

Overlay network assignment in PlanetLab with NetFinder

Yong Zhu and Mostafa Ammar
Networking and Telecommunications Group
Georgia Institute of Technology, Atlanta, Georgia
Email: {yongzhu,ammar}@cc.gatech.edu

Abstract—PlanetLab has been widely used in the networking community to test and deploy user-defined overlays. Serving as a meta testbed to support multiple overlay networks, PlanetLab has significantly lowered the barriers to build new overlays. However, PlanetLab users always face the problem of selecting a set of nodes and interconnecting them to form the desired overlay network. Unfortunately, such a task is usually carried out manually by individual users and sometimes in an ad-hoc manner. In this paper, we develop NetFinder, an automatic overlay network configuration tool to efficiently allocate PlanetLab resources to individual overlays. NetFinder continuously monitors the resource utilization of PlanetLab and accepts a user-defined overlay topology as input and selects the set of PlanetLab nodes and their interconnection for the user overlay. Experimental results indicate that overlay networks constructed by NetFinder have more stable and significantly higher bandwidth than alternative schemes and near optimal available CPU.

I. INTRODUCTION

PlanetLab has been widely used in the networking community to test and deploy new network technologies [1]. It can serve as a testbed for overlay networks. Research groups are able to request a PlanetLab slice in which they can experiment with a variety of wide-area networks and services, such as file sharing, content distribution networks, routing and multicast overlays, QoS overlays, and network measurement tools. One problem faced by PlanetLab users is selecting a set of PlanetLab nodes and interconnecting them to form the desired overlay network.

PlanetLab also serves as a meta testbed on which multiple, more narrowly-defined virtual testbeds can be deployed. For example, the “Internet-in-a-Slice”(IIAS) service aims at recreating the Internet’s data plane and control plane in a PlanetLab slice [2]. Network researchers can use this infrastructure to experiment with modifications and extensions to the Internet’s protocol suite. Currently, IIAS does not provide resource discovery and overlay assignment services and the overlay configuration has to be given explicitly.

Having an automated overlay network assignment service for PlanetLab is important for two reasons. From the PlanetLab operator’s perspective, efficient assignments would result in better resource utilization so that the PlanetLab could accommodate more user overlays with limited resources and avoid hot spots or congestion. From the overlay user’s perspective, having good assignment service would provide overlay users better choices in building their overlays by avoiding congested links/nodes.

Unfortunately, such a task is usually carried out manually by individual users and sometimes in an ad-hoc manner. Manual configuration is time consuming (especially for networks) and prone to human error. It is not efficient even for small overlay networks since the underlying PlanetLab network has hundreds of nodes and its state fluctuates over time due to failures, congestion and load variation. In this work, we are motivated by the above considerations and focus on the on-demand overlay network assignment problem: Upon the arrival of a user-defined overlay topology and associated overlay resource requirements, find the set of PlanetLab nodes and their interconnections to satisfy the user request.

The set of physical resources provided by PlanetLab to support multiple overlays can be divided into two categories: node resources and link/path resources. Node resources such as memory, disk space, number of active slices, and available CPU are all related to a particular PlanetLab node. They enable individual overlays to compute, modify and store their data. In contrast, link/path resources enable an overlay network to send data from one node to another. Common metrics for link/path resources include available bandwidth, delay and loss rate, which indicate how well the user can send data within its overlay.

A recent study reported that available resources vary significantly in PlanetLab, suggesting that wise placement of application instances can be beneficial [3]. Among different node resource metrics, available CPU is the most important one since most users will suffer if the CPU resources are poorly utilized. Unlike the link performance metrics such as delay, loss rate or TCP throughput, available bandwidth directly indicates the extra amount of traffic that can be carried by the link before it becomes congested. Based on these considerations, we focus on the CPU usage and available bandwidth. However, our service model can be easily extended to support other metrics.

There are two challenges in providing an automatic overlay network assignment service: the lack of PlanetLab bandwidth monitoring services and the difficulties in developing a good overlay network assignment algorithm. Despite the growing literature and tools for end-to-end bandwidth measurement, many tools have practical limitations and do not work well in PlanetLab. Even for tools that work within PlanetLab such as Pathload [4] and Iperf [5], running pair-wise measurements between hundreds of Planetlab nodes is a challenging task.

Another challenge comes from the dual objective of the overlay assignment. Specifically, upon the arrival of a overlay request, we want to achieve load balancing on both PlanetLab nodes and links. A special case of the this problem can be formulated as an unsplittable flow problem which is NP-hard. Therefore, the overlay assignment problem is intrinsically difficult and heuristics will be used to solve the problem.

In this paper, we develop NetFinder, an automatic overlay network configuration tool in PlanetLab. This tool continuously collects information about the resource utilization of PlanetLab and accepts a user-defined overlay topology as input. NetFinder then performs overlay network assignment by selecting the set of PlanetLab nodes and their interconnection for the desired user overlay.

The rest of this paper is organized as follows. Section II discusses the related work. Section III gives the network model and presents the detailed methodology used in NetFinder. The implementation is described in Section IV. Performance evaluation is presented in Section V and the paper is summarized in Section VI.

II. RELATED WORK

Our overlay network configuration tool is motivated by the findings in [3], which examines PlanetLab resource utilization data from the perspective of the designer of a resource discovery system. The authors find that, for some resources, the quantity available at a fixed time differs significantly across nodes, suggesting a potential benefit to using a resource discovery system to wisely place application instances.

NetFinder is based on two technical foundations: end-to-end bandwidth/throughput measurement results and overlay network assignment algorithms. Lee *et al.* assess the capabilities of several existing bandwidth measurement tools and describe the difficulties in choosing suitable tools as well as using them on PlanetLab [6]. Specifically, the authors report that pathchar [7], pchar [8], pathChirp [9] do not work with the current PlanetLab. bprobe, cprobe [10] and SProbe [11] can partially run on PlanetLab.

There are a number of existing tools for node selection in PlanetLab. SWORD is a scalable resource discovery tool for wide-area distributed systems [12]. The particular type of resource that SWORD is intended to discover is the set of nodes on which to deploy a service. However, SWORD focuses on selecting the set of nodes that satisfy user requirement without considering the effects of the selection on the overall PlanetLab performance. Furthermore, SWORD has little support on the inter-node performance and currently does not support available-bandwidth performance between nodes.

Our work is based a number of PlanetLab performance monitoring services. Specifically, CoMon provides a monitoring statistics for PlanetLab at both a node level and a slice level [13]. The data gathered by CoMon can be directly accessed through CoMon daemon on port 3121 at each PlanetLab node. Our link performance results are currently obtained from the S^3 project [14], which periodically provides

end-to-end latency, bottleneck bandwidth capacity, and end-to-end available bandwidth between 400+ PlanetLab nodes. We should note that the contribution of this work is to the NetFinder tool that configure the user-defined overlay network in a efficient and on-demand manner. Instead of developing new measurement schemes for PlanetLab, we rely on existing services to collect measurement data. Although the current service using the measurement results form CoMon and S^3 , our service is not restricted by these services and it can easily adopt new monitoring services and measurement results when they are available.

The overlay assignment problem is challenge since we are trying to optimize both the node performance and link performance. The algorithm used in NetFinder is based on our previous work on the virtual network assignment [15], which developed theoretical algorithms for virtual network assignment aiming at reducing the stress among substrate nodes and links [15]. The abstract definition of stress can not be easily mapped to realistic network performance metrics such as the CPU usage and available bandwidth. Therefore, modifications and extensions need to be made for the algorithm to work in our scenario.

The output of NetFinder could be used directly as IAS inputs [2]. Which in turn would start an overlay data plane for PlanetLab following the calculated results.

III. NETWORK ARCHITECTURE

PlanetLab is composed of a number of nodes connected to the Internet at different locations. Therefore, there is an end-to-end native Internet path connecting each pair of PlanetLab nodes. From the view point of PlanetLab users, the only resources available are those at individual Planetlab nodes and along paths between PlanetLab nodes. Network situation on all other links and nodes are transparent to PlanetLab users. Therefore, the PlanetLab network itself can be viewed as a full-mesh overlay on top of the native Internet.

The PlanetLab network in turn serves as a shared substrate to support multiple user-defined overlays. Each user would treat the underlying PlanetLab network just like any physical network.

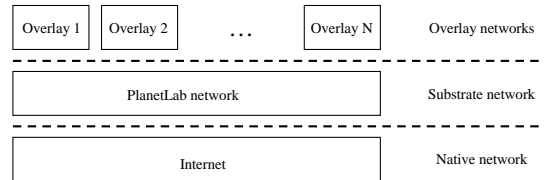


Fig. 1. PlanetLab network layered architecture

Based on the above consideration, we model the PlanetLab overlay network configuration using three layers of network: a native Internet, a PlanetLab substrate network and a number of user-defined overlay networks. The lowest layer is the native Internet composed of native routers and links. On top of that, there is a common PlanetLab substrate network, composed of PlanetLab nodes and the interconnection between them.

Finally, a number of user-defined overlays are built on top of the PlanetLab substrate such that each overlay node is a substrate node and each overlay link is a substrate path, which could, in turn, be a concatenation of a set of Internet paths. In this work, we assume that all PlanetLab nodes are already deployed and their interconnection are determined by native IP routing mechanisms, and focus only on issues between the substrate layer and user overlays.

The flowchart of NetFinder is illustrated in Figure 2. In the background, the system periodically collects the PlanetLab resource utilization information including both available bandwidth data and available CPU data from external sources. NetFinder then accepts user-defined overlay requests. When the user requests include constraints such as the minimum CPU or minimum bandwidth requirements, infeasible parts of the PlanetLab network are removed through preprocessing. Finally, the overlay assignment algorithm is used to calculate the overlay configuration.

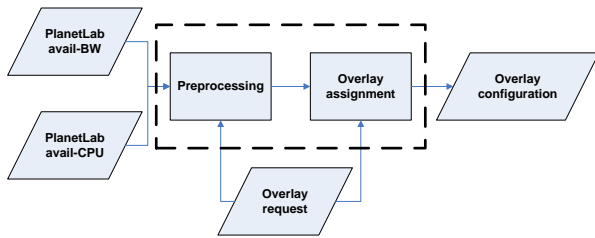


Fig. 2. System diagram

The rest of this section discusses each component of the system in detail.

A. Overlay network request format

To setup an overlay network in the PlanetLab, a user submits a request specifies the topology of the overlay network and a set of resource constraints. An example to setup a 4 node 8 link overlay network is shown in Table I. The request also indicates that each overlay link should have minimum available bandwidth of 10Mbps and each selected PlanetLab node should have at least 20% free available CPU resource.

TABLE I
OVERLAY REQUEST FORMAT

# NUM_NODES, MIN_BW MIN_CPU	4, 10 20
# LINKS: SRC DEST	0 1 1 0 0 2 2 0 0 3 3 0 1 2 2 1

B. PlanetLab substrate performance monitoring

NetFinder collects measurement results from other reporting services. It then preprocesses and stores the resulting data into its own database. The current version of NetFinder obtains node CPU usage information by periodically polling the CoMon daemon on each PlanetLab node. This probing also serves as a failure detection mechanism, where NetFinder determines that a node is down if the probe times out.

The bandwidth measurement in PlanetLab is a challenging problem due to the scale of the network and practical limitations of various tools. NetFinder uses the bandwidth measurement result from the S^3 project, which reports the all-pair available bandwidth measurement results of 400+ PlanetLab nodes every four hours. NetFinder then processes the raw measurement results by removing all inactive nodes and links and keeping only relevant bandwidth and CPU data.

Although NetFinder relies on the PlanetLab measurement results, it is not restricted by any specific tools or reporting services. We choose S^3 results since it is the only all-pair bandwidth measurement service available for the large scale PlanetLab network. However, NetFinder has the flexibility to adapt to measurement results of new techniques when they become available in the future. The only thing that needs to be done is to convert the format from the raw measurement data to NetFinder's internal data format.

C. Overlay network assignment

NetFinder's overlay network assignment is based on our recent work on network virtualization. The problem of assigning PlanetLab nodes/paths to the overlay network is similar to the virtual network (VN) assignment problem [15]. However, to be able to work with PlanetLab, we need the following modifications and enhancements:

- Using bandwidth as the Link-state metrics instead of substrate link stress: Unlike the scenario of VN assignment in [15], where stress is used to model the utilization of the substrate node and substrate link, the meaning of stress in the PlanetLab context is not obvious.
- Constraints in overlay network assignment: The overlay assignment algorithm should consider a number of constraints in making the assignment decision. Basic constraints include bandwidth constraints (e.g., the available bandwidth of the overlay link should exceed 10Mbps) and CPU constraints.

Based on the above consideration, we modify our original VN assignment algorithm as follows:

1) *Preprocessing*: After obtaining the measurement results, given an overlay request with bandwidth or CPU constraints. The algorithm first removes all nodes and links that do not have enough resources.

2) *Cluster center localization*: In the PlanetLab overlay configuration, we use the more realistic metrics such as CPU usage and available bandwidth instead of the abstract ideas of node/link stress to represent the resource availability in the substrate network.

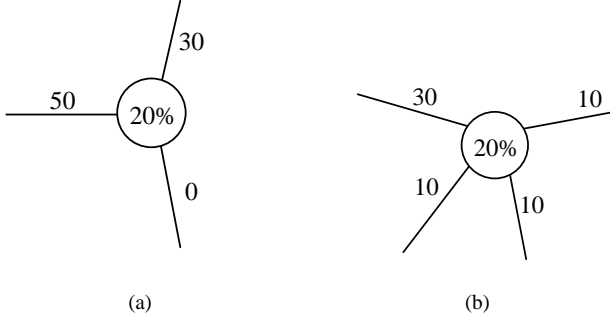


Fig. 3. Neighborhood resource availability, numbers beside links and nodes represent the available bandwidth and CPU respectively

Specifically, we modify the definition of neighborhood resource availability (NR) in [15] as follows:

$$\text{NR}(t, v) = C(t, v) \cdot \sum_{e \in L(v)} B(t, e) \quad (1)$$

where $C(t, v)$ is the available CPU at substrate node v , $B(t, e)$ is the available bandwidth on substrate link e , and $L(v)$ is the set of substrate links directly connected to substrate node v .

The above definition of NR captures both the node resources and neighboring link resources in the sense that a substrate node with a high NR means that both the node itself and some of its directly connected links are lightly loaded. The cluster center, which is the first selected substrate node, is, therefore, identified as the substrate node with the highest NR value. In the example shown in Figure 3, the NR for the substrate node in Figure 3(a) and 3(b) are 16 and 12 respectively. Therefore, the former substrate node is preferred even though it has a lower node degree.

3) *Substrate node selection*: After identifying the cluster center, the next step is to select the rest of the substrate nodes for the overlay request. We redefine the distance of a single substrate path p at time t as follows:

$$d(t, p) = \sum_{e \in p} \frac{1}{B(t, e) + \delta_L} \quad (2)$$

where δ_L is a small positive constant to avoid dividing by zero in computing the distance. And the distance between two substrate nodes u and v is defined as the minimum distance of all paths between them, *i.e.*,

$$D(t, (u, v)) = \min_{p \in P_S(u, v)} d(t, p) \quad (3)$$

Since the overlay topology could be arbitrary, there could be an overlay link between any pair of overlay nodes. Therefore, to minimize the overall distance, the weight of an overlay node is set to be the summation of distances to all previously assigned overlay nodes. Combining the distance and node stress, we use the following *node potential* as the criteria for substrate node selection:

$$\pi(t, v) = \frac{\sum_{u \in V_A} D(t, (v, u))}{C(t, v) + \delta_N} \quad (4)$$

where V_A is the set of selected substrate nodes for the same overlay and δ_N is a small positive constant to avoid dividing by zero. The set of substrate nodes with the minimum potential are then selected.

After all substrate nodes are determined, we need to map the overlay nodes into the selected substrate nodes such that overlay nodes with higher degree are mapped to substrate nodes with higher NR. The intuition behind this is that overlay nodes with higher degree will setup more overlay links. All of these overlay links will go through some of the substrate link in the neighborhood of the corresponding substrate node. Therefore, the above matching tends to reduce the congestion and balance the link load.

4) *Substrate path selection*: The last step of the overlay assignment scheme is to connect the selected substrate nodes based on the overlay topology. This is a standard routing problem and we use the shortest-distance path algorithm [16] again to select the path with minimum distance (defined in Eq. (2)) between corresponding substrate nodes. Therefore, it is consistent with Eq. (3) in our node selection process. The detailed overlay assignment algorithm is given in Algorithm 1.

Algorithm 1 Overlay assignment algorithm (Upon the i 'th overlay arrival at time a_i)

INPUTS:

$G_s = (V_S, E_S)$: substrate topology;
 $C(a_i^-, v), \forall v \in V_S$: current available CPU;
 $B(a_i^-, e), \forall e \in E_S$: current available bandwidth;
 $G_v^i = (V_V^i, E_V^i)$: Overlay topology;

OUTPUTS:

$f_N^i(\hat{v}), \forall \hat{v} \in V_V^i$ and $f_L^i(\hat{e}), \forall \hat{e} \in E_V^i$

$V_A = \emptyset$ {Note: V_A is the set of selected substrate nodes}

$V_A = V_A \cup \{\arg \max_{v \in V_S} \text{NR}(a_i^-, v)\}$

for $i = 2, \dots, |V_V^i|$ **do**

$V_A = V_A \cup \{\arg \min_{v \in V_S - V_A} \pi(a_i^-, v)\}$

end for

Assign nodes in V_A to overlay nodes such that

$\text{NR}(f_n^i(\hat{u})) \geq \text{NR}(f_n^i(\hat{v}))$, *iff* $\text{Degree}(\hat{u}) \geq \text{Degree}(\hat{v})$

Find the shortest-distance path for all overlay links

IV. IMPLEMENTATION

We implement the above service to include all 412 nodes in the S^3 data base. Latest NetFinder results detects 352 alive nodes and 68,966 substrate links with bandwidth measurement results.

The high level service model of NetFinder is shown in Figure 4, which is composed of three entities: a PlanetLab network (composed of nodes and paths connecting them) served as the substrate for all user-defined overlays, a central database responsible for collecting measurement data of PlanetLab and answering overlay setup requests, and users submitting their requests to setup overlay networks. Events in the centralized model evolve as follows: 1) Each PlanetLab node perform

periodic performance measurement of its own as well as paths leading to the neighboring PlanetLab nodes, this information is stored at each PlanetLab node as local data ¹. 2) The centralized database periodically contacts each PlanetLab node to get a copy of the local measurement data and merge them into a global view of the current PlanetLab network status. 3) The software at PlanetLab user sends a request for the current substrate states. 4) Central database returns the data to the user software. 5) The user software calculates the assignment and generates suitable outputs. 6) IIAS script is launched to start the overlay network based on the output from step 5) ².

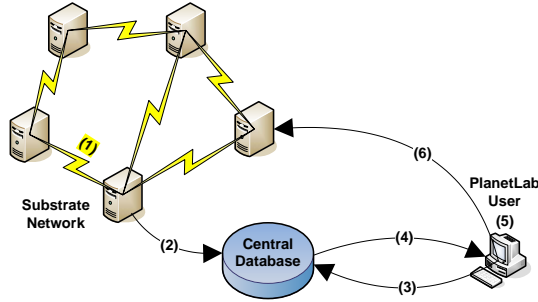


Fig. 4. Centralized NetFinder service model

V. PERFORMANCE EVALUATION

To evaluate the performance of NetFinder from its user's perspective, we use the following metrics: 1) average overlay node available CPU, defined as the average available CPU of the selected PlanetLab nodes, and 2) average overlay link available bandwidth, defined as the available bandwidth along the path connecting the corresponding PlanetLab nodes. Since each overlay node is assigned to a PlanetLab node, the available CPU on the selected PlanetLab node represents the available computational resources for the corresponding overlay node. Therefore, these metrics are more relevant to individual user.

In the first experiment, we took a snapshot of the PlanetLab network and run NetFinder to assign 5,000 randomly generated overlay networks (with evenly distributed size from 2 to 40 and average link probability 0.5). Each overlay network is assumed to be given the identical PlanetLab snapshot. Therefore, this experiment evaluates the average performance of NetFinder on different overlay topologies. For comparison purposes, we also show the results for two alternative overlay assignment schemes: 1) Least load scheme and 2) random selection. The least-load scheme represents the scenario where node selection tools such as CoMon or SWORD is used to find the set of nodes with maximum available CPU. The selected nodes are then connected to form the overlay network. The random selection scheme represents the case where a user totally ignores the performance of PlanetLab and selects nodes randomly.

¹This step is not needed when NetFinder obtains performance data from third-party sources, such as S^3 .

²This step is needed when we use IIAS to directly startup the overlay.

Figure 5 shows the cumulative distribution function (CDF) of single hop overlay path available bandwidth (*i.e.*, the direct overlay link available bandwidth). Comparing the results of NetFinder with least-load and random selection heuristics, we can clearly see that NetFinder significantly improves the available bandwidth. Specifically, more than 90% of the overlay links have the available bandwidth higher than 60Mbps. In contrast, more than 50% overlay links assigned by the least-load scheme have available bandwidth lower than 60Mbps. The random selection scheme performs slightly better than the least-load scheme since it spreads the load among all PlanetLab nodes. However, it is still significantly worse than NetFinder.

A potential use of the overlay network is bandwidth based overlay routing, where data are sent from the source to the destination through the overlay path with the maximum available bandwidth. To show how assigned overlay networks perform under this scenario, we collected the widest overlay path between each pair of overlay nodes and show their CDF in Figure 6. The result has a similar pattern as the direct overlay link available bandwidth. But the difference between these schemes are even more significant.

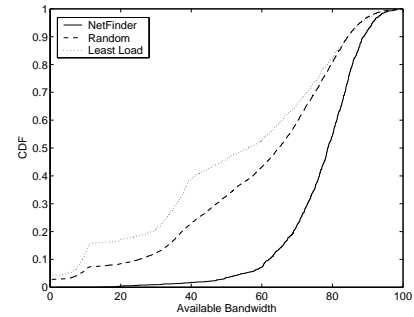


Fig. 5. CDF of single hop overlay path available bandwidth

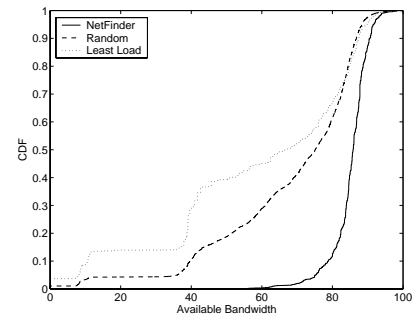


Fig. 6. CDF of widest overlay path available bandwidth

Figure 7 shows the CDF of available CPU on overlay nodes. As expected, the least-load scheme has the best node CPU performance since it performs node selection exclusively on the node performance optimization [15]. However, the difference between NetFinder and the optimal least-load scheme is small. This indicates that NetFinder can achieve near optimal available CPU performance.

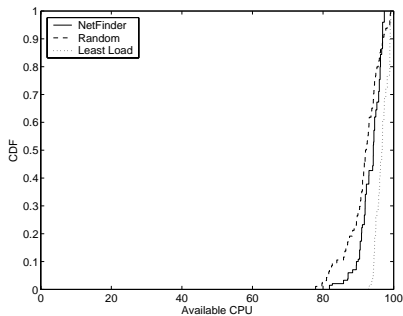


Fig. 7. CDF of overlay node available CPU

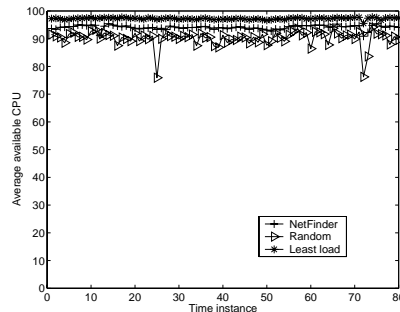


Fig. 11. Average available CPU of overlay nodes

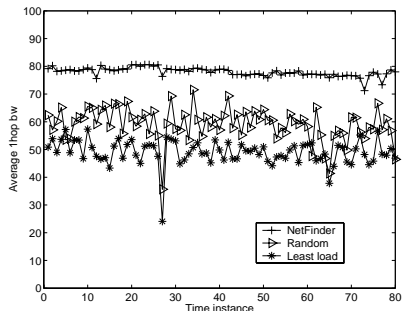


Fig. 8. Average available bandwidth of single hop overlay path

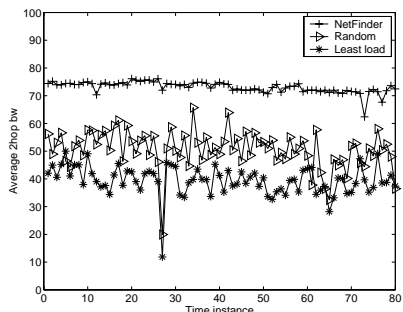


Fig. 9. Average available bandwidth of 2-hop overlay path

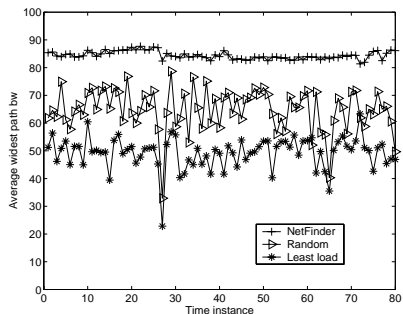


Fig. 10. Average available bandwidth of the widest overlay path

The second experiment studies the performance of assigning a single overlay network at different time instances. We collect the PlanetLab measurement data during the week of February 13-17, 2006 and plot the results of assigning the same 20 node 176 link overlay network. Figure 8, 9 and 10 show the average

available bandwidth of all 1-hop overlay paths (*i.e.*, direct overlay path without going through any intermediate overlay node), 2-hop overlay paths (*i.e.*, indirect overlay paths that go through exactly one intermediate overlay node), and widest overlay paths (*i.e.*, overlay paths with the highest available bandwidth) respectively. All these results show that NetFinder can allocate significantly higher available bandwidth than the other two schemes. Furthermore, the performance improvements are consistent throughout all time instances. Another interesting observation is that selecting the PlanetLab nodes at random and totally ignoring the network states, performs better than the least-load scheme in terms of available bandwidth. The reason is that the low loaded areas in the substrate network tend to be isolated, and therefore, they may be separated by some heavily loaded links. In terms of available CPU, the Least load scheme which is optimal outperforms NetFinder. However, the difference is marginal.

Another observation in this experiment is that the performance curves of NetFinder are much smoother than the other two schemes. This indicates that NetFinder is capable of achieving consistent performances in dynamic network situations. As a summary of this experiment, the results indicate that the NetFinder service can achieve significantly higher available bandwidth than the both least-load and random schemes.

VI. CONCLUSIONS

PlanetLab has been widely used in the networking community to test and deploy user-defined overlays. One problem always faced by PlanetLab users is how to select a set of PlanetLab nodes to form a user-defined overlay. In this chapter, we develop NetFinder, an automatic overlay network configuration tool for PlanetLab. This tool continuously collects information about the resource utilization of PlanetLab and accepts a user-defined overlay topology as input. NetFinder then selects the set of PlanetLab nodes and their interconnection for the user overlay. Performance evaluation using realistic PlanetLab data demonstrates the advantage of NetFinder by comparing it with alternative heuristics or existing tools. The results indicate that NetFinder is highly effective in locating the available PlanetLab resources and avoiding overloaded nodes and links.

REFERENCES

- [1] PlanetLab, <https://www.planet-lab.org/>.
- [2] IIAS, "Internet In a Slice," <https://wiki.planet-lab.org/wiki/bin/view/PlanetLab/InternetInASlice>.
- [3] D. Oppenheimer, D. A. Patterson, and A. Vahdat, "A case for informed service placement on planetlab," PlanetLab Consortium, Tech. Rep. PDN-04-025, December 2004.
- [4] M. Jain and C. Dovrolis, "End-to-end available bandwidth: measurement methodology, dynamics, and relation with tcp throughput," *IEEE/ACM Transaction on Networking*, vol. 11, no. 4, pp. 537–549, 2003.
- [5] "Iperf," <http://dast.nlanr.net/Projects/Iperf/>.
- [6] S.-J. Lee, P. Sharma, S. Banerjee, S. Basu, and R. Fonseca, "Measuring bandwidth between planetlab nodes," in *Proc. PAM 2005*, 2005.
- [7] V. Jacobson, "pathchar: A tool to infer characteristics of Internet paths," <ftp://ftp.ee.lbl.gov/pathchar/>.
- [8] B. A. Mah, "pchar: A tool for measuring Internet path characteristics," <http://www.kitchenlab.org/www/bmah/Software/pchar/>.
- [9] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell, "pathChirp: Efficient available bandwidth estimation for network paths," in *Passive and Active Measurement Workshop*, 2003.
- [10] R. L. Carter and M. E. Crovella, "Server selection using dynamic path characterization in wide-area networks," in *Proc. IEEE Infocom*, 1997.
- [11] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "SProbe: A fast technique for measuring bottleneck bandwidth in uncooperative environments," in *Proc. IEEE Infocom*, 2002.
- [12] SWORD, <http://www.cs.berkeley.edu/~davidopp/sword/>.
- [13] CoMon, <http://comon.cs.princeton.edu/>.
- [14] P. Yalagandula, P. Sharma, S. Banerjee, S.-J. Lee, and S. Basu, " S^3 : Scalable sensing service for planetlab," <http://networking.hpl.hp.com/s-cube/PL/>.
- [15] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *Proc. IEEE INFOCOM*, 2006.
- [16] Q. Ma and P. Steenkiste, "On path selection for traffic with bandwidth guarantees," in *Proc. IEEE ICNP*, 1997, pp. 191–202.