End-to-End QoS Measurement: Analytic Methodology of Application Response Time vs. Tunable Latency in IP Networks

E. S. H. Tse-Au
AT&T Labs
Advanced Technologies Lab
Middletown, New Jersey, USA
esta@att.com

P. A. Morreale
Advanced Telecommunications Institute
Stevens Institute of Technology
Hoboken, New Jersey, USA
pat@ati.stevens-tech.edu

Abstract

In this paper, we discuss one aspect of the measurement issue: how to measure end-to-end application response time (ART) relative to aggregated "tunable" network latency, or tunable latency. The goal is to enhance our understanding of the relationship between these two metrics for database access applications. Tunable latency is defined as follows: the sum of the "round trip" queuing delay & data transmission/Insertion delay from beginning to end of the application transmission. Our problem space concentrates on developing a methodology to graphically characterize response time as a function of tunable latency for existing database access applications in a wired, single-threaded, multi-user, post-deployment client / server environment. A number of tools were used in developing this methodology which was not obvious from the tools' documentation. To test it's feasibility before actual field use, we used an experimental setup to emulate the real user environment. In so doing, we now have two proposed methodologies: one for the experimental setup version (of the in-service scenario) and one for the "actual" in-service scenario. We present results obtained from the *experimental* method. The resulting graphs can be a consultation tool for network tuning and control, classification of user applications by priority class-of-service, SLA negotiation and manual QoS provisioning.

Keywords

End-to-End QoS Measurement, Network Performance, Latency

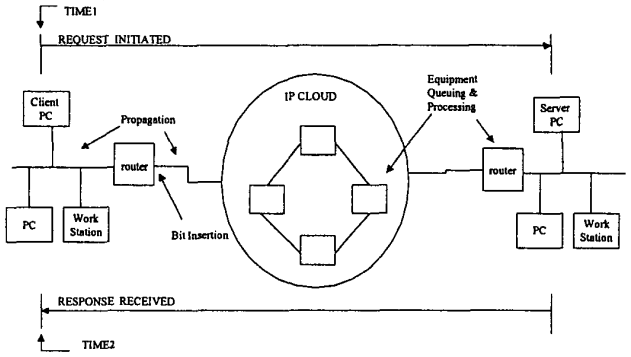
1. Introduction

Today's Internet is held together by the services of a connectionless network layer protocol: IP – Internet Protocol, for the transport of user information using packet switching technology. IP offers a scalable network architecture that allows flexible interconnection of different networks, and yet can support a diverse set of services ranging from bulk data transfer to real-time voice and video transmission.

Compared to mainframes and private line networks, IP offers relatively inexpensive services for the same type of connectivity to remote locations. It also offers robustness such that connections remain intact as long as the source and destination machines were functioning, even if some hosts or transmission lines in between were suddenly put out of operation [1]. Corporations with enterprise networks have become attracted to these beneficial features of the IP network, especially those that use expensive mainframes and point-to-point private lines. Many corporations have begun migration of their enterprise networking solution to that of the IP networking solution. However, a drawback of an enterprise IP network is that it offers only one service class – best effort (resulting in unpredictable service level performance). Hence, corporate customers who had always enjoyed predictable performances of private lines were suddenly faced with unpredictable IP service performance and began demanding guaranteed service quality in network metrics – currently in the form of static and pre-determined (vs. dynamic[1]) SLAs – Service Level Agreements. For corporations with wired, distributed wide area IP data networks, the most requested QoS metrics for business-critical applications are network latency and especially, application response time (ART) [2]. The provisioning of these QoS metrics raises interesting questions, including:

- What is an appropriate definition for each of these QoS metrics?
- How should each metric be measured? etc.

This paper discusses one aspect of the measurement issue: how to measure ART relative to aggregated "tunable" network latency, or tunable latency, so that the relationship between the two metrics can be understood for database access applications. A general, all encompassing definition of network latency is as follows (see Figure 1):



(1) NETWORK LATENCY = PROPAGATION DELAY + BIT INSERTION DELAY + QUEUING DELAY + PROCESSING DELAY [3][4]

(2) APPLICATION RESPONSE TIME (ART) = TIME2 - TIME 1 [2]

Figure 1: a General Definition of Network Latency & Application Response Time

[1] Dynamic in a sense that from customer's viewpoint, the agreed upon service levels within the SLA could be triggered by certain time of the day, unpredictable internal network events such as congestion conditions, or externally-driven events such as IPO in the financial world.

- Network Latency – the time required for one bit to propagate round-trip between two nodes. It includes propagation delay, bit insertion delay, and queuing / processing contributed by routers and switches in the path [3][4].

In this paper, we adopt two *slightly different* definitions for the delay caused by the network portion of the ART by separating it into two quantities – aggregated "un-tunable" network latency & aggregated "tunable" network latency as follows (see Figure 2):

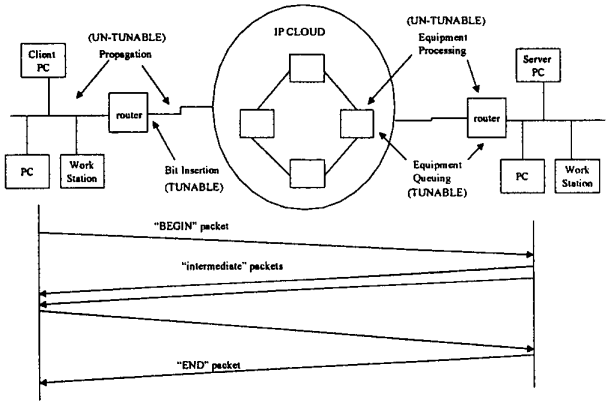


Figure 2: Aggregated Un-tunable Latency & Aggregated Tunable Latency: Over the Entire Application Duration

- Aggregated "un-tunable" network latency, or "un-tunable latency" – the one-bit delay due to *signal propagation* and *equipment processing* in the path between two nodes from the beginning to the end of the application's transmission. This may include a number (> 1 round-trip) of negotiations between two communicating nodes before the entire application is completed and can be expressed as follows:

  "un-tunable latency" = Signal propagation delay + Equipment processing delay

  This part of the network delay is considered "un-tunable" from the service provider (SP) perspective, since propagation and equipment processing are intrinsic characteristics respectively of the transmission medium and electronic & mechanical equipment.
- Aggregated tunable network latency, or "tunable latency" – the total time required for one bit to be *inserted* into the transmission medium and go through *queuing delay* of all equipment in the path. Again, the "total" time signifies the time from the beginning to the end of the application transmission. It can be expressed as follows:

  "tunable latency" = Queuing delay + Data transmission (insertion) delay

  This part of the network delay is considered "tunable" from the SP perspective: SP can adjust these parameters, i.e., the bandwidth effect on transmission and

equipment queuing delay for specific traffic, via router queuing discipline, QoS priority class separation, and / or bandwidth upgrade.

We also adopt the definition for ART as in Figure 1[2]: The time that a user sees between that which a network operation is requested and that which a response is received by the requester.

Our problem space concentrates on developing a methodology to graphically characterize response time of existing in-service DB access applications as a function of "tunable latency" in a wired, single-threaded[2], multi-user[3], post-deployment client / server environment. Since ART cannot be readily controlled, whereas "tunable latency" can be, via network capacity and flow control engineering, such graphs become invaluable "predictive tools" for the current static and pre-determined SLA negotiations for ART and have a number of uses as follows:

(a) Network (re-)engineering / tuning: a typical scenario involves a corporation that needs to have an upper bound guarantee on the ART of a specific (business critical) application. With a graph of ART vs. "tunable latency" (established for the application), SP could quickly translate the ART into the associated latency objective for a specific set of network conditions. This enables SPs to determine necessary network parameters, such as router queuing disciplines, that must be tuned to arrive at the "tunable latency" objective for the application.
(b) An aid to "realistic" SLA negotiation for both the enterprise customers and SPs.
(c) An aid to the classification of user applications by router priority "class of service" according to user need, somewhat similar to a "consultation" tool.
(d) An aid to manual QoS provisioning for network managers and system administrators, for determining the parameters that must be tuned to achieve specific objective tunable latency values when ART threshold-exceeded alarms come in.

This type of graph would be most appropriately developed for business critical, time-sensitive applications such as stock market information retrieval. Hence, database (DB) access is selected as the type of application for our study. To test the feasibility of this methodology before actual field use, we used an experimental setup to emulate the real user environment. In so doing, we now have two proposed methodologies: one for the experimental version of the in-service environment and one for the "actual" in-service environment. We present results obtained from the *experimental* method.

This paper is organized as follows: Section 2 surveys related work in the area of measurement techniques for ART and network latency, and presents our proposed

[2] Single-threaded means that there's one link available between two communicating entities; e.g., a client & a server

[3] By multi-user we mean that router queuing delay becomes non-zero and server response time includes an element of server contention.

measurement methodology for an *in-service* environment. Section 3 presents an experimental setup that emulates this in-service environment, test conditions, and the associated ("experimental version" of the) measurement methodology. Section 4 presents quantitative results from the experimental setup, and test observations. Section 5 presents conclusions and future work. Acknowledgements are shown in Section 6; and we conclude the paper with a list of references and acronyms.

2. Related work / proposed methodology

A literature search was conducted on the subject of measurement methods for end-to-end network latency and ART. There are currently four main techniques in use to obtain ART:

1. API – Application Programming Interface – [5], it requires application vendors to add triggers to the beginning and end of transactions in programming codes to capture the ART. It provides an accurate measurement of the ART, but could be quite intrusive and expensive to do, for reasons such as code tracking and revision, and software upgrades.
2. Passive Listening / empirical observation – [2], [6], and [7] all use some types of software programs to passively listen for the "begin" and "end" trigger of application transactions and empirically measure the ART. These programs can measure ART accurately but require intense instrumentation overhead. Nonetheless, passive listening is popular, as can be seen by its use in many probe products today [8]. Passive listening / empirical observation is also most widely used for network latency measurements. In [9], the delta of the time between the end of the send on host A and the end of the receipt on host B is measured, to estimate the end-to-end one-way network delay. In [10] & [11], probe packets are sents out and the delta of the time between the end of the send and the beginning of the receipt of the same probe packets is measured in the same host. One type of probe packets is the ICMP PING message or its variants. [12] used PING, [13] used FING – a variant of PING. [6] used yet another type: UDP probe packets in the "traceroute" software tool.
3. Synthetic Transaction ™ Technology – This technology uses agents to issue "test" transactions for each application to be monitored and the ART of this test transaction is then measured [14][15].
4. Analytical Modeling – [16] describes a modeling approach to estimate ART from the server with limited client side information. Modeling allows ART prediction: an advantage for those scenarios that may be hard to simulate in a laboratory environment and hard to measure in an in-service situation. A different kind of analytic model is proposed in [17]: it predicts performance of web transport over several protocols. The main difference between [16] and [17] is that, in [17], the model does not address the application level but [16] does. In this paper, like that in [16], application level modeling is one of our main concerns.

All the tools and techniques described in the literature, as well as a number of the tools surveyed in the market place today [14][15][8], can measure either the ART or network latency relatively well. However, few can easily measure and correlate ART with its own network delay, especially the tunable part of the network delay. This paper addresses this issue, it proposes a methodology to do such for DB access applications by using an off-the-shelf tool: Optimal's "Application Expert" [18]. This tool is capable of measuring a 'baseline' ART for existing applications and then project the ART for these applications in various "prediction" scenarios via its built-in modeling capability. A prediction scenario consists of a set of network variables – application's available bandwidth, network's background traffic load, and round-trip inherent latency. Different combinations of these 3 variables produce different prediction scenarios.

Our methodology to generate the ART vs. "tunable latency" graph for existing DB access applications in an *In-Service* environment is as follows:

1. Use Application Expert's packet capture procedure [18] to capture a packet trace of a DB access application from either the client or server side. This will yield the "baseline" ART profile, which includes the ART and its associated time components on the client, server, network propagation and "data transmission".
2. Use Application Expert's recommended procedure, Method 1 of [19], to find the baseline adjustment value[4] that represents the impact of the network on the measured "baseline".
3. The graph of ART vs. "tunable latency" is deduced as below
   (a) Use the procedure in Application Expert's Response Time Predictor tool [18] to project the ART in different prediction scenarios; i.e., by varying different values of the percent background traffic load, e.g., 0 – 99 %, while holding the network latency value and link speed *constant*, and
   (b) For each such scenario,
   - "subtract out" the "baseline adjustment value" from the associated ART value and from either the client (if measured at the server side) or server (if measured at the client side) value
   - plot the (adjusted) ART vs. "data transmission" value obtained in each prediction scenario.

A clarification: the term "data transmission" shown in the steps above, quoted directly from [18], is the "tunable latency" we are referring to in this paper. In

[4] The "baseline adjustment value" is the sum of the *average network propagation* delay and *average bit insertion* delay experienced over a WAN. Average network propagation delay is determined from the round-trip time required for a single-bit to propagate from the source to the destination. Average bit insertion delay is determined from the formula: a / b, where a = average (message) burst size in bits, b = average bandwidth in the network path between client & server (in bps).

Section 4, we explain in detail the reason for our choices of terminology and the associated definitions used in this paper.

3. Experimental setup / Test conditions and Measurement Methodology

To closely emulate a single-threaded / multi-user post-deployment environment, below is a list of components we used for the end-to-end ART of existing DB access applications:

DB ART = Initial client processing / think time + source LAN contention + un-tunable latency + tunable latency + server response time + subsequent client processing time

where:

- *Initial client think / processing delay*: the time it takes the client to hit the "enter" key at the start of the application.
- *LAN contention*: the time it takes for a packet to successfully gain access to the medium.
- *"Un-tunable latency"* and *"tunable latency"*: as previously definded.
- *Server response time*: the time it takes for a request packet of an application to enter the server pool and be processed by the server, plus the time it takes for the server to send the last packet of the reply.
- *Subsequent client processing time*: the time it takes for the client to update and / or process a response, e.g., a retrieved record.

For the experiment, we consider the "Initial client processing / think time" is not a significant part of ART, and that LAN speed is high enough such that source LAN contention is negligible. The technique for measurement of server response and subsequent client processing time is an important *separate* issue in a multi-user environment. However, for our methodology development, we are not concern with in-depth emulation of the various client and server load conditions in our experimental setup. The tool we used measures the server and subsequent client processing time components as part of the captured packet trace of an application and treats them as constants. These client and server time components are *additive* components to the overall ART. They are not affected by network topology and do not affect the "un-tunable latency" and "tunable latency" components. Hence, our experimental setup emphasizes the proper emulation to obtain the "un-tunable latency" and "tunable latency" time components for DB access. The following are the main activities we produce in our experimental setup for the multi-user, single-threaded environment:

- DB access application traffic over a WAN from a client to a server
- Background traffic load in the WAN in a multi-user environment

To emulate the client-server DB access traffic, we used the built- in traffic scripting capability offered by an off-the-shelf tool called Chariot® [20] which consists of a

console and two End-points. Driven by the built-in traffic script execution capability, client-server traffic pattern is exchanged between the two end-points. To emulate background traffic load in the WAN, we used another off-the-shelf tool: Netperf [21], which is capable of generating TCP or UDP traffic.

The experimental setup (shown in Figure 3) consists of the following equipment:

- A router acting as a frame-relay switch hooked between a source and destination router consists of our "WAN backbone"
- Three subnets / hubs connect to the "WAN backbone" with two on side 1 and one on side 2
- Subnet 1 of side 1 consists of a Chariot® End Point 1 (emulated as server) and Chariot® console
- Subnet 2 of side 1 consists of a station running Netperf "traffic generator"
- Subnet 3 of side 2 consists of a Chariot® End Point 2 (emulated as client) and another remote destination host to receive traffic generated by Netperf.
- Optimal's Application Expert can be placed in either Subnet 1 next to the "server" (Chariot® End Point 1) or in Subnet 3 next to the "client" (Chariot® End Point 2)

To emulate in-service DB access applications, we *assume* the following:

- We have a higher probability of accessing the server side of the network
- The speed of the link connecting a client to a database server is an E1 WAN link (this speed will be used in our prediction scenarios)

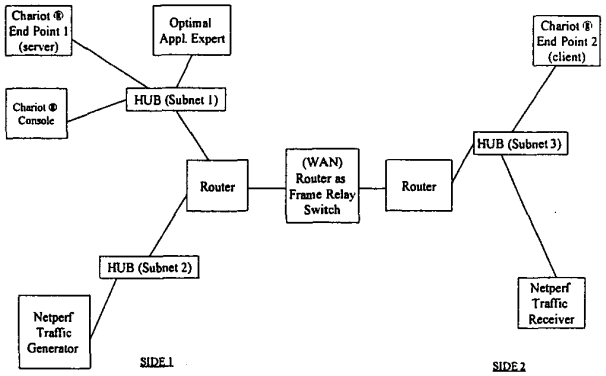


Figure 3: Experimental Setup For Database Access Application Using Off-the-shelf Tools

- Background WAN traffic consists mainly of highly bursty UDP traffic
- The most often used "business critical" application is the traditional DB record retrieval and update using TCP

- A client request packet size of 100 bytes, a record size of 2048 Bytes, and a subsequent client processing time to update a record to be 12 seconds
- A user with a task of updating ten fields, each of which resides in a separate record in the same remote DB server. Hence, the user consecutively accesses the remote server ten times to do ten record retrieval and update operations

Based on the above assumptions and experimental setup, below is our *experimental measurement methodology* for our graph construction:

- Connect Optimal's Application Expert at the server side (Subnet 1), and set up Application Expert for the packet capture mode (with capture filter set to only the relevant client and server IP addresses);
- Modify one of Chariot's® built-in database access scripts[5] to the desired packet sizes and user think time as listed in the assumptions and set up Chariot® End Points to run the script to completion;
- Set up Netperf command line to generate background UDP-stream traffic across the WAN link (to simulate a multi-user environment);
- Activate respectively: Netperf, Application Expert capture, and Chariot® script;
- When packet trace is captured, edit the trace to include only those packets relevant to the DB access operation (i.e., filter out the Chariot's script initialization packets and miscellaneous packets used for communication between the Chariot console and end points);
- Follow Application Expert's recommendation to calculate the "baseline adjustment value", Method 1 of [19]; taking care that the same background traffic stream continues to be present across the WAN link during each step of this calculation; and
- Deduce the graph of "ART vs. tunable latency" using Application Expert's Response Time Predictor Tool, in accordance with that specified in step 3 of the methodology for the *In-Service* environment in Section 2.

4. Test results summary

Before presenting sample test results, we explain factors that led to the choice of the definitions and terminology used for our data presentation. In Application Expert's Response Time Predictor Tool, three parameters can be manipulated in each prediction scenario for ART: (1) percent background traffic load, (2) limiting speed of the link, and (3) round trip network latency. In addition, the tool represents the network portion of the delay by two values: Data Transmission & Network Propagation. Below are our observations of Application Expert's Response Time Predictor tool's prediction behavior:

By keeping all other parameters constant in a prediction scenario

- the higher the percent background traffic load, the longer the data transmission time(while network propagation remains constant)

[5] This DB access traffic emulates a record retrieval and update action.

- the slower the link speed, the longer the data transmission time(while network propagation remains constant)
- the higher the network latency, the higher the propagation delay (while data transmission remains constant)

From these dependencies, we conclude:

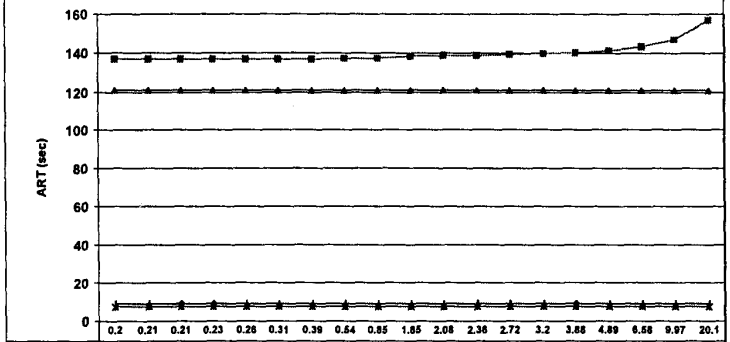
1. the tool's network latency affects only network propagation (and router processing) in a prediction scenario
2. the percent background traffic load affects waiting time in the queue, which in turn affects the length of data transmission in a prediction
3. the speed of the link affects how fast the number of bits is being clocked out onto the transmission medium, therefore, link speed affects the length of data transmission in a prediction
4. from 2 and 3 above: "data transmission" value includes bandwidth effect on transmission time and network queuing delay
5. from 1 above: "network propagation" value includes propagation and router processing delay

This set of conclusions led to the two definitions in this paper to represent the network portion of the delay in the overall ART value; i.e., "un-tunable latency" and "tunable latency", to match the behavior of the tool in a prediction. Even through the two terms can be counted separately, slight *double-counting* of router queuing delay occurs when this tool is used to do predictions. The tool measures *its own version* of the network latency using the ICMP PING message. Hence, the measurement inevitably includes signal propagation, some router processing (which is negligible [22]), and some existing queuing delay in a production link. However, the tool's definition of data transmission delay is actually where added queuing delay is projected ([22], see also 4 above). This double-counting (of queuing delay) will have minimal influence on the projected ART value vs. projected data transmission value *if* the tool's network latency value is held constant in the prediction scenario. In such cases, it will not affect the intended use of our proposed graph, i.e., to determine the range of network tuning needed for specific ART objectives.

Using the experimental setup and methodology listed in Section 3, we present below sample data of end-to-end ART vs. "tunable latency". Figures 4 – 6 are sample results of the "ART vs. Tunable Latency" graph. For each of these graphs, we held the network latency value and link speed constant, and obtained the ART value by varying the percent background load from 0, 10 to 90, 91 to 99 (a total of 19 different prediction scenarios). For each of these different percent background load values, we obtained the tool's predicted data transmission value and ART value. (This ART is then adjusted by subtracting out the baseline adjustment value.) The graph is produced by plotting the tool's predicted (and then adjusted) ART against the predicted data transmission value (i.e., our "tunable latency"). For example, holding the network latency to a constant 600ms (Figure 6), one can achieve an ART value of between 141.3 and 142 seconds if one can keep the

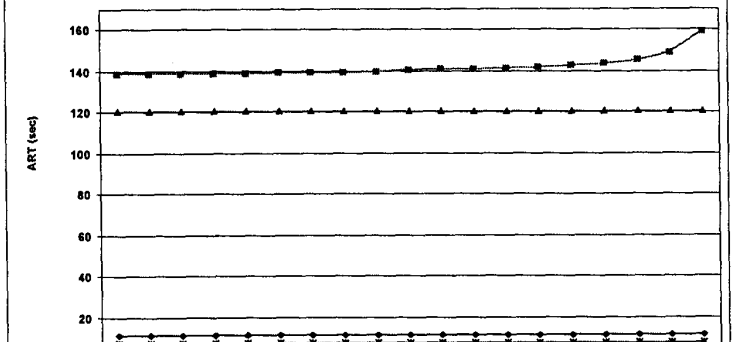
(aggregated) tunable part of the network delay between 200 and 850ms. One should bear in mind that the subsequent client processing time takes > 120 seconds of this ART value for the overall task. This is due to the subsequent client processing time of 12 seconds needed per each of 10 record / field updates. If users feel that with the same server and client load conditions, it will be acceptable if the entire task can be completed in 3 minutes instead, then SPs have a larger value to tune the tunable latency to fit the 3 minute ART objective (e.g., > 20 seconds of "tunable latency" in Figure 6). Note that in Figures 4 – 6, the client, server, and network propagation time components are shown as constants in all prediction scenarios (see Section 3). Figure 7 is a summary graph showing the ART vs. Tunable Latency when the tool's network latency value was held constant at 300ms, at 400ms, at 500ms, and at 600ms.

Figure 4: ART vs. (Aggregated) Tunable Latency, N/W Latency =400ms



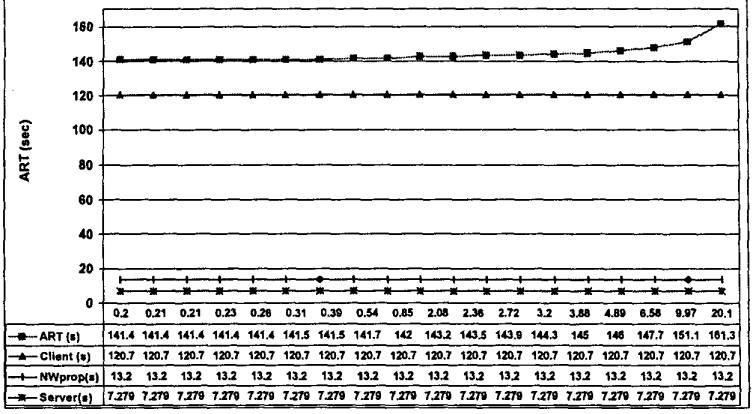
	0.2	0.21	0.21	0.23	0.26	0.31	0.39	0.54	0.85	1.85	2.08	2.36	2.72	3.2	3.88	4.89	6.58	9.97	20.1
ART (s)	137	137	137	137	137	137.1	137.1	137.3	137.8	138.6	138.8	139.1	139.5	139.9	140.6	141.6	143.3	146.7	156.9
Client (s)	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7
NWprop(s)	8.8	8.8	8.8	8.8	8.8	8.8	8.8	8.8	8.8	8.8	8.8	8.8	8.8	8.8	8.8	8.8	8.8	8.8	8.8
Server(s)	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279

Figure 5: ART vs. (Aggregated) Tunable Latency, N/W Latency=500ms



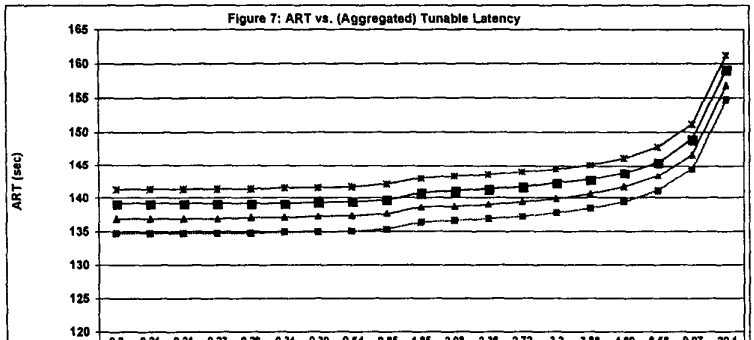
	0.2	0.21	0.21	0.23	0.26	0.31	0.39	0.54	0.85	2.08	2.36	2.72	3.2	3.88	4.89	6.58	9.97	20.1
ART (s)	139.2	139.2	139.2	139.2	139.2	139.3	139.3	139.5	139.8	141	141.3	141.7	142.1	142.8	143.8	145.5	148.9	159.1
Client (s)	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7
NWprop(s)	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11
Server(s)	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279

Figure 6: ART vs. (Aggreagated) Tunable Latency, N/W Latency=600ms



	0.2	0.21	0.21	0.23	0.26	0.31	0.39	0.54	0.85	2.08	2.36	2.72	3.2	3.88	4.89	6.58	9.97	20.1
ART (s)	141.4	141.4	141.4	141.4	141.4	141.5	141.5	141.7	142	143.2	143.5	143.9	144.3	145	146	147.7	151.1	161.3
Client (s)	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7	120.7
NWprop(s)	13.2	13.2	13.2	13.2	13.2	13.2	13.2	13.2	13.2	13.2	13.2	13.2	13.2	13.2	13.2	13.2	13.2	13.2
Server(s)	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279	7.279

Figure 7: ART vs. (Aggregated) Tunable Latency



	0.2	0.21	0.21	0.23	0.26	0.31	0.39	0.54	0.85	1.85	2.08	2.36	2.72	3.2	3.88	4.89	6.58	9.97	20.1
NL=300ms	134.8	134.8	134.8	134.8	134.8	134.9	134.9	135.1	135.4	136.4	136.6	136.9	137.3	137.7	138.4	139.4	141.1	144.5	154.7
NL=400ms	137	137	137	137	137	137.1	137.1	137.3	137.6	138.6	138.8	139.1	139.5	139.9	140.6	141.6	143.3	146.7	156.9
NL=500ms	139.2	139.2	139.2	139.2	139.2	139.3	139.3	139.5	139.8	140.8	141	141.3	141.7	142.1	142.8	143.8	145.5	148.9	159.1
NL=600ms	141.4	141.4	141.4	141.4	141.4	141.5	141.5	141.7	142	143	143.2	143.5	143.9	144.3	145	146	147.7	151.1	161.3

5. Conclusion and future work

In this paper, we have presented two simple methodologies using off-the-shelf technology to construct a graph that is of practical use in a number of scenarios. One is for the experimental setup and the other for an in-service scenario. From the experimental results, we concluded that graphical characterization of ART as a function of tunable latency can be done, and the prospect for using the actual in-service methodology is highly feasible and promising. Conclusion regarding the usefulness of the tool in modeling the ART in actual world situations can be drawn once we had a chance to verify the methodology for the in-service scenario.

Numerous other challenges remain in this area of methodology and technology for end-to-end ART measurement. Below are a few major ones that we've encountered

during this initial effort:

- finding appropriate measurement technique(s) to correlate the ART vs. "tunable latency" for multi-threaded environments, where multiple servers are accessed due to the initiation of a single client request message;
- finding appropriate technique(s) to clearly break down and measure ART in terms of its various time components, such as the client side and server side contention we've discussed; and
- finding a corresponding methodology to construct the "ART vs. tunable latency" graphs for new applications. This is an important area for corporations as many do have its' own internal test labs that are specifically dedicated to testing new applications before actual field use.

With our tried-out experimental methodology, we felt that the most important item to address as part of the continuation of this work would be two-fold:

- verification of the methodology in an in-service scenario (i.e., using that which proposed in Section 2); and
- modification of this methodology for new applications.

6. ACKNOWLEDGMENT

The authors would like to thank D. Heyman, G. Mouradian, D. Jones & R. Oppenheim of AT&T Labs for their valuable comments on the work of this paper.

References

[1] A. S. Tanenbaum, "Computer Networks", 3rd Edition, Prentice Hall PTR, 1996.

[2] M. Maccabee, F. Mosinger, "Decomposition of End-to-End Response Time: Addressing a Major Corporate Requirement", CMG Proceedings, vol.2 , 1997. pp. 873-880.

[3] K. Van DerWal, M. Mandjes, H. Bastiaansen, "Delay Performance Analysis of the New Internet Services with Guaranteed QoS", Proceedings of the IEEE, Vol. 85, No. 17, 12/97.

[4] J. Heideman, K. Obraczka, J. Touch, "Modeling the Performance of HTTP Over Several Transport Protocols", IEEE/ACM Transactions on Networking, Vol.5, No. 5, 10/97.

[5] J. Butler, "A Call to ARM", Capacity Management Review, vol. 25, No.10, pp.1, 3-11, 14-19.

[6] M. Stadelmann, "Performance Instrumentation of WEB Applications", CMG Proceedings vol.1, 1997, pp. 168-75.

[7] B. Cahoon, K. S. McKinley, "Performance Evaluation of a Distributed Architecture for Information Retrieval", SIGIR Forum Conference Title: SIGIR Forum special issue, pp.110-18, ACM, 1996.

[8] Websites for Ecoscope SuperMonitor, NetXRay & Network Monitor are

respectively: www.compuware.com, www.nai.com, www.microsoft.com
[9] W. Bai, P. Zarros, M. Lee, T. Saadawi, "Design, Implementation and Analysis of a Multimedia Conference System Using TCP / UDP", SUPERCOMM/ICC '94, pp.1749-1753, vol.3.
[10] J. Bolot, "End-to-End Behavior of the Internet: Measurements, Analysis, and Applications", 2nd Asian Computing Science Conf. ASIAN '96 Proceedings pp.362-76.
[11] Sanghi, Agrawala, Gudwundsson & Jain, "Experimental Assessment of End-To-End Behavior on Internet", IEEE INFOCOM '93, pp.867-74, vol.2.
[12] R. Carter, M. Crovella, "Server Selection Using Dynamic Path Characteristics in Wide-Area Networks", Proceedings IEEE INFOCOM '97, pp.1014-21, vol.3.
[13] A. Fasbender, P. Davids, "Measurement, Modeling and Emulation of Internet Round-Trip Delay", 8th GI/ITG Conf. On Measuring, Modeling & Evaluating Computing and Communication Systems, MMB '95 Proceedings, pp. 401-15.
[14] NextPoint Networks, "NextPoint S3 Technical Overview Draft", 1998.
[15] http://www.nextpoint.com
[16] Dilley, Friedrich, Jin, Rolia, "Measurement Tools and Modeling Techniques for Evaluating WEB Server Performance", Computer Performance Evaluation, Modeling Techniques and Tools, 9th Int'l Conference Proceedings, pp.155-68.
[17] J. Heidemann, K. Obraczka, and J. Touch, "Modeling the Performance of HTTP Over Several Transport Protocols', IEEE Transactions on Networking, vol. 5, No. 5, 10/97.
[18] Optimal Networks, "User's Guide, Optimal Application Expert", Optimal Network Corporation, http://www.optimal.com, 1996 – 1998.
[19] G. Kaiser, "Using Optimal Application Expert To Predict Response Time Changes for Post-Deployment Applications", Optimal Networks Corporation, 1998. http://support.optimal.com/oae/kb.shtml
[20] Chariot ® Product Information & Technical Documentation in: http://www.GanymedeSoftware.com/html/downloadinfo.htm
[21] Hewlett packard Company, "Netperf: A Network Performance Benchmark Revision 2.0", Information Networks Division, Hewlett-Packard Company, February 15, 1995; http://www.netperf.org/netperf/training/Netperf.html
[22] Conversation with Optimal Networks, August, 1998

List of Acronyms

ART	Application Response Time	PING	Packet Internet Groper
DB	Database	SLA	Service Level Agreement
QoS	Quality of Service	SP	Service Provider
ICMP	Internet Control Message Protocol	TCP	Transport Control Protocol
IP	Internet Protocol	UDP	User Datagram Protocol
LAN	Local Area Network	WAN	Wide Area Network