Published in the proceedings of the
IEEE International Conference on Networking, ICN' 2001,
Colmer, France, June 2001, pp628-638.

# AN ACTIVE PROGRAMMABLE HARNESS FOR MEASURMENT OF COMPOSITE NETWORK STATES

Javed I. Khan and Asrar U. Haque

Internetworking and Media Communications Research Laboratories
Department of Math & Computer Science, Kent State University
233 MSB, Kent, OH 44242

**Abstract** In this paper we present the **active harness**, a scalable and versatile means for deep probing of network local states. The approach makes a clear separation between the "communication" from the "information" part of the probing process. The composition of the "information" component is handled by means of network embedded harness plug-ins. It can facilitate not only a new generation of network aware applications but also network problems.

## 1.Introduction

Measurement and exchange of state information inside a network is a meta service. The ability to construct any global optimization service, whether in network layer or above, depends on how well the decision systems are aware of and up-to-date on the relevant local states in pertinent network elements. For example, local link state information is the building block for routing services and BGP, OSPF, or MPLS [2] all require sophisticated cost metric from local nodes following complex propagation patterns. There are also host of other advanced network services (such as dynamic QoS provisioning, mobile IP forwarding), and new generation of applications (such as distributed server, proxy prefetch) all requiring exchange of such network state information. A prerequisite to any QoS provisioning, whether by dynamic resource allocation or by reservation, is the knowledge about the resources. Distributed caching requires information such as current congestion and current delay to assign best mirror server to a request. Clearly, with the complexity of the Internet and the push for optimized services and applications it is becoming increasingly more important to find the means for monitoring and propagating various complex network state measurements. In this paper we present the concept of **active harness** designed for provisioning this crucial meta internet service. Before presenting the proposed system below a brief account of the current state of the Internet probing technologies is given.

### 1.1. Related Work

*Simple Network Management Protocol* (SNMP), designed in 1990 and (RFC 1157 and RFC 1155) latest updated in 1995 was the first to provide a simple but systematic way of monitoring the Internet states. It's current version SNMPv2 [2,11] now provides a simple point-to-point measurement framework. It defines a set of important variables into a database called management information base organized into 10 categories of information (Current version MIB-II) and allows nodes to send simple point-to-point UDP based queries. SNMP has been very successful in providing the required standardized organization, syntax and representation (ASN.1) for a base set of critical

Published in the proceedings of the
IEEE International Conference on Networking, ICN' 2001,
Colmer, France, June 2001, pp628-638.

state information. However, its communication model is still point-to-point. Consequently, composite measurements cannot be made easily using SNMP. Because of the difficulty in its scalability, SNMP has been used more successfully in network management and monitoring than in dynamic run-time querying. Optimizing systems that need network wide information propagation therefore generally build their own custom mechanism. Routing information propagation in IGP, OSPF, BGP all are classical examples of this situation. Video multicast group management and synchronization require tracking of jitter and round trip delay. RTP and RTCP [12] have been proposed to collect custom time related statistics over a multicast tree. Similarly, *pathchar* [7], *cprobe, bprobe,* [1,9], *woodpecker* [3], and host distance estimation tool [5] have been proposed targeting various specific measurements. In this context, this research proposes a novel system that can address two emergent issues in network probing-- scalability and versatility, and these are illustrated below.

### 1.2. Scalability

Point-to-point mode of communication often severely limits scalability in a large network. For example, for measurement of path statistics the requesting node is required to send individual SNMP messages to all intermediate nodes. Consequently, redundant information flows inside the network increasing the overhead, severely reducing the transparency of the measurement process. Dissemination/aggregation of information with only a point-to-point communication means creates excessive traffic on the network severely limiting the scalability. It seems much of the limitations arises because SNMP cannot extract any intelligence from the intermediate nodes. Since there is no means for in network composition, all compositions must be done at the end-points, only after polling all state information there.

### 1.3. Versatility

On the other hand, the specific probing kits provide greater scalability but hardly can be reused for other measurements. Nevertheless, the trend suggests that versatility of the information is becoming equally important. Exactly, what measurement is useful depends on the optimization objective. For example, in a video server scenario, whether the jitter or the hard delay is more important is dependent on the specific video repair algorithm. In a different scenario, a server before sending data may want to poll information about the speed of only the last link to the home user's computers. In some other scenario the min/max of the path downstream delays and jitters from various junction nodes can help in strategically placing jitter-absorbing buffers in a multimedia streaming virtual private network tree. Emerging tele-interaction applications (such as tele-surgery, remote instrument control) will require handle on the delay incurred at the video frame level, which is exactly not the same as the packet delay. The trend suggests that as more advanced, and complex netcentric applications are being envisioned more versatile network state information would have to be exchanged.

### 1.4. Our Approach

Can scalability and versatility both be retained simultaneously? Apparently, there may not be any efficient answer in an end-to-end paradigm. In the general case, a network can choke with polynomial messaging at the end-points. However, the recent advent of **Active Network** technology seems to offer an innovative way out from this dichotomy

Published in the proceedings of the
IEEE International Conference on Networking, ICN' 2001,
Colmer, France, June 2001, pp628-638.

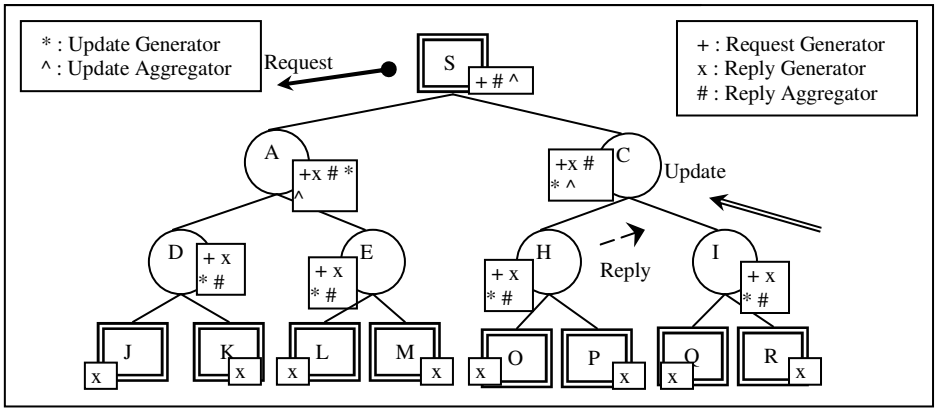


Fig-1 Components of Active Harness

[13,14,8]. Active network allows programmable modules to be embedded inside network junctions. In this research we propose an experimental dynamic mechanism for state information polling and propagation inside network with similar embedded information synthesizers, which seems to be both scalable and versatile. The approach first makes a clear separation between the "communication" from the "information" of the state exchange and propagation process. Communication is handled by the component called "harness". Harness propagates all information via coordinated messaging. On the other hand the "information" component of the process is controlled by a set of soft programmable plug-ins. These plug-ins decide the content of the messages propagated by the harness. In this paper in section 3 first presents the architecture of the proposed system. Section 4 then illustrates the operation of the harness for a typical application. Finally, in section 5 we share both analytical and empirical results depicting the performance of the proposed harness system.

## 2.Harness Architecture

The harness is responsible for initiating, propagating and responding to a series of well-coordinated messages between the nodes in a network. The harness once installed in network nodes, can act in three roles-- *initiator*, state *synthesizer*, and *terminals*. The initiator acts as the communication agent in the network layer for the application that actually requires the information. The synthesizer propagates the state requests and processes the returning states from the terminals.

The harness controls the communication pattern and thus deals with the efficiency of messaging. Harness system accepts a set of plug-ins, which determines the content of these messages, and how they are propagated and aggregated at the junction points.

### 2.1. Messaging

The harness system has been designed to operate with a novel request-reply-update messaging scheme. It has three types of messages *request*, *reply* and *update*. A request message contains fields indicating what information it needs, and dictating how far down the network the probing session should propagate, i.e. level, and any information needed by the receiver to compute required data, e.g. to compute the jitter a receiver needs to know the time stamp of sending successive data. The *request initiator* decides how often a request is generated. The request messages are sent to the *terminals* if they are immediately connected, or to synthesizers for further downstream propagation. A synthesizer upon receiving a request propagates the query by generating a new request message to the down-stream nodes. However, at the same time it might also generate

Published in the proceedings of the
IEEE International Conference on Networking, ICN' 2001,
Colmer, France, June 2001, pp628-638.

an immediate reply for the requestor. The replies from synthesizers may contain current local state and/or past remote states. The terminal nodes send replies to their respective requestors. The terminal reply contains locally retrieved current states. In the return trip of information, the synthesizer nodes aggregate the information and at each stage generate update messages for their requestors. Once a node receives all or specific number of update messages from its immediate down-stream nodes or on timeout, it updates the network local state variables and generates a new update message. The update message contains a synthesized summary of information calculated from all its immediate downstream nodes.

This three-part request-reply-update communication model, if needed, allows the information to be collected without working in lockstep. Even if a node downstream is delayed or silent, it does not hold the entire system; the estimation process can proceed for remaining nodes. The update phase is further equipped with optional and configurable timers to avoid update lockup. In essence, the request-reply phase allows collection of local immediate states. The reply mechanism allows immediate probing into current local states and past synthesized remote states, while the update message retrieves latest remote states.

### 2.2. State Composition

Harness system accepts a set of five plug-ins which are called *request generator*, *reply generator*, *reply aggregator, update generator,* and *update aggregator.* These modules together determine the content of these messages, and how they are aggregated at the junction points. They work via a virtual *slate*. A copy of which is maintained in each of the nodes. The slate works as the local abstract data structures. The slate is programmable and is defined at the session initiation phase. The *request generator* specifies the request message describing the fields it wants from the slate of its down-stream node. At individual nodes the model supports MIB-II and thus acts as a superset of SNMP. The terminal nodes can read/copy MIB variables (or their processed combination) existing in the local slate into variables marked for reply. The harness then invokes the reply messages with the designated slate variables. *Reply aggregator* (or *update aggregators*) in a similar fashion is invoked each time a reply (or update) is received by the harness. They perform domain specific processing of the reply message fields and similarly update their own slate variables. The *update generator* is invoked when a special trigger variable becomes true. The trigger variable is a set of conditions such as all, any, or a specified number of down-stream nodes have updated/replied, or a timer fires. The update generator sends the slate variables synthesized by the update aggregators to the upstream node. Fig-1 describes the architecture of the proposed harness system. It shows the roles, the typical locations of the plug-in modules and the direction of the messages.

At the heart of the composition ability is the transfer functions of the intermediate synthesizers. The request and update phase can be represented by equations:

$$S_t^j = \Phi(\vec{\mathrm{E}}(Q_t^{i,j,-}), M_t^j, S_{t-1}^j), \ \textit{and}\ Q_t^{-,j,k} = \vec{F}(S_t^t) \qquad \ldots.(1)$$

$$S_t^j = \Psi(\overleftarrow{F}(P_t^{-,j,k}), M_t^j, S_{t-1}^j), \ \textit{and}\ P_t^{i,j,-} = \overleftarrow{E}(S_t^t) \qquad \ldots.(2)$$

Published in the proceedings of the
IEEE International Conference on Networking, ICN' 2001,
Colmer, France, June 2001, pp628-638.

Here $S_t^j$ is the local *slate state* at event time t at node j, $\vec{E}$ is the *request receiving* filter (RRF), M is the *local network state* (such as MIB variable), $\vec{F}$ is the *request forwarding* filter (RFF). $Q_t^{i,j,-}$ is the arrived request from parent i, to node j and $Q_t^{-,j,k}$ is the propagated requests to children k. $\overleftarrow{E}$ is the *update forwarding* filter (UFF), $\overleftarrow{F}$ is the *update receiving* filter (URF). $P_t^{-,j,k}$ is the arrived update from child k, and $P_t^{i,j,-}$ is the propagated update to parent i. While, the filters determined the information propagation rules, composition functions $\Phi()$ and $\Psi()$ together determine the message content. While, in principle each of these components for each of the individual harness sites can be programmed differently, however the associated management will be intractable. In this harness we divide the network nodes into subsets based on their role in the topology. Nodes in the topological subsets then inherit uniform programmed behavior. Thus, we need only five distinct programmed modules (plug-ins) to be supplied by the harness programmer.

## 3.Harness Execution Model

The harness operates through 8 states. Fig-2 shows the state transition diagram. The oval shaped boxes describe activities and the square boxes indicate plug-in modules used for those activities. The initiator have states 1-3-4-5-1 if it generates a request of level 1 and if reply is expected, otherwise if level is 1 and reply is not expected then it has states 1-3-1. If the initiator generates a request message of level more than one, and reply and update is expected, then it has states 1-3-4-5-6-7-1. The terminal nodes have states 1-2-1. Parents of terminal nodes (if both reply and update is expected) have states 1-2-3-4-5-8-1. Other synthesizers (if both reply and update is expected) have states 1-2-3-4-5-6-7-8-1. As fig-2 suggests, other combinations are possible depending on whether reply and/or update is not expected.

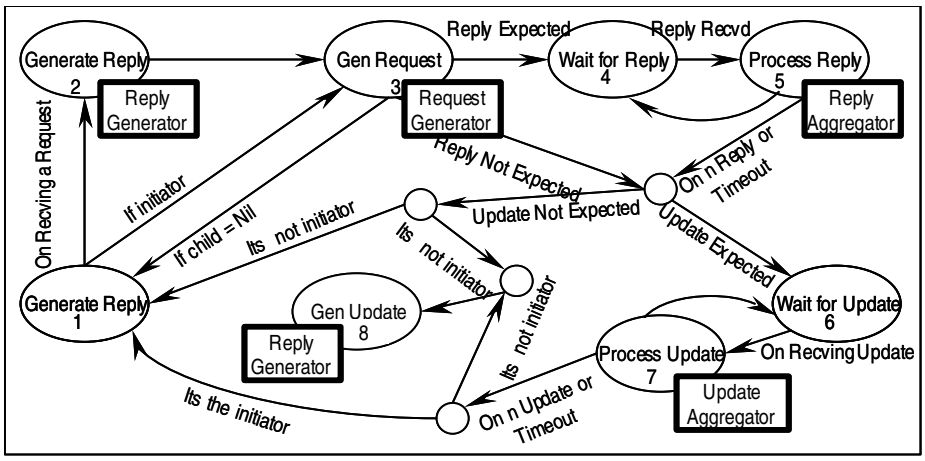


Fig-2 State Transition Diagram of Active Harness

Published in the proceedings of the
IEEE International Conference on Networking, ICN' 2001,
Colmer, France, June 2001, pp628-638.

## 4.Example Probing: Transcoder Step Computation

Below we present an example of custom harness service that helps in configuring an embedded video transcoding tree network. In a multicast distribution tree a video transcoder [15] sits in the junction nodes and steps down the video rate to match the downstream path capacities. The rate of the individual links however cannot be determined just with node or link local information. The step-down parameters are quasi-global state dependent. The optimum rate assignment is a function of the

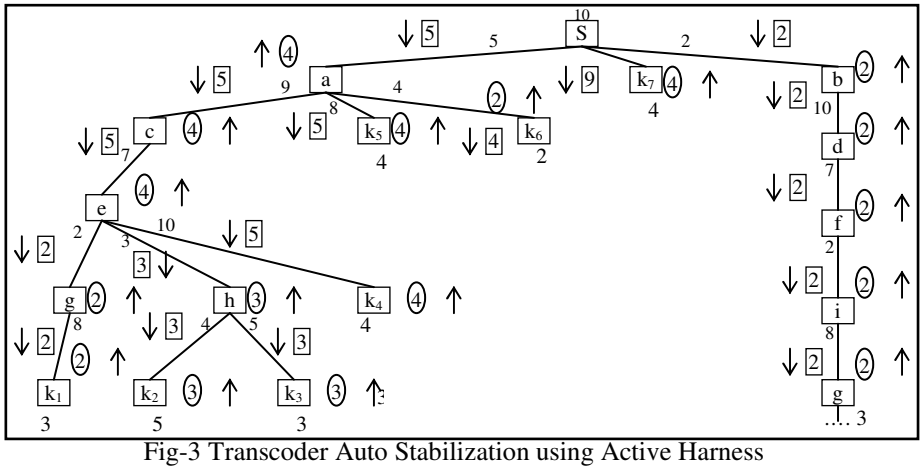


Fig-3 Transcoder Auto Stabilization using Active Harness

upstream and downstream paths, requirements of the sinks as well as the delivery capacity of the source. Below we show how the harness system can be 'programmed' to configure the transcoding network in a two-phase request/update cycle.

Let us consider the transcoding network of Fig-3. Each transcoder junction node has to know the amount of video it is getting from its parent via up-link and the maximum possible amount of video that can flow through all its down-links. The values have to be determined optimally so that it does not receive more than what it can deliver or what is required down the stream.

We map the initiator at the video source and the terminals at the video sinks and the synthesizers in each of the intermediate nodes. The slate, messages and the five plug-in modules are shown in Fig-4 (a)-(c). The process begins from the source. It sends a request, which contains its maximum delivery capacity. Each of the synthesizers regenerates a new downward request, which shows the *maximum deliverable video rate*, recomputed by the request generator plug-in modified by its local downward link

| | |
|---|---|
| AppxChildBW [ ] : float,<br>UsableChildBW [ ] : float,<br>TreeLevel : int,<br>MaxDownLinkTreeBW : float,<br>UpLinkBW : float,<br>SinkOrSourceCapacity : float<br>***(All are initialized to –1)*** | ***RequestMsg:*** MsgType:HarnessVariable, DeliverableRate: float,<br>Level: int<br>***ReplyMsg:*** MsgType: HarnessVariable, Recvable: float<br>***UpdateMsg:*** MsgType: HarnessVariable, MaxRecvable:float |
| Fig-4(a) Slate | Fig-4(b) Message Definitions |

Published in the proceedings of the
IEEE International Conference on Networking, ICN' 2001,
Colmer, France, June 2001, pp628-638.

```
Request Generator Module {
   if (it is a source)
      UpLinkBW= SinkOrSourceCapacity;
   for each child  I {
      RequestMsg.DeliverableRate
        = min (UpLinkBW, AppxChildBW[I]);
      >> Send RequestMsg to child [I]; <<
   }
}
Reply Generator Module {
   UpLinkBW=RequestMsg.DeliverableRate;
   if (Terminal Node)
      ReplyMsg.Recvable = min (UpLinkBW,
      SinkOrSourceCapacity);
  >> Send Reply Message; <<
}
Reply Aggregator Module {
   if (it parent of terminal I){
      MaxDownLinkTreeBW = max
      (MaxDownLinkTreeBW, ReplyMsg.Recvable);
      UsableChildBW[I] = ReplyMsg.Recvable;
      }
}
Update Generator Module {
   UpdateMsg.MaxRecvable = MaxDownLinkTreeBW;
   >> Send Update Message; <<
}
Update Aggregator Module {
   MaxDownLinkTreeBW = max
   (UpdateMsg.MaxRecvable,MaxDownLinkTreeBW);
    UsableChildBW[I] = UpdateMsg.MaxRecvable;
}
(Note: The messages enclosed by << and >> are
            send  by the harness)
```

Fig-4(c) Relevant Pseudocode of Modules

capacity. When, the terminal receives the request it then compares the value with its sink capacity, and determines the *bandwidth receivable* based on the deliverable and its sink capacity. The synthesizers above the terminals collect the *bandwidth receivable* values and compute their own bandwidth receivable based on the maximum demand and past value of *bandwidth deliverable*. The information eventually propagates upward to the source. In this example (fig-3) there is a source – S, a, b, etc. are intermediate nodes, and $k_1$, $k_2$, etc are video sinks. The numbers beside the edges denote bandwidth of that link. For example, the bandwidth of link ac is 9 Mbps. The source, S, can generate video at a maximum rate of 10 Mbps and the capacity of sink $k_1$ is 3 Mbps. The numbers in rectangles indicate bandwidth deliverable that the parent sends to respective child in the request message. The numbers inside the circles indicate the bandwidth receivable that can be absorbed by down-links and are carried by the update messages. The update messages propagate from the sink towards the source carrying the maximum units that can be absorbed. Before a request is generated, the concerned nodes approximate the bandwidth of all the children using SNMP.

Each node maintains two lists. The list AppxChildBW initially stores the approximated bandwidth of respective child and the other list UsableChildBW finally has the maximum bandwidth that respective child should receive. MaxDownLinkTreeBW stores maximum bandwidth up to level of TreeLevel. UpLinkBW stores the bandwidth of the link connecting it to its parent. SinkOrSourceCapacity stores the amount produced or absorbed by it if it is a source or a sink respectively.

Here we have demonstrated an instance how an application can benefit from the harness in deep probing. The harness system can also facilitate collection of other common forms of global information such as jitter, and end-to-end delay. It can also

Published in the proceedings of the
IEEE International Conference on Networking, ICN' 2001,
Colmer, France, June 2001, pp628-638.

probe variety of precise and detail pseudo-global and local network states which are not easily accessible today, such as "end-points of the most constrained link in a path", "jitter across the m-th hop down-steam", "aggregate of outgoing bandwidth from a target node" etc. All it needs are different composition rules.

## 5.Performance

Below we provide estimates of the impact on the links and on the nodes due to the harness operation respectively for one execution/propagation wave. Here L and B stand for a probing depth level and branching factor of the context network tree respectively. The sizes of the Request, Reply, and Update are of r, p, and u bytes respectively and computational impact due to *request generator*, *reply generator*, *reply aggregator, update generator* and *update aggregator* are rg, pg, pa, ug, and ua respectively. Table-1 shows the network wide traffic, message per link and the byte density. The terminals do not generate updates hence terminal links do not carry update messages. Table-2 shows the computational impact on the network nodes due to the plug-in. A particular state probing session may be launched with a subset of capabilities (such as no reply, but update). The design objective is to provide the least impact communication for the given application scenario.

**Table-1 Link Traffic Impact**

| Type | Byte | Max Msg | Msg/link | Bytes/link |
|---|---|---|---|---|
| Request | r | $B^L$ | 1 | r |
| Reply | p | $B^{L+1}$ | 1 | p |
| Update | u | $B^L$ | 1 | u |

**Table-2 Node Processing Impact**

| Node | Request | Reply | | Update | |
|---|---|---|---|---|---|
| | Generator | Generator | Aggregator | Generator | Aggregator |
| Initiator | θ(B,rg) | 0 | θ(B,pa) | 0 | θ(B,ua) |
| Synthesizer | θ(B,rg) | θ(B,pg) | θ(B,pa) | θ(B,ug) | θ(B,ua) |
| Terminal | 0 | θ(B,pg) | 0 | 0 | 0 |

We have also performed statistical simulation to project the performance of the harness system under various constraints. The performance depends on the characteristics of the programmable components (complexity of the plug-ins, message size etc.) as well as on the network (such as bandwidth, topology, probing depth etc.) and platform characteristics (scheduling delay, messaging delay, etc). Fig-5(a) shows how one request/response cycle time varies for various plug-in execution time (x-axis) of the update for three orders of message sizes (1K, 100 bytes, and 10 bytes). In cyclic mode this also represents worst case bound on the information recency. Fig-5(b) shows the update delay with respect to the transport layer messaging delay. Another important metric from the network point of view is the background traffic that the harness operation creates for information collection. Fig-5(c) compares the scalability of the harness system as compared to a point-to-point mechanism. It plots the network "hum" (the background traffic generated by the probing process) for various depth and

Published in the proceedings of the
IEEE International Conference on Networking, ICN' 2001,
Colmer, France, June 2001, pp628-638.

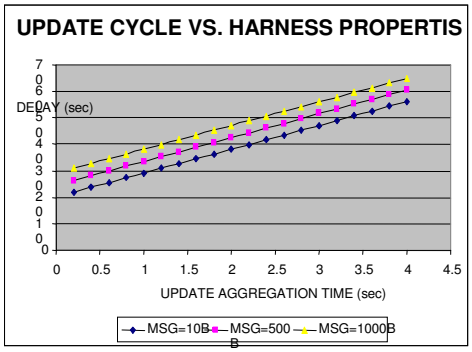


Fig-5(a)

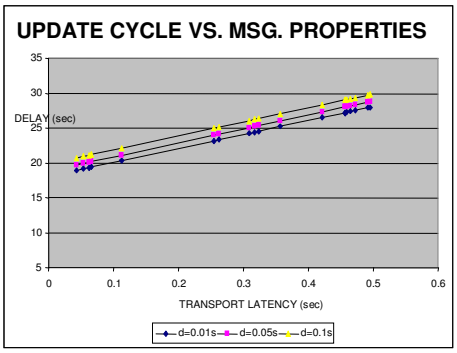


Fig-5(b)

branching factor of the topology. It plots nine cases. The first three curves shows how the *hot-spot* "hum" (y-axis) near the probing root grows in the case of point-to-point mechanism with the increase in the depth of probing (x-axis) for three branching factors (BF=2,4,8). The second set of three curves shows the *average* hum for the same three cases, which is somewhat lesser but still grows rapidly with the probing depth. The average hum provides a measure that how much traffic on the average will be contributed by random spontaneous probing processes appearing at various network locations. On the other hand, the 'hot-spot' hum indicates the change that the application trying to probe, itself has, to be chocked with excessive surrounding probing traffic. The last three plots show the "hum" due to harness process (for three message sizes) (right y-axis). The dramatic scalability of the harness probing is quite apparent. As, can be seen that the hum in harness scheme is not only low but also flat. The result is not unexpected. The network embedded synthesis removes traffic information redundancy.

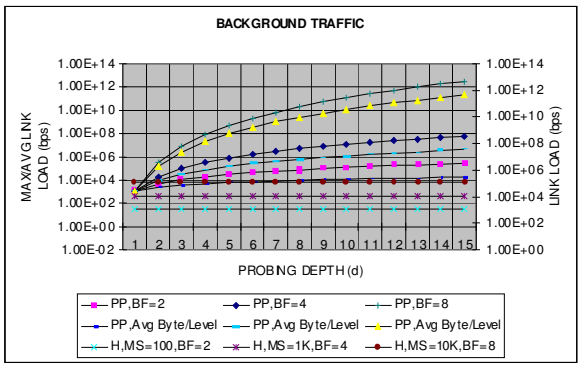


Fig-5(c)

## 6.Conclusions

The key to the system's **scalability** and **versatility** are the embedded aggregators. Since local state dependent aggregation is performed inside a network, it reduces communication and thus enhances the system's scalability. Aggregators also provide the ability to compute network relative deep composite statistics, over the elementary MIB-II variables, thus enhancing the versatility of its ability to collect network states.

The scope of this paper does not permit discussion on implementation. It is non-trivial nevertheless can be realized at user space as *deamons*. Embedded implementation can cut down some overhead and will be critical for sub-second range probing cycles. Implementation on some form of active platform [8,13,14] can further facilitate matters such as remote deployment, and seamless secured execution of the plug-ins. The harness plug-ins require very limited form of programmability compared to general

Published in the proceedings of the
IEEE International Conference on Networking, ICN' 2001,
Colmer, France, June 2001, pp628-638.

active net proposal. Also, the read-write suggestions are through local slate variables only. These characteristics assuage many of the security concerns. The proposed harness is perhaps one of those cases where provisioning even very low-grade programmability can be highly rewarding. The harness increases state visibility of network. In effect it facilitates high pay off smart optimizations for numerous applications, which are not possible today due to the black box nature of current network. Interestingly, such a network layer utility is not only crucial for building a new generation of network aware applications but it is also vital for many of the current problems internet is grappling with. Interestingly many of which are arguably artifacts of the opacity of current network design. Currently, we are exploring its active network based simulation. The work is being supported by the DARPA active network Research Grant F30602-99-1-0515.

## 7.References

1. Carter, Robert L., Mark E. Crovella. Measuring Bottleneck Link Speed in Packet-Switched Networks. Performance Evaluation 27 & 28 (1996), 297-318.
2. Comer D. E., Internetworking with TCP/IP, Principles, Protocols, and Architectures, 4th Ed, Pretice Hall, New Jersey, USA, ISBN- 0-13-018380-6, 2000
3. Dong, Yingfeng, Yiwei Thomas Hou, Zhi-Li Zhang, Tomohiko Taniguchi. A server-based non-intrusive measurement Infrastructure for Enterprise Networks. Performance Evaluation 36-37 (1-4), 1999, 233-247.
4. Downey Allen B., Using Pathchar to Estimate Internet Link Characteristics. http://ee.lbl.gov/nrg-talks.html, April 1997.
5. Francis, P., Sugih Jamin, Vern Paxson, Lixia Zhang, Daniel F. Gryniewicz, Yixin Jin. An Architecture for a Global Internet Host Distance Estimation Service. Proceedings IEEE INFOCOM New York, 1999.
6. Jacobson, V., Traceroute, [URL: ftp://ftp.ee.ibl.gov/traceroute.tar.Z,] 1989.
7. Jacobson, V, Pathchar- a tool to infer characteristics of Internet paths, [URL: http://ee.ibl.gov/nrg-talks.html ], April 1997.
8. Javed I. Khan, S. S. Yang, Medianet Active Switch Architecture, Technical Report: 2000-01-02, Kent State University, [available at URL http://medianet.kent. edu/ technicalreports.html, also mirrored at http:// bristi.facnet.mcs.kent.edu/medianet]
9. Matt Mathis, Jamshid Mahdavi. Diagnosing Internet Congestion with a Transport Layer Performance Tool. Proceedings INET, 1996, Montreal Canada.
10. Paxson, V., Jamshid Mahdavi, Andrew Adams and Matt Mathis. An Architecture for Large-Scale Internet Measurement. IEEE Communication Magazine, August 1998, 48-54.
11. Rose, M.T., & McCloghrie, K. How to manage your Network Using SNMP, Englewood Cliffs, NJ, Pretice Hall, 1995.
12. Schulzrinne, H., S. Casner, R. Frederick, and V. Jacobson. RTP: A Trasport Protocol for Real Time Applications, RFC 1889, 1996.
13. Tennenhouse, D. L., J. Smith, D. Sincoskie, D. Wetherall & G. Minden.. A Survey of Active Network Research. IEEE Communications Magazine, Vol. 35, No. 1, Jan 97, pp 80-86
14. Wetherall, Guttag, Tennenhouse. ANTS: A Tool kit for Building and Dynamically Deploying Network Protocols. IEEE OPENARCH'98, San Francisco, April 1998. Available at: http://www.tns.lcs.mit.edu/publications/openarch98.html
15. Javed I. Khan & S. S. Yang. Resource Adaptive Nomadic Transcoding on Active Network, Applied Informatics. AI 2001, February 19-22, 2001, Insbruck, Austria, [available at URL http://medianet.kent.edu/, also mirrored at http:// bristi.facnet.mcs.kent.edu/medianet] (in press).