Fuzzy Rule-based Adaptation Framework for Wireless Thin-clients

Mohammad AL-TURKISTANY
Computer and Information Science and Engineering Department, University of Florida
Gainesville, FL 32611, USA

and

Abdelsalam (Sumi) HELAL
Computer and Information Science and Engineering Department, University of Florida
Gainesville, FL 32611, USA

ABSTRACT

The thin-client architecture has been shown to offer a promising utility for mobile computing. By delivering any application through a single, small footprint client (the thin client) on a mobile device, it is possible to mobilize all applications without the need for building wireless application gateways (e.g., WAP Gateways). To this end, the thin-client is very promising. However, for certain applications in which the display changes rather frequently, sending display updates frequently and inefficiently could challenge the case for thin-clients and their use in mobile computing environments. Such application behavior would result in performance penalties and costly connection charges. Also, it results in high transmission latency of thin-client's display updates. The effect of active applications could be further compounded by the wide variability of the wireless network and mobile device resource parameters. We present a proxy-based thin-client adaptation framework, which utilizes wavelet-based image coding to enable variable and scalable compression of display rectangles. It offers application-level adaptation by employing a rule-based fuzzy engine that dynamically adapts to bandwidth and client's processing power variability. We describe our framework and its implementation, and we report on measured system performance under variable application and network parameters.

Keywords: thin-clients, adaptive framework, wireless networks, fuzzy control.

I. INTRODUCTION

The concept of *application-level adaptation,* which was introduced by M. Satyanarayanan (surveyed in [5]), allows mobile applications to make trade-off decisions to favor certain qualities of service over others, using application semantics or knowledge. Such QoS parameters include bandwidth, latency, error rate, and usage cost, to mention just a few. For a given mobile device with limited resources and for a specific wireless link quality, mobile applications are allowed to adapt and manifest different requirements to the underlying system. For instance, the variations could be due to a sudden source of signal interference or change in the number of users sharing resources at a cell location.

To enable dynamic adaptation of applications, communication protocols need to discover wireless link parameters. This allows application level protocols to adjust their behavior accordingly. For instance, the thin-client system server should be able to dynamically change the type or level of compression used to transmit screen updates to the client over a wireless link. This is particularly important since thin-client systems place a large traffic load on wireless links when compared to standard client-server computing model.

Furthermore, there is usage cost incentive for optimal utilization of wireless bandwidth since most wireless data service providers charge customers based on the amount of data they transmit and receive over their network. Battery energy constraints of mobile and portable devices may be the single most important constraint on wireless thin-client system. There is urgent need to optimize thin-client's usage of battery power through optimizing the encoding schemes used to send screen updates. It is essential for these encodings to have scalable complexity as much as possible to control energy consumption. This enables trade-offs between energy consumption and quality of screen updates offered by an encoding scheme.

In this paper, we describe a proxy-based adaptation framework for wireless thin-client systems. We base our work on the Virtual Network Computing (VNC) thin-client system from AT&T Cambridge Labs [1]. We describe our framework and show how it dynamically adapts using dynamic context discovery through a fuzzy rule-based inference engine. We first give a very brief background on VNC (Section II) and then describe a wireless thin-client performance model on which we base our framework and inference engine (Section III). The framework and its implementation are presented in Section IV.

Experimental evaluation of the effectiveness of the framework and its adaptability is given in Section V. Finally, Section VII ends with concluding remarks.

II. Virtual Network Computing Thin-client System

VNC is an open-source thin-client system from AT&T. VNC's Remote Frame Buffer (RFB) protocol enables the VNC server to send frame buffer updates to remote client. The basic primitive in RFB protocol is placing a rectangle of pixel data at position (x, y) in the client's frame buffer. Each frame buffer update consists of a number of screen rectangles. The RFB protocol offers poor compression when it handles applications that display complex graphics (e.g., natural images). Consequently, the VNC thin-client system generates huge amount of traffic on network infrastructure. This dramatically degrades the performance of wireless thin-client system and makes the user unsatisfied due to excessive transmission latencies. Furthermore, the RFB protocol has a limited ability to adapt to changing wireless link conditions since the rate of screen updates requested by the thin-client (client pulling model) depends on link bandwidth.

III. Wireless Thin-client Performance Model

We model the wireless thin-client system by considering the factors that affect its performance, such as wireless link bandwidth, client processing power, and server processing power. We model it using a simple model of three cascading M/M/1 queues . Our model assumes incremental screen updates in which the server only needs to send screen rectangles that recently have changed. To simplify the analysis, we assume the server processing power is highly controllable and can be made arbitrarily large compared to the thin-client processing power. Therefore, we drop the effect of the server part in the model. For the communication channel, the average latency is

$$T_d = \frac{1}{B\mu - \lambda} \tag{1}$$

where λ is arrival rate in rectangles/sec, $1/\mu$ is average rectangle size in bits/rectangle, and *B* is the link bandwidth in bps. For the thin-client, the average latency is

$$T_c = \frac{1}{\mu D(\alpha) - \lambda} \tag{2}$$

where $D(\alpha)$ is the decoding rate in bps, $0<\alpha<1$ is the compression ratio. Therefore, the average total latency for the thin-client system is

$$T_{total} = T_p + \frac{1}{\frac{\mu B}{\alpha} - \lambda} + \frac{1}{\mu D(\alpha) - \lambda} \tag{3}$$

T_p is the communication link latency. In Eq. (3), stability condition for the system requires that $\mu D(\alpha) > \lambda$ and $\mu B/\alpha > \lambda$. It is to prevent infinite delay in the thin-client system. In this model, the tuple (λ, μ) represents application's screen update characteristics. Generally, decoding rate $D(\alpha)$ is a function of several variables, such as screen rectangle content, decoding algorithm being used, client's processing power, and target compression ratio. Generally, this function is non-linear and hard to model mathematically.

In Eq. (3), if $B/\alpha >> \lambda$ and $D(\alpha) >> \lambda$, which is the case when the system operates in the client pull mode, there is no queuing latency. Hence, Eq. (3) becomes

$$T_{total} = \frac{1}{\mu}\left(\frac{1}{B/\alpha} + \frac{1}{D(\alpha)}\right) = \frac{1}{\mu} \cdot \frac{1}{BW_{virtual}} \tag{4}$$

Therefore,

$$BW_{virtual} = \frac{B \cdot D(\alpha)}{\alpha D(\alpha) + B} \tag{5}$$

The virtual bandwidth, $BW_{virtual}$, represents the combined effect of transmission latency and processing latency in the system. The virtual bandwidth is the target of our optimisation. By maximizing it, we minimize the system's total latency. If the decoding rate function $D(\alpha)$ is monotically decreasing over the domain of α (such that $D(\alpha) \approx D_0$ at $\alpha \approx 0$), which is the case for the GWIC wavelet-based decoder [10] we are using, then

$$BW_{virtual} \leq D_0$$

Update Quality-latency Trade-off

The maximum virtual bandwidth is achieved (best-case latency) when $BW_{virtual} \approx D_0$ and this happens when $\alpha \approx 0$. This corresponds to thin-client's worst screen quality and highest compression when using lossy encoder. Our goal is to trade off between $BW_{virtual}$ and screen update quality (or compression ratio α). For lossy wavelet-based encoder, increasing the compression level (i.e., smaller α) results in the deterioration of thin-client's screen quality. Therefore, we set the target virtual bandwidth according to the quality of service acceptable to the thin-client user. For instance, if we have screen update quality requirement that dictates maximum value for target $BW_{virtual}$ (e.g., $BW_{virtual} = 3D_0/4$), then from Eq. (5), we get corresponding value for α (e.g., $\alpha = B/(3D_0)$). Dynamic adaptation is achieved by controlling α at the server (or proxy) side using fuzzy controller to compensate for thin-client's processing power and wireless link bandwidth fluctuations. For example, if $B/\alpha >> D(\alpha)$ (i.e. client's processing power is the bottleneck) then we adapt by increasing α until $\alpha \approx B/(3D_0)$. Otherwise, if $B/\alpha << D$ (wireless bandwidth is the bottleneck) then we adapt by decreasing α until $\alpha \approx B/(3D_0)$.

Update Quality-energy Consumption Trade-off

Conceptually, a wireless thin-client needs processor, memory, display, and transceiver. Energy consumed by the processor and the transceiver can be a target of optimization. This is especially important with modern mobile processors that dynamically throttle their power consumption by changing processor's frequency-voltage operating point. Also, modern wireless transceivers offer powerful power management functionalities. Those combined with a highly complexity-scalable decoder present an attractive power optimization mechanism for wireless thin-clients. Thin-clients can dynamically trade off between quality of screen updates and power usage. The thin-client machine is able to detect available battery energy and adjust accordingly the processor's operating frequency. Processor's frequency variations affect the decoding rate $D(\alpha)$. Therefore, if processor's frequency is decreased, then the system should adapt by decreasing the compression ratio (assuming the decoding rate would increase). Consequently, this leads to lower quality of screen updates. The average total energy consumed by single screen rectangle is

$$E_{total} = \frac{k_c}{\mu D(\alpha)} + \frac{k_d \alpha}{\mu B} \tag{6}$$

where k_c is the energy cost per unit time for the processor and k_d is the energy cost per unit time for the transceiver in receiving mode.

IV. The Proxy-based Adaptation Framework

The motivation behind our work is to provide the mobile thin-client user with the best quality of screen updates that can be supported by a wireless link in a user-transparent way. In other words, we want to achieve optimal use of available wireless resources at any given time and any given location without user intervention. We are focusing on one possible optimization: to minimize the average latency observed by the user. Other possible optimizations include power optimization and monetary cost optimization of the wireless thin-client.

The objective of the adaptation framework (Fig. 1) is to achieve minimum total latency of the system by controlling the compression ratio (α) of a wavelet-based encoder at the proxy. This is achieved by making the virtual bandwidth follow a target value (QD_0), which is chosen based on the quality of service requirement. Basically, this is a trade-off between total latency and quality of screen updates. Control action takes place in response to dynamic changes in the wireless link bandwidth and the thin-client processing speed. These variations cause the operating point to change and require a new compression ratio. An error signal (difference between the current virtual bandwidth $BW_{virtual}$ and the target value) is used to drive a fuzzy controller that outputs a new value for compression ratio (α), as shown in Eq. (7) where Q is the screen quality factor.

$$Error = BW_{virtual} - QD_0 \tag{7}$$

The proposed adaptation proxy for wireless thin-clients is shown in Fig. 1. The server sends screen update rectangles to the thin-client through the adaptation proxy. The system follows client-pull protocol where the server sends available screen update only after the client asks for it explicitly. The proxy then encodes these update rectangles using wavelet-based coding and sends them to the thin-client over a wireless link. Wavelet-based image coding has superior rate-distortion performance and scalability compared to the lossy JPEG standard. Its highly scalable rate control allows the thin-client system to offer graceful degradation of quality of service when trading off different performance parameters. Specifically, wavelet-based coding enables trade-offs between encoded image quality and encoded image size (or the computational complexity of decoding).

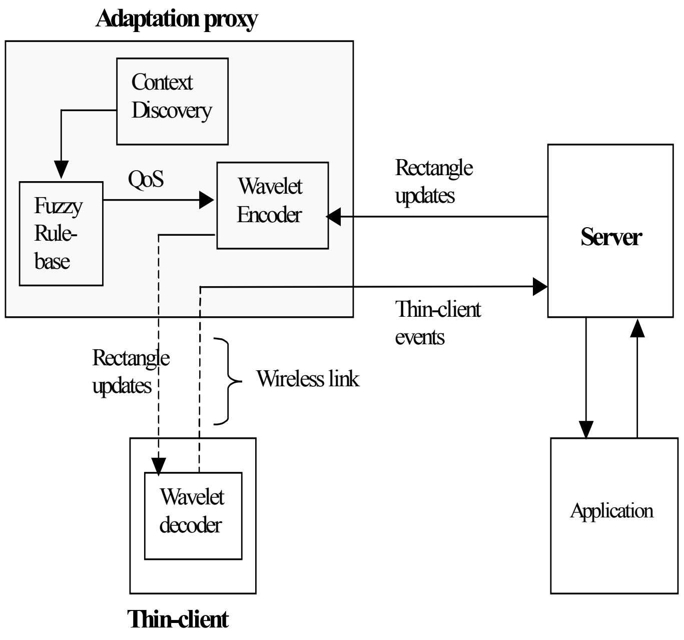

Fig. 1. Thin-client adaptation framework

Fig. 1 shows a context discovery module as part of the adaptation proxy. Its function is to discover the context in which the thin-client and the proxy are operating. This context information may include the characteristics of the wireless link and the thin-client. Generally, the context discovery module is able to elicit the characteristics of the thin-client, such as processing power, memory size, display size, color depth, and battery power. In addition, it is capable of discovering wireless link characteristics, such as bandwidth, latency, and error rate. It feeds collected context information to a fuzzy inference engine, which makes decisions about how to change control actions on the thin-client system to affect the quality of service offered to the user.

Our implementation of this framework targets the case where the thin-client is presenting active media objects that are continuously updating. Examples for such applications could be an animated GIF image file on the Web, Flash Web animation, or Java applet.

One important advantage of our technique is that it does not need to measure the available wireless bandwidth (B) or the exact processing speed of the thin-client. Instead, we only need to estimate the virtual bandwidth. The virtual bandwidth represents the combined effect of wireless link bandwidth, decoding speed of the thin-client, and encoding speed of the proxy. To estimate virtual bandwidth, we measure the time period between two successive, wavelet-encoded, full screen rectangles sent to the thin-client. This approximates the sum of transmission latency, client's decoding latency, and proxy's encoding latency (T_{total}).

Fuzzy Rule-based Controller

Fuzzy logic uses imprecise empirical or expert knowledge instead of differential equations to describe dynamic systems. An important area for applying fuzzy logic is to control complex and non-linear systems because it does not need a mathematical model for the controlled system. Instead, fuzzy control employs a fuzzy rule-based inference engine to capture the dynamic behavior of such systems. The reason for using fuzzy controller in our system is to avoid direct measurement of available wireless bandwidth. Available bandwidth measurement is difficult to implement and is an expensive process. The fuzzy controller enables us to achieve high degree of adaptability with minimum overheads.

Table I FUZZY RULE-BASE FOR THE ADAPTATION MECHANISM

$\dot{BW}_{virtual}$ / $BW_{virtual}$	Neg_Small	Near_Zero	Pos_Small
Neg_Med	Pos_Med	Pos_Med	Pos_Small
Neg_Small	Pos_Small	Pos_Small	Near_Zero
Near_Zero	Pos_Small	Near_Zero	Neg_Small
Pos_Small	Near_Zero	Neg_Small	Neg_Small
Pos_Med	Neg_Small	Neg_Med	Neg_Med

We now present the fuzzy rule-base used to control the latency of our wireless thin-client system. In Table 1, the fuzzy rule that corresponds to the cell located in the first row and first column would read: **If** virtual bandwidth is Neg_Medium **and** rate of change of virtual bandwidth is Neg_Small **then** compression ratio (α) should be Pos_Medium. The fuzzy rules that would contribute to the fuzzy decision-making process depend on the fuzzy input values fed to the fuzzy engine. The results of the fired fuzzy if-then rules overlap to produce an overall output fuzzy set. To get a crisp value that represents the output fuzzy set, the *defuzzification* process is applied to it. The resulting crisp value is then fed back to the system to affect its performance.

V. EXPERIMENTAL RESULTS

Fig. 2 shows decoding time curve for the GWIC wavelet-based decoder. The thin-client tested on Dell Pentium 4, 1.8 GHz with 512 MB RAM. Thin-client's screen size is 800x600 and the color depth is 24 bit/pixel. Decoding time information is needed to characterize the behavior of decoding rate function for the wavelet-based decoder. Fig. 2 shows that the wavelet-based decoding is computationally expensive. It also shows that the GWIC decoder has limited complexity scalability.

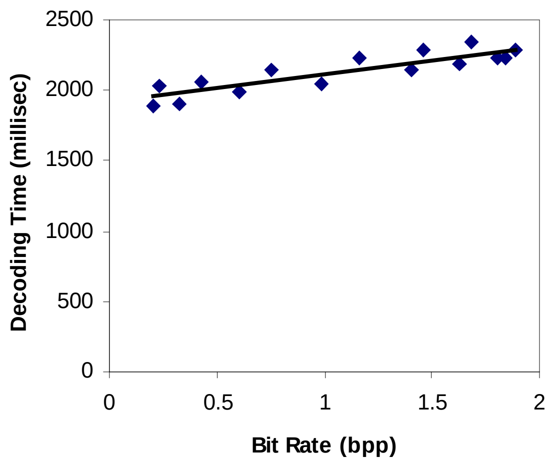

Fig. 2. GWIC decoder's decoding time

We use information inferred from Fig. 2 in designing the adaptation framework. The decoding rate is dependent mainly on the wavelet decoder's processing complexity. Based on decoding time curve, we assume that the decoding time is a linear function of bit rate over the range we considered in our tests. In addition, we need to estimate the thin-client's decoding rate (D_0). For this purpose, we measure the virtual bandwidth when $\alpha \approx 0$, which effectively eliminates the transmission latency contribution. We implement this by sending the first couple of screen updates encoded with a very small compression ratio (e.g., $\alpha = 1/126$). In this case, the decoding rate is $D_0 \approx 1/(\mu T_{total})$, which is used to determine the target virtual bandwidth ($Q \cdot D_0$) for the fuzzy controller.

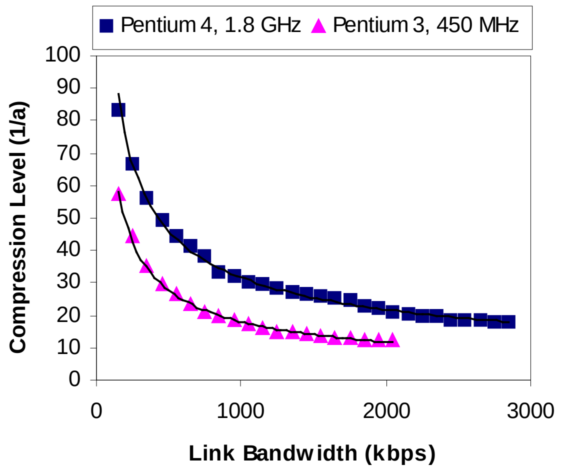

Fig. 3. Compression level control

Fig. 3 shows the performance of the adaptation mechanism for two machines. The square-marked curve represents Dell Pentium 4, 1.8 GHz with 512 MB RAM PC while the triangle-marked curve represents Dell Pentium 3, 450 MHz with 256 MB RAM PC. The screen rectangle size is 800x600 and the color depth is 24 bit/pixel (true color). This figure shows two adaptation aspects. First, it shows how our system adapts to changes in link bandwidth by controlling the compression level to maintain target total latency. For instance, a sudden decrease in the link bandwidth causes the fuzzy controller to output a higher compression level. The target latency for the Pentium 4 machine is 1.7 seconds while for the Pentium 3 machine is 3.36 seconds. The fuzzy engine increases the compression level to adapt to the decrease in link bandwidth. Second, it shows how the system adapts and responds to different thin-client processing speeds. For the fast machine (Pentium 4), the fuzzy controller compresses more (which reduces transmission latency) to keep up with the fast decoding rate of the thin-client. This prevents the transmission latency from being the performance bottleneck.

We emphasize here an important advantage of our adaptation mechanism. It does not need to measure the real link bandwidth. Bandwidth estimation is costly and difficult to implement and it introduces overheads. We instead rely on total latency of the system to estimate the virtual bandwidth ($BW_{virtual}$).

Simulating bandwidth variation is achieved by setting the link bandwidth between the thin-client and the proxy to the desired value using CBQ-based traffic control [9] available on Linux operating system. We then observe the compression ratio outputted by the fuzzy controller at steady-state condition.

Fig. 4 shows the adaptation action using bit rate notation to express the amount of compression applied to screen update rectangles (Bit rate= $24 \cdot \alpha$). Therefore, similar conclusions are drawn from this figure. However, it shows a linear relationship between bit rate and link bandwidth.

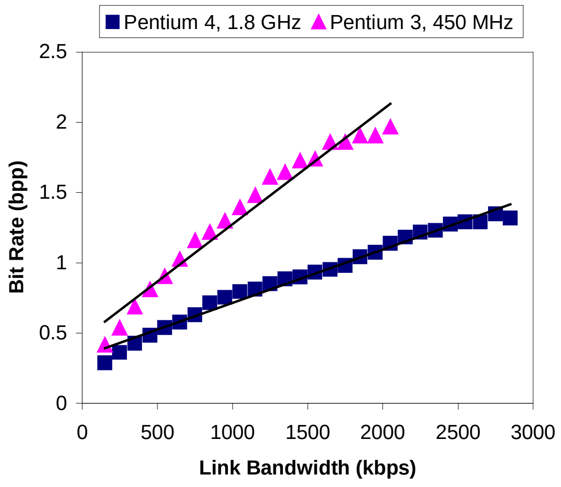

Fig. 4. Bit Rate control

VI. Related Work

The Transend system of UC Berkeley [4] is a proxy-based adaptation system that uses dynamic transcoding of multimedia Web objects. It employs lossy compression of images to reduce bandwidth usage and to enable low-latency Web surfing on handheld devices such as Palm devices. This project demonstrates that on-demand adaptation using transformational (transcoding) proxy is practical and economic. A major objective for this project is to address the need to adapt to network and client variations.

Data-specific transcoding enables application-level network resources management by controlling the transcoding level. Transend's proxy architecture can support dynamic adaptation to changing network conditions. Transend project researchers point out that their system has potentially the best performance when utilizing a network connection monitor. They suggest an automatic adaptation mode where a network monitor can discover effective bandwidth and roundtrip latency. However, they did not present performance results for the automatic adaptation mode.

Compared to Transend project, our system is not limited to Web browsing and applicable to any application running on the server. In addition, our system enables user-transparent adaptation of active media presentation (such as active Flash presentation or Java applet). We use the virtual bandwidth concept, which represents the combined effect of the wireless link bandwidth and the client processing speed. Based on this information, the fuzzy controller fires to decide on the compression level that is needed to achieve target latency and quality of screen updates. Our system does not need to measure directly the available link bandwidth or the client processing speed.

VII. Conclusion

We propose a proxy-based adaptation framework for wireless thin-client systems. It dynamically adapts the performance of wireless thin-client using dynamic context discovery. This context information is used by a fuzzy rule-based inference engine to optimize wireless resources usage by trading off among different quality of service parameters offered to end users. It uses highly scalable wavelet-based image coding technique to provide high scalability of quality of service that degrades gracefully. This framework shields the user from the ill effects of abrupt variability of the wireless network and mobile device resources.

VIII. Acknowledgments

We would like to thank the developers of the following open-source projects used in the implementation and testing of our proposed adaptation framework: Virtual Network Computing system of AT&T Labs [1], GWIC-GNU Wavelet Image Codec [10], and RFB Proxy [13].

IX. References

[1] AT&T Laboratories Cambridge, **Virtual Network Computing**, http://www.uk.research.att.com/vnc, 2003.

[2] B. Badrinath, A. Fox, L. Kleinrock, G. Popek, P. Reiher, and M. Satyanarayanan, "A Conceptual Framework for Network and Client Adaptation", **Mobile Networks and Applications**, Vol. 5, No. 4, 2000, pp. 221-231.

[3] S. Chandra, C. Ellis, and A. Vahdat, "Application-Level Differentiated Multimedia Web Services Using Quality Aware Transcoding", **IEEE Journal on Selected Areas in Communications**- Special Issue on QOS in the Internet, Vol. 18, No. 12, 2000.

[4] A. Fox, S. Gribble, Y. Chawathe, and E.A. Brewer, "Adapting to Network and Client Variation using Infrastructural Proxies: Lessons and Perspectives", **IEEE Personal Communications**, Vol.5, No. 4, pp. 10-19, August 1998.

[5] J. Jing, A. Helal, A. Elmagarmid, "Client-Server Computing in Mobile Environments", **ACM Computing Surveys**, Vol. 31, No. 2, 1999, pp. 117-157.

[6] D. Johnson, and D. Maltz, "Protocols for Adaptive Wireless and Mobile Networking", **IEEE Personal Communications**, Vol. 3, No. 1, 1996, pp. 34-42.

[7] R. Katz, "Adaptation and Mobility in Wireless Information Systems", **IEEE Personal Communication**, Vol. 1, No. 1, First quarter 1994.

[8] T. Kunz, and J. Black, "An Architecture for Adaptive Mobile Applications", **Proceedings of the 11th International Conference on Wireless Communications**, 1999, pp. 27-38.

[9] A. Kuznetsov, CBQ.init Traffic Management Script, https://sourceforge.net/projects/cbqinit, 2003.

[10] J. Lehtinen, GWIC: GNU Wavelet Image Codec, http://jole.fi/research/gwic, 1998.

[11] S. Seshan, M. Stemm, and R. Katz, "SPAND: Shared Passive Network Performance Discovery", **Proceedings of 1st Usenix Symposium on Internet Technologies and Systems**, 1997, pp. 135-146.

[12] M. Stemm, S. Seshan, and R. Katz, "A Network Measurement Architecture for Adaptive Applications", **Proceedings of IEEE INFOCOM 2000**, Vol. 1, 2000, pp. 285-294.

[13] T. Waugh, RFB proxy, http://cyberelk.net/tim/rfbproxy, 2002.