OnTimeMeasure: A Scalable Framework for Scheduling Active Measurements

Prasad Calyam, Chang-Gun Lee, Phani Kumar Arava, Dima Krymskiy, David Lee
The Ohio State University, Columbus, Ohio 43210.
pcalyam@oar.net,cglee@ece.osu.edu,{arava,krymskiy,lee}@cse.ohio-state.edu

Abstract

In order to satisfy and maintain Service Level Agreements (SLAs) which demand high network availability and good network health, ISPs have started instrumenting their networks with Network Measurement Infrastructures (NMIs) that are composed of dedicated measurement servers. Active measurements are frequently used in NMIs to regularly monitor network health and analyze the experience of end-user application traffic traversing the network. However, active measurements initiated by measurement servers need to be regulated. Unregulated active measurement traffic can cause an unpredictable negative impact on the actual application traffic. Also, running simultaneous conflicting active measurements on measurement servers could result in misleading reports of network performance. In this paper, we describe our active measurements scheduling framework called ”OnTimeMeasure” that allows ISPs to regulate the amount of active measurement traffic injected into the network and also prevents conflicts in ongoing active measurements between measurement servers. OnTimeMeasure provides a simple scripting language interface to specify various measurement requirements such as physical topology of measurement server clusters, periodicity of the measurements, and properties of measurement tools. For a given measurement requirements script, OnTimeMeasure uses an efficient heuristic bin-packing algorithm to generate measurement timetables for orchestrating active measurements for a network involving multiple measurement servers, each hosting multiple measurement tools.

Keywords

Active Measurements, Scheduling, Heuristic Bin-packing, Network Monitoring

1. Introduction

Since some of the early and basic network protocols did not address many essential requirements for network measurements in their own specifications, it has become necessary to have Network Measurement Infrastructures (NMIs) in today’s networks. NMIs are used to monitor Internet traffic characteristics on an ongoing basis so as to understand the interactions and performance issues of various Internet protocols. NMIs support active measurement data collection, which requires injecting test packets between the various network paths being monitored. The active measurement data obtained provide network related information such as: topology, available or bottleneck bandwidth, packet one-way delay, round-trip delay, loss, jitter and reordering, which can be analyzed to report levels of end-to-end network performance and to identify end-to-end performance bottlenecks.

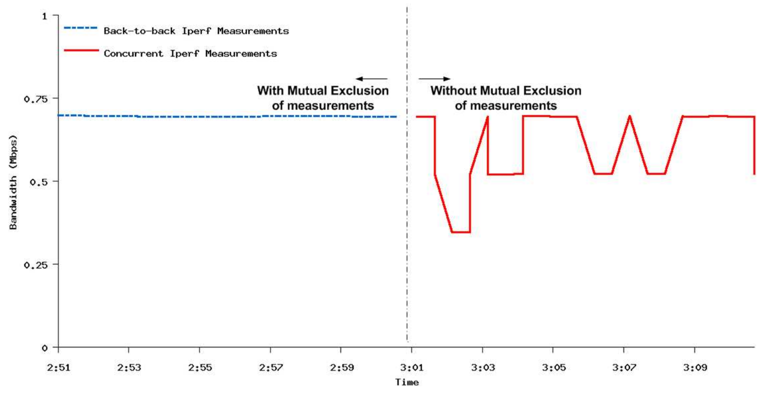


Figure 1: Iperf test results with and without mutual exclusion of measurements

Traditionally, active measurement tools such as Ping and Traceroute were used to determine round-trip delays and network path topology by using ICMP packets. Recently, active measurement tools such as H.323 Beacon [1], Multicast Beacon [2] and BGP Beacon [3] have been developed that emulate application-specific traffic and use the resulting measurement of the emulated traffic to estimate network health from the application perspective. Tools such as Pathchar [4] and Pathrate [5] have also been developed to determine available bandwidth and bandwidth capacity respectively in network paths using sophisticated packet probing techniques. Many of the above active measurement tools and other tools such as OWAMP [6], Iperf [7] and ABwE [8] have been integrated into NMIs [9] [10] [11] [12] used by ISPs and researchers to periodically perform active measurements between a set of measurement servers located at strategic points in both academic and commercial networks.

Since obtaining active measurement data is at the expense of consuming network bandwidth which might be required by actual application traffic, active measurements need to be regulated in NMIs. Also, the methodology used for performing periodic active measurements needs to be addressed carefully. If proper care is not taken, the results of active measurements might falsely indicate the existence of end-to-end performance problems in the network when actually the results are influenced by the error-prone procedures used to collect the active measurement data. The error-prone procedures involve system and network resource contention problems which arise due to lack of mutual exclusion of active measurements that are CPU and channel intensive.

Figure 1 illustrates the effect of scheduling Iperf measurements back-to-back and concurrently in an isolated LAN Testbed between two measurement servers with WAN emulation by NISTnet [13] in the intermediate path. To obtain realistic results, traffic of an H.323 Videoconferencing session at 768Kbps dialing speed was used as cross traffic in the testbed while the active measurements were being initiated using the Iperf tool. It can be observed in the case of concurrently scheduling Iperf measurements (shown in the right-half of Figure 1), the lack of mutual exclusion causes incorrect representation of the intermediate network path health in comparison to the network path health correctly indicated by the Iperf measurements scheduled in a back-to-back fashion (shown in the left-half of Figure 1). In [14], a similar set of observations have been presented to demonstrate the incorrect reporting of bandwidth measurements when measurement probes are intentionally made to collide with each other. The above reasoning for the need for mutual exclusion of measurements can also be extended in cases of other tools such as the H.323 Beacon and Pathchar.

In addition to the need for ”correctness” as described in the previous paragraph, ”efficiency” of scheduling active measurements also needs to be addressed in NMIs. Efficiency refers to the design goal of a scheduler to minimize the total schedule time for facilitating fast repetition of the measurement schedules. A faster repetition of measurement schedules permits more frequent probing to obtain network health information for any given network path in an NMI. A higher frequency of probing leads to a better understanding of the actual performance of the network path.

In this paper we propose a novel scheduling framework that not only regulates but at the same time enables correct and efficient scheduling of active measurements initiated in NMIs. The proposed scheduling framework called ”OnTimeMeasure”, which has actually been implemented in the NMI we developed to monitor our ”The Third Frontier Network” (OARnet Network Backbone), consists of a scripting language interface and a scheduling engine. The scripting language interface provides a generic way to specify various measurement requirements in a form the scheduler engine can comprehend. For a given set of measurement requirements, the scheduling engine solves the scheduling problem using a heuristic bin packing algorithm and automatically generates measurements timetables for a specified number of measurement servers and measurement tools. The measurement timetables are used to activate jobs that do not misreport active measurement results while still minimizing the total schedule time; i.e., maximizing the measurement repetition frequency.

The remainder of the paper is organized as follows: Section 2 details the related work on active measurement scheduling, Section 3 discusses the various scheduling requirements for active measurements in NMIs, Section 4 describes the OnTimeMeasure framework, Section 5 presents the performance evaluation of the heuristic bin packing algorithm used in OnTimeMeasure in comparison with the adhoc and round-robin scheduling schemes and Section 6 concludes the paper.

2. Related Work

As described in Section 1, NMIs need to use network measurement ”schedulers” while setting up periodic or sporadic active measurements to regulate active measurements and to efficiently avoid pitfalls caused by resource contention among multiple simultaneous measurements. NMIs developed to date, however, have not paid much attention to the problem of active measurements scheduling. Many of the todays NMIs use adhoc or round-robin cron job scheduling. In adhoc scheduling, the cron jobs that initiate active measurements are configured without any considerations for avoiding collisions of measurements between multiple measurement servers. This approach results in erroneous measurement results that do not accurately reflect the network conditions. NMIs which are more advanced, use round-robin scheduling in view of avoiding collisions in initiating active measurements. This approach is not scalable in cases where the number of measurement servers in the NMI increases over time. NMIs such as [9] use resource broker scheduling for every active measurement tool on a measurement server. Such a solution again fails to scale in terms of optimum schedulability of measurements as the number of measurement tools and measurement servers increase.

NMIs used in applications such as Network Weather Service (NWS) [15] use token-passing mechanisms for active measurements scheduling. Since NWS utilize active measurements to forecast network performance using time-series models, the token passing mechanisms are required to maintain consistency in periodicity of active measurements in addition to ensuring mutual exclusion of measurements between multiple measurement servers; only a server in possession of a token is permitted to initiate measurements to all the other measurement servers in the NMI. Such token-passing schemes are better in terms of scalability in comparison with the round-robin and resource broker schemes when the number of measurement servers increase. However, the token-passing schemes are not scalable and can become complex and unreliable, when there are multiple tools on each measurement server and when each of these tools needs to be synchronized using tokens with multiple measurement tools of other measurement servers.

3. Scheduling Requirements for Active Measurements

In addition to the consistency in periodicity and mutual exclusion of measurements as required in applications such as NWS, there are many other requirements that need to be considered while designing measurement schedulers. The following subsections describe the requirements, which have not been well-addressed and in some cases not-addressed at all, by the existing scheduling schemes mentioned in Section 2.

3.1 Frequency of Scheduled Measurements

In order to be able to forecast network health or to capture significant network health disorder events or to regulate the amount of active measurements in the network, a certain frequency of an active measurement task may need to be set between measurement servers. E.g. a Ping test may be required to run once every 5 minutes whereas an Iperf test may be required to run once every hour. Hence, the scheduler must schedule various measurements such that a requirement of fixed periodicity of active measurements is satisfied between a set of measurement servers

3.2 On-demand tests Scheduling

In addition to the regularly scheduled tests in the network, there may be times when customized on-demand tests might need to be initiated between certain measurement servers without disrupting the regularly scheduled tests. For example, if tests of H.323 Beacon using G.711 codec have been setup as regularly scheduled tests between two measurement servers, a certain authorized network engineer might need to initiate a specialized test of the H.323 Beacon with G.723 codec between the same two measurement servers. Such sporadic active measurement requests may need to be handled by the scheduler to

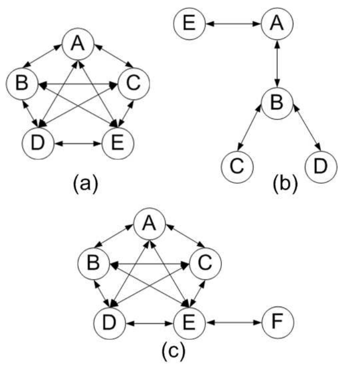


Figure 2: Measurement Cluster Topologies- (a) Full-Mesh (b) Tree (c) Hybrid

make online scheduling decisions that affect the overall measurement schedule of the measurement servers.

3.3 Measurement-Cluster Topology based Scheduling

For the target network that needs to be monitored, we can specify strategic measurement sites forming different measurement-cluster topologies such as full-mesh, tree and hybrid topologies. Figures 2(a) - 2(c) show the full-mesh, tree and hybrid measurement cluster topologies respectively. A full-mesh topology involves testing from a measurement server at a site to every other measurement server at other remote sites; a tree topology involves testing between measurement servers in neighboring sites only and a hybrid topology involves supporting subsets of measurement servers that need full-mesh and tree topology type measurements. The topology specification determines the set of measurements that the measurement scheduler has to schedule. Consequently, the scheduling mechanism needs to be designed so as to accommodate any generic specification of measurement-cluster topology.

3.4 Adherence to Measurement Level Agreements

Since identifying end-to-end performance bottlenecks could involve analyzing data along network paths of multiple ISPs, we can envisage ”measurement federations” in which many ISPs participate in inter-domain measurements based on some Measurement Level Agreements (MLAs) for reaping the mutual benefits of performing end-to-end path measurements. MLAs could specify that only a certain percentage (1 - 5) % or only a certain number of bits per second (1-2) Mbps of the network bandwidth in ISP backbones could be used for measurement traffic that can ensure that actual application traffic is not seriously affected by measurement traffic [*]. Thus, when active measurements in a network or between networks are required to adhere to MLAs, a scheduler must be able to accordingly generate measurement schedules.

[*]Since most active measurement tools have options to specify packet sizes and bandwidth usage of a measurement test, simple calculations can be used to determine how much of a network’s bandwidth will be used by a given set of active measurements, over a certain period of time.

4. OnTimeMeasure Framework

In this section, we describe our active measurements scheduling framework, which we call ”OnTimeMeasure” that can systematically and more efficiently address the various scheduling requirements discussed in Section 3, in comparison to other existing active measurements scheduling schemes.

4.1 Heuristic Bin Packing Algorithm

4.1.1 Terminology:

(a) Measurement Tasks and Jobs - A measurement task T$_i$ consists of a sequence of jobs $J^i_{S_x-S_y}, J^i_{S_x-S_z}, J^i_{S_y-S_x}, ...$ that need to be executed periodically between measurement servers *x,y,z,...* for *n* cycles between a start time t$_{start}$ and an end time t$_{end}$, such that-

$$t_{end} > t_{start} + n\delta \qquad (1)$$

where δ is defined as ”cycle time”, which is described in the next subsection. To better understand the above definition of tasks and jobs, consider an example of a measurement task as follows- Given a cycle time of 120 minutes, let task T$_1$ involve running a full-mesh Iperf test between 3 measurement servers:S_1, S_2 and S_3. Consequently, $J^1_{S_1-S_2}, J^1_{S_1-S_3}, J^1_{S_2-S_1}, J^1_{S_2-S_3}, J^1_{S_3-S_1}, J^1_{S_3-S_2}$ represent the corresponding jobs of T$_1$, which get initiated once every 120 minutes.

(b) Cycle Time - ”Cycle time” δ parameter is the time-window during which, a complete round of all scheduled jobs are executed; i.e., it’s the time within which one unique set of all the measurement jobs that were scheduled between all the measurement servers are performed using the appropriate tools specified in the measurement requirement specifications. Cycle time is the parameter that dictates the ”efficiency” design goal of a scheduler as described in Section 1. Larger the number of measurement servers or tools in the measurement requirement specifications, larger is the cycle time. Also, shorter the cycle time, the more frequently we can obtain network health information for any given network path, which results in a better understanding of the performance of the network paths in an NMI. Hence, the goal of an efficient scheduler must be to minimize the cycle time as much as possible.

(c) Scheduler Bin - A scheduler bin corresponds to a time fraction of the cycle time within which measurement jobs are packed. The concept of a bin is useful in applying different heuristic bin packing algorithms where multiple jobs can be scheduled within a bin in an overlapped fashion. The criterion for overlapping jobs within the same time frame or bin, for a given set of measurement servers and measurement tools, is based on the structure of the ”Tool Conflict Graph” and ”Link Conflict Graph”, both of which are explained in the following subsections. In comparison to the jobs scheduled using heuristic bin packing algorithms, jobs scheduled using round-robin bin schemes are ordered sequentially in time without any overlap of jobs.

(d) Tool Conflict Graph - A Tool Conflict Graph, created for a given set of measurement tools, is used for making scheduling decisions in the process of execution of our heuristic bin packing algorithm. This graph indicates if any two tools should or shouldn’t be run with mutual exclusion on a given measurement server. The reason for ensuring

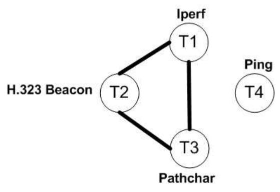


Figure 3: Tool Conflict Graph for tools in Table 1

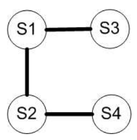


Figure 4: Link Conflict Graph for a tree-type measurement cluster topology

Task#	Measurement Tool	CPU Intensive	Channel Intensive
T1	Iperf	Yes	Yes
T2	H.323 Beacon	No	Yes
T3	Pathchar	Yes	No
T4	Ping	No	No

Table 1 Measurement Tools CPU and Channel Resource Consumptions

mutual exclusion (as discussed in Section 1) is to avoid ”CPU” and ”Channel” resource contention when initiating tests from measurement tools that are- either CPU or channel intensive or are both CPU and channel intensive.

Table 1 shows measurement tools that either consume or don’t consume CPU and channel resources based on the measurement technique used in a given tool. For example, Pathchar does not consume significant channel resources at any given point of time, since it uses a series of ICMP packets for probing. However, Pathchar uses complex statistical analysis based on the ICMP packet responses and determines: Latency and bandwidth of each link in the path, distribution of queue times at intermediate hops along a path and the probability that a packet is dropped. In our tests involving Pathchar, we have observed that the calculations involving the reporting of the above measurement metrics consume significant CPU resources.

Figure 3 shows the tool conflict graph for the measurement tools listed in Table 1. An ”edge” connecting two tasks implies mutual exclusion has to be ensured in executing the two tasks on a given measurement server; i.e., an Iperf measurement task cannot be concurrently scheduled along with an H.323 Beacon measurement task. However, it is acceptable if an Iperf measurement task is concurrently scheduled along with a Ping task. A tool that has an edge to any other tool is considered to have an edge to itself.

(e) Link Conflict Graph - A Link Conflict Graph also uses a similar ”edge-based” mutual exclusion concept seen in the Tool Conflict Graph, except, the decision of ensuring mutual exclusion is done at the link level when scheduling measurements between two measurement servers versus the decision that is made at a single measurement server level in the case of a Tool Conflict Graph. For example, the Link Conflict Graph shown in Figure 4 for a tree-type cluster topology of measurement servers implies that- if a mea-

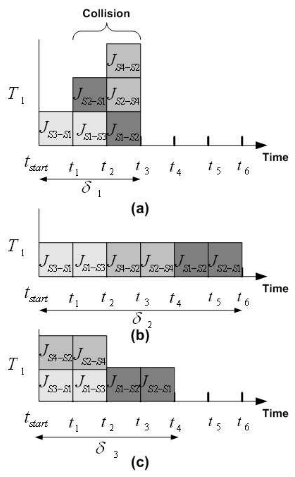


Figure 5: Active Measurements Scheduling Schemes (a) Adhoc Packing (b) Round Robin Packing (c) Heuristic Bin Packing

surement task has been scheduled between S1 and S3 (S1-S3), the following measurement tasks: S3-S1, S1-S2 and S2-S1, cannot be concurrently scheduled in the time frame within which the S1-S3 task is being executed. However, either of the measurement tasks: S2-S4 or S4-S2 will be allowed in the same time frame as the S1-S3 measurement task.

4.1.2 Methodology:

Figures 5(a)-(c) show the active measurement jobs scheduling by the adhoc, round robin and heuristic bin packing schemes respectively for the measurement cluster topology shown in Figure 4. For simplicity in explanation, we assume in Figures 5(a)-(c) that jobs of only one measurement task T_1, which is both CPU and Channel intensive, need to scheduled between the measurement servers S1, S2, S3 and S4.

As shown in Figure 5(a), in the adhoc packing schemes where there are no considerations for mutual exclusion of measurements, jobs are randomly scheduled leading to collisions in the bins present in between the t_1 and t_3 time frame. Figure 5(b) shows the same set of jobs being scheduled using the round robin packing scheme, so as to ensure mutual exclusion between the measurements. The jobs are sequentially placed without any overlap of jobs in any time frame.

In Figure 5(c), the jobs of measurement task T_1 have been scheduled using the heuristic bin packing scheme, so as to ensure mutual exclusion between the measurements. A job is placed in a bin even if there is a job that has already been scheduled in the same bin on the condition that there is no ”edge” present in both the tool conflict graph and link conflict graph involving the two jobs. As seen in Figure 5(c), jobs $J_{S_4-S_2}$ and $J_{S_2-S_4}$ are scheduled in the same time frame as the jobs $J_{S_3-S_1}$ and $J_{S_1-S_3}$ since both S2 and S4 don’t have edges to S1 and S3.

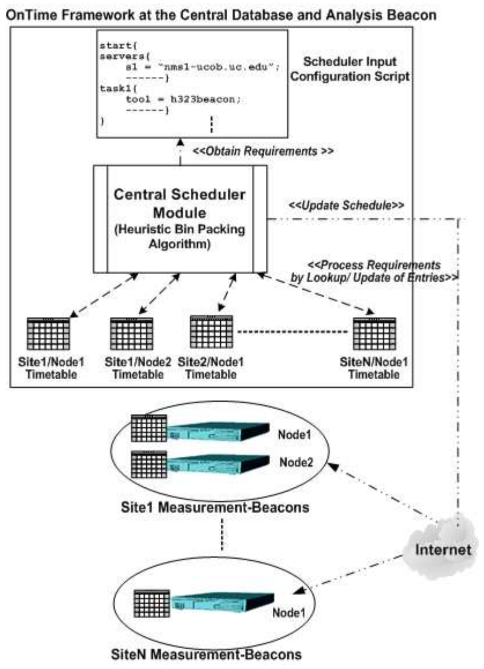
δ_1, δ_2 and δ_3 shown in Figures 5(a) - (c) represent the cycle times obtained by scheduling the jobs using the adhoc packing scheme, round robin packing scheme and heuristic bin packing scheme, respectively. Although, we can observe that the adhoc packing scheme performs best in terms of cycle time compared to the round robin and heuristic bin packing schemes ($\delta_1 < \delta_2 < \delta_3$), the adhoc packing scheme is inefficient since it results in erroneous measurement results. Also, it is obvious from Figures 5(a)-(c) that the round robin packing scheme though achieves mutual exclusion of measurements, it is inefficient in terms of minimizing the cycle time in comparison with the heuristic bin packing scheme. Hence, the heuristic bin packing scheme provides an efficient approach to measurement jobs scheduling in terms of both correctness of measurements and in terms of minimizing the cycle time for any given set of measurement scheduling requirements.

4.2 System Implementation

OnTimeMeasure has been designed from our experience of designing an active measurements scheduler for an end-to-end network performance measurement testbed spanning campus, regional and national academic backbone network paths. We initially implemented a round robin scheduling scheme for our measurement testbed. When we started expanding our testbed into an NMI for our Third Frontier Network (OARnet Network Backbone), we realized the round-robin scheme was inefficient. This led to our design and consequent implementation of OnTimeMeasure in our NMI which is explained in detail in the following subsections.

4.2.1 Preliminary Implementation:
We initially developed a preliminary version of OnTimeMeasure for a pilot testbed [12] we built as part of the Third Frontier Network Measurement Project which has been funded by the Ohio Board of Regents. Our pilot testbed consists of paths interconnecting The Ohio State University, University of Cincinati and North Carolina State University campuses. The pilot testbed has been built to study empirical end-to-end network performance bottlenecks in campus, regional and national academic backbone networks. Each measurement site has two measurement servers connected to the routers at strategic points in the network. CDMA time sources have been deployed at each of the sites, making them Stratum-1 Network Time Protocol (NTP) Servers, for obtaining precise global clock synchronization for one-way delay measurements. Each of the measurement servers is equipped with a network measurement toolkit that comprises of many open-source measurement tools. Our preliminary version of OnTimeMeasure consisted of a ”central scheduler” that generates measurement ”timetables” for each of the measurement servers in the pilot testbed to orchestrate active measurements initiated between the measurement servers. The timetable for each measurement server specifies the times at which a cron job executes a particular tool test on the measurement server. Detailed explanation of the workflow of the preliminary implementation of our scheduling framework can be obtained from [12].

4.2.2 Revised Implementation Architecture:
There were many limitations in the design of our preliminary version of OnTimeMeasure. There was no mechanism to systematically specify scheduling requirements in terms periodicity or measurement cluster topology or MLAs that needed to be incorporated into the scheduling decision process. This caused a lack of flexibility in the design of the scheduler to automatically address various requirements such as frequency and consistency of


OnTime Framework at the Central Database and Analysis Beacon
Scheduler Input Configuration Script
<<Obtain Requirements >>
Central Scheduler Module (Heuristic Bin Packing Algorithm)
<<Update Schedule>>
<<Process Requirements by Lookup/ Update of Entries>>
Site1/Node1 Timetable
Site1/Node2 Timetable
Site2/Node1 Timetable
SiteN/Node1 Timetable
Node1
Node2
Site1 Measurement-Beacons
Internet
Node1
SiteN Measurement-Beacons

```
start{
servers{
    s1 = "nms1-ucob.uc.edu";
    ------}
task1{
    tool = h323beacon;
    ------}
}
```

Figure 6: OnTimeMeasure Framework.

periodicity of active measurements in the testbed paths. Also, there was no support for incorporating on-demand tests into the timetables.

Towards addressing the above limitations, we extended the OnTimeMeasure framework with a ”Scripting Language Interface” and the ”Heuristic Bin Packing Algorithm”, described in Section 4.1, to generate the measurement timetables as shown in Figure 6. The scripting language interface provides a generic and automated way to specify various measurement requirements such as periodicity information, measurement cluster topology, MLAs and any on-demand tests. The measurement requirements specifications collected using the scripting language interface are used by the heuristic bin packing algorithm in determining timetables for each of the measurement servers in the network being monitored. The heuristic bin packing algorithm is part of the ”Central Scheduler Module” (CSM) that resides in the ”Central Database and Analysis Beacon” server as shown in Figure 6. The roles of the ”Central Database and Analysis Beacon” server are to get the input specification of active measurements, centrally schedule the specified active measurements, collect all the measurement data from the measurement servers and also to analyze, summarize and visualize the collected measurement data in real-time.

4.2.3 Scripting Language Interface:

We have developed a simple scripting language that can be used for writing scheduler-input configuration scripts. We have also developed a script-interpreter that parses the scheduler input configuration scripts to organize the various measurement requirement specifications such as: measurement server topology description, measurement tools de-

```
//OnTimeMeasure scheduler-input config script for-
//Third Frontier Network Beacon Infrastructure Testbed
//Last Modified: 12th September 2004
start {
          servers{
               s1 = "nms1-ucob.uc.edu";
               s2 = "nms2-osul.osu.edu";
               s3 = "nms1-ncsl.nc.edu";
               s4 = "nms3-oual.ou.edu";
                s5 = "nms2-utob.utol.edu";
          }

          cluster{
               topology = hybrid; //cluster topology
               fullmesh = s1, s2, s3, s4; //servers in full-mesh topology
               tree = s4, s5; //servers in tree topology
                treelinks = s4-s5; //adjacent tree links
          }

           task1 {
                tool = h323beacon;
                tooloptions = -a, -text;
                impact = high;
                characteristic = vvoip; //measures voice and video performance
                direction = bi; //bidirectional path data generated
                duration = 300; //5 minutes per job
                mutualexclusion = yes; //mutual exclusion must be strictly enforced
           }

           task2 {
                tool = ping;
                tooloptions = -n 100;
                impact = low;
                characteristic = rtt; //measures round-trip delay for a path
                direction = uni; //unidirectional path data generated
                duration = 60; //1 minute per job
                mutualexclusion = no; //mutual exclusion need not be strictly enforced
           }

          //choose any above specified task for scheduling jobs
           periodic{
               task1_start_time = 2004-09-14:00.00.00; //yyyy-mm-dd:hh.mm.ss
              task1_end_time = infinite; //run forever
              task2_start_time = 2004-09-14:00.00.00;
              task2_end_time = infinite;
           }

           ondemand{
                //specify task#, start and end time to add ondemand jobs
               //blank tag implies, there are no ondemand jobs to be scheduled
           }
} //end of scheduler-input config script
```

Figure 7: Sample scheduler-input configuration script to specify active measurement requirements

scriptions, periodicity desired for the measurement tasks, MLAs and on-demand measurement tasks related information.

Figure 7 illustrates the format of a sample scheduler-input configuration script. Using a ”servers” tag, an end-user can specify the measurement servers for which measurement timetables need to be generated. A ”cluster” tag indicates the measurement cluster topology for all the measurement servers that include details of specific interconnections of the topology. A ”task#” tag is used to describe tool-specific information that is used for both generating the measurement timetables and for initiating the tests appropriately. Multiple task numbers can be created for a single tool that has multiple switch options. Just specifying a task in a configuration script does not suffice inclusion of the task into the measurement timetables; the task also needs to be included in either the ”periodic” or ”ondemand” tags along with start and end times for the initiating multiple jobs of that particular task. An ”mla” tag is also part of the scripting language syntax where, information regarding bandwidth limits allowed for consumption can be specified for each/all tasks.

5. Performance Evaluation

In this section, we evaluate the performance of the heuristic bin packing scheme in comparison with the adhoc packing and the round robin packing schemes. Collision Rate is used as a metric to compare the correctness of measurements performed using the heuristic bin packing scheme, with the adhoc packing scheme. Collision Rate provides an estimate of the number of incorrect measurements caused due to lack of mutual exclusion between a set of measurements initiated with a tool which is either/both CPU and Channel intensive. Cycle Time Economy is used as a metric to compare the savings obtained in the overall cycle time using the heuristic bin packing scheme over the round robin packing scheme.

5.1 Collision Rate

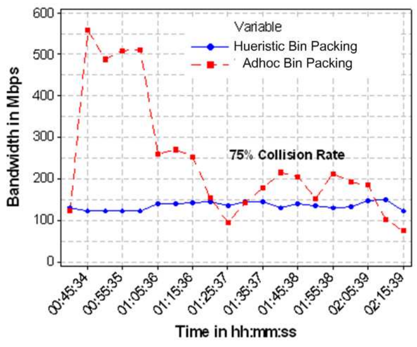


Figure 8: Pathchar measurement results when running Pathchar and Iperf measurements with and without mutual exclusion

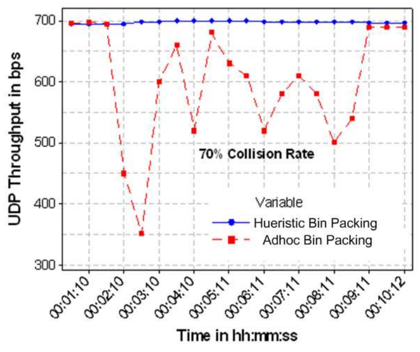


Figure 9: Iperf UDP Throughput measurement results when running two Iperf m easurements with and without mutual exclusion

The results plotted in Figure 8 have been obtained by initiating Pathchar and Iperf measurements in the same ”LAN testbed with WAN emulation” described in Section 1. A total of 40 Pathchar tests were initiated during a test period. The first 20 tests that took about 2 hours to complete, involved initiating Pathchar tests alone ensuring mutual exclusion between successive tests using the heuristic bin packing scheme for the testbed. The results thus obtained as shown in Figure 8 are consistent with the actual network conditions affected by the cross traffic. The next 20 tests, which also took 2 hours to complete, involved randomly initiating Iperf tests in conjunction with Pathchar tests using the adhoc packing scheme. We observe that in this particular set of random tests, 15 of the 20 tests (75%) resulted in collisions. Given the almost ideal conditions in our LAN testbed for such a rate of collisions, we believe that the misrepresentation of the actual network conditions could be even more pronounced in the real Internet where there are more variables that could skew the correctness of active measurements.

Figure 9 shows the UDP throughput results obtained from initiating 40 Iperf tests in

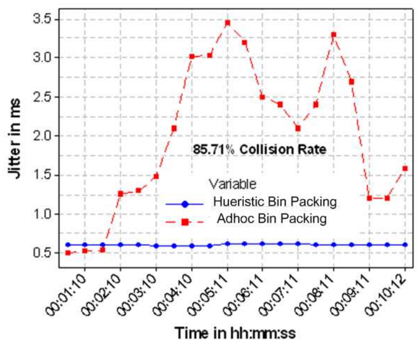


Figure 10: Iperf Jitter measurement results when running two Iperf measureme nts with and without mutual exclusion

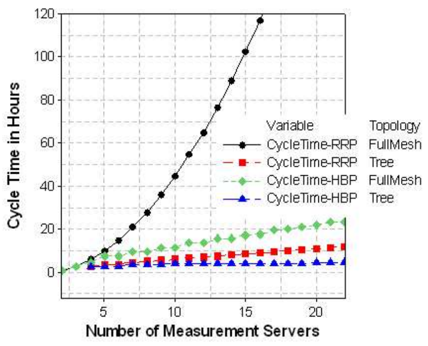


Figure 11: Comparison of Cycle Times for Round Robin Packing and Heuristic Bin Packing Schemes for the full-mesh and tree measurement cluster topologies

the same LAN testbed with WAN emulation that had a cross traffic of a 768 Kbps H.323 Videoconferencing session. Figure 10 shows the corresponding jitter values reported for the above 40 Iperf tests. The first 20 tests that took about 10 minutes to complete, involved initiating Iperf tests alone by ensuring mutual exclusion between successive tests using the heuristic bin packing scheme. The next 20 tests, which also took about 10 minutes to complete, involved randomly initiating Iperf tests in conjunction with the other Iperf tests, by using the adhoc packing scheme. We observe that in this particular set of random tests, the interference caused by conflicting tests resulted in 70% and 85.71% erroneous measurements of UDP throughput and jitter, respectively.

5.2 Cycle Time Economy

Figure 11 shows the simulation results for cycle times obtained, with increase in the number of measurement servers, by using the round robin and heuristic bin packing schemes. For the above simulation, jobs of 3 tools were used for scheduling. All the 3 tools were assumed to be both CPU and Channel intensive with job execution times of 5 minutes, 5 minutes and 20 minutes, respectively. A bin size of 20 minutes was chosen so as to fit the largest of the jobs to be scheduled.

We can observe that significant savings were obtained in cycle time for the full-mesh cluster topology measurements that require *n(n-1)* measurements, by using the heuristic bin packing scheme instead of the round robin packing scheme; *n* being the number of measurement servers. However, the savings in cycle time for the tree cluster topology measurements, though were noticeable, were not as significant as in the case of full-mesh cluster topology measurements. Simulations for hybrid cluster topology measurements were in accordance with the results obtained for the full-mesh and tree cluster topology measurements. For a hybrid cluster topology with a dominant number of full-mesh cluster measurement nodes, the savings in cycle time were significant. Whereas, for a hybrid cluster topology with a dominant number of tree cluster measurement nodes, the savings in cycle time were relatively less.

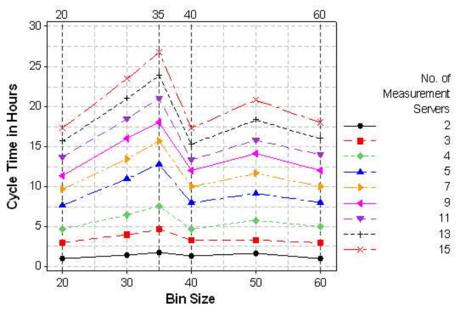


Figure 12: Effects of various bin sizes on the Cycle Times for the Heuristic Bin Packing Scheme

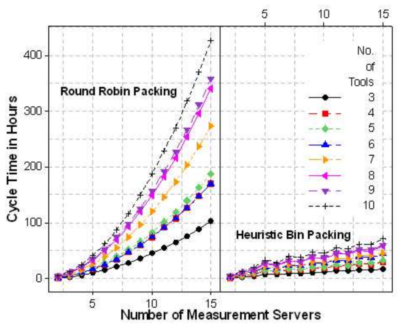


Figure 13: Effects of increase in the number of measurement tools on the Cycle Times for the Heuristic Bin Packing Scheme

Figure 12 shows the simulation results obtained for the same set of 3 tools scheduled on a full-mesh measurement cluster topology, with varying bin sizes. It can be noted that choosing an optimum bin size, based on the execution times of the jobs to be scheduled, can further increase the savings in cycle time; i.e., choosing a bin size of 20, 40 or 60 results in a much less cycle time than choosing a bin size of 35. The above result as shown in Figure 12 is true irrespective of the number of measurement servers for which measurement jobs need to scheduled. It should be noted that the optimum bin size for a given set of jobs depends on the individual execution times of the jobs.

Figure 13 shows the simulation results obtained for a full-mesh measurement cluster topology, with a fixed bin size of 20 and with an increasing number of tools. We can note that in both the round robin and heuristic packing cases, as the number of measurement tools increase for a given number of measurement servers, the cycle time increase has a trend; The cycle time increase is almost exponential in the case of round robin packing scheme whereas the cycle time increase is almost linear in the case of the heuristic bin packing scheme.

6. Conclusion and Future Work

In this paper, we detailed the various motivations and scheduling issues concerned with using active measurements in NMIs. We discussed various scheduling schemes, being used in existing NMIs, and their limitations. To address our identified limitations, we proposed a novel active measurements scheduling framework called ”OnTimeMeasure”. Lastly, we showed how a scripting language interface and a heuristic bin packing algorithm can be used to address active measurements scheduling requirements in a systematic, less error-prone and efficient fashion for a network involving multiple measurement servers, each hosting multiple measurement tools.

We are currently investigating techniques such as distributed hierarchical scheduling to extend our our heuristic bin packing algorithm to function in a distributed scheduler environment. Further, we are planning on integrating techniques such as shared private keys and MD5 cryptographic checksums into the distributed scheduling framework of OnTimeMeasure. Such security techniques will be particularly suitable for inter-ISP measurements and have the potential to prevent abuse of NMI federations by network intruders.

ACKNOWLEDGMENT
This work has been supported in part by The Ohio Board of Regents.

References

[1] P. Calyam, W. Mandrawa, M. Sridharan, A. Khan, P. Schopis, ”H.323 Beacon: An H.323 application related end-to-end performance troubleshooting tool”, ACM SIGCOMM Network Troubleshooting Workshop, 2004.
[2] Multicast Beacon - http://dast.nlanr.net/Projects/Beacon
[3] Z. Mao, R. Bush, T. Griffin, M. Roughan, ”BGP Beacons”, Internet Measurement Conference, 2003.
[4] Pathchar - http://www.caida.org/tools/utilities/others/pathchar
[5] C. Dovrolis, P. Ramanathan, D. Morre, ”Packet Dispersion Techniques and Capacity Estimation”, IEEE/ACM Transactions on Networking, 2004.
[6] S. Shalunov, B. Teittelbaum, ”One-way Active Measurement Protocol (OWAMP)”, IETF RFC 3763, 2004
[7] Iperf - http://dast.nlanr.net/Projects/Iperf
[8] J. Navratil, L. Cottrell, ”ABwE: A Practical Approach to Available Bandwidth Estimation”, PAM, 2003.
[9] Internet2 piPES Project - http://e2epi.internet2.edu
[10] SLAC Pinger Project - http://www-iepm.slac.stanford.edu/pinger
[11] NLANR AMP Project - http://watt.nlanr.net
[12] P. Calyam, D. Krymskiy, M. Sridharan, P. Schopis, ”TBI: End-to-end network performance measurement testbed for empirical bottleneck detection”, IEEE TRIDENTCOM, 2005.
[13] NISTnet network emulation package - http://snad.ncsl.nist.gov/itg/nistnet
[14] B. Gaidioz, R. Wolski, B. Tourancheau, ”Synchronizing Network Probes to avoid Measurement Intrusiveness with the Network Weather Service”, IEEE High-performance Distributed Computing Conference, 2000.
[15] R. Wolski, N. Spring, J. Hayes, ”The network weather service: A distributed resource performance forecasting service for metacomputing”, Future Generation Computer Systems, 1999.