your provider or peers. It can be written like the following statement as in the case for AS22:

```
export: to AS4302 announce AS22 AS5855 AS5303
```

AS5855 and AS5303 are customers of AS22. The customers of an AS can be collected in an AS-Macro. The name of a macro can be freely chosen. It may be written as in Table 2.3.

The AS macro announced the most to other ASes we interpreted as the macro containing the customers. Querying the database for AS-GLOBAL returns the output displayed in Table 2.3, which means that the listed members

```
export:to AS21409  announce AS-GLOBAL
export:to AS8729   announce AS-GLOBAL
                 AS-SWISS AS-NATIONAL
export:to AS8387   announce AS-GLOBAL
export:to AS8404   announce AS-SWISS
                          AS-NATIONAL
export:to AS8406   announce AS-GLOBAL
```

Table 2.1: Example for AS6730

are customers of AS6730. In this example one would lose data by only parsing

```
as-set  :  AS-GLOBAL
descr   :  SUNRISE, TDC Switzerland AG
descr   :  Customers with
                   global connectivity
members :  AS3092
members :  AS3291
members :  AS5398
members :  As5426
members :  AS6730
```

Table 2.2: AS-GLOBAL

AS-GLOBAL in Table 2.3, because the macro AS-SWISS contains additional customers. But parsing all macros announced to other ASes can lead to mistakes as well. The macro AS-NATIONAL, for example, contains only peers, see Table 2.3, so it should not be interpreted as a macro containing customers.

```
descr   :  sunrise National (equal)
               peers in Switzerland
descr   :  connected to AS6730
members :  AS513
members :  AS559
members :  AS3291
members :  AS3303
```

Table 2.3: AS-NATIONAL

The registry entry is also parsed for the occurrence of macros like AS-PROVIDER, AS-TRANSIT, AS-PEERS, AS-CUSTOMERS, which can be quite fruitful. To decide, in the case of nested macros, whether it is a direct or indirect customer, there needs to be a check in an undirected graph which represents all possible connections in the routing table.

Very recently, it came to our attention that the paper of G. Siganos and M. Faloutsos [7] also describes an approach of parsing the registry. Their main goal is to quantify the quality of current Internet Routing Registries. They identify common mistakes and problems in IRR registries. But they also inferred the relationships between ASes.

2.4 The Simplified Robban Tool

A very simple idea to infer the type of relationships between ASes is only looking at the last two ASes of a path and considering only paths consisting of at least three ASes. The last AS in the path is considered a customer of the next to last AS. The occurrences of every pair are counted from the routing tables seen at Oregon Exchange BGP Route Viewer. Only pairs that are seen at least three times are considered, otherwise the pair is not relevant. We can deduce a peering relationship if there are pairs where AS A provides transit for AS B and AS B is providing transit for AS A as well. The original idea for this tool comes from the website netlanis.org [8], where the implementation is more complex than in the version described above. They fix the highest occurrence of a link, where the considered AS is the last AS in the path, as a customer-provider relationship. Then they compare the ratio between the highest and the second highest occurrence of a link and deduce from this ratio whether it is a provider-customer relationship too or a peering. Then the second highest occurrence is compared to the next smaller one and so on. Once deduced a peering, all smaller occurrences of links will be classified as peerings as well.

Chapter 3

Results

In this Section, we provide an overview of the results of all the algorithms discussed. We mention advantages and disadvantages of every approach.

A general difficulty is that all algorithms, however simple or complicated they are, are dependent on a parameter to express a ratio. How to tune this ratio has to be found out by looking at examples and observing when they are interpreted correctly. This means you already need some information about some ASes to set your parameter, and tuning the parameters shows that the percentage of peerings can change considerably.

In Table 3 we see the percentage of peerings produced with different algorithms and the total number of relationships seen by this approach. In this comparison, siblings are counted to the peerings. The rest of the relationships are therefore customer-provider relationships. For the degree and rank based algorithms, the results are averaged over 10 runs. For algorithm [2], we took the results published on their website [3] and we also ran the algorithm with the code they provided us, once with the recommended $\delta_1 = 2, \delta_2 = 3$ and again multiplied by a factor of 100. The factor 100 was chosen to achieve a clear effect and an upper limit. There is an upper limit where the parameters still can rise, but do not produce any more effect. This shows how much the percentage of peerings can vary by change the parameters. When talking about the percentage of peerings, it depends so much on which subgraph of the complete internet graph you are watching. The registry can identify local peerings between small ASes whereas all algorithms using routing tables do not even see a connection.

3.1 Valley-free path

In this section, we look at the results of every variant of the approach which assumes that all paths are valley-free. We find that for every consideration of importance errors arise. If you are an AS providing web-content, you probably do not care about huge address space as peering criteria, but you are interested in peerings with ASes where the readers of your websites are located. So the reason how you choose your providers and peerings do not only depend on the address space or the degree of an AS. Thus, the complexity of peering decisions can not be mirrored by one general definition of importance. Every weighting

Algorithm	Peers	Total
Degree (see 2.1.1)R=60	7.2%	33636.66
Degree (see 2.1.1) $R= \infty$	7.98%	33636.66
Addressspace based 2.1.2	7.38%	33636.66
Website based see 2.1.3	7.34%	33636.66
Endusers based see 2.1.4	7.36%	33636.66
Rank 2.2 website	4.03 %	34299.2
Rank 2.2 $\delta_1 = 2, \delta_2 = 3$	1.41%	34814.5
Rank 2.2 $\delta_1 = 200, \delta_2 = 300$	3.86%	34646.7
Registry see 2.3	47.89%	77261
Robban Tool 2.4	0.51%	30428

Table 3.1: Percentage of Peerings
The Total expresses the number of all connections seen by an algorithm, for the valley-free path approaches averaged over 3 runs, for the rank based approaches over 10 runs

mechanism leads to a different ranking of the ASes. Tables 3.1.3-3.1.3, showing the top five providers for each approximation of importance, illustrate this effect.

3.1.1 Degree based

The assumption that the degree correlates with the importance of an Autonomous System in the global hierarchy of ASes may be problematic for the following reasons. A transit AS may have few, but important providers and peers. The customers of this AS may use private AS numbers or an internal routing protocol, so they will not be seen at all in the BGP routing tables.

The input data for this algorithm are gathered by a route collector which receives all routes of peering ASes. These neighbors of the route collector announce all their peer routes to the route collector as well. According to the exporting rule, peer routes are not announced to other peers or providers, so most peerings are only seen locally. An AS with lots of peers cannot be distinguished from an AS with few peers, because the peerings do not influence the degree of the AS seen by the algorithm, except for the ASes directly connected to the route collector.

There is also a difficulty in fine tuning the ratio R in [1]. The value R has been set to 60. The more BGP routing tables you have, the less important is the choice of R (can be set to infinity then). This is because we can eliminate an AS pair from having a peering relationship if the AS pair appears in any BGP routing table as having a transit relationship.

The advantage of this algorithm is that it infers siblings, so we get a more specific analysis. Reducing a sibling to a peer can hurt the exporting rule.

In Table 3.1.1, the results of this algorithm (using $R = 60$) during half a year are displayed. We can see the number of appearing ASes and the number of connections rising over time.

Total	Peers	Percentage	ASes	Date
34154	2617	0.0766235	16540	20031229
33512	2365	0.0705717	16375	20031128
33244	2288	0.0688244	16320	20031113
32623	2163	0.0663029	16221	20031029
32850	2280	0.0694064	16132	20031015
32182	2019	0.0627369	16019	20030930
31328	1657	0.052892	15888	20030902
31111	1609	0.051718	15829	20030818
30584	1422	0.0464949	15742	20030804
30345	1384	0.0456088	15658	20030721

Table 3.2: Results of algorithm [1]

3.1.2 Address space based

The results produced by the algorithm using the new input do not differ so much from the algorithm using the degree: less than 3% of the relationships change. And the new results show no drift towards more correctness. The number of improvements and the number of changes for the worse are about the same. For AS3303, more peers are correctly recognized, whereas for AS4589, the peers wrongly become customers. The shift of the ASes in the ranking causes these changes. AS4589 had lots of wrongly interpreted customers before, so they were all counted to AS4589's address space which makes the AS more important, and so the old peers become customers. The problem is that old mistakes are carried over and, of course, new mistakes may be generated when giving too much or too little influence to an AS due to the difficulty in assigning the used address space.

3.1.3 Important IP addresses based

In this section the results of the important websites, see 2.1.3, and the IP addresses of the endusers, see 2.1.4, are summarized, because the arising problems are very similar.

Compared to the results of the degree-based approach, the main difference is that there are over 10'000 ASes to which no IP address out of our sample belongs to. Since we have no information about them, they are all weighted equally. There are a lot of ASes not belonging to ISPs, but to companies which neither host websites nor sell Internet access to endusers. ASes that only provide transit for other ISPs, but have no endusers or websites within their address space are not considered as being important which is the same problem when looking only at the address range. To mention an example of what goes wrong, the peerings of AS8210 (Teleonor backbone) disappear and instead AS8210 becomes the provider. This happens because AS8210 has no important website nor gnutella users. There are only five smaller prefixes announced by this AS number. In most cases AS8210 is located next to AS2119. AS2119 (Teleonor internet access) announces a large addresspace where we find some important websites and gnutella users. So, AS2119 always becomes the top provider and according to the valley-free path rule, a peering can only be with the top provider. This means that AS8210 no longer can be interpreted as a peer. When considering the degree of the ASes, AS8210 has degree 205 and AS2119 degree 5.

ASnumber	ASname	Degree
701	UUNET	2382
1239	Sprintlink	1770
7018	AT & T	1627
209	Qwest	999
3356	Level 3	968

Table 3.3: Number of Degree, date 2003/11/28

ASnumber	ASname	Addresspaces
568	DISO-UNRA	72,625,925
3356	Level 3	44,663,587
701	UUNET	35,453,551
7132	SBC	30,510,764
7018	AT& T	28,520,584

Table 3.4: Top five ASes according to addressrange

ASnumber	ASname	Users
1668	AOL	88631
3320	Deutsche Telekom AG	30330
7018	AT& T	25004
7132	SBC	24004
22909	Comcast Cable	14427

Table 3.5: Number of Gnutella Users per AS

ASnumber	ASname	Sites
2914	Verio	2001
13749	Everyones Internet	1457
11305	Interland Incorporated	896
3356	Level 3	719
701	UUNET	677

Table 3.6: Number of Websites (total 65'000 collected)

3.2 Rank based

The rank-based algorithm sees more connections thanks to multiple vantage points. But the total number of peerings seen is higher when using algorithm [1], although algorithm [2] has multiple vantage points and so theoretically could see local peerings. Table 3.2 shows the number of all connections seen by both algorithms, the number of connections interpreted as peerings from both algorithms and the number of customer-provider relationships which they have in common (siblings are considered as peers for this comparison). The number of relationships interpreted as peerings from algorithm [1] and algorithm [2] is quite low.

The largest difference of the results taken from the website [3] compared to the algorithm [1] is the number of peerings, interpreted as customer-provider relationships by the rank-based approach. This is the case for ASes with open peering policies directly connected to the routecollector like AS3303, AS12956, AS3257. ASes tend to be more liberal for peerings abroad, so both parties can save transit cost, and they do not compete with each other, because their customer are from different regions. Thus, small ASes can have peerings with bigger ASes. Algorithm [2] is error-prone in theses cases, because of the different ranks assigned to ASes not in the same stage. (See Section 2.2). The algorithm [2] is better in peerings between bigger ASes like AS701 (UUnet), AS3356 (Level 3), AS1290 (SprintLinkBackbone).

Common links	Peers	Cust.-Prov.	Date
32634	423	29653	20031229
28660	143	27634	20031128
32094	401	29302	20031113
31576	434	28918	20031029
31649	430	28850	20031015
29915	373	27800	20030930
27665	131	26754	20030902
30362	287	28433	20030818
30260	260	28351	20030804
29842	268	28026	20030721

Table 3.7: Common Connections of algorithm [1] and algorithm [2]

3.3 Registry

In this Section, we list problems which complicated the interpretation of the registry entries.

Several ASes always accept ANY for simplicity and for independence of macro name changes of the neighbouring ASes. When generating filters out of this information, routes announced by mistake will not be detected. This "accept ANY" also leads to misinterpretation because all their peers and customers may be interpreted as providers. When announcing ANY to the customer it is possible to detect the conflict. For peers, there is still a chance to find a conflict when seeing the registry entry of the other AS. But in case there is no policy entry for the other AS, we have no control mechanism.

```
import: from  AS2818  accept ANY
import: from  AS6728  accept ANY
import: from  AS8406  accept ANY
import: from  AS9191  accept ANY
import: from AS16353 accept ANY
import: from AS20915 accept ANY
import: from AS13005 accept ANY
import: from  AS5587  accept ANY
import: from AS25108 accept ANY
import: from AS12621 accept ANY
```

Table 3.8: accept ANY from everyone

The statements below, see Table 3.3, lead to the decision of peers, although AS701 is the customer. With some additional knowledge about the top ten providers on the world, such mistakes would not happen.

One could get much better results out of the registry when not just parsing, but being able to understand the context. An example:

or

```
AS8437
remarks:  ----------------------------
remarks:   Peerings NIX
remarks:  ----------------------------
```

A human would immediately understand the remark, but because everybody is free how or whether at all to write remarks, it is impossible to understand all possible remarks just by parsing.

In the registry, there are data about inactive ASes as well. Some ASes have changed the AS number, but the old entry has not been deleted.

When looking at routing tables, peerings are only seen locally. But when looking at the Registries, you get information about all ASes who have an entry. So we get a much higher percentage of the number of peerings. There are over

```
import:  from AS701    accept AS701
import:  from AS14677  accept AS14677
export:  to AS701      announce AS7046
export:  to AS14677    announce AS7046
```

Table 3.9: One-sided entry

```
remarks:Peers:pref=80,Transit:pref=200
remarks:AS112 Project:
import :from AS112 action pref=80;
                       accept ANY
export :to AS112 announce AS-BIT
remarks: SurfNet:
import :from AS1103 action pref=80;
                       accept ANY
export :to AS1103 announce AS-BIT
```

30'000 peerings registered in the Registry that are not seen in the routing tables used to run the algorithms. Either they are legacy entries or it is a local peering that can not be seen by the other algorithms. For example AS1257 peers with AS16245, which can be verified by the looking glass server of AS1257.

Compared:	Reg./Alg.[1]	Reg./Alg.[2]
Common Links	30285	31257
Peerings	1736	671
Cust.-Prov.	26972	26996

Table 3.10: Comparison of the registry to algorithm [1] and algorithm [2], December 2003

We present a comparison of the results from the registry and algorithm [1] and algorithm [2] in Table 3.3. The authors of the paper [7] stated that 83% of the relationships inferred are identical with results produced by the algorithm [1]. It should be said that they also inferred siblings. It is a nice idea to infer siblings. But only looking at the entry of one AS announcing and accepting ANY from its neighboring AS is not very reliable, because many ASes enter this for their provider as well.(e.g. (AS25899, AS701), (AS25899, AS2914), (AS25899, AS7911), (AS28973, AS701), (AS29748 ,AS701), (AS30107, AS4323)) For ASes mutually announcing ANY and accepting ANY the chance is higher that they act as siblings, but it does not necessarily mean that they are in a sibling relationship.(e.g. AS30916, AS12713). On the other hand, there may be siblings out there which we do not detect, because their route announcements look like peers.

3.3.1 Customers

Compared to other approaches, lots of customer-provider relationships are not cataloged in the Registry. The number of customers not seen exceeds 10'000 which is over 30% of all customer-provider relationships seen by the other algorithms. The number of missing edges is so high because important ASes are missing a policy entry in the registry, namly AS1 (Genuity), AS701 (UUNET) and AS1239 (SPRINT), AS7018 (AT& T), AS209 (Qwest). They have no entry in RADB nor RIPE, so these ASes would add hundreds of customer-provider relationships. On the other side, over 10'000 customer-provider relationships are found by the registry for which no other algorithm even sees a connection there. When only looking at results where the registry-entries of both ASes implicate the same relationship, the number of customer-provider relationships gets a factor 10 smaller. When querying sh ip bgp regex AS *A* AS *B* on some route servers, there is no connection seen. This seems to be legacy entries or backup providers which are only used in case of failure of the connection to the main provider, so they are normally not seen in the global routing table in most cases. AS1755, EBONE, was shutdown on 2nd of July 2002, but there are still 25 former customers who seem to have not updated their registry entry since then.

3.4 The Simplified Robban Tool

The Robban Tool identifies only 0.5% peerings. Only looking at the last two ASes and dropping paths seen less than three times reduces the total numbers of connection. When comparing with other approaches, we see that most of the missing connections are peerings. About 95% of the relationships inferred are identical with other approaches. This implementation has its weekness in finding peerings, which could possibly be improved by collecting data from several vantage points.

Chapter 4

Problems

In this Section, we mention reasons why problems arise for all algorithms. Finding an perfect abstraction from reality is impossible and therefore some misinterpretations occur. If the IRRs would be as accurate as wanted by Georgos Siganos at Nanog [7], it would be possible to construct a graph far completer and more precise than any algorithm could do. We join his demand to practitioners and the related authorities to maintain and use more the IRRS.

4.1 Cycles

When AS B is classified as a customer of AS A and AS C is classified as customer of AS B and AS A is a customer of AS C again, then we found a cycle. It normally makes no sense to be the customer of an AS that is a customer of your customer. There are exceptions where it makes sense from the business point of view. If a big provider wants to enter a new market, it becomes customer of a smaller local AS to get connectivity to the new market. They do not want to set up a peering, because many other local ASes of similar size would then also like to have a peering with the new important AS. But there is no interest to make all your paying customers to peers. This is the point where every algorithm comes to fall, because the assumption of a customer-provider relationship is somehow based on the magnitude of an AS, or its position in the global graph. That a very important AS is customer of a smaller and lower AS in the global hierarchy is not accounted for in these algorithms and thus produces a customer-provider relationship.

4.2 Tier one providers

Wenn man nun die vier Approaches [1], [2], 2.3 und 2.4 betrachtet, stellt man fest, dass nur AS701 und AS11537 von allen einheitlich als Tier one identifiziert werden. Der Registry Ansatz identifiziert über 300 Tier ones, das ist weit mehr als realistisch. Dieser Fehler geschieht einerseits wegen Exchange Points, die ANY zu allen andern announcen, was sie als Provider klassifiziert. Beispielsweise AS30865, Tashkent Internet eXchange Point, AS23741 PIPE Networks

Hobart Internet Exchange, As24688 Kharkov Internet Exchange Point. Andererseits gibt es Fälle, bei denen der vermeintliche Tier one provider gar keinen Eintrag hat, aber von einem anderen AS mit Eintrag als Provider angegeben wird. (Beispiel: AS19440, AS23555). Die Resultate zu den vier Ansätzen finden man in den Datenbanktabellen tieroneregistry, tieronegao, tieronelakshmi, tieronerobban.

4.3 Exchange points

When looking closer at the ASes within a cycle, the following incidents seem to occur the most. Exchange points on level 3 are likely to produce conflicts. An exchange point peers with some AS and announces all the routes it gets from them to everyone. The ASes are seen as peers of the exchange point, although they are paying for the service of this neutral exchange point. Being connected to a layer 3 exchange point means that you do not have full control which of your routes will be announced to whom. Sometimes the exchange point operates servers as well, and needs global connectivity for this reason. So, the exchange point becomes a customer of many of its peers. This leads to a conflict, because the ASes connected to the exchange points should be seen as peers. When the exchange point is on level 2, it is not seen at all and therefore produces no conflicts at all. Today most Exchange Points do not provide any layer 3 service, except IP-address- allocation for the routers.

Examples of Exchange Points causing Cycles:
Table: gao20031229oregoninf AS21356 XchangePointMLP AS4635 Hong Kong Internet Exchange
Table: gao20031113oregoninf AS6695 DE-CIX, the German Internet Exchange

4.4 Common misinterpretations

When comparing the different approaches, we see that a few of the ASes produce most of the incongruities. AS3303 has many peerings, which are all listed in the macro AS-SWCMPEERS. Especially the algorithms [2] and the Robban Tool misinterpret about 50% of the peerings of AS3303 as customer-provider relationship. Swisscom peers directly with the route collector of the Route View Project, so these peerings can be seen, but all paths announced to the route collector have the form $AS3303$ $ASPEER$. This implies for the Robban Tool that AS3303 is the provider, when the peer does not announce its routes to the route collector. For the rank based algorithm [2], if $AS3303$ and $ASPEER$ are both not in the residual graph (see section 2.2), it is likely that $ASPEER$ will be interpreted as customer. The Robban Tool can be prevented from doing this misinterpretation by just looking at paths with at least three ASes. But this also breaks if a customer of AS3303 peers with the route collector. Many times all algorithms infer a customer-provider relationship, but the policy read out of the registry infers a peering. It may be that the policy is defined like in Table 3.3 and just one AS has entered its policy, so no conflict can be detected. The number of inferable relationships decreases strongly when just looking at the results fetched out of the registries where the entry for AS A and the entry for AS B infer the same relationship between the two ASes, so only ASes where both have a registry entry with information about its import and export behaviour are taken into account. But the fraction of the relationships inferred differently by the algorithms stays constant. Mainly the same ASes produce conflicts, no matter if the relationship is decided from one or two registry entries. The effective number of conflicts per AS when not double checked is higher. But seeing that this AS leads to conflicts in the double checked case as well, we can infer that the algorithms misinterpreted the importance of this AS. The main part of conflicts seems to be caused by ASes with open peering policies. (e.g AS3303, AS4589).

4.5 Peerings

As we could see in the previous section, peerings cause most problems. When choosing a measure to express importance, one has the problem that not every AS uses the same benchmark to accept another AS as its peer. Here is a cut-out of Level 3 peering policy:

> Level 3 will continue to monitor the market and traffic conditions and accordingly change the peering policy to suit the market and customer needs.

Therefore, we can not rely on just one definition of importance to decide whether they peer or not. Many other considerations influence the peering decision process: Is there a business competition in the same market or is the peer at an exchange point abroad? Is it an IPV6 or IPV4-peering? Can we improve latency? Do we trust in the abilities of the network operators that they do only announce what they should? Do we need to support the network operator and lose time by doing so? Is the other AS expected to grow or to shrink? Do you have special agreements where the peer has to pay if the traffic asymmetry goes beyond a certain ratio?

Because all the answers to these questions are not available to any algorithm, it seems impossible to detect every peering. The majority of the connections seen when only having some vantage points are customer-providers relationships, because you have no information about local peerings. Table 3.2 clearly shows that the number of common peers is only 1.03% of the common links. This is far away from the 5% peerings of algorithm [2] and up to 7.8% peerings plus up to 1.5% siblings of algorithm [1].

An important aspect is in which tier the peerings are situated. The algorithm of [2] exclusively identifies peerings between ASes in higher tiers. Generally, it is more interesting to know the peerings of the higher levels, because more routes are affected by them.

To summarize, the common peerings of approaches 2.1.1, 2.2, 2.3 are peerings on higher levels.

4.6 Siblings

The concept of sibling has no origin in the world of Internet Service Providers. An ISP probably does not know that sibling stands for mutual transit agreement. To infer siblings is at least as difficult as inferring peerings. When two ASes only have a mutual backup agreement, how do we catch the moment when this backup functionality of the two ASes takes place? Maybe we needed to collect some snapshots at different dates and consider all of them to get a higher chance of inferring mutual backup agreements.

Chapter 5

Conclusion

Algorithms to infer relationships between AS seem to be quite good in identifying customer-provider relationships. Peerings are more difficult to find. Problems arise when encountering unusual or unexpected cases. Our model of the BGP-Internet graph is a forest with several roots and cross connections between nodes on a similar level to represent peerings. Misinterpretation is most likely to happen when rootnodes are connected to leaves, ASes have more peerings than customers, exchange points appear on level 3 or ASes have hybrid forms of relationships. Introducing more fine-grained types could help to get a more precise image. In addition to just knowing the provider or peer role, it would complete the image of relationships if we could determine for which subsets and for which size of prefixes an ISP acts as provider or peer. The registry is able to detect special cases, but the information has to be entered. The Registry actually provides views from many vantage points, namely for any entered AS: No algorithm based on routing tables can have views from all vantage points, due to lacking access to the routing tables of all ASes. Therefore, an up-to-date and correct registry could help to interpret atypical cases and add many of local peering connections to complete the graph of the Internet hierarchy.

Chapter 6

Tools

In this section the tools used will be explained.

6.1 MySQL

The types of relationships are saved into a MySQL Database. (Version used:3.23.58) To create a database and give access to a specific user, login as root.

```
shell> mysql --user=root mysql -p

mysql> create database nameofdb;

mysql> GRANT SELECT,INSERT,UPDATE,
   DELETE,CREATE,DROP
    ->     ON nameofdb.*
    ->     TO username@localhost
    ->     IDENTIFIED BY 'password';
```

To use the database from hosts others than where the MySQL Database is installed, execute the statement above by replacing localhost by the desired hostname. But before the changes are accepted, the command

```
mysqladmin -u root -p reload
```

has to be executed. To start mysql simply type:

```
mysql -u username -p nameofdb
```

If the database is installed on some other host, use option

```
-h hostname
```

Now, we provide a short introduction how to create the tables used. Further information about MySQL is available at www.mysql.com.

```
CREATE TABLE tablename (Column1 type, Column2 type, Column type);
e.g. CREATE TABLE t_gao20031229oregoninf
(AS1 MEDIUMINT, AS2 MEDIUMINT, TYPE SMALLINT);
```

The tables are generally named as a composition out of the name of the algorithm (gao, lakshmi,robban), date (YYYYMMDD), input data used (oregon,rrc) and the values of the peer ratio R (60, inf(=infinity)) used to execute the algorithm.

The coding of the relationships is to understand like this:

```
Type 0 : AS1 and AS2 are siblings.
Type 1 : AS1 is customer of AS2.
Type 2 : AS2 is customer of AS1.
Type 3 : AS1 and AS2 are peers.
```

The databases are available as dump files. To produce a dump file type:

```
mysqldump -u user -p db > file.sql
```

The dump file is created in your current directory. With option -c you choose between complete or incomplete insert statements.

```
example complete: insert into t_user (id, name) VALUES (34,Meier);
example incomplete: insert into t_user VALUES (34, Meier);
```

This text file contains all CREATE and INSERT statements of the database. You can read this back into MySQL with:

```
mysql database < dumpfile.sql
```

6.2 Quagga

Quagga is a routing software, a fork of zebra. We used Quagga to set up a peering with 3 routers. This means we got all routes announced by these 3 routers. We so could dump the routing tables of AS559 to use as input for the algorithm [2].Quagga can be downloaded under www.quagga.net. To set up the peering, the configuration files have to be edited. See below how to create the bgbd.conf file:

```
!
hostname macbea
password zzyyxx
enable password xxyyzz
!
!bgp mulitple-instance
!
! network 10.0.0.0/8
!
router bgp 696
neighbor 130.59.32.30 remote-as 559
neighbor 130.59.32.30 description EBGP zu IX1
neighbor 130.59.32.30 ebgp-multihop 255
!
```

```
!log file /usr/local/etc/bgpd.log
dump bgp routes-mrt /usr/local/etc/dumpfile
!
log stdout
!
```

The daemons, also the bgp daemon, provide a vty interface for Cisco like configuration. This can be reached by the command

```
telnet localhost bgpd
```

or

```
telnet localhost 2605
```

Now some important commands, which also can be used when being connected to a Cisco Routeserver. To see all peers to which you are connected type:

```
show ip bgp summary
```

The output produced shows the state of all sessions to the 3 routers.

```
BGP router identifier 130.59.4.109, local AS number 696
5547 BGP AS-PATH entries
680 BGP community entries

Neighbor        V    AS MsgRcvd MsgSent TblVer  InQ OutQ Up/Down  State/PfxRcd
130.59.32.30    4   559   20597      44      0    0    0 00:04:04    24201
130.59.32.42    4   559   21037      44      0    0    0 00:04:03    24200
130.59.32.60    4   559   20792      43      0    0    0 00:04:00    24201
```

The routes can be seen by entering

```
show ip bgp
```

For every prefix, we will have listed the routes announced from every router.

```
BGP table version is 0, local router ID is 130.59.4.109
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop         Weight Path
 *  12.0.0.0         130.59.32.30          0 559 3549 7018 i
 *                   130.59.32.60          0 559 3549 7018 i
 *>                  130.59.32.42          0 559 3549 7018 i
 *  12.0.48.0/20     130.59.32.30          0 559 20965 11537 10578 1742 i
 *                   130.59.32.60          0 559 20965 11537 10578 1742 i
 *>                  130.59.32.42          0 559 20965 11537 10578 1742 i
 *  12.6.208.0/20    130.59.32.30          0 559 20965 11537 10578 1742 i
 *                   130.59.32.60          0 559 20965 11537 10578 1742 i
```

```
*>                  130.59.32.42          0 559 20965 11537 10578 1742 i
*  12.144.59.0/24   130.59.32.30          0 559 20965 11537 10466 13778 i
*                   130.59.32.60          0 559 20965 11537 10466 13778 i
*>                  130.59.32.42          0 559 20965 11537 10466 13778 i
*  12.161.8.0/21    130.59.32.30          0 559 20965 11537 10466 88 i
*                   130.59.32.60          0 559 20965 11537 10466 88 i
*>                  130.59.32.42          0 559 20965 11537 10466 88 i
```

6.3 Internet Routing Registries

Die verwendeten IRRs sind RIPE und RADB. RADB spiegelt die meisten anderen IRRs, daher umfasst diese Datenbank eigentlich alle eingetragen Policy Einträge der ASe. Auf der Website http://www.irr.net/ findet man eine Zusammenstellung aller existierenden IRRs und Literatur zur Routing Policy Specification Language.

Beispiele für Anfragen an eine solche Datenbank:
Um die Policyeinträge von der ASnummer zu sehen:

```
whois -h host ASnummer (e.g. whois -h whois.radb.net AS559)
```

Um zu sehen, in wessen Bereich eine IP-Adresse fällt:

```
whois -h host IPadresse (e.g. whois -h whois.radb.net 130.59.62.1)
```

Um die ASnummer zu erhalten in dessen Bereich sich die IPadresse befindet:

```
whois -h host -T route IPadresse
(e.g. whois -h whois.ripe.net -T route 130.59.62.1)
```

Um zu sehen, an wen ein AS seinen IPrange verteilt hat:

```
whois -h host -i origin ASnummer
(e.g. whois -h whois.ripe.net -i origin AS3320)
```

Chapter 7

Code

In this section, the functionality of the scripts will be described. As well an example illustrates how and in which sequence to start the scripts.

7.1 Valley-free path

In this section, we will explain how to run the algorithm described in [1] and how to incorporate the different metrics.

7.1.1 Degree based

The code to run the degree based approach [1] is to find on the website http:// www-net.cs.umass.edu/ ratton/AS/, downloadable as a .zip file named LogRelInfer-1.0.tar.gz. To avoid that the downloaded file coredumps, you have to augment the size of the matrix in the file common.h which represents the number of ASes appearing in the routing table. It originally was set to 12000. In the meantime, there are over 16'000 ASes in the global routing involved. In the file compute_relation_matrix.h, you have to change the integer transient to a short. Otherwise the systems can run out of memory (at least in my case).

How to run the algorithm: The file MyGao.pl was adapted to extract the ASpath out of the MRT files. MRT is a binary format, produced by Multi-Threaded Routing Toolkit [13] or Zebra. To start the script type:

```
perl Gao.pl < MRT-file > ASmap_date
```

The target file has to be named ASmap_date, e.g. ASmap_20031229. (When using non compressed routing tables, you can use the convert_RouteView.pl to extract the AS Paths, but we never used it.) The so produced file has to be saved in a directory called data. If you wish to have other paths, you could change the paths defined in the files.

First run the program:

```
./dump_matrix date 1 (e.g .(/dumpmatrix 20031128 1)
```

Second step:

```
./compute_relation_matrix date 1
```

To save the results in a database, call the scripts with the following arguments:

```
 perl resultsgaotodb.pl
```

with the arguments AS_index_mapping.date, AS_relationship_dump.date, tablename. The new table will be created by the script.

If you ever encounter segmentation faults, augment the size of the matrix and the maximum AS number and hope you have enough RAM. Otherwise you have to find further optimizations.

The script `collectripedata_dumpgao2sql.pl` was written to automate the downloading and executing process. The paths have to be adapted manually. The script `startinf.pl` is thought to run the algorithm again, R set to infinity.

Ratio R

To run the algorithm with different ratios, you can change the value of R in the file compute_relation_matrix.c by editing the line

```
int PEER_RATIO = 60;
```

To set R to infinity just comment out the line where the comparison between the deg_vector and PEER_RATIO is done. Don't forget to execute make before starting the next run.

7.1.2 Address space based

In the directory addressrange, you'll find all the scripts needed to count the address pace of all ASes. First you have to save all prefixes announced in a routing table (fullrouting) and save each prefix with its origin AS. This is automatically done by creating a new table in the mysqltable with the columns, `ASNUM (MEDIUMINT), PREFIX VARCHAR(64)`. You have to download a routing table saved as MRT-File. It can be downloaded either from http://data.ris.ripe.net/ or from http://archive.routeviews.org/ Don't forget to gunzip or bunzip2 the downloaded file. To script has to be called like this:

```
perl getprefixtable.pl < MRTfile tablename
(e.g.: perl-5.8.0.806 getprefixtable.pl <bview.20011201.0044 demo)
```

When starting the script twice with the same tablename, everything will be saved to the same table. So be careful when rerunning the script. There will only be a warning displayed that the table already exists.

The script elimination.pl eliminates prefixes announced by the same AS, which have intersections, for example 17.0.0.0/9 is completely contained within the prefix 17.0.0.0/8. Then 17.0.0.0/9 will be removed. This has to be done to count the announced address pace properly.

To count the address pace without including customers, just type

```
perl countspace.pl <tablewithprefix> <tabletosavecountedspace>
(e.g perl-5.8.0.806 countspace.pl demo democount)
```

The table where the counted address pace will be save is created by starting. The result of this calculation is needed to calculate the address pace recursively. (The /8 are weighted as a 4 times a /16, the /9 are weighted as 3.5 times a /16 to give less influences to the unused /8 networks.)

Then we call the recursive calculation:

```
perl findcustomer.pl <relationshipstbl> <customertbl> <counttbl>
```

As first argument, you type in the table already containing relationship to calculate the refinement. The second table will be created to save the results. In the third table you give the measurement of importance for the refinement, this can either be the address pace or the number of important websites.

Now, you have created a table where for each AS stands its range, calculated recursively, and zero for the number of customer. By querying "select * from customercount where KUNDEN!=0", you can identify the cases where loops are involved. Now it has to be found out manually, where to break the cycle. Having broken one cycle, just start the script correction.pl customercount to reconstruct the hash with the values again and runs until the next cycle. When there are no more rows where KUNDEN!=0 , you are finished. Nun hat man eine Tabelle mit der ASNUM und der rekursiv berechneten Grösse, was als Input für die neue Berechnung der Beziehungen benutzt wird. Siehe dazu Section 7.1.6.

7.1.3 Important website based

This section describes the scripts used to collect IP addresses of important websites. It is not allowed to query google with wget.

```
wget "http://www.google.com/search?ie=ISO-8859-1&q=hallo+welt"
--15:22:15--  http://www.google.com/search?ie=ISO-8859-1&q=hallo+welt
           => ‘search?ie=ISO-8859-1&q=hallo+welt’
Connecting to www.google.com[66.102.11.99]:80... connected.
HTTP request sent, awaiting response... 403 Forbidden
15:22:16 ERROR 403: Forbidden.
```

Google provides an API (http://www.google.com/apis) to automate queries, but the obtained license key entitles you only to 1'000 queries per day.

The script needed uses the LWP::UserAgent, a class implementing a simple World-Wide Web user agent, to query google.

```
perl randomgoogle.pl <filewithrandomwords> <tablename>
(e.g. perl-5.8.0.806 randomgoogle.pl words distribution)
```

The file with random words of different languages is available at
ftp://ftp.mathematik.uni-marburg.de/pub/mirror/openwall/wordlists/.

7.1.4 Based on IP addresses of endusers

First, we extracted the IP addresses out of the log files (gnutella user logs) by using cut, e.g.

```
cut -f2 neighbors2.log | cut -d ":" -f1 | sort -u >iponly.txt
```

To insert the Gnutella data into a table, call

```
perl gnutella.pl <filewithipaddresses> <tablename>
e.g perl-5.8.0.806 gnutella.pl neighbors2.txt kaaza
```

7.1.5 Assign IP address to AS

To assign an IP address to its AS, we have to find out to which prefix an IP address matches. I tried several modules and elected the module NetAddr-IP-3.19, which manages IP addresses and subnets.
(available at http://search.cpan.org/ luismunoz/NetAddr-IP-3.19/). Because of the missing rights to install modules, we had to extend the include path by adding the statement:

```
BEGIN {
unshift @INC,
qw(/local/statistic/google/Net-Netmask-1.9007/blib/lib);
}
```

To calculate the number of IP addresses per AS faster, we reduced the number of accesses by introducing a hash.
`perl hash.pl <tblwithiprange> <tblwithipaddress> <tbltocount>`

To check which IP addresses are assigned to one AS, not just the number of IP addresses per AS, start the script checkip.pl.

```
perl checkip.pl <ASNUMBER> <tblgnutellaorgoogledata> <tblprefixes>
(e.g. perl-5.8.0.806 checkip.pl 559 kaaza t_range20040203)
```

7.1.6 Rerun the algorithm

Now we want to use the so calculated ranking to rerun the algorithm of Gao. First, run the program `dump_matrix` , then run the script

```
perl definespace.pl <AS_indexmapping.date> <tablewithnewmeasurement>
> AS_degree.date
```

Then continue like normal, run the program compute_relation_matrix again and then fill the results into a table by calling resultgaotodb.pl.

7.2 Rank based

We got an implementation of the algorithm [2]. The file is named infocom02-code.tgz The text below is the documentation he wrote as an email (lakme@eecs.berkeley.edu).

> Here's an old piece of code. Its not well documented. At a very high level, it contains 3 parts: 1. Processing BGP dumps (script: process_dump and some executable)
>
> 2. Finding the type of relationship between ASes
>
> 3. Finding the hierarchy
>
> Before you use it, do the following: 1. Store all the BGP dumps in gzipped format in one directory. Label every BGP dump using its number. For example if you have a dump from AS123, call it 123.gz
>
> 2. Create an `interface` file where you have all the AS numbers and the separators. For example if the AS path in AS123 starts from position 60 in every line, attach the line `123 60` in the script. In some ISPs (like BBNPlanet- AS1), I find the AS path to start from 59. Else I normally find it in position 61. An example ISPs with a list of 8 ISPs and their separators is already provided. Normally for the old cisco dump, the AS path starts at 59, and for the new cisco dump it starts at 61. An example file "interface" is in the tar file.
>
> 3. Create an executable `control` by doing "cc -o control control.c" Finally, execute the following command:
>
> ```
> ./control <interface file> <location of gz dumps>
> <location of tmp dir>
> ```
>
> The results will be present in the tmp directory. Sorry, the code is not yet in releasable stage yet. It needs more work. It started as a bunch of scripts and since we were always inspecting the intermediary results, we maintained it as separate scripts. We later combined these scripts using shell scripts.

First you have to collect data, you can take the data from the website of this project, but unfortunatelly it is not updated every day. So we can collect data on our own, using the script router.pl. The important thing is that *expect* has to be installed. Then just call

```
perl routers.pl <year.month>
```

(e.g perl routers.pl 2003.12) The download starts in parallel for all entered routers. You can check if there are new servers available at the websites:

http://www.netconfigs.com/general/cisco-telnets.htm

http://www.traceroute.org/ under Routeserver

This program does not work properly. We probably got a old version of the code. Not even all connections are seen, although they appear in the routing tables. The number of identified peerings is too small. We tried to find out if we could reach better results by changing the values of δ_1 and δ_2. The script diffparameter.pl changes the parameter RATIOTHRESHOLD, initially defined as 3.0, or PEERTHRESHOLD initially defined as 2.0, and recompiles the code. The achieved peer ratio will be calculated by the script countpeer.pl and writes the results to a file called resultat.txt. To start just type:

```
perl diffparameter.pl <RATIOTHRESHOLD|PEERTHRESHOLD>
```

But the percentage of peerings produced by changing the parameters still were too low to be realistic.

7.3 Registry

To run the scripts you have to install peval, a low level policy evaluation. (Downloadable at http://www.ripe.net/ripencc/pub-services/db/irrtoolset/) It can expand AS macros:

```
./peval -no-as AS-SWITCH
```

results in AS8235 AS12654 AS559. When querying macros, there can also indirect customers be contained in the set of recursively resolved AS macros. For example, AS-VERIO contains AS-VERIO-EUROPE which consists of AS-XARANET and some others. AS-XARANET contains AS-HIGHWAYONE where AS28782 is a customer. peval resolves a set of ASes where AS2914 and AS28782 are contained. But AS28782 is an indirect customer of AS2914, which means they cannot stand side by side in an AS-path, because they have no direct connection.

Therefore, first of all, you have to create a connectivity graph out of all ASes. We want to know all possible links between ASes to be able to decide whether someone is direct or indirect customer. Create a table t_connectivity :

```
CREATE TABLE t_connectivity (AS1 MEDIUMINT,AS2 MEDIUMINT,
UNIQUE(AS1,AS2));
```

Then run the script

```
perl connectivitygraph.pl < MRT-file
```

The check that no AS pair is entered twice in the database is done by the database. The registry querying can just be started by

```
perl registry.pl
```

The versions are named like registry1.pl, registry2.pl, etc. The lower the number the older the version. The actual and newest version is just called registry.pl. The newest version is best documented. In the old versions, the improvements are described upmost in the file.

The output of the scripts checkconsistency.pl and deleteconflict.pl can be redirected to a file. The files detect conflicts and entries that appear twice. They also delete double entries. The script seeconflict.pl reads the outfile generated by the files above to query the IRR again, and display the entries. So you can decide yourself where the mistakes lay. But note that these files are only relevant for the version 1 and 2 of the registry, the newer versions detect and save conflict pairs without additional scripts.

Do not run two instances of the script in the same folder to avoid that they read from the same data files. We query the following IRRs: RIPE and

RADB. RADB mirrors the data of more than 30 other IRR databases therefore we should get all available information. Currently, the RADB receives about 150,000 queries per day from over 7,000 unique hosts. For this reason the technical support of RADB was worried about our query. Here their email:

> I am writing you in hopes that you are the adminstrators or can get me in contact with the adminstrators of the following host.
>
> Name: wilab4.inf.ethz.ch Address: 129.132.130.14
>
> This host is doing whois queries against our server whois.radb.net. Our server is having some serious difficulty handling one query in particular. The host is doing a route-set expansion against AS702:RS-CUSTOMER. The expansion is eating up massive amounts of CPU and causing our other users difficulty querying the RADB. I wish to ask the adminstrator to point their whois queries to our backup server radb3.merit.edu 43. This server should have a better time handling the queries.
>
> I sincerely hope you can help. If contact fails we may be forced to block access to whois.radb.net from wilab4.inf.ethz.ch.
>
> Regards, Chris Frazier Merit Network - RADb 734-647-8927

For this reason, we used the backup server radb3.merit.edu.

7.3.1 Additional stuff

In this section, we list interesting cases and additional problems not mentioned in the paper, because they do not happen very often. Interpreting an AS macro called AS-TRANSIT as a macro containing transit providers is dangerous. In the case of AS17922, they list their transit customers in there.

```
whois -h whois.radb.net AS-PALAPANET-TRANSIT
as-set:     AS-PALAPANET-TRANSIT
descr:      ASN that Approved transit via Palapanet Networks.
members:    AS17826, AS9462, AS17671, AS18112, AS17884, AS18347
members:    AS-SATNET-TRANSIT, AS23695, AS10137
```

Prepending: AS14 want to steer its incoming traffic by prepending itself several times, to avoid that traffic is routed over AS701 and AS1785, because they probably have to pay for these connections.

```
export:        to AS1785 action aspath.prepend(AS14,AS14,AS14);
                      announce AS14:RS-ALL
export:        to AS701 action aspath.prepend(AS14,AS14,AS14,AS14);
                      announce AS14:RS-UNIVERSITY
```

AS14876 ist ebenfalls nicht mehr in den Routing Tabellen zu sehen, genausowie AS6832.

Solange es für IPv6 noch kein eigenes Registry gibt, oder man das nicht anders hervorheben kann als durch Kommentare, kann das ebenfalls zu Missinterpretationen führen. So kann man beispielsweise Kunden sein für IPv4 und Peer für IPv6.

Problematisch sind auch export statements wie das von AS9548:
`export:  to AS3356 announce  202.128.80.0/20  ,`
da man so nicht weiss ob der erwähnte Prefix ANY ist, oder nur ein Teil seiner Routen.

Zudem gibt es einige Einträge, die private ASnummern beinhalten, z.Bsp. AS3356:

```
import:       from AS65000 accept NOT ANY
import:       from AS65432 accept NOT ANY
import:       from AS65517 accept ANY
import:       from AS65533 accept NOT ANY
```

Alle Einträge mit privaten ASnummern müssen aus der Tabelle gelöscht werden.

Example for how free you are to make entries:

```
remarks: .                                 ,,
remarks: .                                (  )   Homer, kids, just keep
remarks: .                               (    )  smiling till these people
remarks: . yaaaaah! oh-my-GOD!          (      ) staring at us go away.
remarks: . we’re trapped inside some    (      )  /
remarks: . computer-dimension-          (      ) /
remarks: .  something-or-another!!      (      ) wow--the     ’Bort’? that’s
remarks: .     \                        (      ) information   BART Simpson,
remarks: .      \        suck-suck-     (      ) super-highway!     Man
remarks: .       ,__     suck-suck!     (      )     \               /
remarks: .      /   ‘‘\      \          (      )      \             }\/\/{
remarks: .      |     |       \         (      )      _/\/\/\_      |    |
remarks: .      (.)(.)|     _/\/\/\_    (,,  ,,)      \,,  ,,/_     (.)(.)
remarks: .       C___ |)    \,>o<,,/_    (.)(.))       (.)(.) /_    | C  |)
remarks: .      /___ \|     \(.)(.) /_    C__, @        C__, @_/    (___ |
remarks: .      __)__/|__    /c    @_/     )  |          )  |/        )__|
remarks: .     /   \/ /  \    C_, )/       oooo          oooo         /  \\
remarks: .    /      /    \    /  |/      /___\\        /   \\       /     \\
remarks: .        Homer       Maggie      Marge          Lisa         Bort
remarks: Cerbernet’s Registry
notify:  noc@bsve.net
notify:  noc@bsve.net
mnt-by:  BSVE-MNT
changed: robincragg@businessserve.co.uk 20020523
source:  RIPE
```

7.4 The Simplified Robban Tool

Die Scripts zum Robban Tool befinden sich im directory robban, das ist weiter unterteilt nach robbantool, netlantis und netlantisdata. Die script im directory netlantis waren dazu beabsichtigt, alle Pfade, wo das gesuchte AS als letztes oder zweitletztes AS im Pfad steht, vom netlanis routecollector zu sammeln und auszuwerten. Weil das ganze aber ziemlich langsam lief, und die Datenbank von Netlantis auch ziemlich belastete, haben wir den Versuch abgebrochen. Im directory netlantisdata sind die Files bis zum Zeitpunkt des Abbruchs gespeichert.

Im Verzeichnis robbantool sind die Files zu finden, welche für den im Paper beschriebenen Ansatz verwendet wurden. Zuerst wird gezählt wie oft zwei ASe am Ende eines Path vorkommen:

```
perl Robban.pl <tablename> <MRT-file>
e.g. perl-5.8.0.806 Robban.pl t_robban < rib.20030630.1655
```

Dannch kann man die Beziehungen in einer Tabelle abspeichern:

```
perl decidenoratio.pl <tablename> <tableresults>
(e.g. perl-5.8.0.806 decidenoratio.pl t_robban t_robbanrelations)
```

Das script deciderobban.pl kann man verwenden, sofern man einen Ratio brauchen möchte, der definiert wie gross das Verhältnis zwischen der Anzahl Pfade, die auf AS1 AS2 enden und derer die auf AS2 AS1 enden, sein darf, um noch als Peer klassifiziert zu werden.

7.5 Another data source

Im Directory RWdata befindet sich das script parse.pl, dessen Aufgabe es war die Daten im File rogerresults.txt, alte Resultate von Roger aus seinen Zeiten bei Microsoft, in eine Tabelle einzulesen.

7.6 Check exporting rules

In Section 1.2, we described the exporting rules. The algorithm [1] tries to infer the AS-relationships via the exporting rules, but the other approaches use different ways to do so. A way to check the results or to find siblings interpreted as peerings is to check where and how many times the exporting rules are hurt. The script checkexportrule.pl counts and lists the cases where exporting rules got hurt. Give a file only containing ASpath, (see Section gao how to generate such files using the script Gao.pl) as first argument and a table of the form (AS1,AS2,type) as second argument.

```
perl checkexportrule.pl <filewithASpath> <tblnamewithrelationships>
```

I did not analyse the results in details to make any relevant conclusion. This would be a field where further investigations could be made.

7.7 In which tiers are the peerings located

Um herauszufinden in welchen Tiers sich die Peerings befinden, braucht man eine Einteilung in verschiedene Tiers. Diese Information wird von Algorithmus [2] generiert oder man kann es von der Website [3] herunterladen. Um die Information dann in die Datenbank zu speichern, verwende

```
perl hierarchy.pl <hierarchyfile> <hierarchytablename>
```

Dannach

```
perl wherearepeering.pl <tableto analyse> <hierarchytable>
```

aufrufen und die Anzahl Peerings zwischen den einzelnen Schichten (und Anteil in Prozent) werden ausgeprintet.

Zusätzlich gibt es noch ein Script, das die Tier ones findet für jeden Ansatz.

```
perl tierone.pl <tablewithapproach> <tabletosavetierones>
```

7.8 Relationship comparison

To compare the results of different approaches or different time, use the scripts in the comparsion folder. The script has to be called

```
perl compare3.pl <table1> <table2> <table3> <comparetable>
```

This script reads all relationships out of the first table and compares them to the other tables. In a second step the relationships of the second tables are compared to all others. If you only like to have the first table compared to the others use the version comparesimple. The results are saved to the comparetable. Using the script the first time, make sure that you have a table called t_compare_parameter_tables otherwise create one.

```
create table t_compare_parameter_tables(id MEDIUMINT NOT NULL
AUTO_INCREMENT, ref_table varchar(64), cmp_table_1 varchar(64),
cmp_table_2 varchar(64), cmp_table_3 varchar(64),PRIMARY KEY(id));
```

It also needs at least one existing table where you want to save the results of the comparison. This table is given as last argument to the script. To create such a comparetable call

```
create table t_compare (AS1 int, AS2 int, ref_val tinyint,
cmp_val_1 tinyint, cmp_val_2 tinyint, cmp_val_3 tinyint,
foreignkey mediumint);
```

Siblings are treated like peers when comparing. 0 means no entry, in the later version 0 was replaced by 6 to avoid misunderstanding. But in the comparetable 0 never means siblings.

Um die Resultate in der comparetable zu sehen, kann man verschiedene Queries verwenden, um die Daten zu analysieren, je nachdem welche Fragestellung man hat. Hier einige Queries, um einen Überblick zu erhalten. In der

Tabelle t_compare_parameter_tables kann man die Id des Vergleichs herauslesen für den man sich interessiert (= foreignkey). Listet die auftretenden Fälle nach Häufigkeit:

```
select count(*), ref_val, cmp_val_1 , cmp_val_2, cmp_val_3
from t_compare where foreignkey=1 group by ref_val, cmp_val_1,
cmp_val_2, cmp_val_3 order by 1;
```

Listet die ASe, welche von einer unterschiedlichen Interpretation betroffen sind, nach Häufigkeit.

```
select count(*),AS1,AS2,ref_val, cmp_val_1 , cmp_val_2, cmp_val_3
from t_compare where foreignkey=1 group by AS1 order by 1;
```

Suche nach spezifischen Fällen, (z.Bsp. welche Beziehungen werden 3 mal als Provider-Customer Beziehung gesehen, und einmal als Peering.)

```
select AS1,AS2, ref_val, cmp_val_1 , cmp_val_2, cmp_val_3
from t_compare where foreignkey=1 and ref_val=cmp_val_2
and cmp_val_3=2 and ref_val=2 and cmp_val_1=3 order by 1;
```

7.9 Java AS graph simulation

At the beginning of the work, we started to create graphs in Java. But first of all, we had to bring the MRT files to an appropriate form. This can be done by calling

```
perl preparejava.pl < MRT-file
```

In the file preparejava.pl are some arrays listed with the IP addresses of the routers peering with one the routecollector of RIPE. You can add an array with the ip address of the routers connected to the routecollector where you obtained the MRT file (See the description on the top of the preparejava.pl file). The script produces files called ipaddresstxt ,like 202_12_28_190txt. These files contain all links between ASes seen from the router having the IP address contained in the file name.

In a next step, you have to read these files to create a graph. When running Java programs set the VM argument -Xmx1000m to have enough memory when creating huge graphs. We used an opensource library called JUNG, the Java Universal Network/Graph Framework, to create graphs. (For information and downloads see http://jung.sourceforge.net/)

7.10 Swiss AS

Allocated address space is address space that is distributed to members for the purpose of subsequent distribution by them. Assigned address space is address space that is distributed to a single end-user for the purpose of actual deployment in that end-user's own network. Address space is also designated as assigned if the organisation holding it uses it for the purposes of addressing their own network or applies it to a pool from which assignments are made dynamically as connections are established. Assignments are made for specific, documented purposes and should not be sub-allocated or sub-assigned.

Eine Liste aller Schweizer ASe ist zum download auf der RIPE Page verfügbar. ftp://ftp.ripe.net/pub/stats/ripencc/delegated-ripencc-latest (Further information: ftp://ftp.ripe.net/pub/stats/ripencc/RIR-Statistics-Exchange-Format.txt) Man beachte das bis lang AS3303 (Swisscom) und AS513 (Cern) interessanterweise nicht in dieser Liste eingetragen sind. Um nun den Adressraum eines jeden Schweizer AS zu erhalten, rufe man die Programme wie folgt auf:

```
perl fetchfiles.pl <ripefile>
perl saveprefix.pl <tablename>
```

Dannach sind die AS nummern mit ihren zugehörigen Prefixen in der Tabelle <tablename> gespeichert. Um überschneidende Adressbereiche zu entfernen benütze man das Script elimination.pl, dass man schon zum Berechnen des addressrange verwendet wurde.

```
perl elimination.pl <tablename>
```

Um die effektive Addressraumgrösse dann zu speichern:

```
perl countspace.pl <tablename> <tablename with addresspace>
```

Das Script conflicttoroutingtable.pl zeichnet Widersprüche auf, die zwischen den zugeteilten Prefixen von RIPE und dem tatsächlichen Routing vorhanden sind.

```
perl conflicttoroutingtable.pl <ripefile> <routingtable>
e.g.perl-5.8.0.806 conflicttoroutingtable.pl ripe t_range20040203
```

Wie man nun diese Information werten will, um seine Kriterien festzulegen, ob ein AS als Peering Partner in Frage kommt, sei dem Peering Koordinator überlassen. Ein weiterer Anhaltspunkt kann sein, ob das AS schon IPv6-fähig ist. Dazu kann man die Tabelle IPV6ASes verwenden. Um diese Tabelle zu erweitern, kann man auch erneut das Script

```
perl ASes.pl <MRT-FILE
```

aufrufen. MRT-File mit nur IPv6 Daten erhält man von Routeview. Bei RIPE ist leider nicht so strikt nach IPv4 and IPv6 getrennt.

7.11 Webinterface

Wir haben die Resultate in einem Webinterface zusammengestellt. Um möglichst wenig rechnen zu müssen während eines Aufrufes des PHP Script, werden die Daten für das Webinterface speziell aufbereitet durch den Aufruf:

```
perl buildwebinterface.pl registrytbl gaotbl lakshmitbl robbantbl
```

mit den vier Tabellen in dieser Reihenfolge, damit es mit der Beschriftung im Webinterface übereinstimmt.

Um die ursprüngliche Wertung des Algorithmus in eine Tabelle abzuspeichern, kann man das script savedegree.pl benutzen.

```
perl savedegree.pl <AS_index.date> <AS_degree.date> <tablename>
```

Um ein Ranking zu produzieren, erstelle man folgende Tabelle.

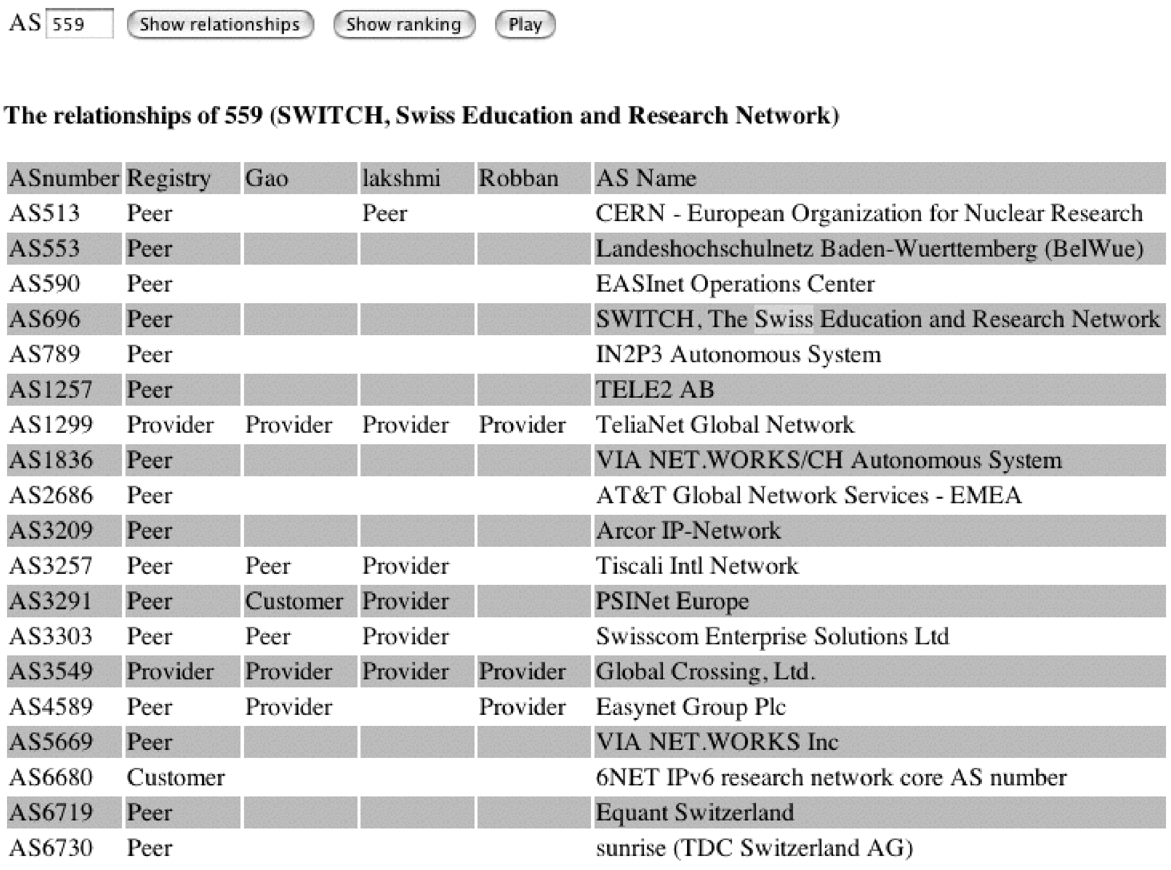
```
create table tablerank(rank INT NOT NULL AUTO_INCREMENT,
NUMBER INT, PRIMARY KEY(rank));
```

und füllt sie in eine Tabelle.

```
insert into tablerank (NUMBER) select COUNTER from table
group by COUNTER desc;
```

Wichtig ist dabei, dass die neue Tabelle genauso heisst wie die Tabelle von der man das Ranking erstellen will, mit dem postfix rank. Beispiel: zu t_count_google gehört t_count_googlerank Um nicht nur die AS Nummern anzuzeigen, sondern auch deren Namen anzeigen zu können, wurden diese in die Tabelle ASnames gespeichert. Dies wird durch den Aufruf `perl fillASnameintable.pl` getan.

Für das Webinterface selbst wurde PHP verwendet. Siehe dazu Codedokumentation im PHP Script.



The relationships of 559 (SWITCH, Swiss Education and Research Network)

ASnumber	Registry	Gao	lakshmi	Robban	AS Name
AS513	Peer		Peer		CERN - European Organization for Nuclear Research
AS553	Peer				Landeshochschulnetz Baden-Wuerttemberg (BelWue)
AS590	Peer				EASInet Operations Center
AS696	Peer				SWITCH, The Swiss Education and Research Network
AS789	Peer				IN2P3 Autonomous System
AS1257	Peer				TELE2 AB
AS1299	Provider	Provider	Provider	Provider	TeliaNet Global Network
AS1836	Peer				VIA NET.WORKS/CH Autonomous System
AS2686	Peer				AT&T Global Network Services - EMEA
AS3209	Peer				Arcor IP-Network
AS3257	Peer	Peer	Provider		Tiscali Intl Network
AS3291	Peer	Customer	Provider		PSINet Europe
AS3303	Peer	Peer	Provider		Swisscom Enterprise Solutions Ltd
AS3549	Provider	Provider	Provider	Provider	Global Crossing, Ltd.
AS4589	Peer	Provider		Provider	Easynet Group Plc
AS5669	Peer				VIA NET.WORKS Inc
AS6680	Customer				6NET IPv6 research network core AS number
AS6719	Peer				Equant Switzerland
AS6730	Peer				sunrise (TDC Switzerland AG)

Figure 7.1: Screenshot of the Webinterface

Chapter 8

Erfahrungsbericht

Durch die Masterarbeit habe ich das Thema Routing und Internet Topology richtig lieb gewonnen. Es war für mich ein total neues Thema und ich habe es sehr interessant gefunden sich mal näher mit der Struktur des Internets zu befassen, so auch mal eine andere Perspektive zu haben. Ich konnte sehr viel lernen, auch Dinge die nicht direkt in mein Paper einfliessen konnten. Dies auch dank der Tatsache, dass ich unvermeidlich so nebenbei noch etwas mitbekam, was meine Bürokollegen beschaftigte. Somit bestand nie die Gefahr, dass ich die Sicht der Praktiker bei meiner Arbeit vergessen würde. Manchmal war es frustrierend, dass andere Leute auf diesem Gebiet schon so viel geforscht und publiziert hatten. Vorallem wenn es erst im nachhinein aufgetaucht ist. Vermisst habe ich zeitweise auch das Arbeiten im Team. Speziell in Phasen,wo man nicht mehr weiterzukommen schien, z.Bsp. beim Umformulieren des Papers, wäre es wertvoll gewesen. Das ich keinen fixen Plan hatte, empfand ich als Vor- und Nachteil zugleich. Einerseits war es toll, flexibel zu reagieren, und anderseits hätte es mir manchmal mehr Sicherheit vermittelt zu wissen wie und wo man steht. Abschliessend kann ich sagen, dass es eine sehr interessante und lehrreiche Zeit war.

Bibliography

[1] Lixin Gao. On Inferring Autonomous System Relationships in the Internet. In *Proc. IEEE INFOCOM, 1997*

[2] Lakshminaraynan Subramanian, Sharad Agrawal, Jennifer Rexford, Randy H.Katz. Characterizing the Inernet Hierarchy from Multiple Vantage Points. August 2001.

[3] http://www.cs.berkeley.edu/~sagarwal /research/BGP-hierarchy/

[4] Iljitsch van Beijnum, BGP. Building Reliable Networks with the Border Gateway Protocol.

[5] Sam Halabi, Internet Routing Architecture. Second Edition.

[6] Craig Labovitz, Abha Ahuja, Srinivasan Venkatachary, and Roger Wattenhofer The Impact of Internet Policy and Topology on Delayed Routing Convergence (INFOCOM), Anchorage, Alaska, April 2001.

[7] Georgos Siganos, Michalis Faloutsos Analyzing BGP Policies: Methodology and Tool In *Proc. IEEE INFOCOM, 2004*

[8] www.netlantis.org

[9] http://antc.uoregon.edu/route-views/

[10] http://data.ris.ripe.net/

[11] C. Alaettinoglu, C. Villamizar, E. Gerich, D. Kessens, D. Meyer, T. Bates, D. Karrenberg, M. Terpstra RFC 2622 - "Routing Policy Specification Language (RPSL)"

[12] Ratul Mahajan, David Wetherall, Tom Anderson Understanding BGP Misconfiguration

[13] Multi-threaded Routing Toolkit http://www.mrtd.net/