2009

Available Bandwidth Inference Based On Node-Centric Clusters

Seetharam Samptur
San Jose State University

AVAILABLE BANDWIDTH INFERENCE BASED ON NODE-CENTRIC CLUSTERS

A Project Report

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Computer Science

by

Seetharam Samptur

April 2009

SAN JOSÉ STATE UNIVERSITY

The Undersigned Project Committee Approves the Project Titled

AVAILABLE BANDWIDTH INFERENCE BASED ON NODE-CENTRIC CLUSTERS

by
Seetharam Samptur

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

Dr. Mark Stamp,	Department of Computer Science	Date
Dr. Robert Chun,	Department of Computer Science	Date
Dr. Praveen Yalagandula,	HP Labs	Date

APPROVED FOR THE UNIVERSITY

Associate Dean	Date

ABSTRACT

AVAILABLE BANDWIDTH INFERENCE BASED ON NODE-CENTRIC CLUSTERS

by Seetharam Samptur

End-to-End Available Bandwidth (AB) is a real-time network metric that is useful for a wide range of applications including content distribution networks, multimedia streaming applications and overlay networks. In a large network with several thousand nodes, it is infeasible to perform all-pair bandwidth measurements as AB measurements could induce traffic overhead along the path. Also because of its dynamic nature, the measurements have to be performed frequently thus imposing significant probe traffic overhead on the network.

In this paper, we discuss a clustering based distributed algorithm to infer the AB between any pair of nodes in a large network based on measurements performed on a subset of end-to-end paths. The algorithm was validated on Planet-Lab and for some nodes, 80% of the inferences were within 50% of the actual value.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Mark Stamp, for his guidance, encouragement and support. I would also like to thank Dr. Robert Chun for his time and effort. A special thanks to Dr. Praveen Yalagandula for introducing me to the interesting field of Internet Network Measurements and also for his valuable suggestions.

Thanks to my parents for their constant encouragement and support with everything I do. Lastly, but most importantly, I want to take this opportunity to thank my wife, Sravanthi, and my daughter, Aishwarya, for tolerating my absence during my course work. This project would not have been possible without their support.

TABLE OF CONTENTS

TABLE OF FIGURES

INDEX OF TABLES

1 Terms and Abbreviations used in this document

Available Bandwidth – Available Bandwidth (AB) between any two nodes at any instance is the maximum throughput on the path between them taking into account the traffic at that time.

Available Capacity – Available Capacity between any two nodes at any given time is the maximum throughput on the path between them assuming there is no traffic.

Boa – Boa is a single-tasking HTTP server (Boa Web Server, 2005).

Content Delivery Network – Content Delivery Network (CDN) is a system of networked computers that deliver content to end users.

Destination Clusters – Destination clusters on a node N are clusters that contain nodes that share the first few hops along the paths from node N.

Emulab – Emulab is a network testbed available for researchers to evaluate their systems (Emulab - Network Emulation Testbed, 2002).

Pathchrip – Pathchrip is an active probing tool for estimating the available bandwidth on a communication network (Vinay J. Ribeiro, 2003).

PlanetLab – PlanetLab is an open platform for developing, deploying, and accessing planetary-scale services (PlanetLab: Global Research Network).

Pathload – Pathload (Dovrolis) is a bandwidth estimation tool.

Pathneck – Pathneck is an active probing tool for identifying bottlenecks along a path (Pathneck, Ningning Hu (CMU), 2004).

Scalable Sensing Service – Scalable Sensing Service (S^3)is a scalable, secure and reliable service that provides the system states for both individual nodes as well as for the network in real time.

Source Clusters – Source clusters on a node N are clusters that contain nodes that share the last few hops along the paths to node N.

Spread PaiR Unused Capacity Estimate – Spruce is an available bandwidth estimation tool.

2 Introduction

A wide range of applications including content-distribution networks, video streaming applications, and peer-to-peer applications are based on overlay network infrastructure. Akamai content delivery network is an effective overlay network that solves the problem of delivering content in a scalable and reliable way (Dilley, Maggs, Parikh, Prokop, Sitaraman, & Weihl, 2002). Akamai's infrastructure works with the content providers and allocates more servers to sites experiencing high traffic and directs client requests to the nearest server. The criteria used in choosing a server include availability and distance. Availability is determined by the server's current load, while distance is determined based on dynamic link characteristics such as end-to-end available bandwidth.

End-to-End Available Bandwidth (AB) between any two nodes is the maximum throughput on the path between them and is highly dependent on the real-time traffic load along the path. However, in a large network with several thousand nodes, it is infeasible to perform all-pair bandwidth measurements for the following reasons:

a. Measuring AB in a network with N nodes would require N^2 AB measurements.

b. AB can vary over short timescales because of its dynamic nature (Shriram, 2007).

c. It is challenging to perform accurate end-to-end pair-wise AB measurements in a large distributed network due to interference of existing traffic (Song & Yalagandula, Jan 2007).

For these reasons, there is a strong need for an AB inference technique that is both scalable and accurate in inferring AB between various network nodes. The problem of designing scalable monitoring services has received considerable attention in the recent past (Praveen Yalagandula, 2006). Also, Broute, a scalable AB estimation system based on a client-server route sharing model, has been proposed by researchers from Carnegie Mellon. Broute uses special nodes called the landmark nodes, and also a per-hop AB estimation tool to monitor all-pair AB measurements. Pathneck (Pathneck, Ningning Hu (CMU), 2004), the tool used in Broute for determining an upper-bound on AB was primarily developed to identify bottlenecks in the internet. The paper by (Alok Shriram S. B., 2007) proposes scalable end-to-end AB inference algorithms that shows better results compared to other solutions. However, the drawback with these algorithms is that the solution is not distributed.

In this paper, we discuss a clustering based distributed algorithm to infer AB between any pair of nodes in a large network based on measurements performed on a subset of end-to-end paths.

3 Available Bandwidth Inference – Applications

Available Bandwidth estimations are useful in many applications including content distribution networks, video streaming etc. In this section, we present an application where the algorithm described in this paper can be used to reduce the number of AB calculations in a large network.

3.1 Content Distribution Network

A Content Distribution Network (CDN), shown in Figure 1, is a system of servers networked together over the Internet in an attempt to deliver content to the end users quickly and efficiently.

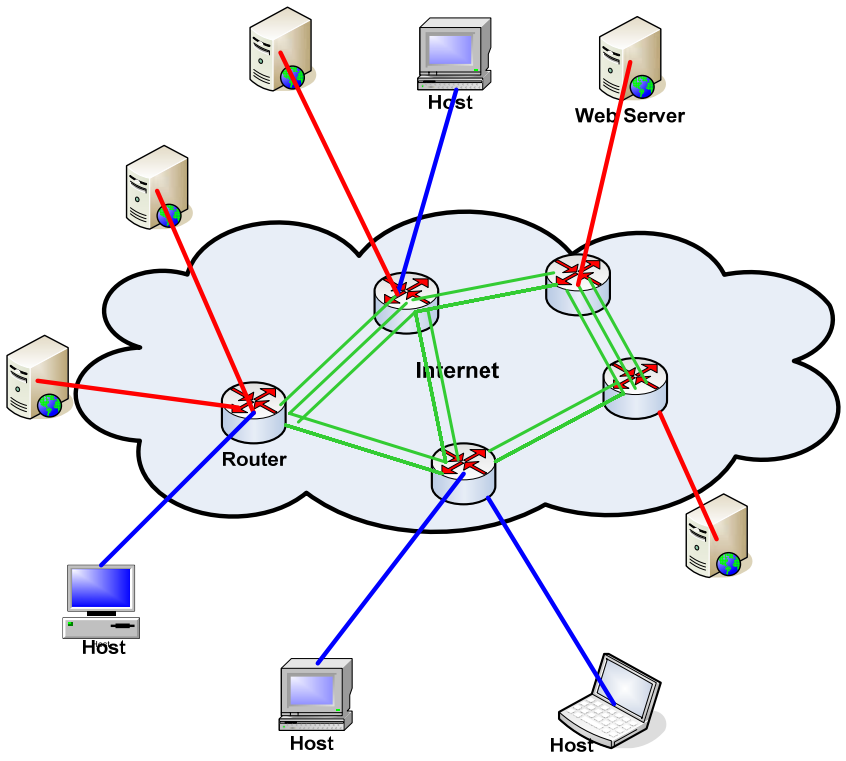

Figure 1 - Content Distribution Network

The servers satisfy requests from clients, in this case, end-users by providing the requested content. In some instances, the server may not have the requested content and has to obtain it from one of the networked servers connected in its CDN. The server can query the AB inference engine described in this paper to identify the destination server with best AB among all the servers.

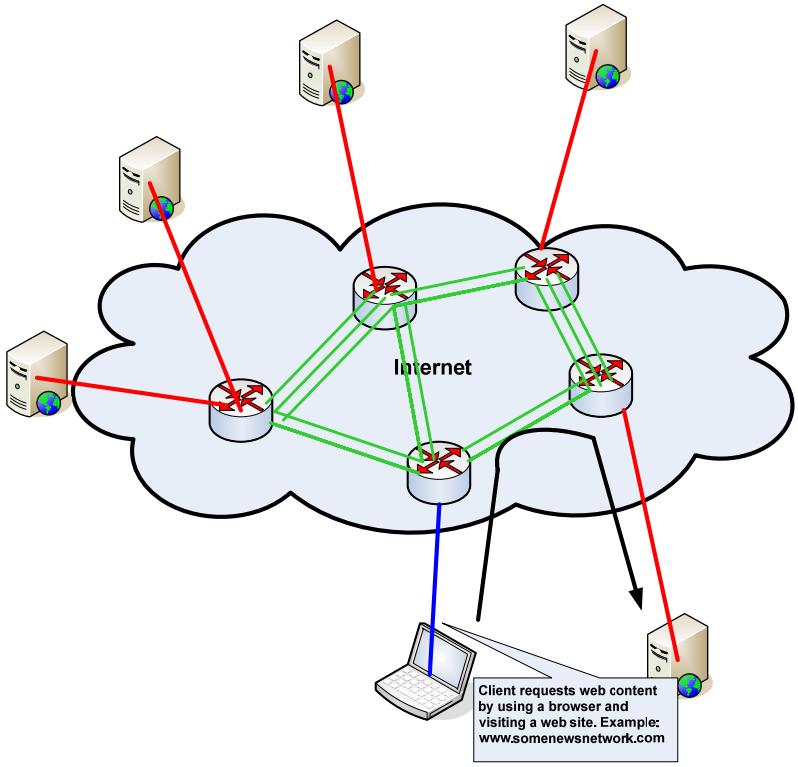

Figure 2 - Client Initiates Request for Content

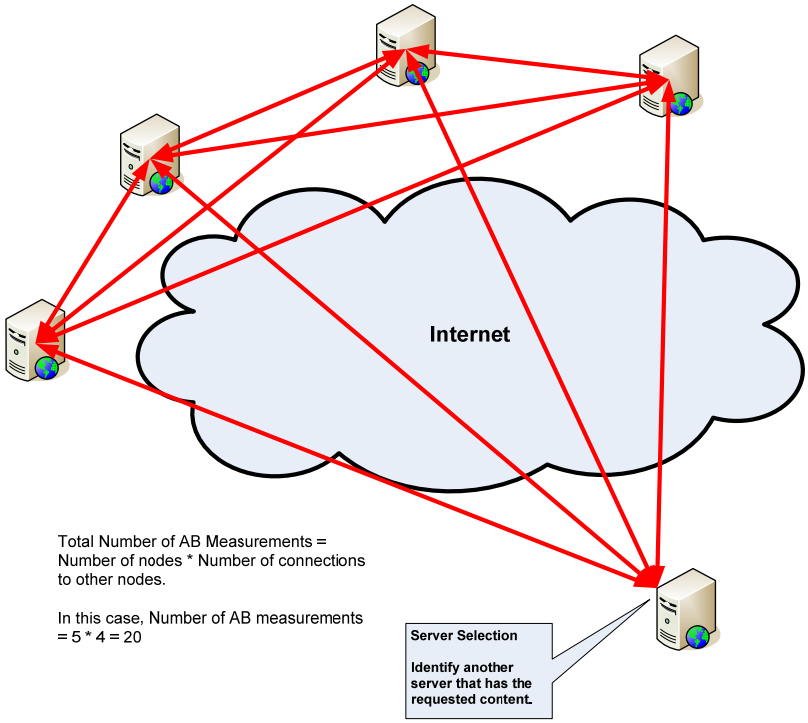

Figure 3 - All-Pairs AB Measurements without AB Inference

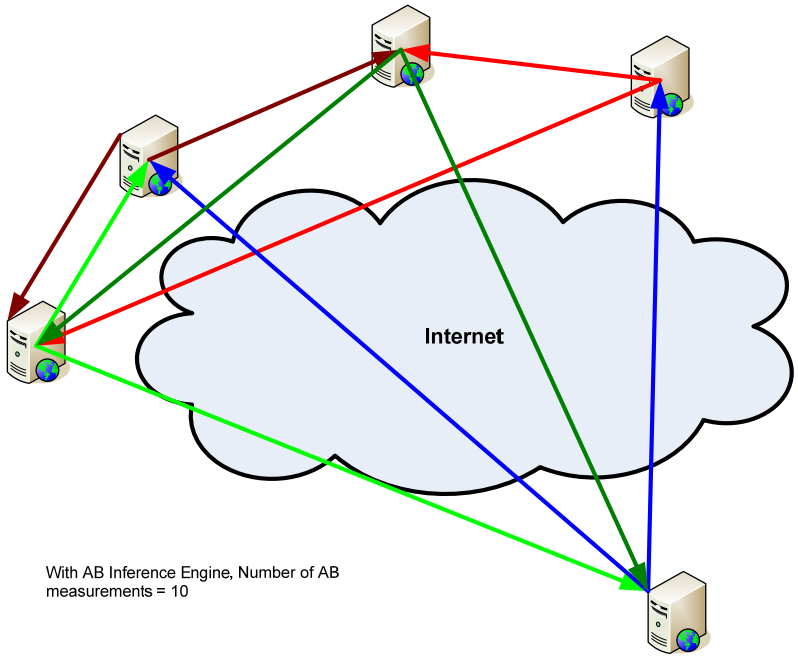

Figure 4 - Few Pairs AB Measurements with AB Inference Engine

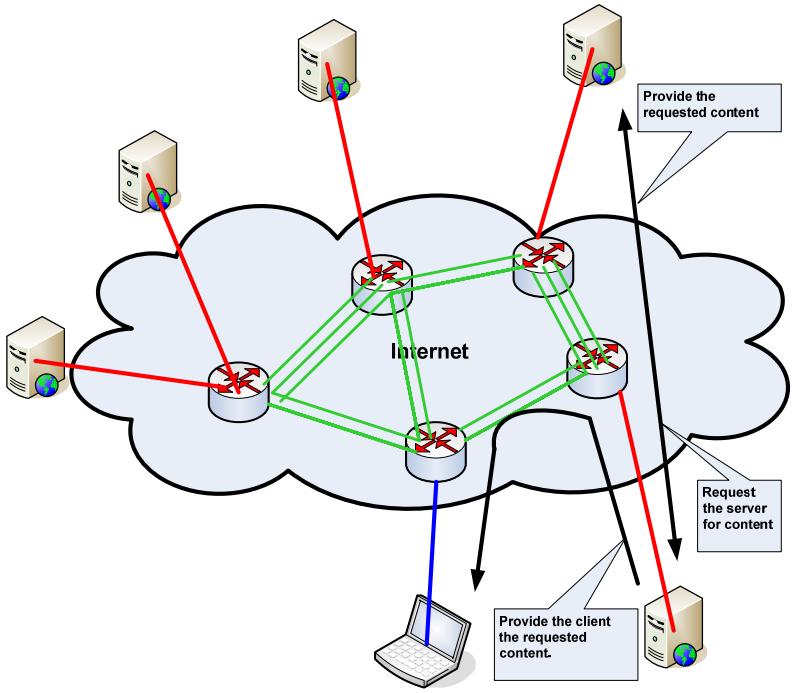

Figure 5 - Server Provides Client with Content

As shown in Figure 5, the actual number of AB measurements was reduced by 50% in the example use case. Certain AB measurement tools induce traffic into the network and the subsequent reduction in AB measurements will translate to an increased AB for rest of the applications.

4 Architecture and Design

This section describes the high level architecture and detailed design of the AB inference algorithm.

4.1 Overview

The AB inference algorithm described in this paper is based on node-centric clusters. This approach involves building clusters dynamically based on nodes in the network. For each node, the network configuration is split into two different cluster views – source clusters and destination clusters. Cluster heads are identified for these node-centric cluster views and AB measurements are performed on a subset of end-to-end paths. These measurements are used to infer the AB metric for any node pair in the network. Since the clusters are node-centric, it is easy for a node to self-adapt to a different cluster view to improve the inference results.

As shown in Figure 6, the inference engine is executed on all nodes in the network. A client node is the node that generates the AB inference request for the distributed system. All nodes in the network assume the role of a client node when generating inference requests. The peer-to-peer architecture is one method to structure the inference application such that identical software components or engines are executed on different nodes in the network. Each engine performs a subset of measurements and communicates the results to its peers using TCP/IP as the communication mechanism. With this approach the problem of inferring AB between any node pair is divided into identical sub-problems that are solved independently by each node. The information

computed on all these nodes is required to infer AB between any node pair and hence the connectivity to the network is vital. Unlike traditional peer-to-peer networks, discovering peer nodes is simple because the inference algorithm is executed in a controlled environment with every node aware of the network topology.

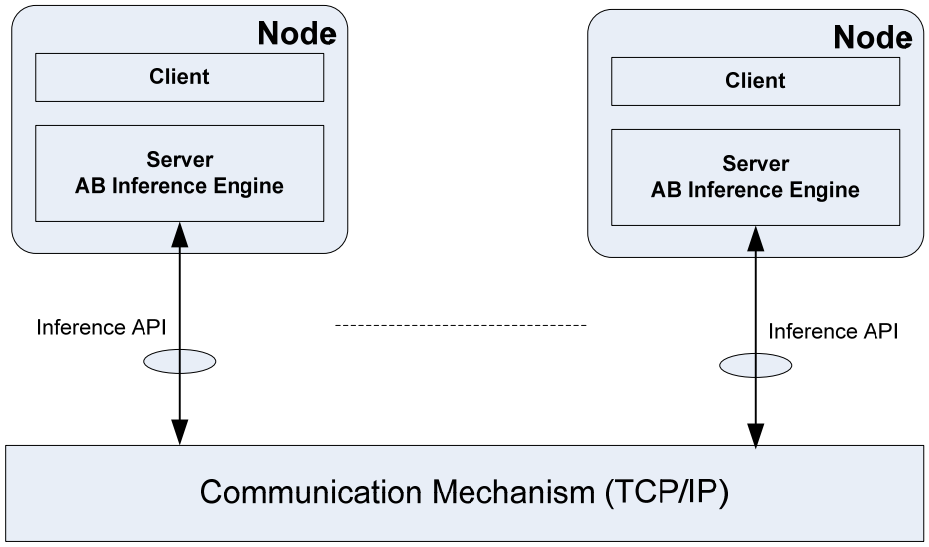

Figure 6 – AB Inference Peer-to-Peer Distributed Architecture

The node-centric approach inherently makes the algorithm distributed thus removing the dependence on a centralized server. The software is logically divided into two components, client and server inference engine. The server on each node has access to information on the AB between the node on which it is executing and the other nodes in the network. For information on AB between other nodes, the server communicates with its peer running on the other nodes. The client can request the local server for information on AB between any two nodes in the network. The server will infer the AB

between the requested nodes based on the information on the local node and based on the information it receives from its peer components. Additionally, this solution is scalable as new nodes are added to the network because the new nodes have access to AB information available on other nodes in the network.

The overall program flow for the server is as shown in Figure 7. The computations performed at each node in the network can be broadly divided into following tasks:

- Network data collection
- Network topology construction
- Cluster formation and Cluster head selection
- AB measurements and Inference

Each of these tasks is described in subsequent sections.

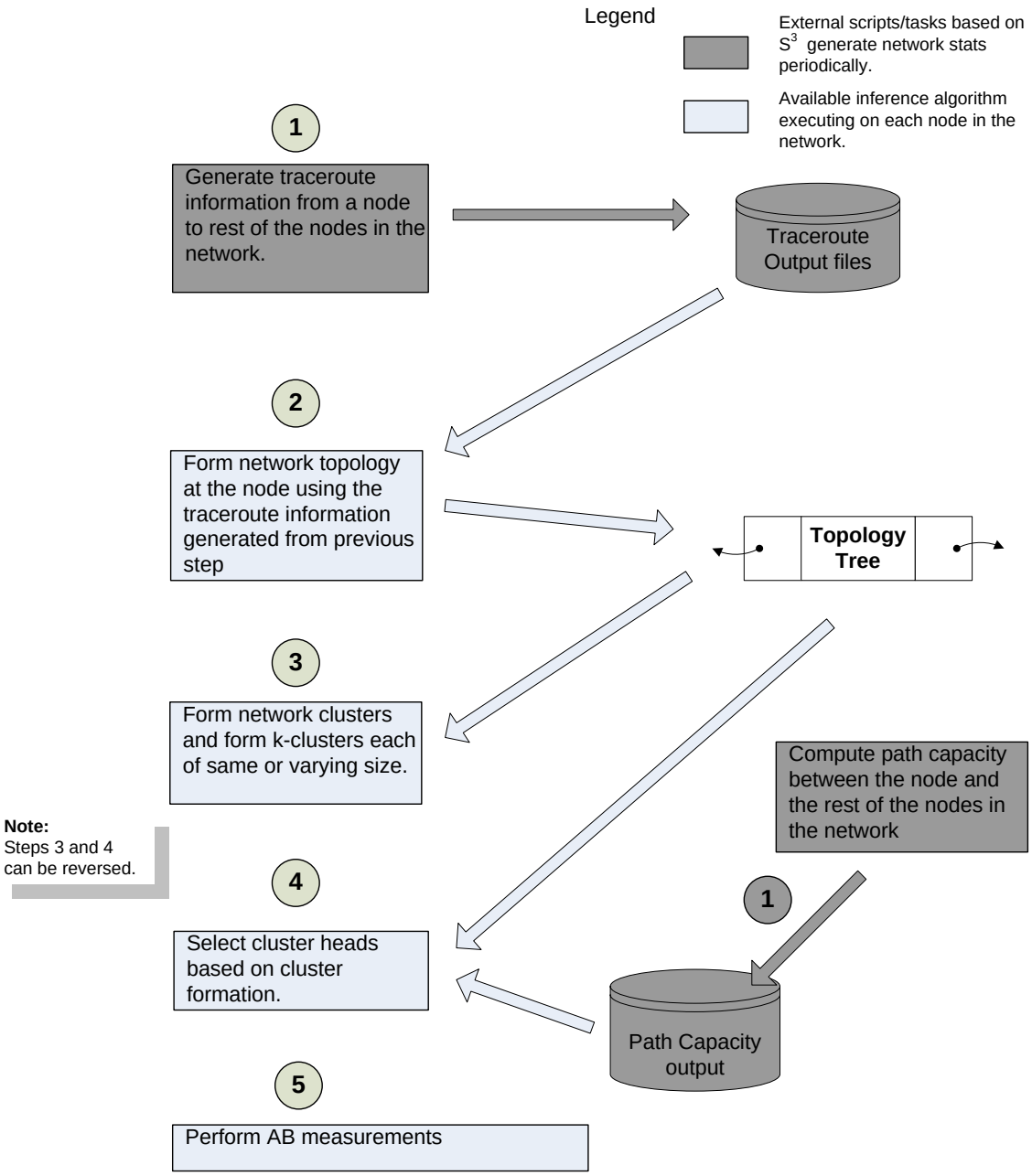

Figure 7 – AB Inference Program and Data Flow Diagram

4.2 Network Data Collection

One of the first steps is to gather various network data for evaluating the algorithm. The network topology is required to determine the connections between various nodes in the network. Additionally, the path capacity, the maximum possible end-to-end throughput that is fixed between two end nodes, is also required as it is used to select cluster heads on forming clusters.

The process of generating the topology information is highly dependent on the number of nodes in the network and the tool used to gather the information. Hence, the data collection time could be quite high since AB inference will be used on a large network. However, these networks are expected to have very few changes, if any, over long periods of time. For these reasons, it is efficient to perform the data collection once at the beginning and to update any changes by an external entity. Hence, the data collection component is developed as Perl scripts that are executed periodically to update the network topology and path capacity metrics.

4.3 Network Topology Construction

Network topology is the interconnection between directly connected nodes in a network (Siamwalla, 1998). The nodes can be either hosts or routers that connect these hosts in the network. In a large network, hosts and routers can be added (removed) to (from) the network thus making it difficult to determine an accurate topology in real-time. Additionally, the tools available to determine the interconnections may introduce some errors in the topology discovery because of complexity in routing protocols. A

typical router can have several IP interfaces to connect to different sub-nets. Based on the tools used, each of these interfaces could end up as a router in the topology. There are some standard protocols such as SNMP that can be used to overcome the multiple interface problems and to deduce an accurate network topology. However, not all nodes have a SNMP agent installed on them to provide the required topology information. The challenge is to identify tools that are widely deployed, impose the least possible overhead and discover an accurate topology.

Most applications that use AB inference to improve performance are deployed in a controlled network environment. For example, a content distribution network will include a number of servers that distribute content and the nodes that host the content are pre-determined and their information is available. However, the physical topology of the network including the routers and the ports that connect the different end hosts is required in order to generate clusters required to infer AB between all nodes in the network.

Topology discovery can be an active or a passive process (R. Siamwalla, July 1998). Active mechanisms require sending/receiving protocol packets to determine the paths between the nodes in the network. Passive techniques rely on the data on the network to populate the topology database. A passive approach can analyze packets that are sent and received over various ports on the device to determine a list of nodes in the network and their interconnections. Since the passive technique relies on network traffic, it is useful in environments where such traffic is available at times to deduce the topology. This project is validated on a research network that does not have predictable traffic at all times. Hence, we consider tools based on active mechanism in this project.

4.3.1 Different Tools/Protocols Considered for Topology Discovery

This section includes the various tools and protocols based on active mechanisms that were considered for identifying the network topology. Three methods are investigated before selecting one of the techniques suitable for solving the inference problem.

4.3.1.1 SNMP based network management tools

Simple Network Management Protocol (SNMP) is a widely used network management protocol used for network monitoring. SNMP agents are deployed on the various nodes in the network and a SNMP manager running on a host extracts the information from the SNMP agents. On devices that support SNMP, in order to gain advantage over competitors, most vendors implement SNMP agents that expose proprietary Management Information Base (MIB) thus making it difficult to develop software that can work with agents from multiple vendors. The physical topology MIB, RFC2922 (Jones, 2000) provides a standardized way to identify connections between network ports and to discover network addresses of the SNMP agents. It describes the various MIB objects that can be used to learn the physical network topology. One of the major drawbacks of SNMP is that not all devices have SNMP support and thus use of SNMP to determine the network topology is restricted to intranets built around SNMP-based devices.

4.3.1.2 Domain Name System (DNS)

The DNS associates information for the different domains in a distributed database system that is based on a client-server model. The nodes that manage the database are called the domain name servers or name servers for short. A DNS query can be initiated from a DNS resolver to retrieve information about the domain managed by a name server. NSLOOKUP is an application that can be used to retrieve various name server records from a name server. This application can be used to initiate a zone transfer to retrieve all the name server (NS) records from a primary name server. Since the NS records contain sensitive information, most name servers are configured to enable zone transfers only between inter-dependent name servers or transfers are protected by enforcing encryption on the payloads (Paul Albitz, 2001).

4.3.1.3 Traceroute

Traceroute is a network utility used to determine the path a packet would take from source machine to destination. It uses the IPv4 protocol time to live (TTL) field or the IPv6 hop limit field to determine the routers/gateways on the path. An UDP request destined to an unused port is sent to the destination with a TTL (or hop limit) set to 1 and increases it by 1 until the max hop value is reached. At each stage, the gateway that receives the request with a TTL (or hop limit) value of 1, will respond with an ICMP TIME EXCEEDED response and the destination will respond with a PORT UNREACHABLE message (Wiki: Traceroute). On receiving each ICMP response, the lists of routers along the path are populated.

Issues with using traceroute:

1. All packets may not take the same path and hence the output could be confusing at times.
2. Some routers on the path may not respond to the ICMP request on the interface
3. Dependence on TTL field leads to dependence on implementations. Some implementations could be buggy; some may not follow the protocol and may end up forwarding packets with a TTL value of 0.
4. The IP address of the router indicates the interfaces on which the packets are received and not the interfaces on which the packets are forwarded subsequently.

5. Since TTL is a TCP field, layer-2 switches in the network will go undetected resulting in a less detailed topology.
6. Sending probe requests to every router along the path results in considerable network overhead.

4.3.1.4 Conclusion - Topology Discovery Tool

In addition to the tools described above, there are also tools based on proprietary protocols such Cisco Discovery Protocol, Foundry discovery protocol etc. However, these can only be used in intranets where all the devices support such protocols.

Since most networks include devices from different vendors, use of SNMP or other proprietary protocols is not an option. For security reasons, the DNS servers may be configured to block any requests to retrieve the name server records. Most nodes respond to traceroute requests for network monitoring purposes. Hence, in spite of some known issues, **traceroute** seemed to be a suitable tool that could be used to discover an approximate topology of a large network.

4.3.2 Forward Topology View

AB inference between two nodes could be different depending on the direction of the path. The AB between two nodes, X and Y, will vary depending on the source node, i.e. AB(X→Y) can be different from AB(Y→X) because of the network topology. The reason for this is that the end-to-end paths between two nodes may be different depending on the route established between the two end hosts. Hence, the forward topology from

each node to every other node in the network is constructed to identify all the routers along the paths to various destinations.

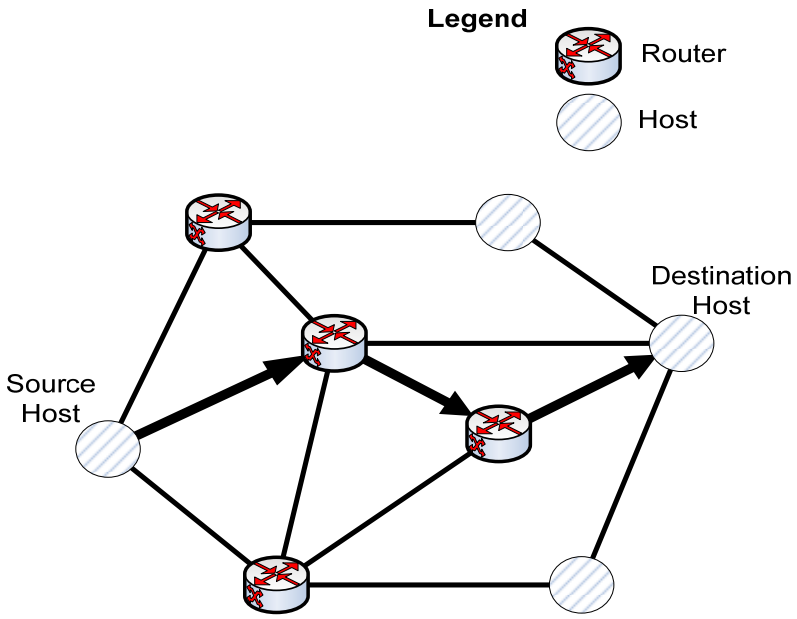

Figure 8 –Topology View from Source to Destination

In Figure 8, the route from a source host to a destination host is shown with two routers along the path. This step also provides information about the different hosts that share routers and hence the same segments along the way from the source node. This forward topology data is used to create destination clusters that will help reduce the number of AB measurements.

4.3.3 Reverse Topology View

Similar to the forward topology view, the view from all the nodes to the source node is essential in inferring the AB from any node to the source node. This view is termed

the reverse topology view from other end nodes to the source node and contains information on all the routers shared by the other nodes when communicating to the source node.

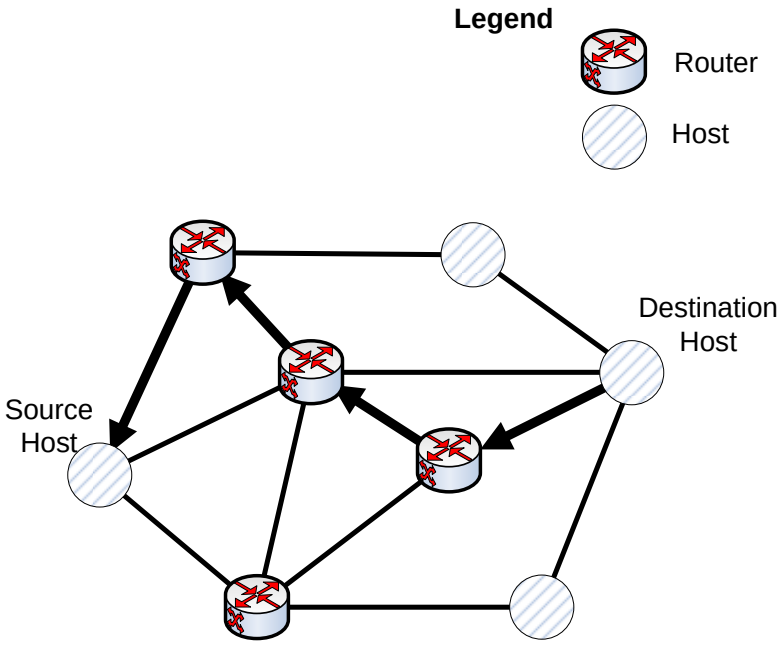

Figure 9 –Topology View from Destination to Source

The route information obtained in this step is used to form source clusters similar to the destination clusters formed using the forward topology view. The route from the destination host to the source host is shown in Figure 9. Note that this reverse route is entirely different from the forward route used to traverse from the source host to the destination host.

4.4 Clustering

The term cluster is overloaded and refers to different things based on the type of application. The applications are diverse and can range from clustering computers, clustering data points for statistical data analysis and clustering network nodes. The clustering algorithms are generic and can be applied to most problems including the AB inference problem. In the context of this problem, clustering is the process of organizing nodes into groups whose members are similar based on certain criteria.

Clustering methods (Wiki: Data clustering, 2008) can be broadly classified as follows:

- Partitioning algorithms
- Hierarchical algorithms
- Density-based algorithms
- Grid-based algorithms

This project involves identifying nodes that have similar characteristics and clustering the nodes into clusters. Of the different clustering methods mentioned above, the partitioning algorithms are ideal for this project. Some of the partitioning clustering algorithms including K-means, K-medoid and Fuzzy-C were investigated.

4.4.1 K-means

The *K*-means algorithm assigns each node to the cluster's centroid. The centroid is a node that forms a good representative of its cluster. A set of K centroids are chosen at random or based on criteria applicable to the problem. The rest of the nodes are added to the clusters based on the distance of the node to one of the K centroids. The "distance"

could be the shortest path between the nodes or based on a path overlap between the nodes.

The algorithm steps for the standard K-means (Wiki: Data clustering, 2008) clustering are:

1) Choose the number of clusters, ***K***.
2) Randomly generate ***K*** clusters and determine the cluster heads, or directly generate ***K*** random objects as cluster heads.
3) Assign each node to the nearest cluster center.
4) Re-compute the new cluster centers.
5) Repeat the two previous steps until the cluster configurations do not change.

4.4.2 Fuzzy-C

The Fuzzy-C clustering algorithm is similar to the K-means algorithm except that each object can be assigned to one or more clusters. The coefficients for each object are computed to determine its distance from the cluster center. The degree with which an object is considered to be part of a cluster is inversely proportional to its distance from the cluster center (Wiki: Data clustering, 2008).

4.4.3 K-medoid

The K-medoid clustering algorithm finds representative objects called medoid, which is the most centrally located object in the cluster.

The algorithm steps for the standard K-means (Wiki: Data clustering, 2008) clustering are:

1) Start from an initial set of K medoids to form K clusters
2) Add each data object to the cluster with most similar medoid.
3) Randomly select a non-medoid in each cluster.
4) Compute the cost of switching the current medoid with the randomly chosen non-medoid. If the cost is low, choose the non-medoid as the new medoid.
5) Repeat steps 3 and 4 until there is no change in medoid.

Adding nodes to multiple clusters will result in complicating the Inference algorithm as it has to then optimally select one of the node clusters. Hence Fuzzy-C was not considered for this project. With the K-medoid approach, the medoid is chosen and replaced iteratively until the appropriate medoid is chosen for the cluster. This process could result in a significant amount of time for large data set. Since we are dealing with a large number of network nodes, this algorithm does not scale well. For these reasons, we chose an algorithm that is based on K-means partitioning algorithm.

4.4.4 Clustering Around Netroids (CAN)

An important component of a clustering algorithm is the distance measured between two data points or nodes in this case. Domain knowledge is required to guide the formulation of a suitable distance measure metric. As described in (Hartuv and Shamir), the goal of any clustering analysis should satisfy two criteria: *homogeneity*: elements in the same cluster should have high similarity and *separation*: elements in

different clusters should have low similarity. Also, the similarity between the cluster head and the other nodes in the cluster should be high. For the networking problem at hand, we can choose the paths shared by nodes as a metric to determine the clusters.

In this project, clusters are built around **Net**work nodes on ste**roids** or **Netroids**. Netroids are nodes with best path capacity from the client node. The rest of the nodes are added to these clusters based on Common Path Index (CPI). CPI is the number of hops shared by the nodes from the client node.

The following clustering algorithm is a **variant** of the K-mean algorithm:

A. Cluster Head Selection Data

Compute the path capacities between the client node and all the other nodes in the topology tree.

Sort the nodes in decreasing order of the path capacities.

B. Compute the router list for each node

The path from the client node to a destination node will include one or more hops through routers.

As part of the topology formation, create a data structure (hash table) that stores the node along with the list of routers on its path.

C. Generate a common path index (CPI) matrix

For each node in the list

- Find the number of routers common in its path to the other node

- Store the number in index (i, j) where i and j are the nodes under consideration

Note: The value for CPI (i, j) = CPI (j, i). Hence we need to run this step for n/2 nodes only and not n.

D. Initialize available nodes list with all nodes

Repeat Steps E and F until all nodes are assigned to clusters

E. Determine Cluster head

Choose the node with highest capacity (**See step A.**) as the cluster head.

F. Scan the row for the cluster head

If the entry for an index is greater than or equal to some value

(For example: r/2 where r is the number of routers)

Then

Add that node to this cluster

Remove the node from the available nodes list

Repeat this step until all entries in the row have been scanned

In our project, cluster heads are chosen and clusters are formed around these cluster heads. The primary reason for this approach is that the cluster head selection is based on the end-to-end path capacity and this information is available thus eliminating any heuristics.

4.4.5 Destination Clusters

The AB Inference algorithm described in this paper is node-centric and hence distributed on all the nodes in the network. The server component on each node executes an instance of the clustering algorithm, CAN, described in section 4.4.4. It is applied on the forward topology of the network from each node. The result is the formation of destination clusters on a node with each cluster containing nodes that have similar CPI from the node. In other words, for any node X, the destination cluster contains nodes that share the first few hops from it. The reason for choosing nodes that share the first few hops is because the AB from the client node to these nodes will have some correlation as they share some hops along the way. The node in each destination cluster with the maximum end-to-end capacity from node X is chosen as the destination cluster head.

4.4.6 Source Clusters

The clustering algorithm is also applied to the reverse topology data to generate source clusters. These clusters have nodes that share a similar CPI to the node on which these clusters are being generated. For any node X, the source cluster contains nodes that share the last few hops to the node X. The node in each source cluster with the maximum end-to-end capacity to node X is chosen as the source cluster head.

The information related to IP aliasing, when available, will yield better source clusters. The IP alias resolution is the process of identifying IP addresses belonging to the same router (Ken Keys, CAIDA). Each router in the network can have two or more interfaces and each interface will have a different IP address. Since traceroute is used for

constructing the topology, the traceroute results can have multiple addresses that point to the same router. Hence, with IP alias information, the clustering algorithm will identify nodes that share the same router even though the IP addresses of the routers in their respective paths are different.

The router connecting nodes A, B and X in Figure 10, has an IP addresses for each of its three interfaces. Hence, without the IP alias information, the source cluster on node X may not contain nodes A and B in the same cluster even though these nodes share the same router on their first hop to node X. This could result in more number of clusters thus increasing the number of AB measurements.

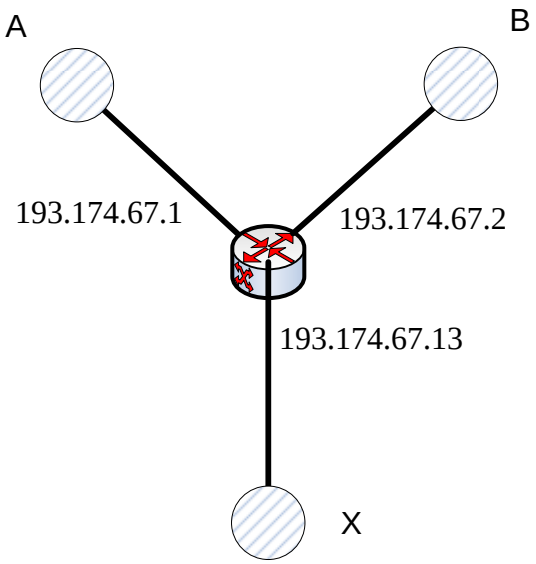

Figure 10 – IP Aliasing

However, the IP aliasing information is not required for destination clusters because the data from node X to nodes A and B is always transmitted through a single interface that is going into the router.

4.5 Inference Algorithm

This section describes the AB inference algorithm executed on all nodes in the network. With this, the AB between any two nodes (A and B) in the network can be inferred in a short time from any node X. The source and destination cluster information will be used to reduce the total number of AB measurements during the inference process.

The following notation is used in this section:

$N = \{X, X_1, X_2, X_3 \ldots X_{n-1}\}$ is the set of '*n*' nodes in the network

D_A: Destination cluster on node A

S_A: Source cluster on node A

$H(D_A)$: Head of a destination cluster on node A

$H(S_A)$: Head of a source cluster on node A

$D_A(B)$: Destination cluster on node A containing node B

$S_A(B)$: Source cluster on node A containing node B

$H(D_A(B))$: Head of destination cluster on node A containing node B

$H(S_A(B))$: Head of source cluster on node A containing node B

$A \rightarrow B$: Available bandwidth from node A to node B.

Each node X_i performs measurements from itself to all the destination cluster heads. Similarly it also requests the heads of all source clusters for the available bandwidth information from the head of source clusters to itself. With this information, the available bandwidth between any two nodes A and B is inferred as follows:

Step – 1: Lookup AB from A and Head of the destination cluster on A containing B

$$A \rightarrow H(D_A(B)) \qquad (1)$$

Step – 2: Lookup AB from Head of the source cluster on B containing A to B

$$H(S_B(A)) \rightarrow B \qquad (2)$$

Step – 3: Infer AB using (1) and (2)

$$A \rightarrow B = min\{ A \rightarrow H(D_A(B)) \text{ and } H(S_B(A)) \rightarrow B \} \qquad (3)$$

The end-to-end path between two nodes will have multiple hops and the available bandwidth is usually equal to the bandwidth on hop that is the minimum of all hops. Hence, we choose the minimum of the two AB measurements and not the average or maximum in equation (3) above.

Consider the scenario of possible overlaps between measurements used for inferring the bandwidth on a path between two nodes A and B as shown in Figure 11. The points E and F are intermediate points on the path from A to B where the path AB intersects with paths CB and AD respectively.

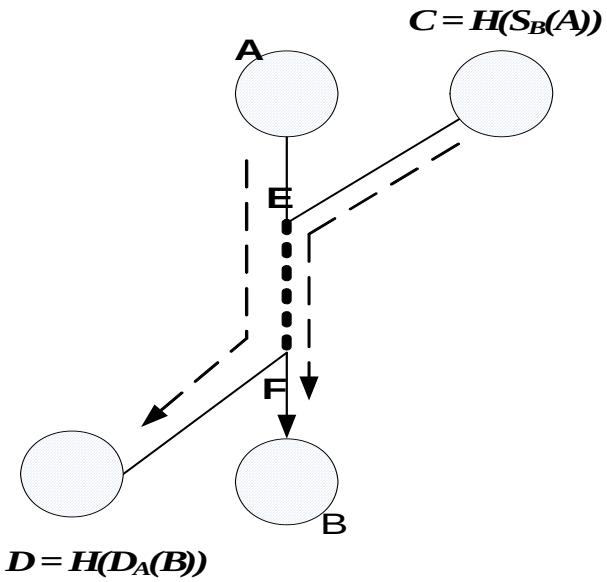

Figure 11 – AB Inference from A to B with Overlapping path EF

Note that in the case of overlapping paths, the paths that do not overlap with the intended path, DF and CE should not contain smaller bandwidth than the overlapped paths because we chose the cluster head nodes that have maximum capacity and hence we expect these links to have higher bandwidth.

In the case where there is no overlap, as shown in Figure 12, the portion of requested path that is not covered by the measured paths, XY, is assumed to be bottleneck free.

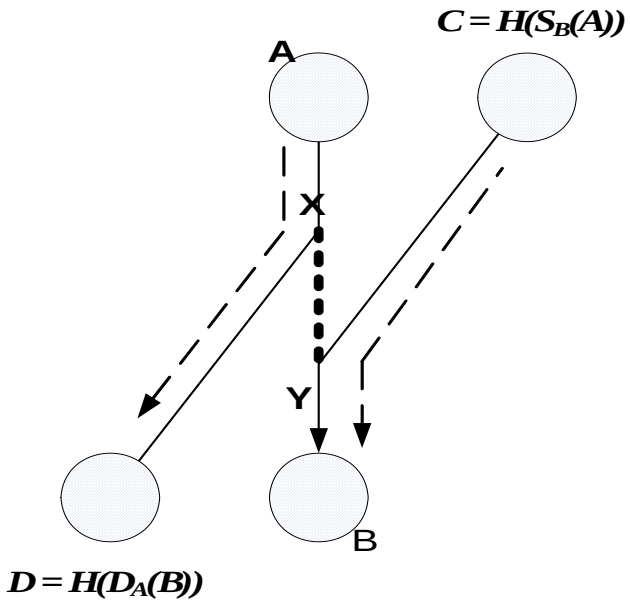

Figure 12 - AB Inference from A to B with Non-overlapping path XY

Also, we assume that the overlapping path EF in Figure 11and the non-overlapping path XY in Figure 12 are not bottlenecks because these are the core links that are expected to be well provisioned fibre optic links that have high capacity and bandwidth as opposed to the last mile links, AE, FB, AX and YB.

4.6 AB Inference Examples

Assume a network N containing nodes, where:

$$N = \{X, X_1, X_2, X_3 \ldots X_{n-1}\}$$

is the set of '*n*' nodes in the network

Assume the AB inference client executing on node X is interested in AB metric between X and some other node X_i in the network.

At each node in the network, the computations shown in (1) and (2) below are performed periodically. For example at node X, the following metrics are computed periodically:

(a) Measure AB from X to head of all destination clusters

$$\mathbf{X \rightarrow H(D_X)} \qquad \mathbf{(1)}$$

(b) Measure AB from head of source clusters to X

$$\mathbf{H(S_X) \rightarrow X} \qquad \mathbf{(2)}$$

Computing **(1)** is straightforward and it can be measured on node X itself.

Computing **(2)** is as follows:

```
For each source cluster Sx on X
    Contact H(Sx) requesting for AB from H(Sx) to X computed
  at H(Sx)
```

Example - 1

Infer AB from node X and say node X_3

Assume $\mathbf{X_3}$ is the head of a destination cluster on node **X**

The AB from **X** to $\mathbf{X_3}$ is a simple lookup as the value was obtained in **(1)** above.

Example – 2

Infer AB from node X and say $\mathbf{X_4}$

Assume $\mathbf{X_4}$ is NOT head of any destination cluster on node X

$$\mathbf{X \rightarrow X_4 = min \{ ABVal\_1, ABVal\_2 \}} \qquad \mathbf{(3)}$$

Computing ABVal_1

ABVal_1 is computed on node **X** as follows:

1) A lookup on $\mathbf{X_4}$ will provide the destination cluster $\mathbf{X_4}$ is part of and also the head of that destination cluster, $\mathbf{H(D_X(X_4))}$.
2) Run pathchirp to get AB from node **X** to $\mathbf{H(D_X(X_4))}$

$$\mathbf{X \rightarrow H(D_X(X_4))} \qquad \mathbf{(4)}$$

Computing ABVal_2

ABVal_2 is computed on node $\mathbf{X_4}$ as follows:

1) A lookup on **X** in the source clusters will provide the source cluster **X** is part of.
2) Using the cluster information, lookup for the head of that source cluster, $\mathbf{H(S_{X4}(X))}$.
3) Run pathchrip to get AB from node $\mathbf{H(S_{X4}(X))}$ to $\mathbf{X_4}$

$$\mathbf{H(S_{X4}(X)) \rightarrow X_4} \qquad \mathbf{(5)}$$

Finally, compute (3) using (4) and (5) to complete inference on AB from node **X** to $\mathbf{X_4}$.

$$\mathbf{X \rightarrow X_4 = min \{ X \rightarrow H(D_X(X_4)),\ H(S_{X4}(X)) \rightarrow X_4 \}}$$

5 Experimental Evaluation

In this section, we discuss the software tools both third-party software as well as software developed for this experiment, a PlanetLab (PlanetLab: Global Research Network) testbed used for evaluating the AB inference algorithm and the test results.

5.1 Software Tools

The project involved processing a large number of traceroute output files generated using Scalable Sensing Service (S^3) (Praveen Yalagandula, 2006). Perl scripts were developed to parse these files to generate information pertaining to routes between different nodes in the system. These routing data files were further analyzed using a distributed AB inference algorithm. The algorithm was implemented using Java as it is suitable for distributed computing.

5.2 PlanetLab

PlanetLab is a network testbed that has evolved over a period of time to aid researchers in conducting distributed experiments in network measurement, peer-to-peer networks, content distribution, resource management, authentication, distributed file systems, and many other areas (Neil Spring, 2006). A wide number of experiments are in progress at any time on around 700 nodes located around the world.

5.3 Scalable Sensing Service – S^3

S^3, a scalable, secure and reliable service was developed to provide the system states, both individual node as well as the network in real time (Praveen Yalagandula,

2006). The S^3 architecture includes web-based sensor pods used to execute and collect data periodically. The sensor pods shown in Figure 13 provide a secure web interface that provides APIs to query, control and notify events. The backend includes a controller that triggers management agents and a repository containing policies and test results.

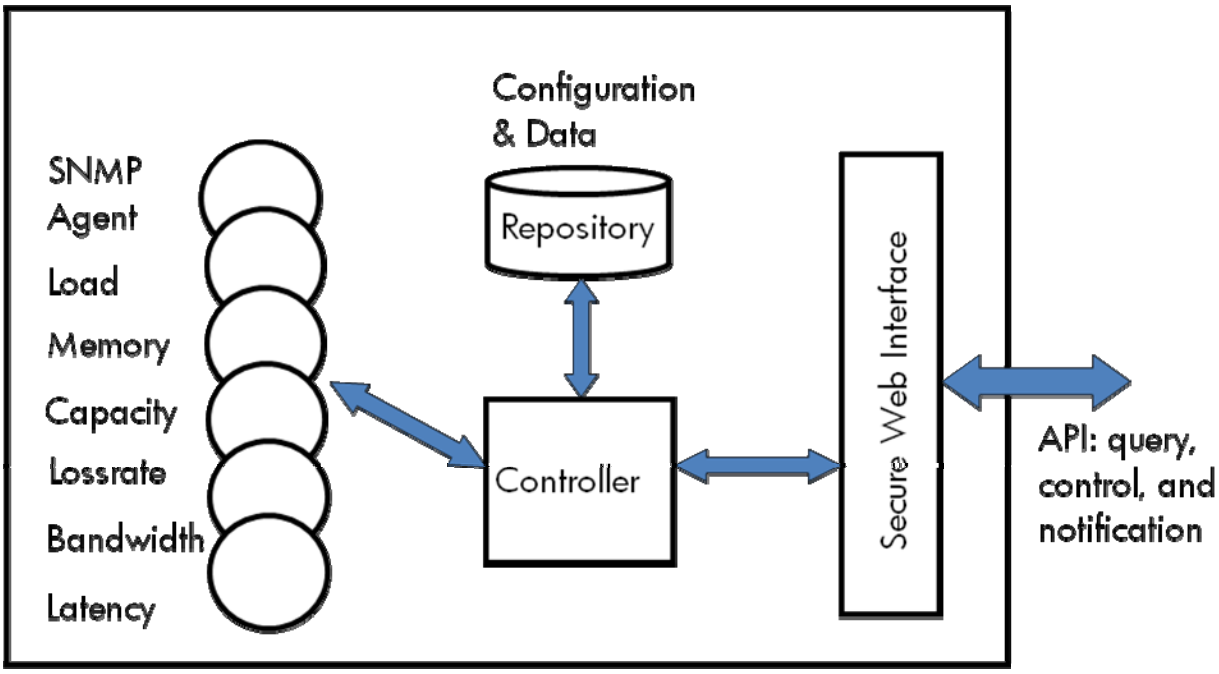

Figure 13 - S3 Sensor Pod

(Praveen Yalagandula, 2006)

An implementation of S^3 module is available on PlanetLab testbed. The secure web interface is provided by BOA (Boa Web Server, 2005), a single-tasking embedded web server, designed for speed and security. It is written in C and has been ported to many UNIX flavors. The sensor pods are implemented as CGI scripts that invoke network

management and measurement applications such as ping, traceroute, pathneck, pathchrip etc.

5.4 End-to-End Available Bandwidth Measurement tools

There are a number of publicly available bandwidth estimation tools based on different methodologies. The different tools include: abing, cprobe, pathchirp, pathload and Spruce. In (Alok Shriram M. M., 2005), the authors compare these tools for accuracy and operational characteristics along with the factors that impact the tools performance. The bandwidth estimation tools have to be very fast and less intrusive as accurate results are required in real-time. This section provides a brief description of some of these tools.

5.4.1 Pathload

Pathload estimates the end-to-end available bandwidth by sending stream of UDP packets at a rate higher than the available bandwidth in the path. The relative one-way packet delays show an increasing trend when the packet stream rate is higher and no delay when the stream rate is lower than the available bandwidth (Dovrolis). It uses a fleet of N streams to estimate the available bandwidth. The drawback with this tool is that it has to be executed on both the sender and receiver to determine the available bandwidth between them.

5.4.2 Spread PaiR Unused Capacity Estimate (Spruce)

Spruce derives estimates of available bandwidth from the amount of delay introduced by the network between paired packets. It sends 14 back-to-back UDP packet pairs with a waiting interval of 160-1400 ms between pair probes (Alok Shriram M. M., 2005). Each packet is time-stamped at both the sender and receiver ends and the sender estimates the available bandwidth based on the packet inter-arrival time. One drawback of this tool is that the internal algorithm requires the available capacity between the sender and receiver.

5.4.3 Pathchirp

Pathchrip is an active probing available bandwidth estimation tool that uses an exponentially spaced chirp probing train (Vinay J. Ribeiro, 2003). The primary advantage of this technique over the packet pair techniques used by pathload and spruce is that the number of packets is reduced by half. It estimates the available bandwidth along a path by launching a number of packet chirps from sender to receiver and then conducting a statistical analysis at the receiver.

5.4.4 Conclusion - AB Measurement Tool

Based on the experimental results described in (Alok Shriram M. M., 2005), pathchirp is considered as one of the better tools for measuring available bandwidth and hence is used in this project.

5.5 Deployment

The experiments were conducted on data collected from PlanetLab network that includes computers located in various parts of the world.

5.5.1 Network Data Collection

The first step involved identifying the topology of the test bed by generating traceroute information from each node to every other node in the network. The process resulted in large number of text files containing the traceroute information at each node.

Here are the steps followed to gather the data on the PlanetLab network:

a. Each node on the PlanetLab network has a BOA web server and a S^3 service sensor pod in the form of a CGI script that supports applications including ping, traceroute, pathneck etc.
b. On each node, start the CGI script with the command "traceroute", destination set to other nodes in PlanetLab and source set to local node.
c. The previous step will result in one traceroute file per destination for each source node. For example: Five nodes will result in permutation(5, 2) or 20 traceroute output files. In general with “n” nodes, we would have permutation(n,2) traceroute files.
d. A Perl script was developed to pre-process the traceroute output file and generate another set of files containing the routing information. The resulting file is per destination similar to the ones generated in step (b)

above but the contents are stripped to contain just the traceroute output starting from the first hop to the last known good hop.

The traceroute application is executed on all nodes and the routing information is gathered to and from every other node in the network because of the possible asymmetry in the results.

5.5.2 Clustering Analysis

The AB inference algorithm can be used in many applications and some of these applications may have limitations on the number of measurements that can be performed periodically. Hence, it is useful for the application to configure the AB inference engine to accept the number of clusters as a configuration parameter and to cluster the nodes into the required number of clusters.

This section describes the results obtained by using various clustering techniques and will be useful for determining the appropriate clustering technique based on the application use case. The clustering algorithm, CAN, described in section 4.4.4, is based on common path index. The CPI, as described earlier, is the number of routers common along the paths between two nodes from the client node.

5.5.2.1 Destination Cluster Analysis

The experiment was run on PlanetLab network and there were 278 active nodes when the traceroute results were captured. The clustering techniques, shown in Figure 14, are numbered 1 to 9, where technique #1 is most conservative method resulting in as few clusters as possible. Technique #1 clusters all nodes that share at least one hop

(Minimum CPI) from the client node while technique #1 attempts to cluster nodes that share the maximum number of hops (Maximum CPI) from the client node. Technique #3 considers the average CPI and the rest of the methods are averages of prior methods.

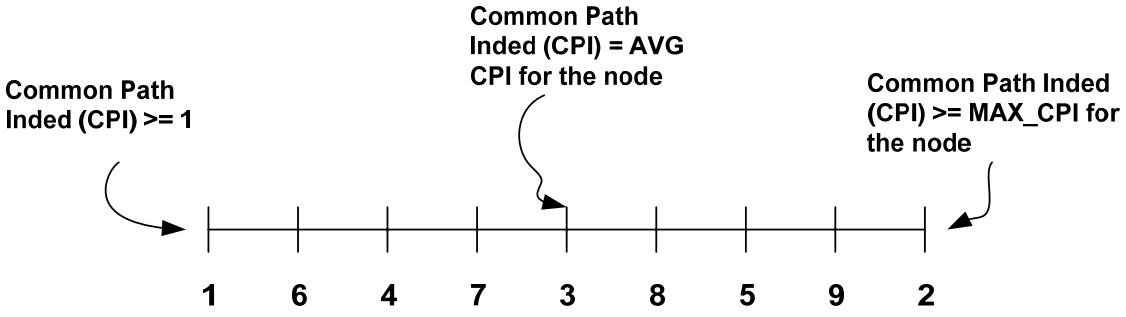

Figure 14 - Clustering Analysis

The number of destination clusters created for each of the 278 nodes was used to compute the average number of destination clusters shown in Table 1. Similarly, the same data was used to determine the maximum number of destination clusters created for each technique.

Clustering technique	1	6	4	7	3	8	5	9	2
Avg. Destination Clusters	1	4	5	5	7	9	14	19	41
Max. Destination Clusters	46	113	116	106	136	187	220	226	156

Table 1- Destination Cluster Data

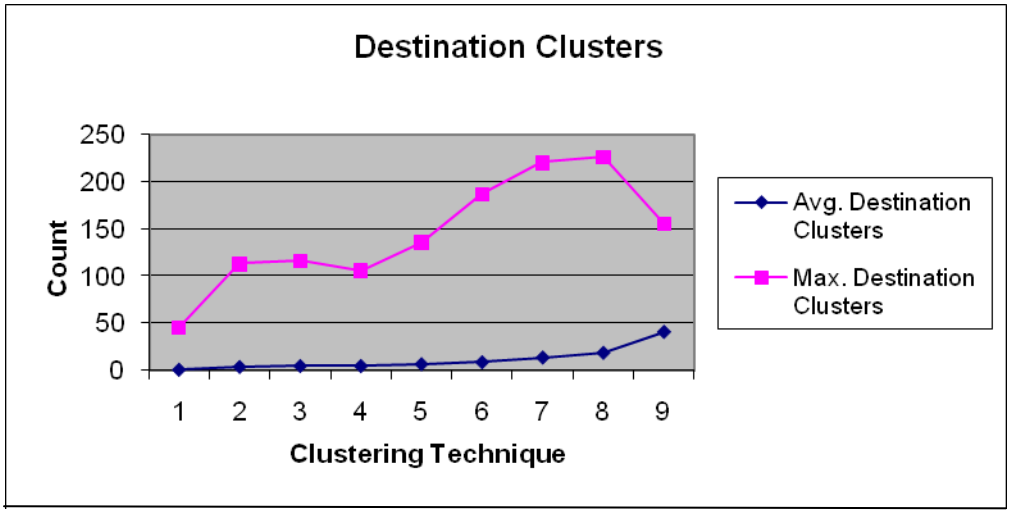

Figure 15 – Destination Clustering Analysis

5.5.2.2 Source Cluster Analysis

Similar to the destination clusters, the number of source clusters created for each of the 278 nodes was used to compute the average number of source clusters shown in Table 2. Also, the same data was used to determine the maximum number of destination clusters created for each technique.

Clustering technique	1	6	4	7	3	8	5	9	2
Avg. Source Clusters	1	19	24	22	30	32	38	44	63
Max. Source Clusters	61	61	65	66	66	68	105	117	166

Table 2 - Source Cluster Data

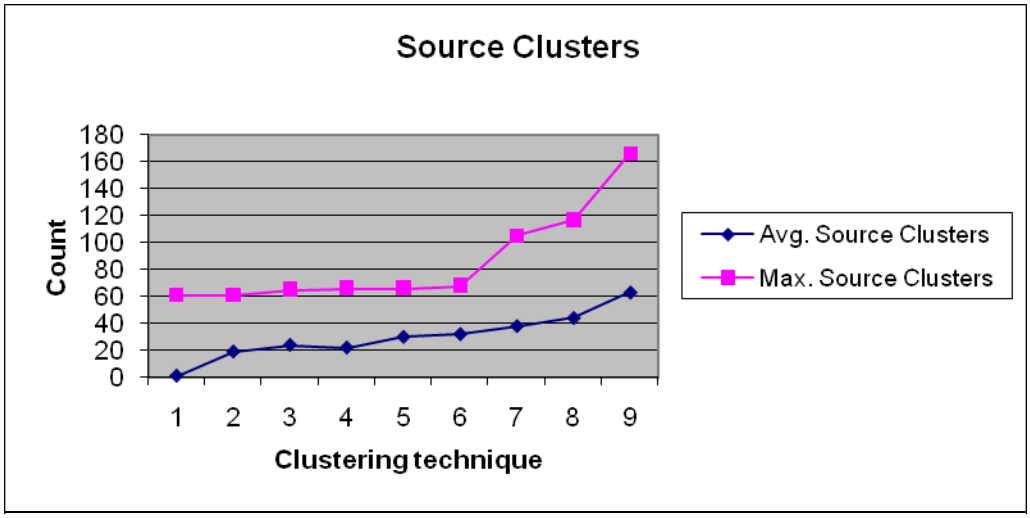

Figure 16 - Source Cluster Analysis

5.5.2.3 Conclusion – Cluster Analysis

From the results for both the source and destination cluster analysis, we see that the average number of source and destination clusters created increased linearly as expected from the most conservative technique, #1 to the most restrictive technique, #9. This information can be used to determine the best technique suitable for a node. Depending on the memory availability on the node, the node may decide to choose one that creates fewer clusters.

5.5.3 Available Bandwidth Inference Measurements

The algorithm described in this paper was tested on PlanetLab (PlanetLab: Global Research Network) and Emulab (Emulab - Network Emulation Testbed, 2002) networks as they provide a geographically distributed platform suitable for this project. The tests included executing the AB inference engine for different nodes located in US and

Europe. The Java application accepted configuration parameters that allowed the user to choose the test node, the clustering technique and the number of measurements to be performed. Clustering technique #3, with average CPI was chosen as the clustering method of these tests. A set of five nodes were chosen based on their geographical location to get a good representation of all the nodes as we are testing a node-centric algorithm. We discuss the results for two of these nodes in this paper.

5.5.3.1 Node Results: planetlab1.xeno.cl.cam.ac.uk

A subset of AB inference measurements for node planetlab1.xeno.cl.cam.ac.uk is shown in Figure 17. For each node, the inferred AB value was computed and compared against the actual value to determine its deviation.

Destination	Actual AB	Inferred AB	Deviation
Planetlab1.ie.cuhk.edu.hk	24.254921	26.68211	-10.007
planet1.zib.de	20.964064	22.809776	-8.80417
plab1-itec.uni-klu.ac.at	16.052383	17.444479	-8.67221
ent1.cs.nccu.edu.tw	8.046409	8.607523	-6.97347
Planetlab1.isi.jhu.edu	27.019152	28.900839	-6.96427
Planetlab1.cs.stevens-tech.edu	25.016403	26.735878	-6.87339
planet2.cs.ucsb.edu	22.918459	24.481089	-6.81822
Planetlab04.cs.washington.edu	4.234247	4.488182	-5.99717
Planetlab2.csres.utexas.edu	23.555752	24.886576	-5.64968
planetlab-01.naist.jp	26.833254	28.292778	-5.43924
planet2.l3s.uni-hannover.de	22.517841	23.686703	-5.19083
planetlab2.elet.polimi.it	23.176973	24.347116	-5.04873
pl4.planetlab.uvic.ca	16.74947	17.570864	-4.904
planetlab-02.naist.jp	24.847546	26.038326	-4.79234
planetlab1.eecs.wsu.edu	26.709494	27.957735	-4.6734
planetlab1.cs.purdue.edu	25.714848	26.90719	-4.63678
planetlab1.een.orst.edu	25.650787	26.756716	-4.31148
planetlab1.ceid.upatras.gr	14.30161	14.838045	-3.75087
planetlab11.millennium.berkeley.edu	16.575321	17.170305	-3.58958
planet-lab1.ufabc.edu.br	23.674725	24.453465	-3.28933
planetlab4.cse.nd.edu	4.916461	5.071445	-3.15235

planetlab1.uc.edu	23.73589	24.281046	-2.29676
planetlab02.erin.utoronto.ca	21.4526	21.890673	-2.04205
planetlab3.mini.pw.edu.pl	2.857143	2.891776	-1.21215
planetlab1.cslab.ece.ntua.gr	27.536064	27.674194	-0.50163
planetlab3.xeno.cl.cam.ac.uk	25.465683	25.536839	-0.27942
planet1.l3s.uni-hannover.de	24.052637	23.909071	0.596883
planetlab4.inf.ethz.ch	5.52935	5.477408	0.939387
planetlab-01.ece.uprm.edu	27.836956	27.443813	1.412306
planetlab1.ewi.tudelft.nl	24.059845	23.709812	1.454843
planetlab-5.cs.princeton.edu	28.467436	27.777857	2.422343
planetlab-03.naist.jp	29.014782	27.502525	5.212023
planetlab1.ics.forth.gr	27.76452	26.308847	5.242925
planetlab1.elet.polimi.it	26.719261	25.275772	5.402429
planetlab1.sfc.wide.ad.jp	27.599812	26.072556	5.533574
planetlab3.hiit.fi	3.887295	3.665266	5.711658
planetlab1.dtc.umn.edu	27.698046	26.111338	5.728592
planetlab-1.di.fc.ul.pt	26.748945	25.152641	5.967727
planetlab1.cse.nd.edu	27.20229	25.52785	6.155511

Figure 17 – AB Inference Measurement Subset (planetlab1.xeno.cl.cam.ac.uk)

Of the 278 available nodes, only 47% or 133 measurements were successful because either the end node was down for maintenance because of which the actual value was unavailable or the clusters heads were down because of which the inferred value was unavailable. Of these successful results, 80.45% of the inferred values lie within $\pm$ 50% of the actual value.

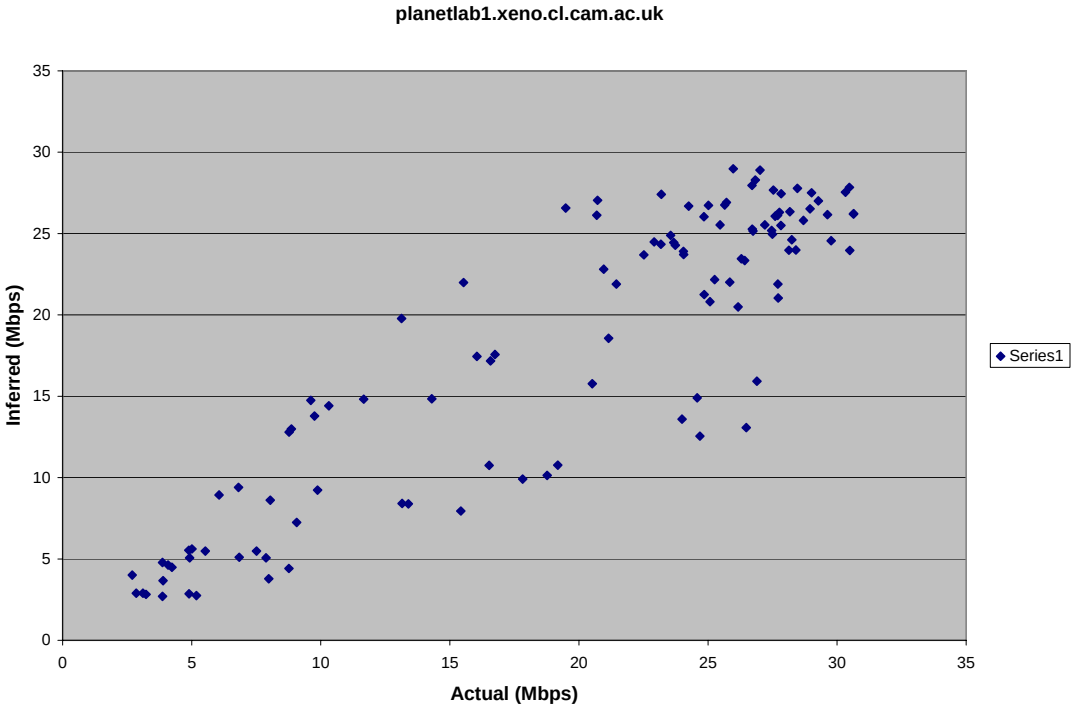

Figure 18 - Actual vs. Inferred Available Bandwidth (planetlab1.xeno.cl.cam.ac.uk)

We plot the actual and inferred values of AB in Figure 18. The results are scattered but we see that the actual values are clustered around two points, 5Mbps and 30Mbps, and in these cases, the inferred value closely matches the actual value. Figure 19 is the plot of the cumulative distribution function.

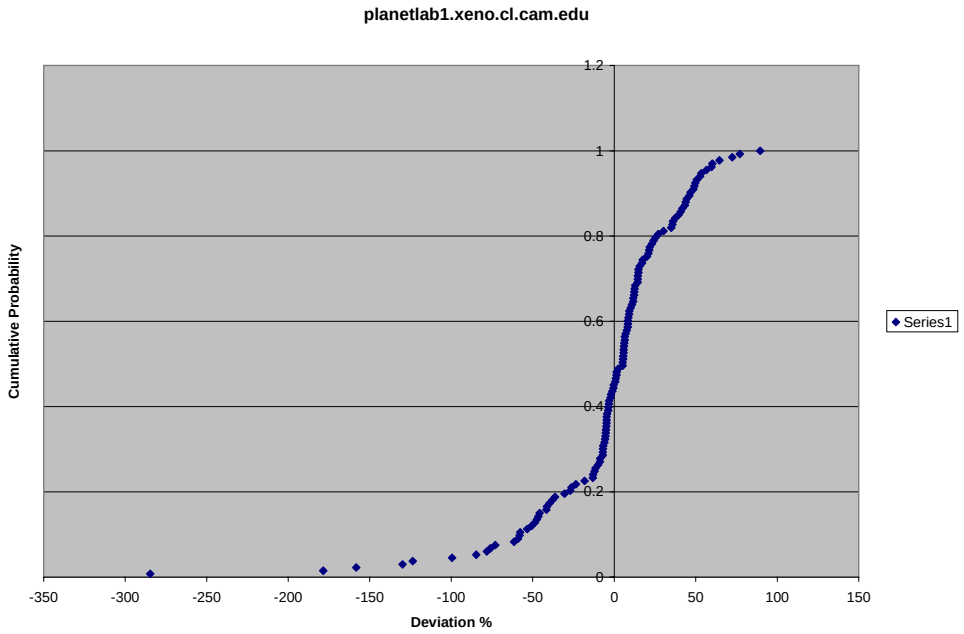

Figure 19 - CDF of Deviation in Inferred AB (planetlab1.xeno.cl.cam.ac.uk)

Spearman Rank Order Correlation Coefficient

The correlation coefficient is a number that can be used to determine the strength of association between two variables. We use the Spearman rank order correlation coefficient to determine the association between the actual AB and the inferred AB values.

To determine the Spearman's rank correlation coefficient, we rank both the actual AB and the inferred AB values from the node **planetlab1.xeno.cl.cam.ac.uk** to all the other nodes in the network in ascending order. Let the actual rank and inferred ranks of an i^{th} pair of nodes with actual AB a_i and inferred AB i_i be r_i^a and r_i^i. The Spearman's rank correlation (Wiki: Spearman's rank correlation coefficient, 2008) can be computed using the equation shown below :

$$r_s = 1 - \left(\frac{6 \sum d^2}{n^3 - n} \right)$$

Where Σ = summation,
d = (r_i^a - r_i^i) and
n = number of measurements

The value of r_s will be between ±1, where a negative value indicates strong negative correlation and a positive value indicates a strong positive correlation.

Interpreting Spearman's Rank Correlation Coefficient

The spearman's coefficient, r_s, is compared against the critical values shown in Figure 20. The value N is the number of pairs of values used to compute the coefficient and the values 0.05, 0.02 and 0.01 indicate the significance level. For ex: if r_s = 0.71 with N=16, then the value r_s is likely to occur by chance less than 1 out of 100 attempts indicating a strong correlation between the pair of values used to compute r_s.

N (the number of pairs of values):	*0.05*	*0.02*	*0.01*
5	1	1	
6	0.886	0.943	1
7	0.786	0.893	0.929
8	0.738	0.833	0.881
9	0.683	0.783	0.833
10	0.648	0.746	0.794
12	0.591	0.712	0.777
14	0.544	0.645	0.715
16	0.506	0.601	0.665
18	0.475	0.564	0.625
20	0.45	0.534	0.591
22	0.428	0.508	0.562
24	0.409	0.485	0.537
26	0.392	0.465	0.515
28	0.377	0.448	0.496
30	0.364	0.432	0.478

Figure 20 - Critical values for r_s (Wiki: Rhotable)

The table in Figure 21 shows a subset of the rank ordering and difference 'd' calculations for some of the measurements shown in Figure 17.

Actual Rank	Inferred Rank	D1	D2
21	98	-77	5929
1	30	-29	841
3	29	-26	676
39	79	-40	1600
37	73	-36	1296
41	69	-28	784
5	22	-17	289
50	123	-73	5329
48	94	-46	2116
35	60	-25	625
32	52	-20	400
30	51	-21	441
59	131	-72	5184

Figure 21 - Spearman's Rank Order Correlation Coefficient Subset (planetlab1.xeno.cl.cam.ac.uk)

The total number of measurement, n = 133 and **Σ** is 94673. Hence the value computed, r_s **= 1-(6*94673 / (133*(133*133-1))) = 0.7585.** Based on the information in Figure 20, the value of 0.7585 suggests a fairly strong positive correlation between the actual and inferred AB for the node **planetlab1.xeno.cl.cam.ac.uk.**

5.5.3.2 Node Results: vn1.cs.wustl.edu

The results for node, vn1.cs.wustl.edu, is briefly discussed in this section. A subset of measurements for this node is shown in Figure 22.

Destination	Actual AB	Inferred AB	Deviation
plab-2.sinp.msu.ru	5.567828	6.015182	-8.03462
planetlab1.cse.nd.edu	4.5641	4.913298	-7.65097
planetlab5.cs.duke.edu	4.264096	4.565757	-7.07444
planetlab2.byu.edu	7.183977	7.650455	-6.49331
planetlab-3.imperial.ac.uk	6.408414	6.799259	-6.09893
phil.cc.vt.edu	6.426438	6.77126	-5.36568
planetlab06.mpi-sws.mpg.de	5.408039	5.690492	-5.22284
planetlab1.cis.upenn.edu	6.260821	6.563991	-4.84234
planetlab03.cnds.unibe.ch	5.900004	6.158718	-4.38498
planetlab1.fit.vutbr.cz	6.289565	6.516424	-3.60691
planetlab2.inf.ethz.ch	5.610929	5.775923	-2.94058
planetlab-4.cs.princeton.edu	6.480654	6.620143	-2.15239
planetlab04.cnds.unibe.ch	4.439945	4.462431	-0.50645
planetlab1.eecs.wsu.edu	5.602197	5.61886	-0.29744
node-1.mcgillplanetlab.org	5.114754	5.129586	-0.28998
mars.planetlab.haw-hamburg.de	4.306139	4.307919	-0.04134
planetlab4.flux.utah.edu	6.302859	6.283828	0.301942
vn3.cs.wustl.edu	106.26028	105.933716	0.307325
planetlab1.eecs.jacobs-university.de	6.572195	6.536201	0.547671
planetlab01.cnds.unibe.ch	6.619369	6.566646	0.796496
vn2.cs.wustl.edu	105.84175	104.47282	1.293374
planetlab1.unl.edu	6.224894	6.112179	1.810714

Figure 22 - AB Inference Measurement Subset (vn1.cs.wustl.edu)

In this case, of the 278 nodes, only 25% or 72 measurements were successful. Of these successful results, 90.27% of the inferred values lie within $\pm$50% of the actual value. Figure 23 and Figure 24 show the actual vs. inferred and the CDF respectively.

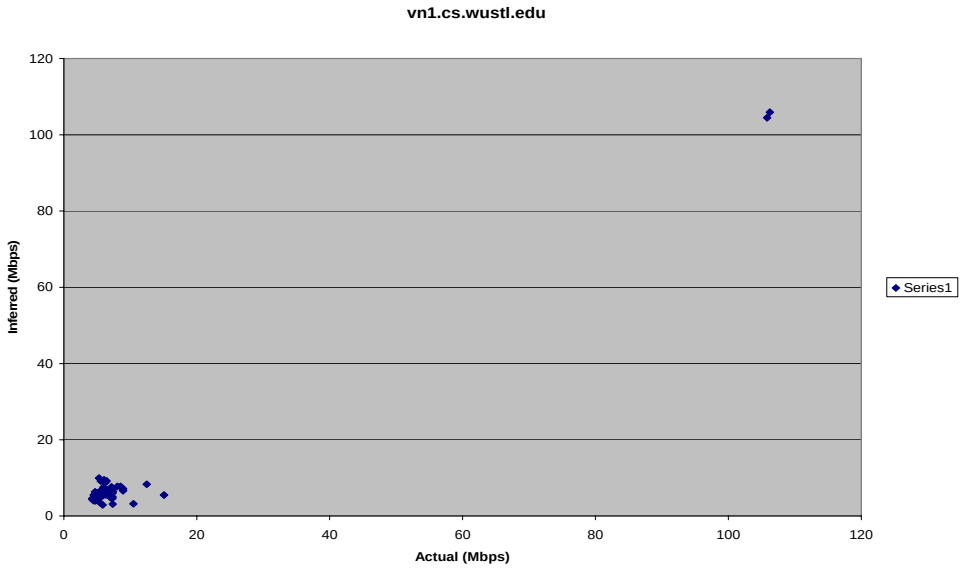

Figure 23 - Actual vs. Inferred Available Bandwidth (vn1.cs.wustl.edu)

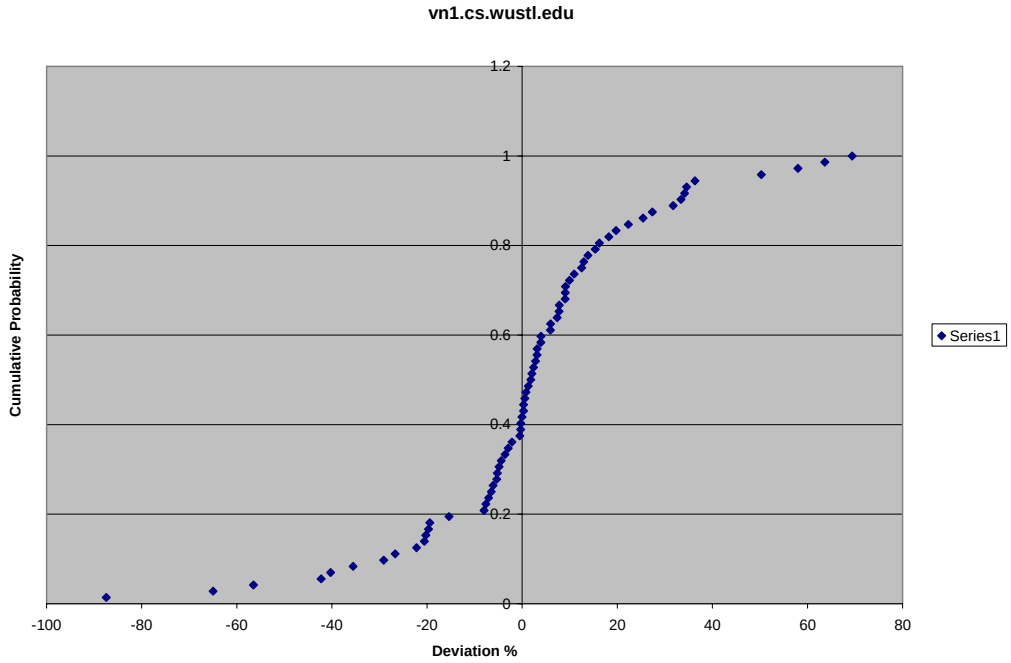

Figure 24 - CDF of Deviation in Inferred AB (vn1.cs.wustl.edu)

Spearman Rank Order Correlation Coefficient

The table in Figure 25shows a subset of the rank ordering and difference 'd' calculations for some of the measurements shown in Figure 22.

Actual Rank	Inferred Rank	D1	D2
16	70	-54	2916
20	67	-47	2209
35	69	-34	1156
45	68	-23	529
36	66	-30	900
8	43	-35	1225
9	37	-28	784
33	61	-28	784
4	26	-22	484
14	38	-24	576
7	27	-20	400
22	54	-32	1024
27	58	-31	961
32	57	-25	625

Figure 25 - Spearman's Rank Order Correlation Coefficient Subset (vn1.cs.wustl.edu)

The total number of measurement, n = 72 and **Σ** is 37100. Hence the value computed, r_s **= 1-(6*37100 / (72*(72*72-1))) = 0.403499.** Based on the information in Figure 20, the value of 0.403499 suggests a fairly strong positive correlation between the actual and inferred AB for the node vn1.cs.wustl.edu**.**

6 Related Work

Estimating end-to-end bandwidth is challenging because of its dynamic nature and a number of tools have been developed to measure it. In a comparison study of bandwidth measurement tools, bandwidth estimation experiments were conducted on a high-speed testbed using publicly available bandwidth estimation tools (Alok Shriram M. M., 2005). The different tools included: abing, pathchrip, pathload and Spruce. The accuracy and operational characteristics of these tools and the factors that impact the tools performance are analyzed. The authors concluded that pathload and pathchirp are the most accurate tools for their experiments.

While estimating AB is challenging, inferring end-to-end AB is more interesting and has a wide range of applications. End-to-End AB is dependent on the available bandwidth along the links that form the path. The bottleneck link, the one with the smallest residual bandwidth is also the weakest link that determines the AB of the entire path. The authors of BRoute claimed that the bottleneck links are primarily the links near the end hosts termed edge segments, and hence only measured the AB on these links to estimate the bandwidth of all paths (Ningning Hu, 2005). BRoute proposed two modes for collecting end segment bandwidth, an infrastructure mode and a peer-to-peer mode. The former used landmarks to which all nodes perform measurements or decide a subset of paths to measure. However, each bandwidth landmark can support only a limited number of nodes. The peer-to-peer mode is designed such that the nodes perform AB measurements in a co-operative fashion. This method scales better than the infrastructure

mode as this decentralizes the measurement process but frequent route changes makes the measurements more complicated. A study by (Alok Shiram, 2003) showed that identifying potential bottlenecks for each path based on links with minimum available bandwidth leads to false positives.

Research by (Alok Shriram S. B., 2007) describes three scalable algorithms with decreasing probe overhead. The algorithm are based on end-to-end AB measurements over a subset of nodes in the network as opposed to AB measurements over last hop links as described in previous approaches. The crux of the algorithms is to group together nodes that are likely to share bottleneck links and to select well provisioned head nodes for each node's cluster. The AB measurements are performed from each head-node to nodes outside the cluster and the AB from other members of the cluster is then inferred using the measurement from the head-node. The techniques described in (Alok Shriram S. B., 2007) are evaluated on PlanetLab using the scalable sensing network service.

This paper takes one step further in developing a scalable AB inference algorithm that is also distributed. By distributing the computation across the various nodes in the network, the actual number of AB measurements is reduced and hence the computation time. The tests are performed using the current AB estimation tools that are more accurate compared to tools developed in the past. The combination of current AB estimation tools and a distributed algorithm for inferring AB has resulted in 80% of the values within a deviation of $\pm$50% for some nodes.

7 Conclusion

The primary goal of this project was to reduce the total number of AB measurements in a large network and at the same time lower the error rate on AB inference compared to existing techniques. The results on the PlanetLab network were very promising but not stellar. Since the inference algorithm described in this paper is node-centric, the results were mixed based on tests conducted on a set of nodes. The number of successful measurements was only around 50% after repeated attempts because of network topology problems. For few nodes, 80% of the measurements were in the deviation range of $\pm$ 50% which matches the results from an existing inference algorithm described in (Ningning Hu, 2005). However, for some of the nodes, there was a weak correlation between the actual AB value and the inferred AB value. Since the algorithm is node-centric, the weak correlation is observed for some nodes and is related to the selection of cluster heads. The correlation can be improved by adapting a different clustering technique for each node. Since the technique is based on distributed computing, the solution is highly scalable and seamlessly integrates new nodes that are added to the network. The results on the PlanetLab testbed were promising for some nodes and hence the engine can be deployed by websites that distribute content from multiple geographical locations.

7.1 Future Work

The AB inference algorithm described in this paper is a first step towards providing a distributed and scalable AB estimation solution. As mentioned earlier, the results look

promising for certain nodes and there is scope for improvements for nodes that had weak correlation between the actual and inferred values. First, this paper proposes various clustering techniques that are tuned based on the common path index between all nodes in the network. For the case study on PlanetLab, one of the techniques was chosen for all nodes. However, since the algorithm is node-centric further investigations are needed to evaluate the best clustering technique for each node. This may require some trial and error method to find the best cluster head(s) using the techniques described in this paper until the inference error is significantly reduced.

Second, the clustering method discussed here determines the cluster head and then forms clusters around the cluster head. An alternative approach that needs further investigation is to first group nodes that share a common metric and then select a cluster head that is superior compared to other nodes in terms of the chosen metric. This could result in better cluster formation thus leading to reduction in inference error.

Finally, the clustering algorithm discussed in this paper is a variant of K-means, one of the partitioning algorithms. The software developed for this project is modular and designed to dynamically swap the clustering algorithm. Future research in this field can take advantage of this feature by reusing the software to test other clustering algorithms in an attempt to reduce the inference error.

References

Alok Shiram, J. K. (2003). Identifying Bottleneck Links Using Distributed End-to-end. Available Bandwidth Measurements. *ISMA 2003 Bandwidth Estimation Workshop.* San Diego: ISMA 2003.

Alok Shriram, M. M. (2005). Comparison of Public End-to-End Bandwidth Estimation Tools on High Speed Links. *Passive and Active Network Measurement* (pp. 306-320). Boston: SpringerLink.

Alok Shriram, S. B. (2007). *Scalable End to End Available Bandwidth Inference.*

Bill Cheswick, H. B. (2000). Mapping and visualizing the Internet. *Proceedings of the General Track: 2000 USENIX Annual Technical Conference* (pp. 1-12). San Diego: USENIX 2000.

Boa Web Server. (2005, Febraury 23). Retrieved 11 12, 2008, from http://www.boa.org/: http://www.boa.org/documentation/boa-1.html#ss1.1

Dilley, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman, R., & Weihl, B. (2002). Globally distributed content delivery. *Internet Computing, IEEE* (pp. 50 - 58). IEEE Internet Computing Magazine .

Dovrolis, C. (n.d.). *Pathload tutorial*. Retrieved 12 19, 2008, from www.cc.gatech.edu/: www.cc.gatech.edu/fac/Constantinos.Dovrolis/pathload_tutorial.html

Emulab - Network Emulation Testbed. (2002 , October 1). Retrieved 9 21, 2008, from http://www.emulab.net/: http://www.emulab.net/

Fiuczynski, M. E. (2006). PlanetLab: overview, history, and future directions. *ACM SIGOPS Operating Systems Review* (pp. 6-10). New York: ACM.

Jones, A. B. (2000, September). RFC2922 - Physical Topology MIB. U.S.A.

Ken Keys, C. A. (2009). *IP Alias Resolution Techniques.* San Diego: IP Alias Resolution Techniques.

Lee, S.-J., Sharma, P., Banerjee, S., Basu, S., & Fonseca, R. o. (2005). Measuring Bandwidth between PlanetLab Nodes. *Passive and Active Network Measurement* (pp. 292-305). Boston: SpringerLink.

Matteucci, M. (n.d.). *Tutorial: Clustering*. Retrieved Oct 5, 2008, from http://home.dei.polimi.it/: http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/

Neil Spring, L. P. (2006). Using PlanetLab for network research: myths, realities, and best practices. *ACM SIGOPS Operating Systems Review* , 17-24.

Ningning Hu, P. S. (2005). Exploiting internet route sharing for large scale available bandwidth estimation. *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement* (pp. 16-16). Berkeley: USENIX Association.

pathChirp: Efficient Available Bandwidth Estimation for Network Paths. (2003). Retrieved 2 3, 2009, from http://www.spin.rice.edu/Software/pathChirp/: http://www.spin.rice.edu/Software/pathChirp/

Pathneck, Ningning Hu (CMU). (2004). Retrieved 2 4, 2009, from http://www.cs.cmu.edu/~hnn/pathneck/: http://www.cs.cmu.edu/~hnn/pathneck/

Paul Albitz, C. L. (2001). *DNS and Bind.* O'Reilly.

PlanetLab: Global Research Network. (n.d.). Retrieved October 2008, from http://www.planet-lab.org/: http://www.planet-lab.org/

Praveen Yalagandula, P. S.-J. (2006). S3: A Scalable Sensing Service for Monitoring Large Networked Systems. *Proceedings of the 2006 SIGCOMM workshop on Internet network management* (pp. 71-76). Pisa, Italy : ACM.

R. Siamwalla, R. S. (July 1998). *Discovering Internet Topology.*

Shamir, E. H. (March 10, 1999). *A Clustering Algorithm based on Graph Connectivity.*

Shriram, A. K. (2007). Empirical Evaluation of Techniques for Measuring Available Bandwidth. *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE* (pp. 2162-2170). Anchorage: IEEE.

Song, H. H., & Yalagandula, P. (Jan 2007). Real-time End-to-end Network Monitoring in Large Distributed Systems.; . 2nd International Conference on 7-12 Jan. 2007 Page(s):1 - 10. *Communication Systems Software and Middleware* , 1-10.

Vinay J. Ribeiro, R. H. (2003). pathChirp: Efficient Available Bandwidth Estimation for Network paths. *SLAC-PUB-9732* .

Wiki: Data clustering. (2008, September 12). Retrieved Nov 14, 2008, from http://en.wikipedia.org: http://en.wikipedia.org/wiki/Data_clustering

Wiki: Rhotable. (n.d.). Retrieved March 2, 2009, from http://www.sussex.ac.uk/: http://www.sussex.ac.uk/Users/grahamh/RM1web/Rhotable.htm

Wiki: Spearman's rank correlation coefficient. (2008, August). Retrieved Feb 2009, from http://en.wikipedia.org/wiki/Spearman's_rank_correlation_coefficient: http://en.wikipedia.org/wiki/Spearman's_rank_correlation_coefficient

Wiki: Traceroute. (n.d.). Retrieved June 18, 2009, from http://en.wikipedia.org/wiki/Traceroute: http://en.wikipedia.org/wiki/Traceroute