Resource Management considerations in Collector Overlay Networks

Mukundan Venkataraman, Shamik Sengupta, Mainak Chatterjee
School of EECS
University of Central Florida
Orlando, FL 32816
E-mail: {mukundan,shamik,mainak}@cpe.ucf.edu

Raja Neogi
Radisys Inc.
15797 NW Andalusian Way
Portland, OR 97229.
E-mail: raja_neogi@ieee.org

Abstract

Streaming monitorables on an overlay plane orthogonal to the data-path is emerging as a promising way to monitor, detect and isolate network impairments and outages. However, keeping the overlay stable and efficient poses acute challenges, since the overlay network itself shall not be monitored for outages. A linear mapping of exporter elements onto collectors often leads to a sub-optimal resource allocation, frequently saturating a few collectors leading to gaps in monitored data. Such gaps could prove costly for network operators, and a most common reaction to this is expensive hardware upgrades. In times of such outages, however, it is observed that not all collectors are saturated. More often than not, balancing the load and efficiently distributing available resources was all that was needed to prevent saturation related outages, and increase overall capacity of the overlay. We discuss ways to construct a scalable, robust, and resilient overlay that can function autonomously given the constraints of a typical real world access network.

1 Introduction

Network service providers of VoD and IPTV have a pressing need to detect, isolate and correct network impairments within an access network. An access network is a protected series of nodes that acquire or capture streaming content for distribution to end-users, with each access network maintained by an individual service provider. Access networks typically run the UDP/IP protocol stack. UDP is a best effort service, which provides no service guarantees. Further, it provides no feedback to the sender, which makes it impossible to detect packet loss without the addition of explicit feedbacks. IP is a lightweight network layer protocol, and there is no guarantee that the network will not lose, re-order, fragment or drop packets.

Some solutions have tried to add explicit feedbacks mechanisms about packet delivery, and mostly work by round trip time estimation and time-outs upon a failure to receive an acknowledgement. Accurate RTT estimation has been proven to be non-trivial. As a result, false timeouts or long failure detections are not uncommon. Besides, RTTs may be too long to learn about, or recover, from an error. Since continuous playback demands a steady arrival of packets for consumption, a delayed packet is as good as a lost packet.

Thirdly, existing quality evaluation schemes are either housed on end nodes or require the original frame for reference to calculate degradation. Examples would be metrics like PSNR, VQM [3], MPQM [2] etc. PSNR requires the original frame for reference, and schemes like MPQM are too complex and require extensive hardware. Further, though these schemes can provide an estimation of degradation given the necessary inputs, they can never say what caused the degradation or where in the network it occurred.

We propose a hierarchical network of decentralized nodes. A schematic diagram of the overlay is as shown in Fig. 1, with the nodes organized as a tree. Designated nodes on the data path (called "exporters") gather and export various network usage and anomaly statistics about a given flow, like loss, delay, jitter etc. on a plane orthogonal to the data-path. These flows are mapped to a series of "collectors" found one level higher on the overlay tree. Collectors take incoming packets from various exporters, and convert them to some form of a health score. Examples of such conversions could include computing a mean opinion score (MOS [6, 5, 3]), which on a scale of 0 to 5, can provide an interpretative health score of a given network segment. After aggregation, and certain forms of suppression, MOS scores from various collectors make it to the root of the overlay tree: the analysis and control (ANCON) node, providing a global perspective of the entire access network here. With an even placement of exporters, root cause analysis can now take the granularity of delay rates or loss rates in individual segments of the access network.

Such an architecture could prove to be the basis for performing a host of actions at ANCON. Such actions could include: terminating flows, suggest workarounds, provide feedbacks to source nodes reporting a problem in the net-

0-7695-3094-X/08 $25.00 © 2008 IEEE
DOI 10.1109/ICNS.2008.42

work, alert service providers about outages, and possibly implement autonomous self-healing algorithms. Deploying an overlay of such a nature would not normally require a huge investment on a service providers part as well, since such an infrastructure is already present to gather billing information. In other words, we simply leverage the existing infrastructure, making such a scheme easy to deploy.

Streaming monitorables on the orthogonal plane poses stiff networking challenges [1, 7]. Such a network must be economical, robust, resilient and minimalistic to be embraced by service providers. Further, this overlay itself shall not be monitored by any other external network, which necessities the upkeep of the overlay to be an autonomous procedure.

Some questions on the feasibility of such an architecture arise: (i) How scalable is the architecture with number of data-flows? (ii) How will traffic within the overlay be engineered?, (iii) How will outages within the overlay itself be handled?; and (iv) What are the implications of the this design to a service provider, or, what minimum set-up is required to deploy this service?

The purpose of this paper is to provide guidelines for the design of a scalable, robust and resilient overlay architecture that can autonomously monitors parameters, even in the face of outages. Such an architecture would be minimalistic, and would significantly increase performance capacity with a given deployment of collector nodes. In our experience, there is a need to correctly map traffic out of exporters to an available pool of collector nodes. While a straightforward choice of streaming to a nearest collector node may be simple to implement, such an allocation may inadvertently saturate a given collector node very fast. To most service providers, this translates to costly upgrades of the collector nodes to make them more capable. More often than not, an even distribution of load across the available pool of collector nodes sufficiently solves the saturation problem by allocating newer flows to unsaturated collector nodes. A distributed load that prevents collector saturation in turn increases the overall capacity of the overlay, keeping deployment costs to a minimum.

2 Architectural Overview

The topology can be loosely classified in two broad categories: (i) exporters, or elements that have information to stream, and (ii) collectors, elements that gather incoming parameters, and potentially pass them to higher nodes. A schematic overview is shown in Fig. 1, where the white nodes are exporters and the gray ones collectors. The plant portion of the access network terminates at about the EQAM as shown in the figure, and what follows beyond that is a series of passive nodes interconnected by passive HFC elements, which form the connection between the EQAM and an end user's set-top-box (STB). In our case, the STB's are the only elements capable of acting like an exporter for all the nodes between the EQAM and the end user. This means that a service provider can actively monitor all points from source to EQAM, and beyond that, just the STB. Such a design is justified, since errors potentially creep in active elements of the access network, and occasionally in the connection between the EQAM and the STB.

N	Number of data flows in the data path
S	Set which contains a list of all collectors
S_{sat}	Set of collectors that are saturated
S_{can}	Set of collectors that are candidates
E	Set of active exporters
e	Number of data flow exporters (pre-E-QAM, static)
s_i	Number of active set-top-boxes exporting parameters
c_k	The kth collector, from the superset S
p	Number of parameters per flow
n_i	Ingress flows at collector node i, $i \leq k$
e_{CPU}	% of CPU at exporters for extracting parameters
b_{CPU}	% of CPU at STB for extracting parameters
s_{CPU}	% of CPU at collector for processing *one* flow
b_{ui}	Bandwidth required for streaming p params
B_{ui}	Uplink bandwidth (egress) available at Collector i
b_{di}	Downlink (ingress) bandwidth for p params at node i
B_{di}	Downlink (ingress) bandwidth available at collector i

Figure 2. Tabular column of parameters used in modeling

We also observe that failures in segments beyond the plant boundary (after EQAM and before STB), can be easily derived by analyzing the signature of failed STBs, connected to a particular EQAM. Hence, for all outages within the access network, failure detection is fast because of an even placement of exporters. Beyond the EQAM, outages are detected as reported by STB's.

We abstract the loose bunch of exporters as a set E, and the collecting bunch of collector nodes as a set C. While the collector node pool is static, the exporter pool shows a variation in number captured by the following equation: $E = e + s_i$, which states that with an inception of a new flow, a new set-top-box is initiated as an exporter, and is subsequently added to the list E. Similarly, the set S denotes all set of all collector nodes in the topology. The elements of set S make up two other sets, $S_{saturated}$ and $S_{candidate}$, the set of collectors that are saturated and the set of collectors which are potential candidates for traffic mapping. Consequently, the following holds: $S = S_{saturated} + S_{candidate}$. The set of variables used in our algorithm is tabulated in Fig. 2. The allocation algorithm is shown as pseudo-code in Fig. 3.

3 Simulation results

We designed a discrete event system simulator in Java which closely models an access network as well as the collector overlay. The exporters in the access network

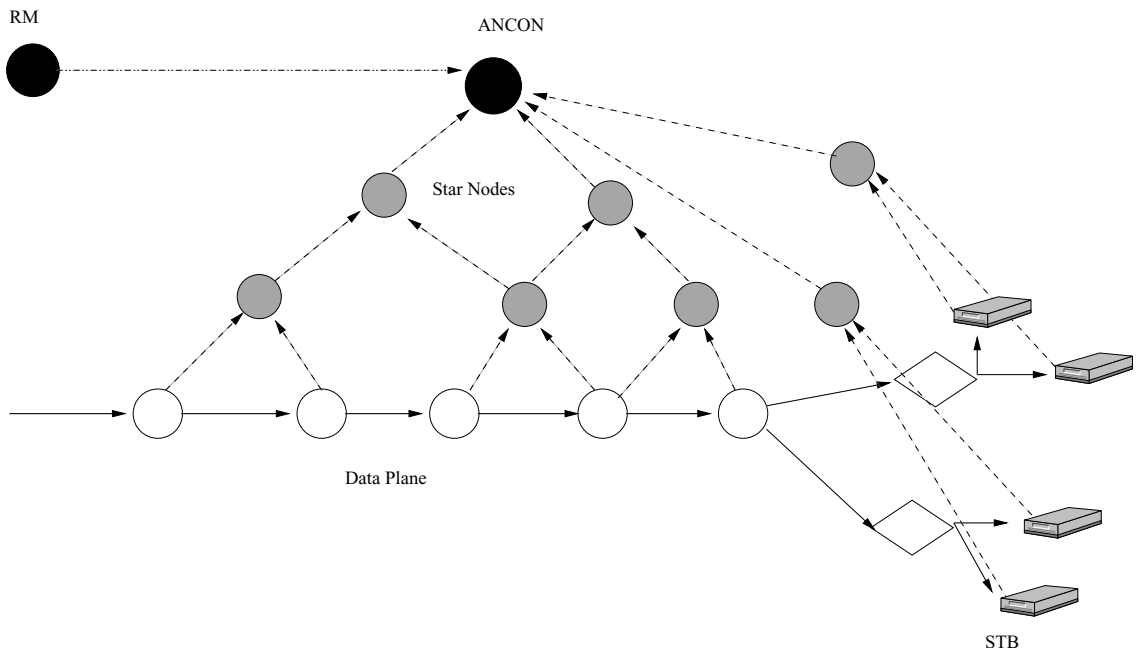

Figure 1. Architectural overview of the topology.

stream [4] parameters bundled in one single message per flow. It is possible to specify a "rate" (akin to a packet per given time interval) for the exporters.

Our model of simulation works as follows. Flows are introduced to the source at the access network, and the flow arrival is modeled as a poisson process. As a flow is admitted into the access network (which in conventional networks is done by the Resource Manager), exporters are configured to stream statistics for the given flow in question. The ANCON, in turn, begins the mapping algorithm for the given flow. In particular, it scans the list of candidate collectors to find a suitable mapping for all exporters to suitable collectors for that given flow. The allocation algorithm is as shown in Fig. 3.

The allocation algorithm scans the list of collectors in the set of candidate of potential collectors (i.e., collectors whose resources are not saturated). Upon a comparison of various parameters (like CPU saturation, buffer occupancy and bandwidth utilization), a decision on the choice of collector is reached. This is done for every list of exporter that is on the data path. The algorithm runs in $O(n^3)$ time, because a scan is made for all these cases: the number of flows, the exporters and the collectors. Note that this only happens one time: allocation at the start of the flow, and the deallocation (which runs in $O(1)$) at the flows termination.

3.1 Effect of increasing exporters

As the number of active exporters increase, we seek to understand packet delivery performance of the overlay. We simulate the case of 25 exporters, and 1000 active flows. Each flow generates a monitorable every 5 secs. In other words, this means that each exporter injects a monitorable every 5 ms into the overlay. Since the topology layout is such that all packets converge to one destination (the ANCON), the receiving capacity of this node must be sufficiently large to avoid gaps in monitoring. ANCON was configured to run continuously with little or no gaps, with packet reception and packet processing (processing a received packet to correlate) modeled as separate threads. If packets are sent to ANCON faster than it can receive and process, the input buffer gets saturated. Ingress traffic beyond this point is tail dropped, until the buffer begins to free. We measure ANCON's performance in terms of delivery ratio. This is defined as the ratio of the number of packets received and processed at ANCON, to the total number of monitorables generated at *all* the exporters for all the flows through the access network. As can be seen (Fig. 4), the delivery ratio (DR) drops with an increase in the number of active exporters. The nature of the drop in DR values is also noteworthy: while the drop is steep with an initial increase in exporters, the value begins to stabilize after a certain point in time. Also shown in the figure are the effects of varying the ANCON buffer size. The input queue size was varied from a capacity of 50 packets to a 100 packets in steps of 25. The plot shows two things: (i) the nature of DR variation remains the same, and (ii) the values of DR for various buffer sizes converge.

3.2 Variations in ANCON buffer size

To better understand the effect of variations in the ANCON buffer size, we conducted an experiment where we varied the buffer size from 10 packets to a 100 packets in steps of 10. The experiment had 50 exporters and 5 collectors. The number of flows were 1000 and the reporting

STEP 1. Resource manager grants requests for data flow $N+1$
STEP 2. ANCON initiates collector node mapping
STEP 3. Initialize export list $E \leftarrow e + s_{j+1}$
STEP 4. Identify collector with monitored parameters p_i
for $k = 1$ to $|E|$ **do**
  select e_k from E
  for $j = 1$ to $|S_{candidate}|$ **do**
    Select **candidate collector**, such that
    $B_{uj} > \frac{p_i}{l}$ **and** $B_{dj} > \frac{p_i}{l}$ **and** $E_{k,CPU} > e_{k,CPU}$
    if Candidate found **then**
      Map $E_k \rightarrow c_k$
      Update $S_{candidate}$, **do**
      $B_{uj} \leftarrow B_{uj} - b_{uj}$
      $B_{dj} \leftarrow B_{dj} - b_{dj}$
      $E_{k,CPU} \leftarrow E_{k,CPU} - e_{k,CPU}$
      if $B_{uj} < b_u$ **or** $B_{dj} < b_d$ **or** $E_{j,CPU} < e_{CPU}$ **then**
        move $c_k \rightarrow S_{saturated}$
      end if
    else
      Decrement parameter: $p_i \leftarrow p_i - 1$
      if $p_i < p_{min}$ **then**
        Throw *No collector match* exception
      else
        Go back to STEP 4, with p_i as input.
      end if
    end if
  end for
end for

Figure 3. Algorithm for flow allocation, which runs in $O(n^2)$

frequency was once every 5 secs per flow. We continue to measure performance as the delivery ratio (DR) defined earlier. As seen in the plot (Fig. 5), delivery ratio increases monotonically with larger buffer sizes. However, it begins to hit a saturation and continues to retain a steady value. Also, because of the high frequency of reporting, the delivery ratio is stuck at around 0.2 with increasing buffer sizes. This implies that little is gained by simply increasing buffer sizes, which is a common reaction when monitoring gaps are observed. This also indicates that a buffer size of at around 100 is mostly optimum to support the overlay traffic for the sort of scenarios that we consider.

3.3 Role of Collectors

Collectors are an intermediate hop between the exporters and the ANCON. If all the active exporters were to stream packets directly to ANCON, they would soon overwhelm ANCONs capacity to both receive and process information. The role of collectors as an intermediate is hence twofold: (i) they can quantitatively aggregate traffic towards ANCON by suppressing self-similar messages (for example,

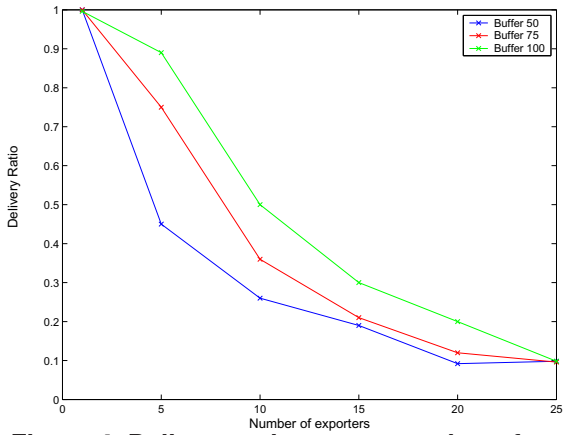

Figure 4. Delivery ratio versus number of exporters, for a given set of 10 collectors

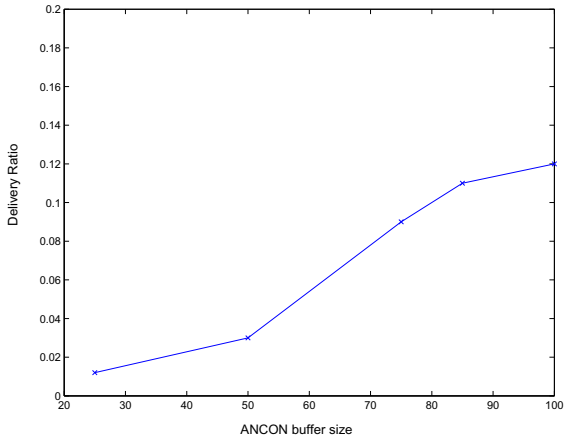

Figure 5. Delivery ratio versus the size of the ANCON input buffer

if an exporter reports the same statistic consecutively, the collector might decide to send just one packet by suppressing redundant ones), and (ii) collectors can perform qualitative suppression: by extracting various parameters from the packets, like delay, loss and jitter, they can compute interpretative health scores. An example os such a score would be a mean opinion score, or MOS.

A natural question with such a design is: how many collectors are ideally required for a given set of exporters? We seek to discover such a relationship by varying the number of collectors in the overlay by keeping the other parameters static and measuring delivery ratio. In our experiment, we chose 50 exporters, 1000 flows and a monitoring rate of 1 packet every 5 secs per flow. The variation in delivery ratio with an increasing number of collectors (Fig. 6) for the given set of 50 exporters is interesting. Delivery ratio improves with rising number of collectors to a maxima, and then drops in a non-increasing fashion. This clearly estab-

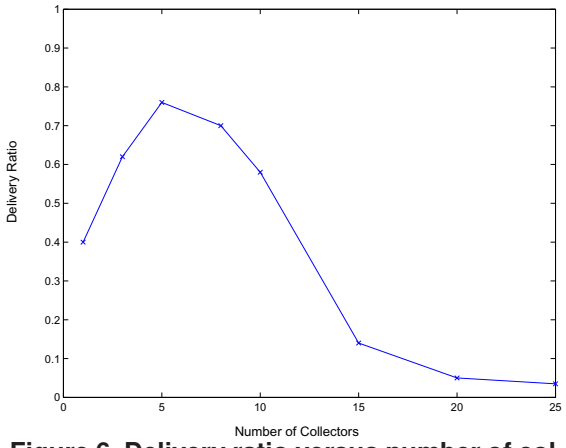

Figure 6. Delivery ratio versus number of collectors, for a given set of 50 exporters

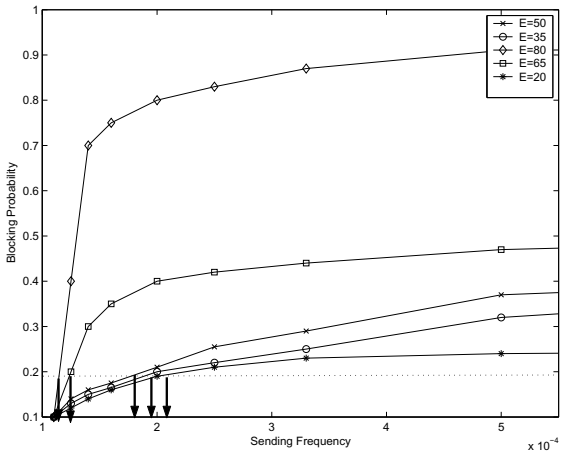

Figure 7. Blocking probability against source rate and number of exporters

lishes that there is a particular number of collectors that are indeed optimum for a given set of exporters.

Upon extensive investigation with various other values, we offer this following guideline as a simple rule of thumb. The optimum number of collectors is roughly one-tenth the number of active exporters within an access network.

3.4 Monitoring gaps: specifying minimum requirements

Blocking probability is defined as the probability that the demanded service is blocked, averaged over a sufficiently long interval of time. In our case, it is the probability that a monitored information is lost in the overlay before it makes it to ANCON. Lost information relates directly to gaps in monitoring: something a service provider would not want to see. What is ideally required is a form of quality assurance: given a requirement that the blocking probability be less than a certain value, what is the right number of exporters for such a set-up. Note that knowing the number of exporters is usually sufficient, since the right number of collectors can be arrived at by choosing one tenth the number of exporters.

To understand the nature of sampling on the number of exporters for a given threshold of , we first understand the behavior of blocking itself for a given streaming rate. We conducted an experiment where we measured the blocking probability for rising rate of monitoring at exporters. Blocking probability is defined as the probability that a packet generated at an exporter does not make it to ANCON. We choose a sufficiently large number of packets (5,00,000) over a 6 hour long simulation period to understand the nature of blocking within the overlay. We run each experiment for different set of collectors at least 10 times to smoothen out variations in performance. This first run is illustrated in Fig. 7. Blocking probability for any given set of exporters increases sharply with a slight increase in streaming rate. The value, however, stabilizes with further increase in the streaming rate, almost hitting a saturation beyond a certain point. Shown in Fig. 7 are cases for various number of exporters (20 to 80 in steps of 15).

Given such a behavior, we now apply quality assurance. We ask: what is the sampling frequency at ANCON if a service provider demands a blocking probability of less than 0.2 (i.e., at least 80% of monitorables should be received and processed at ANCON). We draw a horizontal line across the plot in Fig. 7 at around the value of 0.2 for blocking probability, and subsequently measure the correct sampling frequency given the maximum blocking probability.

The sampling frequency for such a case for various values of exporters is as shown in Fig. 8. The curve exhibits a parabolic trend, with the rate falling more steeply beyond a certain point (in this case, for $E > 50$). This study implies a few properties of this overlay: (i) it is possible to specify design with maximum blocking probability in mind, since there is an optimum number of exporters all blocking probabilities, (ii) there is an optimum number of collectors given a set of exporters; and, (iii) buffer sizes, and increasing them to improve capacity, does not necessarily work very well

3.5 Comparing load allocation to randomized allocation

Having studied the nature and dynamics of the overlay, we finally turn our attention to ensuring a balanced load distribution given the following: (i) a maximum blocking probability demanded, or in other words, a minimum quality assurance in monitoring, (ii) a correct set of exporters for such a case, and (iii) the correct set of collectors for this set-up. Having established the first three requirements in the previous sections, we now run our load-balancing algorithm to study the variations in load distribution across various collectors for a given set of 250 exporters, 25 collectors,

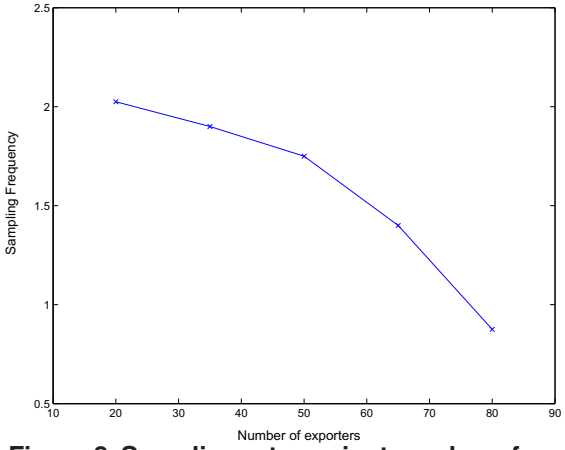

Figure 8. Sampling rate against number of active exporters, for a blocking probability of less than 0.2

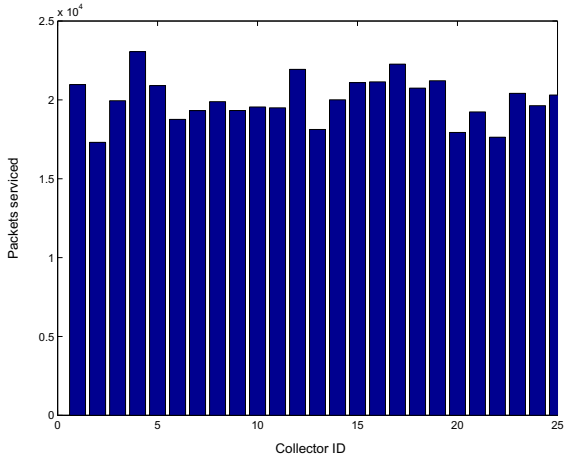

Figure 9. Distribution of number of packets serviced by a collector, for a case of 25 collectors

a monitoring rate of 1 packet every 5 secs per flow, for a set of 1000 flows. The simulation runs reported here run for a total of approximately 5,00,000 packets, for a simulation time of 6 hours. For 25 collectors, this means that a perfectly distributed load would result in approximately 20000 packets serviced per collector. The plot for this experiment is as shown in Fig. 9. We run our load distribution algorithm as flows arrive and depart (Fig. 3), and measure the actual number of packets serviced by every collector. As can be seen, the variance of load distribution is nearly minimum. While it is theoretically impossible to achieve a perfectly balanced system, our algorithm outperforms other schemes (like random allocation or a round-robin allocation), both in terms of load distribution as well as maximizing delivery ratio.

4 Conclusions

We have extensively investigated the feasibility of a hierarchical overlay of collectors that can monitor the health and usage in access networks. We have justified a two tiered architecture, with embedded exporters at the leaf and collectors at the middle layer.

Since ANCON is the single destination of choice for all monitorables, there is a cap on the number of exporters possible in an access network, for a given frequency of collection. Given the number of exporters, there is an optimum number of collectors which can maximize the capacity of the overlay. Roughly, it is about one-tenth the number of exporters.

Increasing buffer sizes in to improve performance is a weak alternative, and only provides marginal improvements. Blocking probability (BP), or gaps in monitoring, can be a design nucleus for such overlays. Specifically, it is possible to specify a tolerance for BP for an overlay. Likewise, the sampling rate as a parameter can be adjusted to guarantee the specified BP.

With most existing deployments unable to detect, isolate or diagnose failures at a network segment level, we believe that an architecture such as this could prove to be an excellent starting point to designing and implementing very effective fault-diagnosis mechanisms.

References

[1] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, R. Morris, "Resilient Overlay Networks", *Proc. 18th ACM SOSP*, Oct 2001.

[2] C. J. van den Branden Lambrecht and O. Verscheure, "Perceptual quality measure using a spatio temporal model of the human visual system", *Proc. IST/SPIE Conference Digital Video and Compression: Algorithms and Technologies 1996*, vol 2668, Feb 1996

[3] M. H. Pinson and S. Wolf, "A New Standardized Method for Objectively Measuring Video Quality", *IEEE Trans. on Broadcasting*, 50(3). Sept 2003.

[4] Internet Protocol Detail Record, `www.ipdr.org`

[5] D.S. Hands, "A basic multimedia quality model", IEEE Transactions on Multimedia, Volume 6, Issue 6, Dec. 2004, pp. 806 – 816.

[6] M. Yabe, S. Yokota, T. Nakajima, "A Quality Evaluation of High-Speed Video Streaming on Congested IP Networks", 6th Asia-Pacific Symposium on Information and Telecommunication Technologies, Nov. 2005, pp. 53 - 58.

[7] N. Miller and P. Steenkiste, "Collecting network status information for network-awareapplications", *IEEE Infocom'00*, Tel Aviv, Israel. March 2000. pp. 641 – 650.