Intelligent Adaptation Framework for Wireless Thin-client Environments

Mohammad Al-Turkistany and Abdelsalam (Sumi) Helal

Computer and Information Science and Engineering Department
University of Florida, Gainesville, FL 32611, USA

helal@cise.ufl.edu

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, 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 its 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 compression technique to enable variable and scalable compression of display rectangles. It enables 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 implementation, and report on measured system performance under variable application and network parameters.

Keywords: *thin-clients, adaptive framework, wireless networks*

1. Introduction

The concept of *application-level adaptation,* which was introduced by M. Satyanarayanan (surveyed in [Jing99]) allows mobile applications to make tradeoff 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, networking 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 a client over a wireless link. This is particularly important since thin-client systems place large traffic load on wireless links when compared to a standard client-server computing model.

Furthermore, there is usage cost incentive for optimal utilization of wireless bandwidth since most of the wireless data service providers charge customers based on the amount of data they transmit and receive over their network. Battery energy limitations of mobile and portable devices may be the single most important constraint in wireless thin-client system. There is a need to optimize the thin-client usage of battery power through optimizing the encoding schemes used to send screen updates. It is essential for these encodings to have low computational complexity as much as possible to minimize energy consumption. This may require a trade-off between computational complexity and quality of service (compression level) of 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 UK Labs. 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 1.1) and then describe a wireless thin-client performance model on which we base our framework and inference engine (Section 2). The framework and its implementation are presented in Section 3. Experimental evaluation of the effectiveness of the framework and its adaptability is given in Section 4. Finally, Section 5 ends with concluding remarks.

1.1 Virtual Network Computing (VNC) Thin-client System

VNC is an open-source thin-client system from AT&T. VNC's Remote Frame Buffer (RFB) protocol enables VNC server to send frame buffer updates to remote client. The basic primitive in RFB protocol is putting 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 limited ability to dynamically adapt to changing wireless link conditions.

2. 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 can model it using a simple model of three cascading M/M/1 queues as shown in Figure 1. To simplify the analysis, we assume the server processing power is highly controllable and can be made arbitrary large compared to thin-client's processing power. For now, we drop the effect of server part in the model.

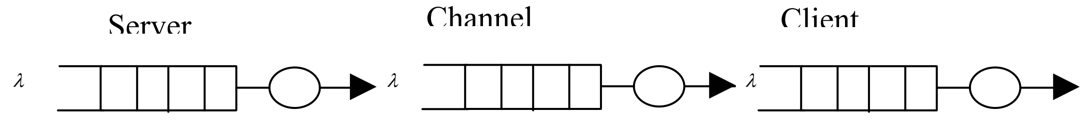

Figure 1. Queuing model for wireless thin-client system

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

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

Therefore, the average total latency is

In equation (3), stability condition for the system requires that $\mu D(\alpha) > \lambda$ and $\mu B / \alpha > \lambda$. This 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. This function is highly non-linear and hard to model mathematically.

For further analysis of Equation (3), we assume that $\frac{B}{\alpha} >> \lambda$ and $D(\alpha) >> \lambda$ which means that we are ignoring for now the effect of queuing latency. Hence, Equation (3) becomes

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

Therefore,

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

The virtual bandwidth, $BW_{virtual}$, is the target of our optimisation since by maximizing it, we minimize the system's total latency. If a decoder has $D(\alpha)$ that changes slowly or is almost constant over the domain of α (i.e. $D(\alpha) \approx D_0$), which is the case for GWIC wavelet-based decoder we are using, then

$$BW_{virtual} \leq D_0$$

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

Figure 2 shows the operating point for the adaptation mechanism. It shows the intersection of decoding rate curve with effective bandwidth curve at $\alpha = B / D_0$. This results in $BW_{virtual} = D_0 / 2$. In our implementation, we have chosen a target value of $BW_{virtual} = 3D_0 / 4$. As stated before, this value is dependent on the quality of service expected by users of thin-client system.

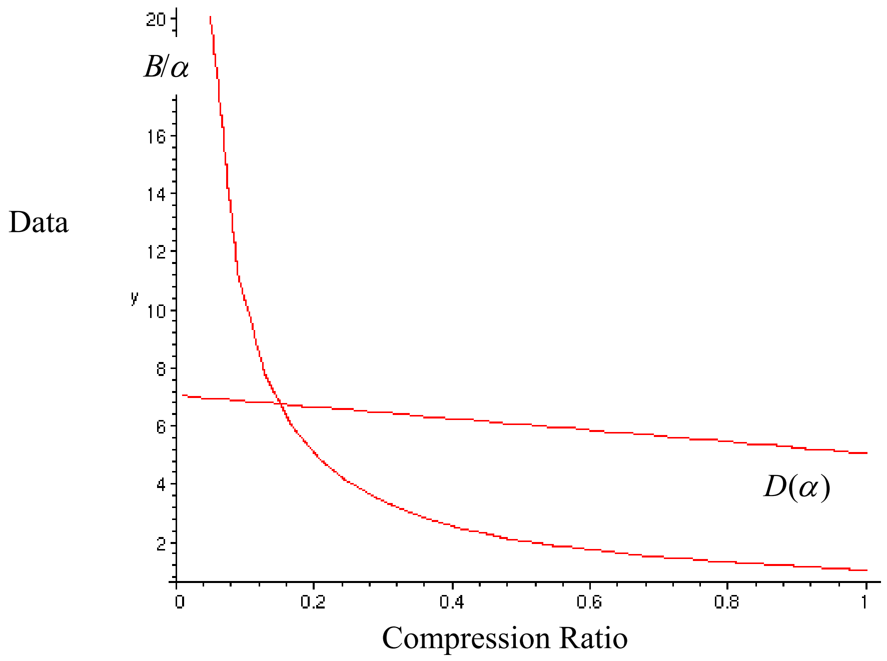

Figure 2. Data rates for the wavelet-based decoder and wireless effective bandwidth

The objective of the adaptation framework is to minimize the total latency of the system by controlling the compression ratio (α) of wavelet encoder at proxy side. This is achieved by making the virtual bandwidth reach a target value (QD_0) chosen based on quality of service requirement. Basically, this is a trade-off between total latency and screen updates quality. Control action takes place in response to dynamic changes in wireless link bandwidth and thin-client processing power. These changes cause the operating point to move and requires new compression ratio. An error signal (difference between current virtual bandwidth $BW_{virtual}$ and the target value) is used to drive a fuzzy engine that outputs a new value for compression ratio (α) as shown in equation (6) where Q is the quality of service coefficient.

$$Error = BW_{virtual} - QD_0 \tag{6}$$

We propose a power cost model for the pure thin-client system. A wireless thin-client only needs a processor, memory, display and a transceiver. Energy consumed by processor and transceiver can be target of optimisation. This is especially important with modern mobile processors that can throttle dynamically their power consumption by changing processor's frequency-voltage operating point. Also, modern wireless transceivers offer powerful power management functionality's. Those combined with a highly complexity-scalable decoder present an attractive power optimisation mechanism for wireless thin-clients. Thin-clients can dynamically trade-off between quality of service and power usage. The average total energy consumed by single screen rectangle is

$$E_{total} = \frac{k_c}{\mu D(\alpha)} + k_d \frac{\alpha}{\mu B} \tag{4}$$

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.

3. The Proxy-based Adaptation Framework

The main objective of our architecture is to provide mobile user with the best quality of service that can be supported by wireless link in 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. One possible optimisation, which we will focus on, is to minimize the average latency observed by the user. Other optimisations include power optimisation and monetary cost optimisation of wireless thin-client.

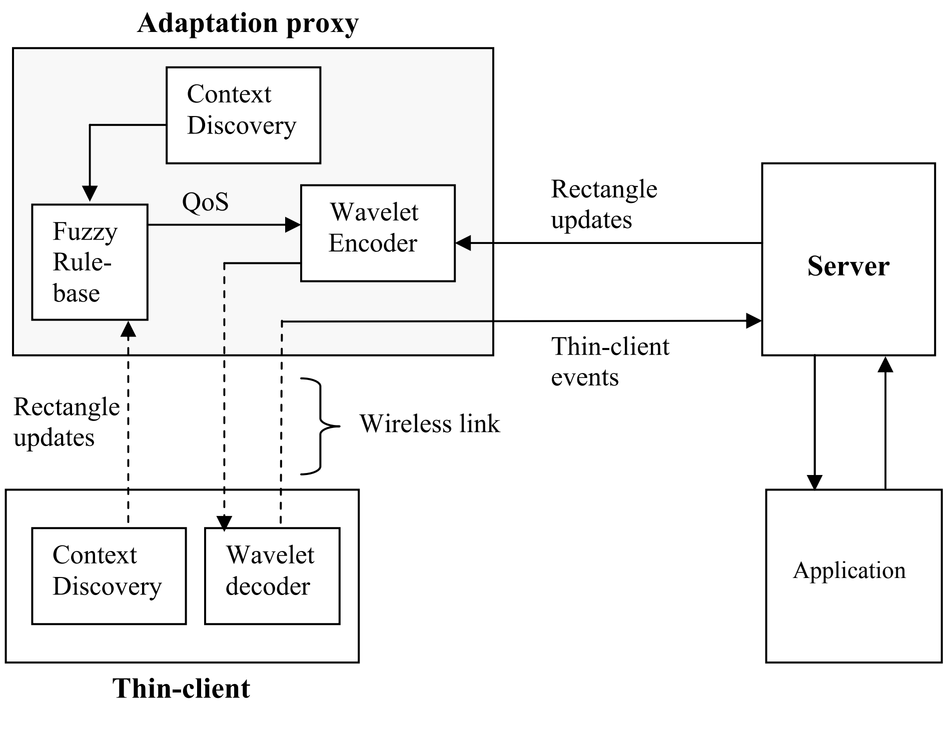

Figure 3. Thin-client adaptation framework

We propose an adaptation framework for wireless thin-clients as shown in Figure 3. The 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 encoder enables trade-offs between encoded image quality and encoded image size (or the computational complexity of decoder).
Figure 3 shows context discovery module as part of proxy and thin-client modules. Its function is to discover the context in which the thin-client and proxy are operating. This context information may include wireless link and thin-client characteristics. Generally, context discovery module is able to elicit the characteristics of thin-client such as its processing power, memory size, display size, color depth, and battery power. In addition, it is capable of discovering wireless link's characteristics such as bandwidth, latency, and error rate. It feeds context information it collects to fuzzy inference engine, which processes it and makes decisions about how to change control policies of thin-client system to affect the quality of service offered to thin-client user.

3.1 Fuzzy Rule-based Adaptation

Fuzzy logic uses empirical or expert knowledge instead of differential equations to describe a dynamic system. An important area for applying fuzzy logic is to control complex and highly non-linear systems. This is because it does not need a mathematical model for a controlled system. Instead, it employs fuzzy rule-based inference engine to capture the dynamic behavior of such systems. Now, we present the fuzzy rule-base used to control and optimize the latency of our wireless thin-client system.

In Figure 4, the fuzzy rule that corresponds to cell located at first row and first column would read: **If** virtual bandwidth is Neg_Medium **and** virtual bandwidth rate of change is Neg_Small **then** compression ratio (α) should be Pos_Medium. The fuzzy rules that would contribute to fuzzy decision-making process depend on fuzzy inputs to the fuzzy engine. The fired fuzzy if-then rules results overlap to produce an overall output fuzzy set. To get a crisp value that represent the output fuzzy set, the *defuzzification* process is applied to it. Then, the resulting crisp value is fed back to the system to affect its behavior.

		Bandwidth rate of change		
		Neg_Small	Around_Zero	Pos_Small
Band-width	Neg_Medium	Pos_Medium	Pos_Medium	Pos_Small
	Neg_Small	Pos_Small	Pos_Small	Around_Zero
	Around_Zero	Pos_Small	Around_Zero	Neg_Small
	Pos_Small	Around_Zero	Neg_Small	Neg_Small
	Pos_Medium	Neg_Small	Neg_Medium	Neg_Medium

Figure 4. Fuzzy Rule Base for wireless thin-client system

4. Experimental Results

Figure 5 shows decoding time curve for GWIC wavelet-based thin-client decoder. The thin-client tested on Dell Pentium 4, 1.8 GHz with 512 MB RAM. Thin-client's screen size is 800x600 and colour depth is 24 bit/pixel. Decoding time information is needed to characterize the behaviour of decoding speed function for wavelet-based decoder.

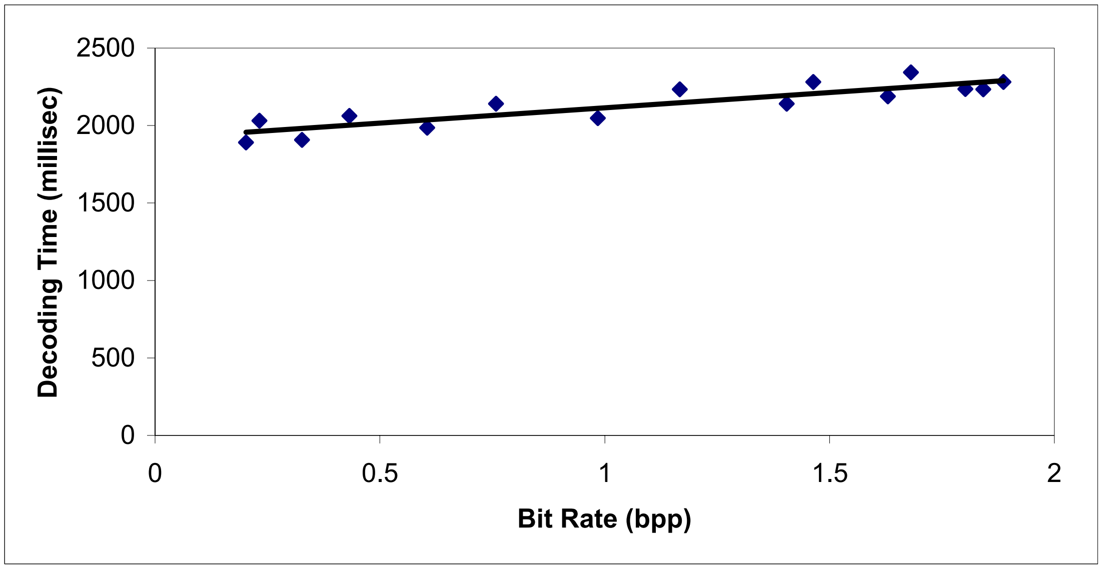

Figure 5. GWIC decoder's decoding time

We used information inferred from Figure 5 in designing our adaptation framework. Specifically, we used it to identify decoding speed function D (α). This is mainly dependent on wavelet decoder's processing complexity. Based on this curve, we assume that decoding time is a linear function of bit rate over the range we considered in our tests. Our framework designed for the case where the thin-client is presenting active media object that is continuously updating. Examples for such applications could be an animated GIF image file on the Web, Flash Web animation, or streaming video screen.

One strong advantage in our technique is that it does not need to measure the actual wireless bandwidth (B). Instead, we need only to estimate the virtual bandwidth. Virtual bandwidth represents the combined effect of wireless link's bandwidth, decoding speed of thin-client, and encoding speed of proxy. To estimate it, we measure the time period between two successive, wavelet-encoded, screen rectangles sent to thin-client. This approximates the sum of transmission latency, client's decoding latency, and proxy's encoding latency (T_{total}). In addition, we need to estimate thin-client's decoding rate (D_0). For this purpose, we measure virtual bandwidth when $\alpha \approx 0$ which effectively eliminate the transmission latency contribution. We implement this by sending the first couple of screen updates with maximum compression possible (i.e. $\alpha = 1/126$). In this case, the decoding rate is $D_0 \approx \frac{1}{\mu \cdot T_{total}}$ which used to determine the target virtual bandwidth ($Q \cdot D_0$) for the fuzzy controller.

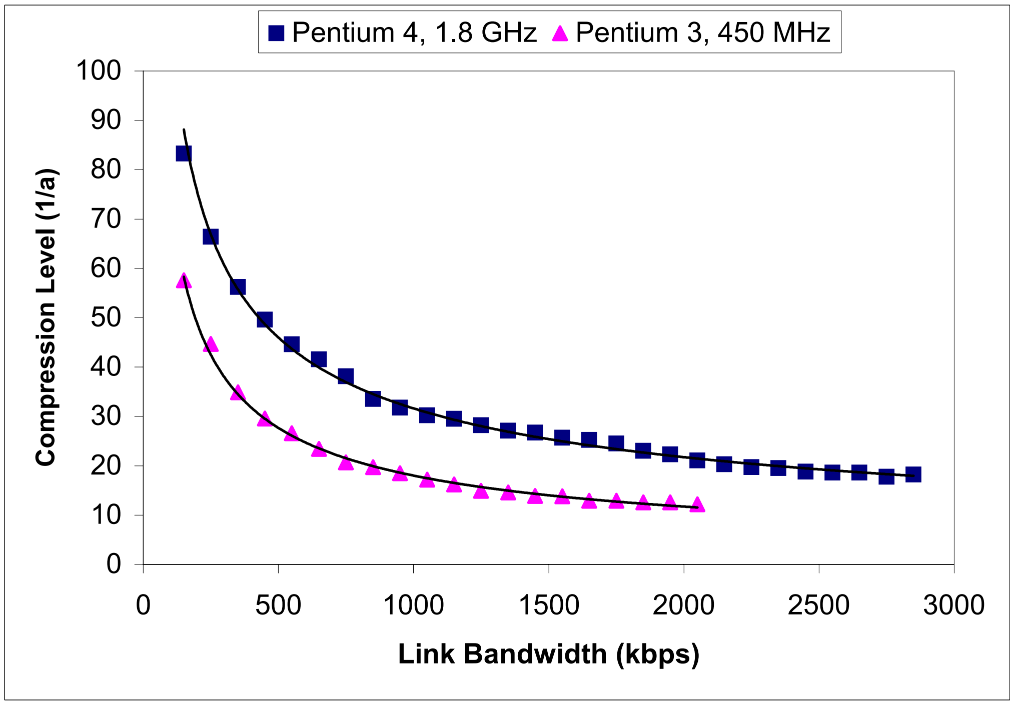

Figure 6 Compression level control process

Figure 6 shows the adaptation process for two machines. The square-marked curve represents Dell Pentium 4, 1.8 GHz with 512 MB RAM while triangle-marked curve represents Dell Pentium 3, 450 MHz with 256 MB RAM. The screen rectangle size is 800x600 and color depth is 24 bit/pixel. This figure shows two adaptation aspects. Firstly, it shows how our system adapts to changes in link bandwidth by controlling compression level to maintain minimum total latency. The target latency for Pentium 4 machine is 1.7 seconds while for Pentium 3 machine is 3.36 seconds. As shown, the fuzzy engine increases the compression level in response to decrease in link bandwidth. Secondly, it shows how the system adapts and responds to different thin-client processing speeds. For fast machine, the fuzzy engine has to compress more (which reduces transmission latency) to keep up with the fast decoding rate of thin-client to avoid being the performance bottleneck as represented by total latency (T_{total}).

We note here a big advantage of our adaptation mechanism. It does not need to measure the real link bandwidth. Bandwidth estimation is costly and difficult process and can introduce an overhead if not done passively. Instead, we rely on total latency of the system to estimate the virtual bandwidth ($BW_{virtual}$).

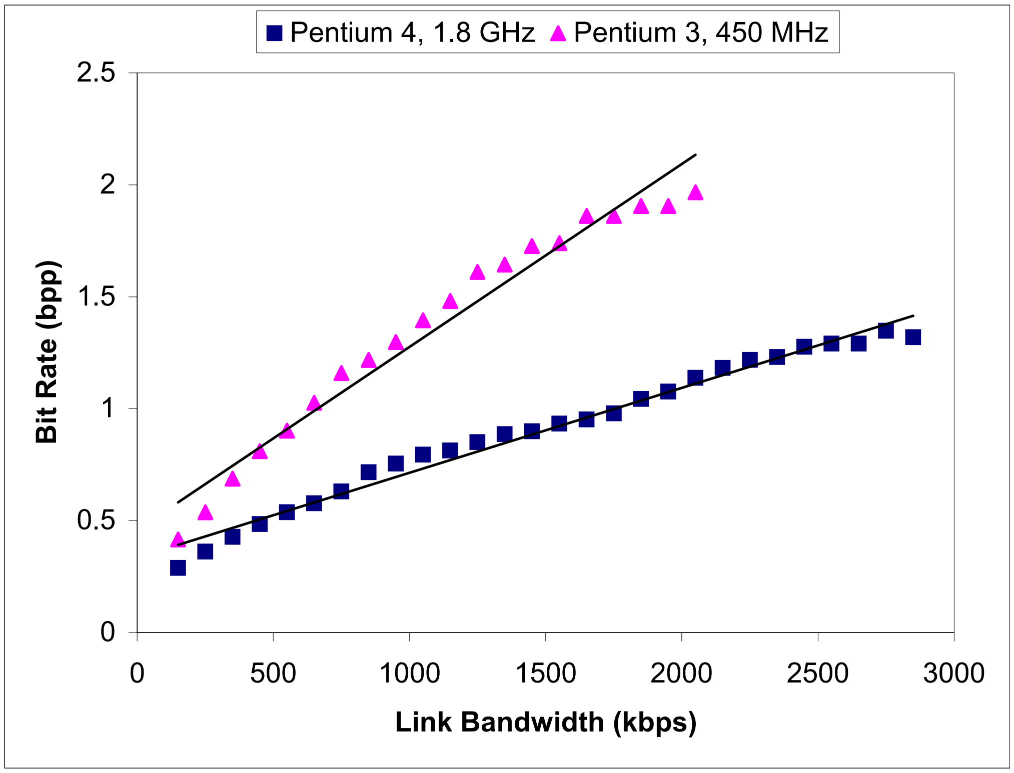

Figure 7 Bit Rate control process

Figure 7 shows the information in figure 5 using bite rate notation to express the amount of compression applied to screen update rectangles (Bit rate= $24 \cdot \alpha$). Therefore, the same conclusions are drawn form this figure. However, it shows a linear relationship between bit rate and link bandwidth.

5. Related Work

The Transend system at UC Berkeley [Fox98] is a proxy-based adaptation system based on transcoding web objects and images. It employs lossy compression of images to reduce bandwidth usage and enables low-latency web surfing on handheld devices such as Palm devices.

This project demonstrates that on-demand adaptation using transformational proxy is practical, and economic. A major objective for this project is to address the need to adapt to network and client variations.

The core of their system is on-demand datatype-specific distillation using lossy compression. It enables application-level network resources management by controlling the transcoding level. Transcoding multimedia objects to lower quality representation reduces end-to-end latency perceived by the client. Transend's proxy architecture can support dynamic adaptation to changing network conditions. Transend project researchers point out that their system has best performance when utilizing a network connection monitor. They suggest an automatic adaptation mode where network monitor can discover effective bandwidth, and roundtrip latency. However, they did not present implementation for the automatic adaptation mode.

Also, the Quality Aware Transcoding project [Chan00] uses image object transcoding to enable web servers to manage their available bandwidth. It enables dynamically changing the size of multimedia objects to match different client classes. It employs image quality versus size tradeoffs to transcode JPEG images. This system uses proxy-based Transcoding servers to offer differentiated services by dynamically adapting media objects size to the bandwidth of different network classes. They present some Transcoding polices and demonstrated how informed Transcoding technique can offer good quality of multimedia objects while reducing network bandwidth usage. Essentially, this facilitates application-level network resources management based on defined client classes while delivering a high quality media content.

Compared to these two projects, our system enables thin-client transparent adaptation of active media presentation (such as active Flash presentation or video streaming). We use virtual bandwidth estimation technique, which represent the combined effect of wireless link bandwidth and client's processing speed. Based on this information, a fuzzy engine fires to decide on compression level that is needed to achieve target latency and quality of thin-client's screen updates. Our system does not need to measure directly actual link bandwidth neither client's processing power. Both of those research projects do not support automatic adaptation of media objects to client processing power and wireless bandwidth in client transparent way.

6. 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 the end users. It uses highly scalable wavelet-based image coding techniques 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 the mobile device resources.

7. References

[Badr00] Badrinath, B., Fox, A., Kleinrock, L., Popek, G., Reiher, P., and Satyanarayanan, M., A Conceptual Framework for Network and Client Adaptation, Mobile Networks and Applications, Vol. 5, No. 4, 2000.

[Chan00] Chandra, S., Ellis, C. and Vahdat, A., Application-Level Differentiated Multimedia Web Services Using Quality Aware Transcoding, In IEEE Journal on Selected Areas in Communications - Special Issue on QOS in the Internet, 2000.

[Fox98] Fox, A., Gribble, S.D., Chawathe, Y., and Brewer, E.A., Adapting to Network and Client Variation using Infrastructural Proxies: Lessons and Perspectives, IEEE Personal Communications, 5(4):10-19, August 1998.

[Jing99] Jing, J., Helal, A., Elmagarmid, A., Client-Server Computing in Mobile Environments, ACM Computing Surveys Vol. 31, No. 2, pp. 117 - 157, June 1999.

[John96] Johnson, D. and Maltz, D., Protocols for Adaptive Wireless and Mobile Networking, IEEE Personal Communications, 3(1):34-42, February 1996.

[Katz94] Katz, R., Adaptation and Mobility in Wireless Information Systems, IEEE Personal Communication, First quarter 1994.

[Kunz99] Kunz, T. and Black, J., An Architecture for Adaptive Mobile Applications, in Proceedings of Wireless 99, the 11th International Conference on Wireless Communications, Calgary, Alberta, Canada, pages 27-38, July 1999.

[Sesh97] Seshan, S., Stemm, M. and Katz, R., SPAND: Shared Passive Network Performance Discovery, Proceedings of 1st Usenix Symposium on Internet Technologies and Systems (USITS '97), Monterey, CA, December 1997.

[Sesh00] Seshan, S., Stemm, M. and Katz, R., A Network Measurement Architecture for Adaptive Applications, Proceedings of IEEE Infocom 2000, Tel Aviv, Israel, March 2000.