Discovering Topologies at Router Level*

Donato Emma, Antonio Pescapé, and Giorgio Ventre

University of Napoli "Federico II"
{doemma,pescape,ventre}@unina.it

Abstract. Management is an essential task for the correct behavior of networks. In this field, several aspects should be taken into account. Among them, network topology is one of the most important elements to control. This paper proposes an approach to the topology discovery based on a hybrid methodology. We propose a tool, called *HyNeTD* (Hybrid Network-Topology Discovery), that effectively combines active and passive measurements to discover network topologies at router level. Architectural choices are presented and discussed and some preliminary experimental results, carried out over a controlled test-bed, are given.

1 Introduction

Automatic discovery of physical topology plays a crucial role in enhancing the manageability of modern IP networks. Discovering network topologies is an important and inherently difficult task [1] [3]. Network topology knowledge can prove useful in different situations, from "*fault isolation*", to "*performance analysis*", from "*network planning*" to "*service positioning*", and, finally, from "*Traffic Engineering algorithms*" to a new general class of "*topology-aware distributed algorithms*". Many factors contrast with the manual construction of network maps: (i) the number of entities involved in today's network is very large and keeps increasing exponentially; (ii) different parts of the network are managed by different authorities; (iii) network topologies are dynamic. One way to overcome these difficulties is to automate the discovery process by searching more efficient ways to find network maps. Several approaches for topology discovery at Autonomous System (AS) level have been proposed. As for approaches at router level some proposals exist yet, but, to the best of our knowledge, few platforms are available.

In this paper we propose an adaptive and hybrid IP based methodology for network-topology discovery at router level, its implementation in a tool called *HyNeTD* (Hybrid Network Topology Discovery), and its experimental validation.

The rest of the paper is organized as follows. Section 2 presents background and motivations at the base of our work. In Section 3 the state of the art is

* This work has been partially supported by the Italian Ministry for Education, University and Research (MIUR) in the framework of the FIRB Project "Middleware for advanced services over large-scale, wired-wireless distributed systems" (WEB-MINDS) and QUASAR PRIN Project. The authors would like to thank Domenico Dichiarante for his valuable support and hints

T. Magedanz, E.R.M. Madeira, and P. Dini (Eds.): IPOM 2005, LNCS 3751, pp. 118–129, 2005.

discussed. Section 4 presents *HyNeTD* design choices. In Section 5 a preliminary experimental analysis of our proposal is presented. Finally, Section 6 ends the paper with conclusions and issues for research.

2 Background and Motivations

An increasing number of Internet applications attempt to optimize their communications by monitoring network topologies. Hence, the need arises for widely usable, and highly accurate, algorithms and techniques capable to identify network entities using little or no information about them. Many topology discovery methodologies have been proposed in literature. We propose the following classification:
Passive Methodologies – relying on the use of Simple Network Management Protocol (SNMP) and Domain Name System (DNS);
Active Methodologies – based on the use of tools such '*ping*' and '*traceroute*';
Routing Based Methodologies – based on the use of routing information;
Hybrid Methodologies – efficient combinations of the previous methodologies.
Different approaches to the topology discovery may be evaluated on the basis of *efficiency* (i.e. impose the least possible network overhead), *quickness* (i.e. take the least possible time), *completeness* (i.e. discover the entire topology) and *accuracy* (i.e. not make mistakes). *Passive Methodologies* are fast and reliable but also not always usable. *Active Methodologies* are neither reliable nor fast, but they are more widely usable. *Routing Based Methodologies* strictly depend on the routing protocol but they are fast and reliable. Finally, *Hybrid Methodologies* give the opportunity to merge the benefits of the previous methodologies. None of the above approaches outperforms the others. For example in [20] it is shown how studies based only on *traceroute* retrieved information may led to erroneous results. This consideration is pushing research in the topology discovery field towards an *adaptive hybrid methodology* able to follow the current network infrastructure status and configuration. In this paper we propose such an approach, by appropriately combining an active-based technique with a passive information retrieval methodology.

3 Related Work

Fremont system [12] uses an extensible set of discovery modules that are based on a variety of different protocols and information sources. To the best of our knowledge it was the first example of a hybrid discovery approach. In [2] Keshav *et al.* describe several heuristics and algorithms to discover both intra-domain and Internet-backbone topology. The proposed approach combines SNMP, routing information, *traceroute*, and measurements. Mansfield *et al.* in [11] define an SNMP based framework for *Internet mapping*. The main lack of their approach is the use of only passive information sources. In [9] Barford *et. al* study the marginal utility of adding information sources in performing wide-area measurements in the context of Internet topology discovery. They show that the

marginal utility of additional measurement sites declines rapidly even after the first two sites. *Mercator* project [3] explores tools such as *traceroute* to group IP addresses in order to produce an Internet map. The CAIDA's Skitter [4] project has been developed to combine *traceroute* and benchmark-based analysis. This tool uses *traceroute* to find the paths connecting two nodes and to collect performance information from them. *Remos* [10] is an architecture that can be used to provide resources information to distributed applications. To collect information from different networks and their hosts, it provides several collectors that use different technologies, such as SNMP or benchmarking. Anyway, *Remos* gives support only to distributed applications, such for example grid computing, therefore it can not be used as a tool for *general purpose* network topology discovery. At AS level, there have been interesting works [5–8] which leverage *traceroute* measurements and BGP routes to infer AS-Level maps. In [13] Spring *et al.* present Internet mapping techniques that allow to measure directly router level ISP topologies without a significant loss in accuracy. The proposed techniques include the use of (i) BGP routing tables to focus measurements, (ii) IP routing properties , (iii) alias resolution techniques, and (iv) DNS queries. As for proprietary solutions, SNMP-based tools for automatic discovery of network topology are included in many commercial network management systems (i.e. HP's OpenView [14] and IBM's Tivoli [15]). These tools assume that SNMP is widely deployed, but they also send ICMP messages toward not SNMP-capable hosts and routers. Details of their discovery algorithms are proprietary and not available to the authors of this work.

4 HyNeTD Approach

HyNeTD main goal is to discover the router level topology of IP networks via the IP address spaces and the SNMP community names. *HyNeTD* has been designed to be used in scenarios that differ for (i) available information sources, (ii) size, and (iii) administrative policies. In any scenario *HyNeTD* tries to reach the maximum achievable accuracy and completeness, also reducing the produced network overhead.

In this Section we first introduce design choices adopted for *HyNeTD*, then we give a description of the *HyNeTD*'s architecture.

Design Choices. In order to guarantee a high degree of completeness and reliability, with a low network overhead, *HyNeTD* adopts a hybrid methodology by combining both passive and active approaches. SNMP and DNS are the information sources of the *HyNeTD*'s passive methodology. At the basis of the *HyNeTD*'s active approach are both *ping* with record route (*ping-rr* in the following) and *traceroute* utilities. *HyNeTD* applies the active approach only to get information from devices that do not have an accessible SNMP agent.

HyNeTD Architecture. *HyNeTD* implements a multi-step discovery process by adapting its behavior to the state of the network.

In the first step, in order to identify all the *active addresses*, *HyNeTD* sends a *ping* towards all the addresses belonging to the discovered addresses spaces. It assumes as active the responding ones. These addresses are added to a list of active addresses (the *UP list* in the following).

In the second step *HyNeTD* looks for available SNMP information sources. The simplest way to test the SNMP availability is to send an SNMP request and wait for a response. If a response arrives the SNMP availability may be assumed. This technique is simple but may produce a high network overhead if only a small part of the active machines have an accessible SNMP agent. In order to decrease this overhead, *HyNeTD* tries to reduce the number of addresses to be tested. It sends one UDP packet towards the *SNMP port* of any *active addresses*. *HyNeTD* assumes that it is impossible to obtain SNMP information from addresses responding with ICMP *"Host Unreachable-Port"* messages. Moreover, *HyNeTD* inserts such addresses in a list of addresses for which it is necessary to implement an active discovery process (the *ICMP list* in the following). If an address does not respond it is possible that (a) the UDP packet or the ICMP response is lost, (b) the device is configured to not respond, or (c) there is an active SNMP agent. In order to avoid mistakes due to (a), in case of no response *HyNeTD* resends the UDP packet. If still no response is received *HyNeTD* assumes (c) and inserts the address in the the list of addresses that may give SNMP information (the *SNMP list* in the following). Obviously (b) is still possible, *HyNeTD* solves this ambiguity at a later stage.

In the third step *HyNeTD* starts the passive discovery process. For each address belonging to the *SNMP list*, *HyNeTD* sends some SNMP requests to retrieve information belonging to the standard part of the SNMP Management Information Base (MIB) (see Table 1). *HyNeTD* assumes that all the addresses that do not respond to the first SNMP request can not be used as SNMP information sources. Therefore, it does not send any future SNMP request toward such addresses, which are also added to the *ICMP list*. Finally, If the DNS feature is enabled, *HyNeTD* ends the passive discovery process by sending DNS inverse look-up calls to obtain the addresses name.

In the forth step *HyNeTD* starts the active discovery process. This process is based on the use of both *ping-rr* and *traceroute*. In order to obtain information also from routers that do not respond to *traceroute*, *HyNeTD* uses a modified *traceroute* implementation: it uses ICMP Echo-Request messages instead of UDP packets with an invalid destination port. The information retrieved using *ping-rr*

Table 1. Retrieved MIB entries

IP-MIB::ipForwarding	an integer value equal to 1 if the device is capable to forward IP packets
IP-MIB::ipAdEntAddr	a table containing the ip addresses of the interfaces belonging to the device
IP-MIB::ipAdEntIfIndex	a table containing references to other interfaces from which it is possible to obtain SNMP information
IP-MIB::ipAdEntNetMask	a table containing the net masks of each of the device's addresses
IP-MIB::ipNetToMediaNetAddress	the ARP tables of each of the interfaces of the device
RFC1213-MIB::ipRouteDest	a table containing the subnets of all the interfaces reachable from the device
RFC1213-MIB::ipRouteNextHop	a table containing the addresses of the next hop routers
RFC1213-MIB::ipRouteMask	a table containing the net masks of all the reachable subnets
RFC1213-MIB::ipRouteType	a table containing for each of the subnets of the ipRouteMask a flag that indicates if the device is directly connected to the subnet
IF-MIB::ifSpeed	a table containing the speed of each of the interfaces belonging to the device

may be effective in discovering networks of small size. Indeed, by using *ping-rr* it is possible to discover some links that *traceroute* would have not found due to the presence of devices that do not respond to the ICMP Echo-Request.

Finally, during the fifth step, *HyNeTD* merges and elaborates all retrieved information. This process is composed of several sub-steps:
(a) *Passive subnets reconstruction.* Using information obtained from the SNMP, *HyNeTD* identifies the subnet of each address belonging to the *SNMP list*.
(b) *Passive links reconstruction. HyNeTD* identifies all the links between the routers discovered during the third step by using the following observation: "if two routers are connected through the interfaces A and B then those interfaces must belong to the same link".
(c) *Alias resolution.* During the fourth step *HyNeTD* has constructed several lists of addresses. In this sub-step it uses these lists to discover routers' addresses by using the following assumption: "if an address is on the path from the *HyNeTD* host to any destination, *HyNeTD* assumes that this address belongs to a router". Moreover, the discovered routers' addresses are not yet grouped together into routers. The process of grouping is called *alias resolution*. To overcame the drawbacks of a simple *alias resolution* approach based only on information retrieved from the DNS system (e.g. DNS may be mis-configured), *HyNeTD* combine such an approach with the *Ally alias-resolution* heuristic [13].
(d) *Active links reconstruction.* In this step *HyNeTD* uses the results of (c) in order to find out the links not recognized in (b). Starting from the same observation used in (b) and using the results of both *ping-rr* and *traceroute*, for each router's address A, *HyNeTD* searches among the addresses of the *previous hop devices* (devices that are one step before the considered router in either the result of *traceroute* or *ping-rr*) the one that is the *closer* to A (that is the address that produces the minimum value when XOR-ed with A). This address is assumed to be the other end of the link that starts from A.
(e) *Active subnets identification.* Finally, *HyNeTD* finds the subnet masks of each address belonging to the *ICMP list* performing the following operations:

- *Extraction of subnets from links:* For each link $A1 \longleftrightarrow A2$ identified during (d), *HyNeTD* calculates a temporary net-mask using the empirical formula $temp\_mask = NOT[(AND(A1, A2))XOR(OR(A1, A2))]$. $temp\_mask$ is an estimation of the unknown subnet mask. If the estimation is not compatible with $A1$ and $A2$, *HyNeTD* produces a compatible mask by setting to 0 the least significant bits of $temp\_mask$. The resulting mask is associated to both A1 and A2.
  The following sub-steps deal with addresses not associated to any link.
- *Passive information comparison:* HyNeTD verifies if it is possible to include each of these addresses into one of the already identified subnets and sets coherently their subnet masks.
- *Active addresses partitioning:* By assuming that two addresses belong to the same subnet if they have the same *previous hop address*, *HyNeTD* divides the addresses that still are not associated to any subnet in a set of lists. Each of these lists represents a subnet.

– *ICMP mask request: HyNeTD* sends an ICMP mask request towards all the addresses of each list. Each address may (a) respond with a valid mask, (b) respond with 0.0.0.0, or (c) not respond. If all the addresses of a list do not respond, or return an invalid mask, nothing can be done. In the case of just one valid response, *HyNeTD* assumes the given mask as the mask of the list. In case of multiple responses, it implements a *voting* process to select the list's net-mask.
– *Subnet guessing from a cluster of addresses:* For all still pending subnet lists, *HyNeTD* performs the *subnet guessing from a cluster of addresses* heuristic [2]. This heuristic returns an estimation of the net-masks to be assigned to the addresses lists.

HyNeTD Multi-thread Implementation. Both passive and active methods require to wait responses from contacted devices. *HyNeTD* tries to avoid these waits adopting a Multi-Thread Implementation in which all the activities that do not have a serial tie are overlapped: (i) the first four steps are characterized by an internal concurrent architecture; (ii) passive discovery methods and active discovery methods are executed in a parallel fashion.

To summarize, the proposed architecture, stepping from the results of some previous works, integrates different approaches and techniques and introduces some main innovations, such: (i) the use of *ping-rr*; (ii) the implementation of a scan technique to test the SNMP availability; (iii) the implementation of the Ally algorithm; (iv) and the use of *DNS inverse look-up* call in combination with the results of the Ally algorithm. These innovations, as shown in section 5, enable to achieve good performance in terms of efficiency, quickness, completeness, scalability and accuracy.

5 Validation and Experimental Analysis

The main goal of our experimental analysis has been to study the behavior of *HyNeTD* by varying, the discovered topologies (in this paper we present the result obtained analyzing the topologies of Figures 1, 2, and 3), the number of threads, the number of retries and the amount of "cross-traffic". The metrics used to evaluate *HyNeTD* are:
Network overhead – total amount of probing traffic;
Discovery time – discovery's duration;
Accuracy – ratio between the number of discovered routers R_d and the real number of routers.

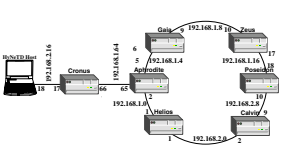

Fig. 1. Ring topology

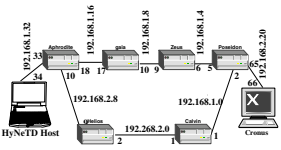

Fig. 2. Backup topology

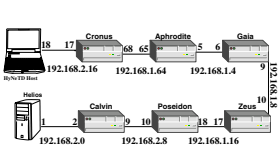

Fig. 3. Linear topologies

5.1 Test-Bed Description

The used test-bed (see Table 2) is composed by 7 routers (Linux Mandrake 9.1 boxes with *ZEBRA Routing Software*), and by a Compaq Evo notebook with Linux RedHat 9.0 used to run *HyNeTD*. As for SNMP, we used the UCD-SNMP [16]. Moreover, we used D-ITG [19] to generate cross traffic.

Table 2. Test-bed description

Router/Host Names	CPU	RAM	Network Interfaces
CRONUS, POSEIDON, APHRODITE, ZEUS	Intel Celeron 500 MHz	64 MB	3 INTEL pro 100+
HELIOS, CALVIN, GAIA	Intel PIII 757 MHz	256 MB	3 INTEL pro 100+
HyNeTD host	Intel P IV 2.4 GHz	256 MB	NIC integrated on the board

Due to its adaptive and hybrid architecture, *HyNeTD* may be used in different network conditions. Here we show experimental results with *HyNeTD* used in the conditions reported in Table 3. In the *Passive* case *HyNeTD* uses only passive methods, so it works like a pure passive discovery tool. Vice versa, in the *Active* case it works like a pure active discovery tool. Clearly, in the *Hybrid* cases *HyNeTD* uses both passive and active methods. As for the comparison with existing approaches, thanks to this *modus operandi* we compared HyNeTD with pure passive and active approaches (implemented in other tools). Before stepping into experimental details, it is worth noting that we repeated each test several times. In the following the mean values across 20 test repetitions are reported and we obtained a confidence interval greater than or equal to 94%. Table 4 shows and describes all the *HyNeTD*'s input parameters and options. Its outputs are the lists of discovered *hosts*, *routers*, links, and *subnets*, provided both in text and XML format.

Table 3. Network Conditions

Name	Description
Passive	SNMP available on all the network routers. "DNS inverse look-up" feature enabled.
Active	SNMP disabled on all routers. "DNS inverse look-up" feature disabled.
Hybrid 1	SNMP disabled on all routers. "DNS inverse look-up" feature enabled.
Hybrid 2	SNMP available only on some network routers. "DNS inverse look-up" feature enabled.

Table 4. *HyNeTD*'s input parameters

Parameter's syntax	Parameter's description	Parameter's syntax	Parameter's description
-b base_address size [base_address size]	base addresses and dimensions of discovered address spaces	-d	enable DNS inverse name lookup
-c community_name [community_name]	SNMP's community names	-e	enable ARP table extraction
-r ret_number	max number of retries in packet sent	-n max_number_of_thread	maximum number of threads
-t time_out	timeout duration		

5.2 Ring Topology Experimental Results

Ring topology is characterized by a loop. Adopting an active methodology, and probing the network from a single point, the discovery of this loop fails. Indeed, in such a topology one router does not forward packets. In this case the

topology can not be correctly reconstructed and such a router is not recognized as a router. This sub-section shows that *HyNeTD*'s hybrid approach allows to correctly reconstruct this loop using a limited amount of passively collected data.

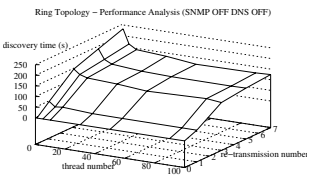

Fig. 4. Ring topology discovery time

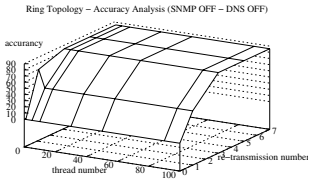

Fig. 5. Ring topology accuracy

Discovery Time and Accuracy Evaluation. Figures 4 and 5 show the measured discovery time and accuracy trends in the *Active* case as function of the number of threads, and of the number of retransmissions. The discovery time is increasing with the number of retries and decreasing with the number of threads. The first dependence can be easily explained taking into account the extra waiting times due to each retransmission. The second dependence is related to a faster execution of the steps 1 and 2. There is no relation with the SNMP information retrieval. Indeed, the current implementation of *HyNeTD* does not parallelize the step 3. In all other network conditions we have found the same behaviour. Table 5 shows, for all network conditions, (i) the minimum and the maximum measured discovery time with and without cross traffic (column 2 and 3), (ii) the maximum theoretical and the achieved accuracy (column 4). Considering that we used a high amount of cross traffic ($100Mbps$), we measured a very light dependence on it. We found the same accuracy, a difference in the discovery time that can be neglected in the worst cases also varying the number of threads (see figure 6), and the same tendency for both these metrics. Therefore the following results represent well both the cases with and without cross traffic.

Table 5. Ring topology discovery time and accuracy

Network Conditions	No cross traffic	Cross traffic	Accuracy
Passive	**min:** $3.74s$ **max:** $141.10s$	**min:** $9.64s$ **max:** $148.45s$	**theoretical:** 100% **max. measured:** 100%
Active	**min:** $21.6s$ **max:** $232.023s$	**min:** $23.64s$ **max:** $232.17s$	**theoretical:** 86% **max. measured:** 86%
Hybrid 1	**min:** $8.5s$ **max:** $160.29s$	**min:** $10.45s$ **max:** 160.33	**theoretical:** 86% **max. measured:** 86%
Hybrid 2	**min:** $9.64s$ **max:** $148.44s$	**min:** $14.4s$ **max:** $175.72s$	**theoretical:** 100% **max. measured:** 100%

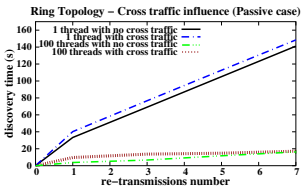

Fig. 6. Ring top. cross traffic analysis

As for the accuracy, we found the highest values in case of *Passive* and *Hybrid 2* conditions. In case of *Active* and *Hybrid 1* conditons the active methods are predominant, therefore there is a greater dependence on packets lost. *HyNeTD* considers that a packet is lost when *time_out* is elapsed. It is possible to accelerate the discovery process by reducing *time_out*, but some packets may be erroneously considered lost. In order to achieve the maximum theoretical accuracy in both *Active* and *Hybrid 1* conditions (that is equal to $\frac{R_d}{R_r} = \frac{6}{7} \simeq 0.86$) retries number was set equal to 3.

Network Overhead Analysis. The overhead produced by *HyNeTD* depends on several factors (i.e. network conditions, address spaces size, etc.). In this experimental analysis two address spaces were explored for a total amount of 90 addresses. Figure 7 shows sent and received data (in bytes) in all the network conditions. The *Passive* case presents the maximum overhead, whereas the lowest overhead is produced in case of *Hybrid 1*. This can be explained with a high amount of information retrieved from the DNS with a "low overhead". As for the produced overhead, considering the worst case (the *Passive* one), *HyNeTD* produces traffic with a mean rate that range from about $0.680 KBps$ to about $10.4 KBps$ (it depends on the maximum number of threads). Considering that the explored topology is composed of $100 Mbps$ links this overhead may be neglected.

5.3 Backup Topology Experimental Results

Backup topology is characterized by the presence of a backup path. We considered this topology in order to verify if *HyNeTD* is capable to discover backup paths in case of network conditions different from the *Passive* one.

Discovery Time and Accuracy Evaluation. Table 6 shows that *HyNeTD* achives the minimum discovery time in the *Passive* case. In the *Hybrid 1* case it reaches similar performances. In the *Active* case it shows the worst discovery time. Finally, in the *Hybrid 2* case *HyNeTD* presents an intermediate discovery time. As for the accuracy, *HyNeTD* discovers correctly all routers, links, and subnets in all the considered network conditions. Such result shows that *HyNeTD* is capable to discover backup paths without using passive information sources.

Table 6. Backup topology discovery time and accuracy

Network Condition	No cross traffic	Accuracy
Passive	**min:** $6.27s$ **max:** $124.17s$	**theoretical:** 100% **max. measured:** 100%
Active	**min:** $24.34s$ **max:** $204.44s$	**theoretical:** 100% **max. measured:** 100%
Hybrid 1	**min:** $7.51s$ **max:** $129.4s$	**theoretical:** 100% **max. measured:** 100%
Hybrid 2	**min:** $18.47s$ **max:** 159.17	**theoretical:** 100% **max. measured:** 100%

Network Overhead Analysis. The higher amount of overhead is produced in both *Passive* and *Active* network conditions (see Figure 8). The overhead produced in the *Active* case grows with the number of retries. Otherwise, in the *Passive* case it can be assumed independent on the number of retries. In the *Hybrid 1* network condition *HyNeTD* produces the minimum overhead. This result confirms the "high quality" of information retrieved using the DNS.

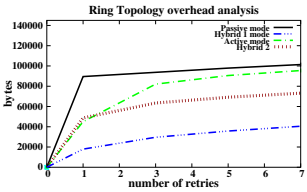

Fig. 7. Ring top. network overhead

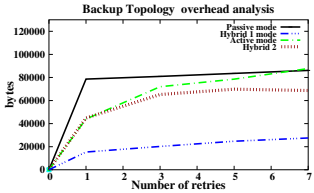

Fig. 8. Backup top. network overhead

5.4 Linear Topologies Experimental Result

In this case we consider a family of *linear topologies* composed of a number of routers that ranges from 1 to 6. We used this family to evaluate the scalability of our tool. In testing how *HyNeTD* scales with respect to number of routers belonging to the discovered topology we considered especially the *Passive* and *Active* network conditions. Indeed, such conditions may be assumed respectively as the lower and the upper bound for the topology discovery times.

Discovery Time Analysis. Figure 9 shows the measured discovery times in case of *Passive* and *Active* conditions. The measured discovery time is quite independent from the number of routers in the *Passive* case, and highly dependent on it in the *Active* case. Hence, *HyNeTD* presents optimal scalability properties when used as a passive discovery tool. In the *Hybrid* cases the discovery time is always lower than the one measured in the *Active* case, and greater than the one measured in the *Passive* case. It is possible to state that *HyNeTD* is scalable when used as a hybrid discovery tool if in this case the discovery time is found to be close to the *Active* one. In order to verify if our hybrid approach is scalable, we considered the *Hybrid 2* network condition. Moreover, to consider a condition that is as much as possible close to the *Active* one we activated the SNMP agent on only one router. Figure 10 shows the discovery time in the case of *Active*, *Passive*, and *Hybrid 2* network conditions (values reported for the *Hybrid 2* are the worst ones measured varying the router on which the SNMP agent is active). In the *Hybrid* case, values are always less than one-half of that measured in the *Active* case. Moreover, they are very close to the *Passive* ones for the first three topologies, and their rates grow slower with respect to the *Active* case. Therefore, this preliminary analysis shows that *HyNeTD* hybrid approach scales better than a pure active approach.

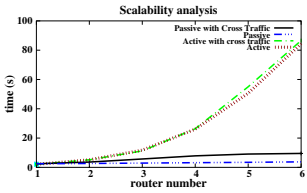

Fig. 9. Passive and Active cases

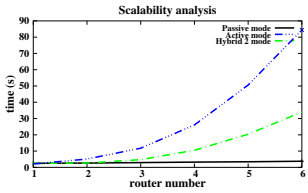

Fig. 10. Hybrid case

Network Overhead Analysis. As for the produced/received traffic in both the *Active* and *Passive* cases, it grows with the number of routers. In the *Passive* case we found a linear growing rate coherently with the constant size of the SNMP messages. In the *Active* case we found a "super-linear" growing rate coherently with the factorial growing of the number of information needed for the alias resolution process. Therefore, it is possible to assume that the *Active* mode is less scalable than the *Passive* one.

6 Conclusions and Future Works

In this paper we proposed a platform for the topology discovery of IP networks at router level. We presented the methodologies which our platform is based on as well as its performance analysis over classical network topologies. The proposed hybrid approach showed good performance in terms of accuracy, network overhead, and discovery time. Obtained results confirmed our hypothesis on the power of hybrid approaches. Indeed, the hybrid approaches are capable to adapt their behavior to the different network conditions. Moreover, due to the combined use of several techniques we showed how our hybrid approach permits to reach better performance overcoming the lacks of both active and passive methods. As for scalability, we found that our hybrid methodology presents an optimal behavior with respect to pure active approaches. Results presented in this paper are related to experiments over a controlled test-bed on a small-scale. Currently, we are testing the system behavior on realistic networks of a much wider-scale (i.e. Universitá di Napoli network). Preliminary results (both in terms of performance and scalability proprieties) confirming the behavior that HyNeTD has shown over a laboratory test-bed. We will extend our platform to discover "Layer 2" topologies too. We plan to adopt a hybrid methodology composed of several techniques [17] [18] able to take into account results from several data sources. Due to the low overhead of *HyNeTD* we plan to use it also in a distributed fashion, placing several monitoring probes over the network under control and partitioning the address space. The distributed monitoring platform will be enhanced with the use of tools we implemented for available bandwidth measurements [21]. Our objective is the development of a unified framework allowing to monitor and manage topology as well as links status.

References

1. C. Gkantsidis, E. Zegura, "Experiment and learn to discover Network Topology", Oct. 99 Georgia Tech.
2. R. Siamwalla, R. Sharma, and S. Keshav,"Discovering internet topology", Tech. Rep., Cornell University, Jul. 1998.
3. R. Govindan and H. Tangmunarunkit, "Heuristics for Internet map discovery", *Proc. of* IEEE Infocom '00, Mar. 2000.
4. K. C. Claffy and D. McRobb. "Measurement and Visualization of Internet Connectivity and Performance", `http://www.caida.org/TOOLS/measurement/skitter/` (As of July 2005).

5. R. Govindan and A. Reddy, "An Analysis of Internet Inter-Domain Routing and Route Stability", *Proc. of* IEEE Infocom '97, Apr. 1997.
6. H. W. Braun and K. C. Claffy, "Global ISP interconnectivity by AS number", `http://moat.nlanr.net/AS/` (As of July 2005).
7. H. Chang, S. Jamin, and W. Willinger, "Inferring AS-level Internet Topology from Router-level Traceroutes", *Proc. of* SPIE ITCom '01, Scalability and Traffic Control in IP Networks, Aug. 2001.
8. H. Chang et al., "On inferring as-level connectivity from BGP routing tables", Technical Report UM-CSE-TR-454-02, 2002.
9. P. Barford et al., "The Marginal Utility of Network Topology Measurements". *Proc. of* ACM/SIGCOMM Internet Measurement Workshop, Nov. 2001.
10. P. Dinda et al., "The Architecture of the Remos System," *Proc. of* 10th IEEE Symp. on High-Perf. Dist. Comp. (HPDC'01), Aug. 2001.
11. G. Mansfield et al., "Techniques for automated Network Map Generation using SNMP". *Proc. of* Fifteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Mar. 1996.
12. Wood et al., "Fremont: A System for Discovering Network Characteristics and Problems". *Proc. of* Winter USENIX Conference, Jan. 1993.
13. N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP Topologies with Rocketfuel". *Proc. of* ACM/SIGCOMM '02, Aug. 2002.
14. HP, "HP Management software", `http://www.openview.hp.com` (As of July 2005).
15. IBM, "Tivoli software", `http://www.tivoli.com` (As of July 2005).
16. University of California at Davis, "The UCD-SNMP project home page", `http://www.ece.ucdavis.edu/ucd-snmp/` (As of July 2005).
17. B. Lowekamp et al., "Topology Discovery for Large Ethernet Networks". *Proc. of* ACM SIGCOMM '01, Aug. 2001.
18. R. Black et al., "Ethernet Topology Discovery without Network Assistance", *Proc. of* 12th IEEE International Conference on Network Protocols (ICNP'04), Oct. 2004.
19. S. Avallone et al., "Performance evaluation of an open distributed platform for realistic traffic generation", Performance Evaluation: An International Journal, Vol. 60 issue 1/4, pp. 359-392, Mar. 2005
20. A. Lakhina, J. Byers, M. Crovella and P. Xie, "Sampling Biases in IP Topology Measurements", *Proc. of* IEEE Infocom, Apr. 2003.
21. A. Botta et al., "BET: A Hybrid Bandwidth Estimation Tool", *Proc. of* IEEE ICPADS, PMAC-PDG'05 Workshop, Vol. 2, pp. 520-524, Jul. 2005.