NetProfiler: Profiling Wide-Area Networks Using Peer Cooperation

Venkata N. Padmanabhan[1], Sriram Ramabhadran[2,⋆], and Jitendra Padhye[3]

[1] Microsoft Research
padmanab@microsoft.com
[2] University of California at San Diego
sriram@cs.ucsd.edu
[3] Microsoft Research
padhye@microsoft.com

Abstract. Our work is motivated by two observations about the state of networks today. Operators have little visibility into the end users' network experience while end users have little information or recourse when they encounter problems. We propose a system called *NetProfiler*, in which end hosts share network performance information with other hosts over a peer-to-peer network. The aggregated information from multiple hosts allows NetProfiler to *profile* the wide-area network, i.e., monitor end-to-end performance, and detect and diagnose problems from the perspective of *end hosts*. We define a set of attribute hierarchies associated with end hosts and their network connectivity. Information on the network performance and failures experienced by end hosts is then aggregated along these hierarchies, to identify patterns (e.g., shared attributes) that might be indicative of the source of the problem. In some cases, such sharing of information can also enable end hosts to resolve problems by themselves. The results from a 4-week-long Internet experiment indicate the promise of this approach.

1 Introduction

Our work is motivated by two observations about the state of networks today. First, operators have little direct visibility into the end users' network experience. Monitoring of network routers and links, while important, does not translate into direct knowledge of the end-to-end health of the network. This is because any single operator usually controls only a few of the components along an end-to-end path. On the other hand, although end users have direct visibility into *their own* network performance, they have little other information or recourse when they encounter problems. They do not know the cause of the problem or whether it is affecting other users as well.

To address these problems, we propose a system called *NetProfiler*, in which end hosts monitor the network performance and then share the information with other end hosts over a peer-to-peer network. End hosts, or "clients", are

⋆ The author was an intern at Microsoft Research during part of this work.

M. Castro and R. van Renesse (Eds.): IPTPS 2005, LNCS 3640, pp. 80–92, 2005.

in the ideal position to do monitoring since they are typically the initiators of end-to-end transactions and have full visibility into the success or failure of the transactions. By examining the correlations, or the lack thereof, across observations made by different clients, NetProfiler can detect network anomalies and localize their likely cause. Besides anomaly detection and diagnosis, this system allows users (and also ISPs) to learn about the network performance experienced by other hosts. The following scenarios illustrate the use of NetProfiler:

- A user who is unable to access a web site can find out whether the problem is specific to his/her host or ISP, or whether it is a server problem. In the latter case, the user's client may be able to automatically discover working replicas of the site.
- A user can benchmark his/her long-term network performance against that of other users in the same city. This information can be used to drive decisions such as upgrading to a higher level of service (e.g., to 768 Kbps DSL from 128 Kbps service) or switching ISPs.
- A consumer ISP such as MSN can monitor the performance seen by its customers in various locations and identify, for instance, that the customers in a certain city are consistently underperforming those elsewhere. This can call for upgrading the service or switching to a different provider of modem banks, backhaul bandwidth, etc. in that city.

We view NetProfiler as an interesting and novel P2P application that leverages peers for network monitoring and diagnosis. Peer participation is critical in NetProfiler, since in the absence of such participation, it would be difficult to learn the end-host perspective from multiple vantage points. This is in contrast to traditional P2P applications such as content distribution, where it is possible to reduce or eliminate dependence on peers by employing a centralized infrastructure. Each end-host is valuable in NetProfiler because of the perspective it provides on the health of the network, and not because of the (minimal) resources such as bandwidth and CPU that it contributes. Clearly, the usefulness and effectiveness of NetProfiler grows with the size of the deployment. In practice, NetProfiler can either be deployed in a coordinated manner by a network operator such as a consumer ISP or the IT department of an enterprise, or can grow organically as an increasing number of users install this new P2P "application".

To put NetProfiler in perspective, the state-of-the-art in end-host-based network diagnosis is an individual user using tools such as ping and traceroute to investigate problems. However, this approach suffers from several drawbacks.

A key limitation of these tools is that they only capture information from the viewpoint of a single end host or network entity. Also, these tools only focus on entities such as routers and links that are on the IP-level path, whereas the actual cause of a problem might be higher-level entities such as proxies and servers. In contrast, NetProfiler considers the entire end-to-end transaction, and combines information from multiple vantage points, which enables better fault diagnosis.

Many of the existing tools also operate on a short time scale, usually on an as-needed basis. NetProfiler monitors, aggregates, and summarizes network performance data on a continuous basis. This allows NetProfiler to detect anomalies in performance based on historical comparisons.

Another important issue is that many of the tools rely on active probing. In contrast, NetProfiler relies on passive observation of existing traffic. Reliance on active probing is problematic due to several reasons. First, the overhead of active probing can be high, especially if hundreds of millions of Internet hosts start using active probing on a routine basis. Second, active probing cannot always disambiguate the cause of failure. For example, an incomplete traceroute could be due to a router or server failure, or simply because of the suppression of ICMP messages by a router or a firewall. Third, the detailed information obtained by client-based active probing (e.g., traceroute) may not pertain to the dominant direction of data transfer (typically server→client).

Thus we believe that it is important and interesting to consider strategies for monitoring and diagnosing network performance that do *not* rely on active probing, and take a broad view of the network by considering the entire end-to-end path rather than just the IP-level path and combining the view from multiple vantage points.

In the remainder of the paper, we discuss the architecture of NetProfiler, some details of its constituent components, open issues and challenges, and related work.

2 NetProfiler Architecture and Algorithms

We now discuss the architecture of NetProfiler and the algorithms used for the acquisition, aggregation, and analysis of network performance data.

2.1 Data Acquisition

Data acquisition is performed by *sensors*, which are software modules residing on end hosts such as users' desktop machines. Although these sensors could perform active measurements, our focus here is primarily on passive observation of existing traffic. The end host would typically have multiple sensors, say one for each protocol or application. Sensors could be defined for the common Internet protocols such as TCP, HTTP, DNS, and RTP/RTCP as well protocols that are likely to be of interest in specific settings such as enterprise networks (e.g., the RPC protocol used by Microsoft Exchange servers and clients). The goal of the sensors is both to characterize the end-to-end communication in terms of success/failure and performance, and also to infer the conditions on the network path.

We have implemented two simple sensors — *TcpScope* and *WebScope* — to analyze TCP and HTTP, respectively. The widespread use of these protocols makes these sensors very useful. We now describe them briefly.

TcpScope: TcpScope is a passive sensor that listens on TCP transfers to and from the end host, and attempts to determine the cause of any performance

problems. Our current implementation operates at user level in conjunction with the NetMon or WinDump filter driver on Windows XP. Since the user's machine is typically at the receiving end of TCP connections, it is challenging to estimate metrics such as the connection's RTT, congestion window size, etc. We outline a set of heuristics that are inspired by T-RAT [1] but are simpler since we have access to the client host.

An initial RTT sample is obtained from the SYN-SYNACK exchange. Further RTT samples are obtained by identifying flights of data separated by idle periods during the slow-start phase. The RTT estimate can be used to obtain an estimate of sender's congestion window *(cwnd)*. A rough estimate of the bottleneck bandwidth is obtained by observing the spacing between the pairs of back-to-back packets emitted during slow start. [1] Using estimates of the RTT, *cwnd* and bottleneck bandwidth, we can determine the likely cause of rate limitation: whether the application itself is not producing enough data or whether an external factor such as a bandwidth bottleneck or packet loss is responsible.

Our initial experiments indicate that the TcpScope heuristics perform well. In ongoing work, we are conducting more extensive experiments in wide-area settings.

WebScope: In certain settings such as enterprise networks, the clients' web connections might traverse a caching proxy. So TcpScope would only be able to observe the dynamics of the network path between the proxy and the client. To provide some visibility into the conditions on the network path beyond the proxy, we have implemented the WebScope sensor. For an end-to-end web transaction, WebScope is able to estimate the contributions of the proxy, the server, and the server–proxy and proxy–client network paths to the overall latency. The main idea is to use a combination of cache-busting and byte-range HTTP requests, to decompose the end-to-end latency.

WebScope produces less detailed information than TcpScope but still offers a rough indication of the performance of the individual components on the client-proxy-server path. WebScope focuses on the first-level proxy between the client and the origin server. It ignores additional intermediate proxies, if any. This is just as well since such proxies are typically not visible to the client and so the client does not have the option of picking between multiple alternatives. Finally, we note that WebScope can operate in a "pseudo passive" mode by manipulating the cache control and byte-range headers on existing HTTP requests.

2.2 Normalization

The data produced by the sensors at each node needs to be "normalized" before it can be meaningfully shared with other nodes. For instance, the throughput observed by a dialup client might be consistently lower that that observed by a LAN client at the same location and yet this does not represent an anomaly. On the other hand, the failure to download a page is information that can be shared regardless of the client's access link speed.

[1] We can determine whether two packets were likely sent back-to-back by the sender by examining their IP IDs.

We propose dividing clients into a few different bandwidth classes based on their access link (downlink) speed — dialup, low-end broadband (say under 250 Kbps), high-end broadband (say under 1.5 Mbps), and LAN (10 Mbps and above). Clients could determine their bandwidth class either based on the estimates provided by TcpScope or based on out-of-band information (e.g., user knowledge).

The bandwidth class of a node is included in its set of attributes for the purposes of aggregating certain kinds of information using the procedure discussed in Section 2.3. Information of this kind includes the TCP throughput and possibly also the RTT and the packet loss rate. For TCP throughput, we use the information inferred by TcpScope to filter out measurements that were limited by factors such as the receiver-advertised window or the connection length. Regarding the latter, the throughput corresponding to the largest window (i.e., flight) that experienced no loss is likely to be more meaningful than the throughput of the entire connection.

Certain information such as RTT is strongly influenced by a client's location. So it is meaningful to share this information only with clients at the same location (e.g., same city).

Certain other information can be aggregated across all clients regardless of their location or access link speed. Examples include the success or failure of page download and an indiction of server or proxy load obtained from TcpScope or WebScope.

Finally, certain sites may have multiple replicas, with clients in different parts of the network communicating with different replicas. As such it make sense to report detailed performance information on a per replica basis and also report less detailed information (e.g., just an indication of download success or failure) on a per-site basis. The latter information would enable clients connected to a poorly performing replica to discover that the site is accessible via other replicas.

2.3 Data Aggregation

We now discuss how the performance information gathered at the individual end hosts is shared and aggregated across nodes. Our approach is based on a decentralized peer-to-peer architecture, which spreads the burden of aggregating information across all nodes.

The process of data aggregation and analysis is performed based on a set of client attributes. For both fault isolation and comparative analysis, it is desirable to compare the performance of clients that share certain attributes, as well as those that differ in certain attributes. Attributes may be hierarchical, in which case they define a *logical* hierarchy along which performance data can be aggregated. Examples of hierarchical attributes are

- *Geographical location:* Aggregation based on location is useful for users and network operators to detect performance trends specific to a particular location (e.g. "How are users in the Seattle area performing?"). Location yields a natural aggregation hierarchy, e.g., neighborhood→city→region→country.

– *Topological location:* Aggregation based on topological location is useful for users to make informed choices regarding their service provider (e.g., "Is my local ISP the reason for the poor performance I am seeing?"). It is also useful for network providers to identify performance bottlenecks in their networks. Topological location can also be aggregated along a hierarchy, e.g., subnet→PoP→ISP.

Alternatively, attributes can be non-hierarchical, in which case they are used to filter performance data to better analyze trends specific to that particular attribute. Examples of non-hierarchical attributes include:

– *Destination site:* Filtering based on destination site is useful to provide information on whether other users are able to access a particular website, and if so, what performance they are seeing (e.g. "Are other users also having problems accessing `www.cnn.com`?"). Although not hierarchical, in the case of replicated sites, destination site can be further refined based on the actual replica being accessed.
– *Bandwidth class:* Filtering based on bandwidth class is useful for users to compare their performance with other users within the same class (e.g. "How are all dialup users faring?") , as well as in other classes ("What performance can I expect if I switch to DSL?").

Aggregation based on attributes such as location is done in a hierarchical manner, with the aggregation tree mirroring the logical hierarchy defined by the attribute space. This is based on the observation that nodes are typically interested in detailed information only from "nearby" peers. They are satisfied with more aggregated information about distant peers. For instance, while a node might be interested in specific information, such as the download performance from a popular web site, pertaining to peers in its neighborhood, it has little use for such detailed information from nodes across the country. Regarding the latter, it is likely to be interested only in an aggregated view of the performance experienced by clients in the remote city or region.

Non-hierarchical attributes such as bandwidth class and destination site are used as filters that qualify performance data as it aggregated up the logical hierarchy described above. For example, each node in the hierarchy may organize the performance reports it receives based on bandwidth class, destination site and perhaps the cross-product. This enables the system to provide more fine-grained performance trends (e.g., "What is the performance seen by dialup clients in Seattle when accessing `www.cnn.com`?"). Conceptually, this is similar to maintaining different aggregation trees for each combination of attributes; in practice, it is desirable to realize this in a single hierarchy as it limits the number of times an end-host has to report the same performance record. Since the number of bandwidth classes is small, it is feasible to maintain separate hierarchies for each class. However, with destination sites, this is done only for a manageable number of popular sites. For less popular sites, it may be infeasible to maintain per-site trees, so only a single aggregated view of the site is maintained, at the cost of losing the ability to further refine based on other attributes.

Finally, mechanisms are required to map the above logical aggregation hierarchies to a *physical* hierarchy of nodes. To this end, we leverage DHT-based aggregation techniques such as SDIMS [2], which exploits the natural hierarchy yielded by the connectivity structure of the DHT nodes. Aggregation happens in a straightforward manner: nodes maintain information on the performance experienced by clients in their subtree. Periodically, they report aggregated views of this information to their parent. Such a design results in good locality properties, ensures efficiency of the aggregation hierarchy, and minimizes extraneous dependencies (e.g., the aggregator node for a client site lies within the same site).

2.4 Analysis and Diagnosis

We now discuss the kinds of analyses and diagnoses that NetProfiler enables.

Distributed Blame Attribution: Clients that are experiencing poor performance can diagnose the problem using a procedure that we term as *distributed blame attribution*. Conceptually, the idea is for a client to ascribe the poor performance that it is experiencing to the entities involved in the end-to-end transaction. The entities could include the server, proxy, DNS[2], and the network path, where the resolution of the path would depend on the information available (e.g., the full AS-level path or simply the ISP/PoP that the client connects to). The simplest policy is for a client to ascribe the blame equally to all of the entities. But a client could assign blame unequally if it suspects certain entities more, say based on information gleaned from local sensors such as TcpScope and WebScope.

Such blame information is then aggregated across clients. The aggregate blame assigned to an entity is normalized to reflect the fraction of transactions involving the entity that encountered a problem. The entities with the largest blame score are inferred to be the likely trouble spots.

The hierarchical aggregation scheme discussed in Section 2.3 naturally supports this distributed blame attribution scheme. Clients use the performance they experienced to update the performance records of entities at each level of the hierarchy. Finding the suspect entity is then a question of walking up the attribute hierarchy to identify the highest-level entity whose aggregated performance information indicates a problem (based on suitably-picked thresholds). The preference for picking an entity at a higher level reflects the assumption that a single shared cause for the observed performance problems has a greater likelihood than multiple separate causes. For instance, if clients connected to most of the PoPs of Verizon are experiencing problems, then the chances are that there is a general problem with Verizon's network rather than a specific problem at each individual PoP.

[2] The DNS latency may not be directly visible to a client if the request is made via a proxy.

Comparative Analysis: A client might benefit from knowledge of its network performance relative to that of other clients, especially those in the same vicinity (e.g., same city). Such knowledge can drive decisions such as whether to upgrade to a higher level of service or switch ISPs. For instance, a user who consistently sees worse performance than others on the same ISP network and in the same neighborhood can demand an investigation by the ISP; in the absence of comparative information, the user wouldn't even know to complain. A user who is considering upgrading from low-end to high-end DSL service could compare notes with existing high-end DSL users in the same locale to see how much improvement an upgrade would actually result in, rather than simply going by the speed advertised by the ISP.

Likewise, a consumer ISP that buys infrastructural services such as modem banks and backhaul bandwidth from third-party providers can monitor the performance experienced by its customers in different location. If it finds, for instance, that its customers in Seattle are consistently underperforming customers elsewhere, it would have reason to suspect the local infrastructure provider(s) in Seattle.

Network Engineering Analysis: A network operator could use detailed information gleaned from clients to make an informed decision on how to reengineer or upgrade the network. For instance, consider the IT department of a large global enterprise that is tasked with provisioning network connectivity for dozens of corporate sites spread across the globe. There is a plethora of choices in terms of connectivity options (ranging from expensive leased lines to the cheaper VPN over the public Internet alternative), service providers, bandwidth, etc. The goal is typically to balance the twin goals of low cost and good performance. While existing tools and methodologies (based say on monitoring link utilization) are useful, the ultimate test is how well the network serves end-users in their day-to-day activities. NetProfiler provides an end-user perspective on network performance, thereby complementing existing monitoring tools and enabling more informed network engineering decisions. For instance, significant packet loss rate coupled with the knowledge that the egress link utilization is low might point to a problem with chosen service provider and might suggest switching to a leased line alternative. Poor end-to-end performance despite a low packet loss rate could be due to a large RTT, which could again be determined from NetProfiler observations. Remedial measures might include setting up a local proxy cache or server replica.

Network Health Reporting: The information gathered by NetProfiler can be used to generate reports on the health of wide-area networks such as the Internet or large enterprise networks. While auch reports are available today from organizations such as Keynote [3], the advantage of the NetProfiler approach is lower cost, greater coverage, and the ability to operate virtually unchanged in restricted environments such as corporate networks as well as the public Internet.

3 Experimental Results

We present some preliminary experimental observations to provide a flavor of the kinds of problems that the NetProfiler system could address. Our experimental setup consists of a set of a heterogeneous set of clients that repeatedly download content from a diverse set of 70 web sites during a 4-week period (Oct 1-29, 2004). The client set includes 147 PlanetLab nodes, dialup hosts connected to 26 PoPs on the MSN network, and 5 hosts on Microsoft's worldwide corporate network. Our goal was to emulate, within the constraints of the resources at our disposal, a set of clients running NetProfiler and sharing information to diagnose problems. Here are a few interesting observations:

- We observed several failure episodes during which accesses to a web site failed at most or all of the clients. Examples include failure episodes involving `www.technion.ac.il` and `www.hku.hk`. The widespread impact across clients in diverse locations suggests a server-side cause for these problems. It would be hard to make such a determination based just on the view from a single client.
- There are significant differences in the failure rate observed by clients that are seemingly "equivalent". Among the MSN dialup nodes, those connected to PoPs with ICG as the upstream provider experienced a much lower failure rate (0.2-0.3%) than those connected to PoPs with other upstream providers such as Qwest and UUNET (1.6-1.9%). This information can help MSN identify underperforming providers and take the necessary action to rectify the problem. Similarly, clients in CMU have a much higher failure rate (1.65%) than those in Berkeley (0.19%). This information can enable users at CMU pursue the matter with their local network administrators.
- Sometimes a group of clients shares a certain network problem that is not affecting other clients. The attribute(s) shared by the group might suggest the cause of the problem. For example, all 5 hosts on the Microsoft corporate network experience a high failure rate (8%) in accessesing `www.royal.gov.uk`, whereas the failure rate for other clients is negligible. Since the Microsoft clients are located in different countries and connect via different web proxies with distinct WAN connectivity, the problem is likely due to a common proxy configuration across the sites.
- In other instances, the problem is unique to a specific client-server pair. For example, the Microsoft corporate network node in China is never able to access `www.nmt.edu` whereas other nodes, including the ones at the other Microsoft sites, do not experience a problem. This suggests that the problem is specific to the path between the China node and `www.nmt.edu` (e.g., site blocking by the local provider). If we had access to information from multiple clients in China, we might be in a position to further disambiguate the possible causes.

4 Discussion

4.1 Deployment Models

We envision two deployment models for NetProfiler: *coordinated* and *organic*. In the coordinated model, NetProfiler is deployed by an organization such as the IT department of a large enterprise, to complement existing tools for network monitoring and diagnosis. The fact that all client hosts are in a single administrative domain simplifies the issues of deployment and security. In the organic model, on the other hand, NetProfiler is installed by end users themselves (e.g., on their home machines) in much the same way as they install other peer-to-peer applications. They might do so to obtain greater visibility into the cause of network connectivity and performance problems that they encounter. This is a more challenging deployment model, since issues of privacy and security as well as bootstrapping the system become more significant. We discuss these challenges next.

4.2 Bootstrapping

To be effective, NetProfiler requires a sufficient number of clients that overlap and differ in attributes to participate, so that meaningful comparisons can be made and conclusions drawn. The coordinated model makes this bootstrapping easy, since the IT department can very quickly deploy NetProfiler on a large number of clients in various locations throughout the enterprise, essentially by fiat.

Bootstrapping is much more challenging in the organic deployment model, where users install NetProfiler by choice. There is a chicken-and-egg problem between having a sufficient number of users to make the system useful and making the system useful enough to attract more users. To help bootstrap the system, we propose relaxing the insistence on passive monitoring by allowing a limited amount of active probing (e.g., web downloads that the client would *not* have performed in normal course). Clients could perform active downloads either autonomously (e.g., like Keynote clients) or in response to requests from peers. Of course, the latter option should be used with caution to avoid becoming a vehicle for attacks or offending users, say by downloading from "undesirable" sites. In any case, once the deployment has reached a certain size, active probing could be turned off.

4.3 Security

The issues of privacy and data integrity pose significant challenges to the deployment and functioning of NetProfiler. These issues are arguably of less concern in a controlled environment such as an enterprise.

Users may not want to divulge their identity, or even their IP address, when reporting performance. To help protect their privacy, we could give clients the option of identifying themselves at a coarse granularity that they are comfortable

with (e.g., at the ISP level), but that still enables interesting analyses. Furthermore, anonymous communication techniques (e.g., [4]), that hide whether the sending node actually originated a message or is merely forwarding it, could be used to prevent exposure through direct communication. However, if performance reports were stripped of all client-identifying information, we would only be able to perform very limited analyses and inference (e.g., we might only be able to infer website-wide problems that affect most or all clients).

There is also the related issue of data integrity — an attacker could spoof performance reports and/or corrupt the aggregation procedure. In general, guaranteeing data integrity would require sacrificing privacy (e.g., [5]). However, in view of the likely usage of NetProfiler as an advisory tool, we believe that it would probably be acceptable to have a reasonable assurance of data integrity, even if not iron-clad guarantees. For instance, the problem of spoofing can be alleviated by insisting on a two-way handshake before accepting a performance report. The threat of data corruption can be mitigated by aggregating performance reports along multiple hierarchies and employing some form of majority voting when there is disagreement.

5 Related Work

In this section, we briefly survey existing tools and techniques for network monitoring and diagnosis, and contrast them with NetProfiler.

Several tools have been developed for performing connectivity diagnosis from an end host (e.g., ping, traceroute, pathchar [6], tulip [7]). While these tools are clearly useful, they have some limitations, including dependence on active probing of routers (which may be expensive and also infeasible in many cases), and a focus on just the IP-level path and the view from a single host. In contrast, NetProfiler correlates on *passive* observations of existing end-to-end communication from *multiple* vantage points to diagnose problems.

Network tomography techniques [8] leverage information from multiple IP-level paths to infer network health. However, tomography techniques are based on the analysis of fine-grained packet-level correlations, and therefore have typically involved active probing. Also, the focus is on a server-based, "tree" view of the network whereas NetProfiler focuses on a client-based "mesh" view.

PlanetSeer [9] is a system to locate Internet faults by selectively invoking traceroutes from multiple vantage points. It is a server-based system (unlike NetProfiler), so the direction of traceroutes matches the dominant direction of data flow. PlanetSeer differs from NetProfiler in terms of its dependence on active probing and focus on just the IP-level path.

Tools such as NetFlow [10] and Route Explorer [11] enable network administrators to monitor network elements such as routers. However, these tools do not directly provide information on the end-to-end health of the network.

SPAND [12] is a tool for sharing performance information among end hosts belonging to a single subnet or site. The performance reports are stored in a central database and are used by end hosts for performance prediction and mirror

selection. NetProfiler differs from SPAND in several ways, including its focus on fault diagnosis rather than performance prediction and use of a P2P approach that encompasses nodes beyond the local subnet or site.

Several systems have been developed for distributed monitoring, aggregation, and querying on the Internet. Examples include Ganglia [13], Slicestat [14], IrisNet [15], PIER [16], Sophia [17], SDIMS [2], and Astrolabe [18]. NetProfiler could in principle leverage these systems for data aggregation, albeit with relaxed consistency and timeliness requirements. The primary focus of our work is on leveraging end-host observations to diagnose network problems rather than on developing a new data aggregation system.

The Knowledge Plane proposal [19] shares NetProfiler's goal of enabling users to diagnose network problems. But it is more ambitious in that the knowledge plane is envisaged as encompassing not only the end users' network experience but also network configuration and policy information. In contrast, NetProfiler is designed to be deployable on today's Internet with only the cooperation of (a subset of) end hosts.

Systems such as NETI@home [20] and Keynote [21] also gather end-host-based network performance data. Although it is unclear in what ways this data is further analyzed, NetProfiler's analyses described in Section 2.4 could easily be applied to such data.

Finally, like NetProfiler, STRIDER [22] and PeerPressure [23] also leverage information from peers to do cross-machine troubleshooting of configuration problems, by comparing the configuration settings of a sick machine with that of a healthy machine. NetProfiler is different in that it explicitly deals with information on specific problems (e.g., DNS lookup failures for a particular server) rather than "blackbox" configuration information. Also, given the focus on wide-area network troubleshooting, NetProfiler requires the participation of a larger number of peers in a diverse set of network locations.

6 Conclusion

We have presented NetProfiler, a P2P system to enable monitoring and diagnosis of network problems. Unlike in many previous P2P applications, the participation of peers is fundamental to the operation of NetProfiler. The results from an initial 4-week experiment indicate the promise of the proposed approach. We believe that the capabilities provided by NetProfiler can benefit both end users and network operators, such as consumer ISPs and enterprise IT departments. In ongoing work, we are also exploring using end-host observations to detect large-scale surreptitious communication as might precede a DDoS attack.

Acknowledgements

We thank our colleagues at the Microsoft Research locations worldwide, MSN, and PlanetLab for giving us access to a distributed set of hosts for our experiments. We also thank Sharad Agarwal for his comments on an earlier draft.

References

1. Zhang, Y., Breslau, L., Paxson, V., Shenker, S.: On the Characteristics and Origins of Internet Flow Rates. In: SIGCOMM. (2002)
2. Yalagandula, P., Dahlin, M.: A scalable distributed information management system. In: SIGCOMM. (2004)
3. : Keynote Internet Health Report. (http://www.internethealthreport.com/)
4. Reiter, M.K., Rubin, A.D.: Crowds: anonymity for Web transactions. ACM Transactions on Information and System Security **1** (1998) 66–92
5. Przydatek, B., Song, D., Perrig, A.: Sia: Secure information aggregation in sensor networks (2003)
6. Downey, A.B.: Using pathchar to Estimate Link Characteristics. In: SIGCOMM. (1999)
7. Mahajan, R., Spring, N., Wetherall, D., Anderson, T.: User-level Internet Path Diagnosis. In: SOSP. (2003)
8. Caceres, R., Duffield, N., Horowitz, J., Towsley, D.: Multicast-based inference of network-internal loss characteristics. IEEE Transactions on Information Theory (1999)
9. Zhang, M., Zhang, C., Pai, V., Peterson, L., Wang, R.: PlanetSeer: Internet Path Failure Monitoring and Characterization in Wide-Area Services. In: OSDI. (2004)
10. Feldmann, A., Greenberg, A., Lund, C., Reingold, N., Rexford, J., True, F.: Deriving traffic demands for operational ip networks: Methodology and experience. In: SIGCOMM. (2001)
11. http://www.packetdesign.com/.
12. Seshan, S., Stemm, M., Katz, R.H.: Spand: Shared passive network performance discovery. In: USITS. (1997)
13. Ganglia. http://ganglia.sourceforge.net/.
14. Slicestat. http://berkeley.intel-research.net/bnc/slicestat/.
15. Gibbons, P.B., Karp, B., Ke, Y., Nath, S., Seshan, S.: Irisnet: An architecture for a world-wide sensor web. IEEE Pervasive Computing (2003)
16. Huebsch, R., Hellerstein, J.M., Lanham, N., Loo, B.T., Shenker, S., Stoica, I.: Querying the internet with pier. In: VLDB. (2003)
17. Wawrzoniak, M., Peterson, L., Roscoe, T.: Sophia: An information plane for networked systems. In: HotNets. (2003)
18. van Renesse, R., Birman, K., Vogels, W.: Astrolabe: A robust and scalable technology for distributed system monitoring, management and data mining. ACM Transactions on Computer Systems (2003)
19. Clark, D., Partridge, C., Ramming, J., Wroclawski, J.: A Knowledge Plane for the Internet. SIGCOMM (2003)
20. Simpson, C.R., Riley, G.F.: NETI@home: A Distributed Approach to Collecting End-to-End Network Performance Measurements. PAM (2004)
21. Keynote Systems. http://www.keynote.com.
22. Wang, Y., Verbowski, C., Dunagan, J., Chen, Y., Chun, Y., Wang, H., Zhang, Z.: STRIDER: A Black-box, State-based Approach to Change and Configuration Management and Support. In: Usenix LISA. (2003)
23. Wang, H., Platt, J., Chen, Y., Zhang, R., Wang, Y.: Automatic Misconfiguration Troubleshooting with PeerPressure. In: OSDI. (2004)