Mapping the African Internet

J.S. Gilmore, N.F. Huysamen and A.E. Krzesinski
Department of Mathematical Sciences
University of Stellenbosch
7600 Stellenbosch, South Africa

Abstract—This paper describes the methods used to generate a router level map of the African Internet. The traceroute utility was used to collect router level information on the Internet. We developed software to automate the sending of traceroute probes to selected IP addresses, to store the information produced by the traceroute data and to transform the data into adjacency matrices. The adjacency matrices, together with geographical data concerning the location of the routers, were used to draw a map showing the Internet topology.

I. INTRODUCTION

Previous work has been done to map the Internet both at the router level [1, 2, 3] and at the Autonomous Systems (AS) level [4, 5, 6]. Work has also been done on inferring AS relationships from router level maps [7]. However, we are not aware of previous work done to map the African Internet.

This paper describes the methods that we used to generate a router level map of the African Internet. As in [1], we used the traceroute utility [17, 18, 9] to collect router level information on the Internet.

II. THE traceroute UTILITY

A. General working

The IP packet header contains a “time to live” (TTL) field whose value denotes the number of routers that a packet may visit before it is discarded. The TTL field is used to prevent packets from being trapped in routing loops: such packets are discarded, rather than have them consume Internet resources indefinitely.

The Internet Protocol is configured with a companion protocol called the Internet Control Message Protocol (ICMP) which defines a set of error messages that are returned to the source host whenever a router or host is unable to process an IP packet successfully. Thus when the TTL expires, an ICMP response is sent back to the host. This response not only informs the host that the TTL has expired, but it also contains the name and IP address of the router where the TTL expired.

From this information, the first router in a path to a specified destination can be identified. The next batch of three probes has a TTL of two and these packets will expire at the second router. When the ICMP response is returned, the second router will be known. This is done repeatedly until a packet reaches the destination host.

The traceroute utility stops when it receives either a Port Unreachable (type 3) or a Host Unreachable (type 1) ICMP response. A Host Unreachable response occurs if a route to the destination host on a directly connected network is not available (does not respond to ARP); the destination host either does not exist, or the destination host is offline.

B. Traceroute, as used in Internet mapping

The traceroute utility sends probes to destination IP addresses, but the destinations themselves are largely irrelevant when traceroute is used for mapping the routers in the Internet. What matters is the route that the probes traverse to reach their destinations. This is because the intermediate routers are mapped, and not the destination hosts. Multiple probes are thus sent out, and the returned routes are used to build a connection graph of the Internet as described in section III-C.

C. Weaknesses of traceroute

Networks often do not respond [4] to traceroute probes because many networks block the ICMP response packets from exiting their network. These packets are needed in order to discover the route. However, our experience is that most intermediate routers attach at least their IP address to the ICMP reply that is returned when a packet’s TTL expires. Most of these routers also attach their name.

III. TRACEROUTE SOFTWARE

A. The database interface

The mapping software generates and processes large volumes of data. File handling must therefore be robust and queries to the data must be handled efficiently. For these reasons it was decided to base the mapping software on a public domain database implementation.

The database that we decided on is called SQLite [10]. SQLite is a C library that implements a self-contained, embeddable SQL database engine. SQLite may be less configurable than MySQL, but it provides greater ease of use. SQLite also appears to incur less overhead so the file that it outputs is smaller than a file from MySQL containing the same data. For example our SQLite database was 75 MB and the corresponding MySQL database was some 100 MB in size.

SQLite is easily interfaced to C. The database can be operated in console mode, or via functions in C that output the console commands to the database directly.

This work was supported by grant numbers 2054027 and 2677 from the South African National Research Foundation, Siemens Telecommunications and Telkom SA Limited.
IP address data

The following functions were implemented in the SQLite database:

- Retrieve IP address ranges: This function returns the next unmarked IP address range from the “IPranges” table. If the traceroute data were successfully written into the database (see section III-A), the IP range is marked as completed. When the retrieve function is called again, the next unmarked IP address range is returned. A variation on this function was also written to get the next IP range for a given country.
- Write/Retrieve traceroute information: This function is invoked with a route structure as a parameter. The route is written into the database in the “route” table. All routes are written into the same table, and a destination IP address field is used to distinguish between the different routes. The retrieve function is used by the program Grapher to retrieve two consecutive router IP addresses along a route.

Geographical data

The GeoLite City database by MaxMind [8] was used to retrieve the geographic information of the router IP addresses. This is a free world-wide database that contains a list of IP ranges mapped to cities with their latitude and longitude shown as well.

This database is said to be 60% accurate to within a 25 mile radius in the USA and 50 to 55% accurate to within a 25 mile radius in South Africa. The accuracy for the rest of Africa is not known. There are other, perhaps more accurate, commercial versions of the geolocation software available [8, 13, 11].

A function was written to query the GeoLite City database and retrieve the latitude and longitude for a given IP address. However, the GeoLite City database does not contain the latitude and longitude positions for all IP ranges. Some IP addresses have latitudes and longitudes of zero. These “zero positions” must be looked up manually. Other websites are queried for the information [12] and if no geographic information can be found on a specific IP address, the “whois” registrar information is used.

B. The Tracer program

The Tracer program queries a database to retrieve an IP address to which it sends out a traceroute probe. The route information generated by the probe is received and stored in the database.

Finding the source code, traceroute-1.4a12

Our initial task was to find the C or java source code of a suitable traceroute implementation. The program that was used is called traceroute-1.4a12-20 [9] where -20 denotes the build number and it was written in C. The program was developed by Van Jacobson at the University of California, Lawrence Berkeley Laboratories. The program has been used in a number of mapping projects [4, 7]. The program executes a single traceroute to a target IP address.

General working

This section describes how Tracer works and how the original traceroute-1.4a12 program was modified and expanded to become Tracer. The program functions as follows:

- Tracer is run by first specifying the country to map; Tracer queries the database for IP ranges from the specified country, see section III-A.
- The IP address range is used to send out probes to two IP addresses in the selected range. These IP addresses are chosen randomly out of the first and second half of the IP range respectively.
- As each “TTL expired” ICMP response is received, the information for that router is saved in a data structure.
- After the information for the whole route has been received, the route data structure is written into the database, see section III-A.
- This process is repeated for every IP address range in the specified country, or in the whole database, depending on the mode Tracer is running in.

After this process has completed, Grapher is run over the “route” database to process the route information, see section III-C.

Modifications to the traceroute-1.4a12-20 program

- The code was made modular. This made it easier to expand the functionality of the program and was necessary in making our program scalable.
- The database interface functions described in section III-A were added to the program.
- The program was modified to execute in a loop, continuously querying the database to obtain the next IP range, see section III-A.
- A data structure was implemented to store the route information sent back by the traceroute probes. This data structure is an array of structs that store the router information.

The following route information is recorded:

- the router IP address,
- the router name, and
- the time taken by the ICMP packet to travel from the previous router to the current router. The time depends on the link medium and is also a measure of the distance between the two routers.

- The original program sent out multiple probes to improve the chances of getting a response. If a probe was successful, all subsequent probes to the same target are unnecessary because the required information has already been collected.

The program was therefore modified to send out one probe and to send out another probe only if no reply was received from the previous probe. The number of retries can be specified.

C. The Grapher program

Once the route data are collected by Tracer, the Grapher program is used to process these data and generate a connec-
tion graph.

General Working

The data that the Grapher program uses are the route information as well as the IP-to-geographic location information, all of which are stored in the database.

First the route information is used to generate an adjacency matrix. An adjacency matrix is a $N \times N$ symmetric binary matrix where $N$ is the number of routers in the database. When a router $i$ is connected to another router $j$, then $N_{ij} = N_{ji} = 1$. The adjacency matrix thus represents an undirected graph. This will always be the case where any concept of “flow direction” is ignored.

The Grapher program continuously queries the database for route information. In response to a query, two consecutive IP addresses in a route are returned. These two addresses are the ones that are connected and thus their routers are connected. The connection is recorded in the adjacency matrix.

The Grapher program uses the geolocation database to find the geographic location of each router in the adjacency matrix. This information is written out together with the adjacency matrix for visualisation purposes.

Role in the mapping project

Multiple sets of traceroutes were executed, with each set generating a separate database with its own tables for storing the relevant information. Because the software that manipulated and used the database was developed as the traceroute data were being accumulated, having multiple databases ensured that we had a complete database to experiment with, while traceroutes were still being executed with another database. The Grapher program was run against each of these databases. There were seven databases in total:

1) Algeria
2) Countries bordering on South Africa: Botswana, Lesotho, Mozambique, Namibia and Swaziland
3) Large countries: Cameroon, Cote d’Ivoire, Egypt, Gabon, Ghana, Guinea, Kenya and Morocco
4) Nigeria
5) Remaining countries: this database was used where Tracer was run in “global” mode for all other remaining African countries
6) South Africa
7) Southern countries: Angola, Malawi, Zambia and Zimbabwe

After all the software was developed, Tracer was run again over the whole of Africa. This gave us an eighth complete database of Africa that had less routers listed as the seven combined databases. This might be due to the fact that it was run over a much shorter period of time: three days instead of more than a week.

D. The Seer program

The Seer program is used to merge the adjacency matrices as well as the list of latitudes and longitudes and IP addresses output by Grapher. Adjacency matrices cannot be straightforwardly added, since the matrices may have different sizes and IP addresses.

The Seer program takes input from two databases. It uses one of the databases as a source and the other as a destination. All IP addresses in the source IP addresses list are compared to the destination list, IP addresses that are not found in the destination list are added to the list.

After the matrix has been converted into lists, the lists are compared to all the rows in the destination matrix. Connections appearing in the lists are checked against the matrix and where a connection appears in the list, a check is done to make sure that a 1 exists in the appropriate element in the matrix. Because the names of all the new routers have already been added into the IP address list, which is directly linked to the destination adjacency matrix, all the connection information of the source routers is also inserted.

The updated destination adjacency matrix is written out, along with an updated IP addresses and latitude-longitude table. Seer can also write out the data in a sparse matrix representation.

E. The Judge program

After the Seer program has processed all the data, the final database contains information on some 3925 routers so that $3925^2 \sim 15^7$ elements are in the adjacency matrix. This gives rise to a visually cluttered Internet map. A program was therefore needed to limit the data range to obtain a clearer picture of smaller parts of the map.

The Judge program extracts a subset of the full database. The subset is defined by minimum and maximum latitudes and longitudes. In effect, a database is generated that consists of a constrained view of the global map and all routers outside the specified range of latitudes and longitudes are ignored.

The Judge program was used to generate area maps for the African Internet generated without having hundreds of international router connections cluttering the map.

F. MATLAB and Xfig

An adjacency matrix represents a connection graph of the network and a method is required to render the graph as a visual map of the Internet.

The adjacency matrix and the latitude and longitude list are space-separated data structures and the “load” function in MATLAB [14] was used to import these structures. The “gplot” graphing function was invoked with the adjacency matrix and a coordinate array as parameters.

The latitude and longitude list provided the coordinates for the “gplot” function. This generated a map of routers which was overlaid on a “world map”. This map was exported as encapsulated post script (eps). Finally Xfig [15] was used to merge the graphed map generated by MATLAB, and a world map, in eps format.

Google Earth [19] helped with the positioning of the routers on the world map. Latitude and longitude position of routers were checked with Google Earth to determine where the routers should be positioned on the map.
IV. MAPS OF THE AFRICAN INTERNET

Fig. 1 presents a map of the African Internet at the router level. The map was generated using the above-mentioned tools and methods. After the Judge program constrained the global dataset, an adjacency matrix was produced that only contained routers in Africa and their connections. This dataset was mapped as described in III-F to generate a map of the African Internet.

This map shows the routers and the links between the routers, as well as the locations of the routers in Africa. Looking at the density of the outbound links at a specific point, the Internet entry points can be identified. The connections along the west coast of Africa are very dense, perhaps because of the SAT3/WASC cable (South Atlantic 3/West African Submarine Cable) [16].

Fig. 2 presents a map of the overseas links used by the African Internet. The links were discovered by sending traceroute generated probes from South Africa to countries in Africa. The most obvious reason for a packet first going to USA or to Hong Kong and then into Africa is to minimise delay.

Fig. 3 shows the locations of the routers that connect Africa to the global Internet. When routing from a country in the south of Africa, for example South Africa, to a country in the north of Africa, for example Algeria, a satellite link to Amsterdam and then a link to Algeria is most likely cheaper (in terms of the IGP metric and BGP policy) than going through many fibre links in Africa. Although a satellite link has a high latency, it involves only one hop and in many cases that will yield a two- or three-hop route to the destination. Many terrestrial links have low latencies, but will yield high hop count routes. Terrestrial routes thus appear to be more expensive end-to-end than satellite routes.

This is however not the only reason for an inter-Africa packet being routed overseas. The first database that we created was for internal South African traffic. The map showed that internal South African traffic was also being routed internationally, mainly to Washington, DC, Honk Kong and Amsterdam. Once again, a satellite link is likely to be cheaper (in terms of the IGP metric and BGP policy) than going through several fibre links in South Africa.

V. CONCLUSIONS AND FUTURE WORK

A. Router and link density

It is interesting to note the density of routers and links in the different African countries. South Africa has the most links. This might be due to the fact that all the traceroutes were sent out from South Africa. The link density might look different if the traceroute probes had been sent out from another country in Africa. On the other hand, significantly more IP addresses are allocated to South Africa than to any other country in Africa, approximately 9.7 million.
Table I shows the top 5 allocation of IP addresses in Africa. From this list it can be seen that Morocco with the second largest number of IP addresses has more that 16 times fewer IP addresses than South Africa.

It should however be pointed out that these numbers refer to allocated IP addresses and not necessarily to IP addresses in use. There is no way of knowing how many IP addresses are in use at any given time in a country. IP addresses are allocated to companies or to Internet Service Providers (ISPs). Some of these IP addresses will be allocated to hosts for Internet access. The only way to know the number of IP addresses in any given range that are allocated to hosts, and the number
of those hosts on-line, would be to ping every host in that range and check whether a response is received. This cannot be done instantaneously, and thus the picture that would be seen would be a time averaged one.

B. Multiple traceroute sources

Improvements in accuracy. As mentioned above, running traceroutes from multiple traceroute servers will improve mapping accuracy. When routing to a certain destination, a defective router may be traversed, or more likely, a network/AS that restricts ICMP packet forwarding. When routing from a different location, that network might be bypassed and the ICMP response packets might reach the sender.

With more than one vantage point, certain techniques can be used to identify IXP’s as well as the multiple interfaces of routers [4].

Another advantage of using multiple locations to route from, is that a location can be chosen that has already been mapped, but is nearer to the destination than the current source location. This would prevent IP packets from going over international routers before reaching their destination and greatly simplifies the coding and mapping process.

Adding multiple sources to Tracer, source routing. The Tracer program might be improved by adding source routing capabilities. All probes can then be sent to a certain location first, and then from there to their respective destinations. This would give the same effect as having multiple physical vantage points. This method was successfully implemented in the past [1].

One drawback of this approach is that most routers have their source routing capabilities disabled in order to prevent abuse. However, some routers do have source routing enabled and not that many vantage points are needed. If a few source routable routers can be found, it would greatly increase the accuracy of the mapping methods.

Another method would be to have several hosts positioned at different geographical locations around the world. The obstacle here is that Tracer needs root access to receive ICMP responses. It might prove difficult to find an international host where root access is possible.

C. Improvements to the Grapher program

Alternative data structures. The Grapher program could use adjacency lists or sparse matrices instead of adjacency matrices as its main data structure. Since the matrices that are created by mapping the Internet are usually sparse it would greatly decrease the amount of memory needed to store the data.

Adjacency lists also decrease the amount of memory needed to store the data. Each IP address has a list of all the IP addresses that it is connected to.

Geographic location database. More accurate commercial geolocation databases might be purchased to increase the accuracy of the geographical mapping. This would improve the accuracy of the generated map. Currently the method used to identify multiple interfaces also uses the fact that IP addresses that belong to the same router will have the same geographic location. When the IP addresses are mapped using the geolocation database, routers with the same positions are mapped to the same location. This automatically identifies the interfaces of routers. A more accurate geolocation database might therefore be able to distinguish between multiple routers.

REFERENCES

[13] IP to location software from the company: IP2Location http://www.ip2location.com

John Gilmore and Nico Huysamen are undergraduate students in the Departments of Electrical and Electronic Engineering and Computer Science respectively at the University of Stellenbosch. Anthony Krzesinski is a Professor of Computer Science at the University of Stellenbosch, South Africa.