

Adaptive Wireless Thin-client Model for Mobile Computing

Mohammad Al-Turkistany, Abdelsalam (Sumi) Helal and Mark Schmalz

Abstract—The thin-client computing model has the potential to significantly increase the performance of mobile computing environments. By delivering any application through a single, small-footprint client (called a *thin client*) implemented on a mobile device, it is possible to optimize application performance without the need for building wireless application gateways. We thus present two significant contributions in the area of wireless thin-client computing. Firstly, a mathematical performance model is derived for wireless thin-client system. This model identifies factors that affect the performance of the system, and supports derivation and analysis of adaptation strategies to maintain a user-specified quality of service (QoS).

Secondly, a proxy-based adaptation framework is developed for wireless thin-client systems, which dynamically optimizes performance of a wireless thin client via dynamically discovered context. This is implemented with rule-based fuzzy logic that responds to variations in wireless link bandwidth and client processing power. Our fuzzy inference engine uses contextual data to dynamically optimize tradeoffs among different quality of service parameters offered to end users. Additionally, our adaptation framework uses highly scalable wavelet-based image coding to provide scalable QoS that can degrade gracefully.

Our thin-client adaptation framework shields the user from ill effects of highly variable wireless network quality and mobile device resources. This improves performance of active applications, in which the display changes frequently. Further, active application behavior may produce high transmission latency for screen updates, which can adversely affect user perception of QoS, resulting in poor interactivity. We report measured adaptive performance under realistic mobile device and network conditions, for several different clients and servers.

Index Terms— mobility adaptations, mobile computing models, thin-client model.

I. INTRODUCTION

THE concept of *application-level adaptation*, which was introduced by M. Satyanarayanan (surveyed in [Jing99]), allows mobile computing applications to select certain types of service over others, via tradeoff analysis based on application

semantics or knowledge. Salient quality-of-service (QoS) parameters include but are not limited to bandwidth, latency, error rate, usage cost, and power consumption. For example, consider a mobile device with limited internal resources and a specific wireless link quality. In this context, mobile applications adapt and manifest different requirements to the underlying system, based on available computing and communication resources. Challenges to adaptation have multiple sources. For example, in communication, QoS variations could result from signal interference or jamming, or from a change in the number of users sharing resources at a cell location. Computation also presents challenges, for example, in graphics applications that require intensive floating-point calculations.

A solution is presented by a *wireless thin client*, which is comprised of a processor, memory, display and transceiver. Performance parameters of the processor and transceiver can be optimised dynamically. For example, modern wireless transceivers offer power management capabilities that, when combined with a complexity-scalable decoder, present an attractive power optimization mechanism for thin clients. In practice, thin clients dynamically trade off quality of screen updates and power usage.

Additionally, to enable dynamically adaptive applications, communication protocols must be able to discover wireless link parameters, allowing application-level protocols to adjust their behavior accordingly. For example, thin-client systems place large traffic load on wireless links when compared to the standard client-server computing model. Thus, a thin-client system server should be able to dynamically change the type or level of compression or error correction used to transmit screen updates to client over a wireless link.

As mentioned previously, there is a usage cost incentive for optimal utilization of wireless bandwidth, because most wireless data service providers charge customers according to the amount of data transmitted and received. With respect to power consumption, battery constraints on mobile device functionality can be the primary constraint in wireless thin-client system. As a result, there is a need to optimize thin-client usage of battery power, for example, by optimizing the encoding schemes that support transmission of screen updates.

In general, across each performance modality (bandwidth, QoS, power, etc.), it is essential for applications to have scalable complexity, to robustly meet performance objectives. This implies multi-parameter optimization supporting the

M. Al-Turkistany is with the Computer and Information Science and Engineering Department, University of Florida, Gainesville, FL 32611 USA (malturk@cise.ufl.edu).

A. Helal is with the Computer and Information Science and Engineering Department, University of Florida, Gainesville, FL 32611 USA (helal@cise.ufl.edu).

M. Schmalz is with the Computer and Information Science and Engineering Department, University of Florida, Gainesville, FL 32611 USA (mssz@cise.ufl.edu).

adaptivity inherent in a thin-client system.

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 paradigm from AT&T Cambridge Laboratories [Att03]. We describe our approach in detail and show how our algorithm adapts dynamically, primarily via context discovery through fuzzy rule-based inference. In particular, we first overview VNC (Section II), then describe a wireless thin-client performance model on which our optimization algorithm is based (Section III). Implementational issues pertaining to the algorithm and its underlying inference engine are discussed in Section IV. Experimental evaluation of algorithm performance and adaptability is given in Section V, related work is discussed in Section VI, and concluding remarks are given in Section VII.

II. 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 a VNC server to send frame buffer updates to a remote client. The basic primitive in the RFB protocol places a rectangle of pixel data at a given location in the client's frame buffer. Each frame buffer update comprises a number of on-screen rectangles. The RFB protocol does not necessarily support a high compression ratio for applications that display complex graphics or natural images. Consequently, the VNC thin-client system requires relatively high bandwidth from a network infrastructure. This dramatically degrades the performance of a wireless thin-client system based on VNC, because the resulting transmission latencies can dissatisfy users. Furthermore, the RFB protocol suffers from limited adaptation 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

In this section, we develop a mathematical model, based on queuing theory, to support analysis of performance bottlenecks in wireless thin-client system.

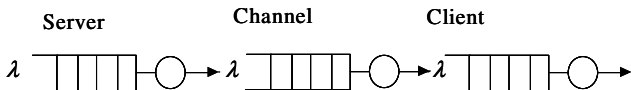


Fig. 1. Performance model using three M/M/1 queues

We model the wireless thin-client system by considering factors affecting its performance, for example, wireless link bandwidth as well as client and server processing power. In practice, the simple model of three cascading M/M/1 queues shown in Fig. 1 suffices. The arrival rate at each queue is the same when the system is running under stable conditions (i.e., steady state operation). Thus, when the system is stable, the average length of each queue remains the same. In such cases, traffic entering any queue must leave the server at the same

rate, to avoid queue overflow.

For efficiency, our model assumes incremental screen updates, in which the server transmits only screen rectangles that have recently changed. To simplify the analysis, we assume the server's computational capabilities are highly controllable, and can be made arbitrary large compared to the thin-client's processing power. Thus, it is possible to neglect the server effects in our model.

In the communication channel model, the average latency is given by

$$T_d = \frac{1}{B\mu - \lambda}, \quad (1)$$

where λ denotes arrival rate in rectangles/sec, $1/\mu$ denotes average rectangle size in bits/rectangle, and B denotes link bandwidth in bits per second (bps). Here, the tuple (λ, μ) represents the application's screen update characteristics.

Similarly, the thin client's average latency is given by

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

where $D(\alpha)$ denotes the decoder throughput in bps, and the degree of compression is given by $0 < \alpha < 1$, with compression ratio $CR = 1/\alpha$. Generally, the decoding rate is a function of several variables such as screen rectangle content, decoding algorithm, client processing power, and specified compression ratio. For realism, the decoding function is assumed to be nonlinear and difficult to model mathematically.

Using queueing theory, and treating the system as two separate servers, the average total latency for the wireless thin-client system is the sum of the average response times given in Equations (1) and (2), that is

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

where T_p denotes communication link latency. In Equation (3), stability constraints require that $\mu D(\alpha) > \lambda$ and $\mu B / \alpha > \lambda$. That is, the service rate at each queue must exceed the arrival rate. Therefore, the utilization factor (the arrival rate divided by the service rate) for each queue must be less than unity.

Implementationally, this constraint prevents infinite delays in screen rectangle transmission, and guarantees that queues do not overflow. The queue with the lower service rate is the performance bottleneck, since (a) arrival rates are the same for both queues and (b) the queue having lower service rate has a higher utilization factor.

A further parameter of interest to performance analysis is effective bandwidth $\underline{B} = B / \alpha$. We next discuss several specific cases of the model given in Equation (3), in particular, bandwidth-constrained or I/O bound mode (in terms of \underline{B}), compute-bound mode, and overloaded mode, then present a technique for design tradeoffs based on performance analysis.

A. Wireless Bandwidth Constrained Mode

In practice, a large number of users can migrate into a wireless LAN coverage area, resulting in decreased effective link bandwidth available to a thin-client device. As a result, the effective bandwidth can become less than the thin client's decoding rate. Consequently, the service rate of the wireless channel falls below the service rate of the thin-client processor. From Equation (3), it is readily seen that latency is mainly dominated by the effect of wireless link bandwidth, which represents the performance bottleneck.

B. Processing Power-Constrained Mode

A thin-client device may decrease its processor speed to conserve power, for example, given a battery level alarm. This can force a wireless thin client to operate within constrained computational abilities. In such cases, the decoding speed of the thin-client device may become less than the transmission bandwidth of the wireless link, that is, $D(\alpha) < B$. Consequently, the client's computational service rate is less than the wireless channel's service rate. In this case, Equation (3) indicates that system latency is mainly limited by the constrained decoding rate of the client device. This means that client processing speed is the performance bottleneck.

C. Overloaded Mode of Operation

Overloaded operation can occur when a thin-client system transitions from stable to unstable operation. For example, consider an increase in application or user activity, such that the client's channel or processor utilization factor approximates or exceeds unity. This forces one of the queues to grow, which means that its service rate is less than the arrival rate of screen update rectangles. This mode is undesirable, because it leads to very high latency that can be observed by the user. In response, the system usually drops some of the arriving screen update rectangles, which adversely impacts display quality.

In Equation (3), if the system operates in client pull mode, such that $B \gg \lambda$ and $D(\alpha) \gg \lambda$, then there is no queuing latency. Hence, Equation (3) becomes

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

where the virtual bandwidth $B_{virtual}$, representing the combined effect of transmission latency and processing latency in the system, is given by

$$B_{virtual} = \frac{B \cdot D(\alpha)}{\alpha D(\alpha) + B}. \quad (5)$$

By maximizing virtual bandwidth, we minimize total system latency. For example, given a nominal value D_0 , if the decoding rate function $D(\alpha)$ decreases monotonically over the domain of α , such that $D(\alpha) \approx D_0$ at $\alpha \approx 0$, then

$$B_{virtual} \leq D_0 \quad (6)$$

This is the case for the GWIC wavelet-based decoder employed in this study [Leh98].

D. Trade-off between Update Quality and Latency

The maximum virtual bandwidth (best-case latency) is achieved when $B_{virtual} \approx D_0$, which occurs when $\alpha \approx 0$. This result corresponds to thin client's worst screen quality and highest compression ratio (e.g., when using a lossy encoder). However, a practical application must satisfy the user's requirement of high quality and fast response. Thus, we have developed a tradeoff between $B_{virtual}$ and screen update quality, approximated by α . For a lossy wavelet-based encoder, decreasing α results in decreased display quality. Therefore, the target virtual bandwidth is set per user-specified QoS constraints.

For example, if the screen update quality constraint indicates maximum virtual bandwidth (e.g., $B_{virtual} = 3D_0/4$), then from Equation (5) we obtain $\alpha = B/(3D_0)$. Dynamic adaptation is achieved by controlling α at the server (or proxy) side using a fuzzy controller to compensate for the thin client's lack of processing power and fluctuations in wireless link bandwidth. For example, if $B/\alpha \gg D(\alpha)$ (the bottleneck is client processing power), then the thin-client system adapts by increasing α until $\alpha \approx B/(3D_0)$. Otherwise, if $B/\alpha \ll D$ (bottleneck is wireless bandwidth) then the system adapts by decreasing α until $\alpha \approx B/(3D_0)$.

E. Trade-off between Update Quality and Energy Consumption

In wireless thin clients comprised of a processor, memory, display and transceiver, processor and transceiver energy consumption should be optimized. This is especially important with modern mobile processors, which can dynamically adjust their power consumption by changing processor's frequency-voltage operating point. As mentioned in the introduction, modern wireless transceivers feature power management strategies that, combined with a complexity-scalable decoder, are attractive for thin-client power optimization.

In practice, thin clients dynamically trade off quality of screen updates and power usage. A thin client machine is able to detect available battery energy and adjust the processor's operating frequency accordingly. Unfortunately, frequency variations affect the decoding rate $D(\alpha)$. Thus, if the processor frequency is decreased, then the system should adapt by decreasing the compression ratio, assuming that increased compression quality would increase decoding speed. As before, the result is lower-quality screen updates.

This effect is modelled by expressing the average total energy consumed by single screen rectangle as

$$E_{total} = \frac{k_c}{\mu D(\alpha)} + \frac{k_d \alpha}{\mu B}, \quad (7)$$

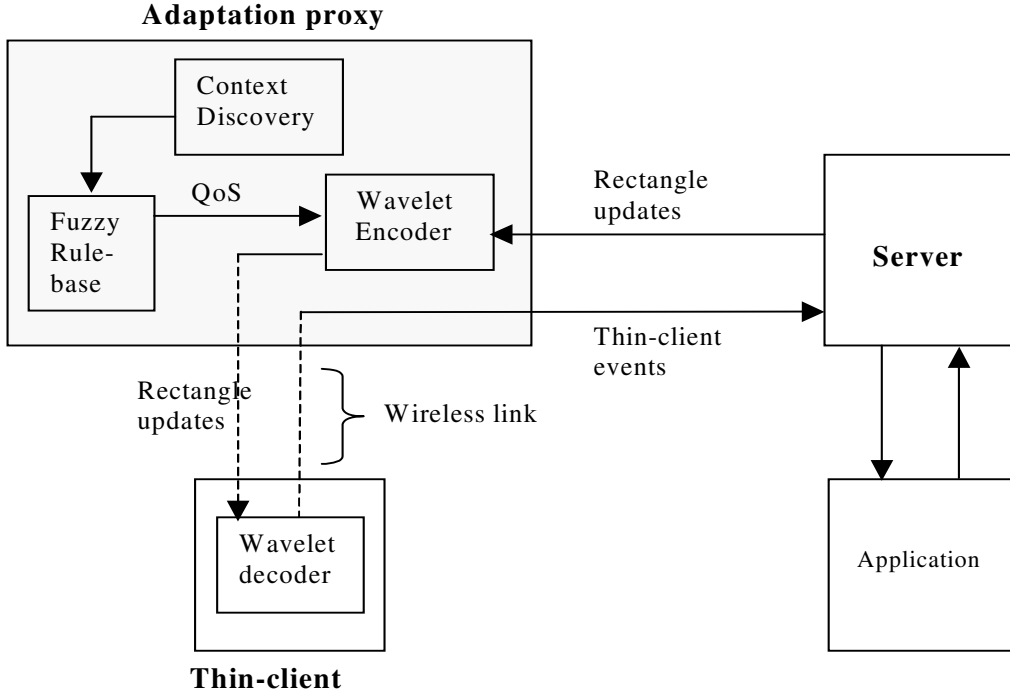


Fig. 2. Thin-client adaptation framework.

where k_c and k_d respectively denote energy cost per unit time for the processor, and for the transceiver operating in receiving mode.

IV. THE PROXY-BASED ADAPTATION FRAMEWORK

In this study, our objective is to maximize the quality of screen updates for a mobile thin client that can be supported by a wireless link in a way that is transparent to the user. In other words, we want to achieve optimal use of available wireless resources at a given time and location without user intervention. One possible solution minimizes the average latency observed by the user. Other possible solutions include optimisation of power consumption and monetary cost.

The objective of the adaptation framework shown in Fig. 2 is to minimize total system latency by controlling the compression ratio α of the wavelet-based encoder at the proxy. This is achieved by making the virtual bandwidth follow a target value, denoted by QD_0 , which is based on a QoS requirement. As before, this is a trade-off between total latency and quality of screen updates.

In terms of control theory, a control action takes place in response to dynamic changes in wireless link bandwidth and thin-client processing power. These variations cause the operating point to change, thereby adjusting the compression ratio. An error signal, which is the difference between the current and specified values of the virtual bandwidth, drives a fuzzy controller that outputs a new value for α . This is illustrated as follows, where Q denotes display quality:

$$Error = B_{virtual} - QD_0 \quad (8)$$

As shown in Fig. 2, the server sends screen update rectangles to the thin client through an adaptation proxy. The system follows the client-pull protocol, where the server sends an available screen update only after the client explicitly requests the update. The proxy encodes the updated screen rectangles using wavelet-based coding, then sends the encoded rectangles to the thin client over a wireless link.

In practice, wavelet-based image coding can exhibit superior rate-distortion performance and scalability, when compared to the lossy JPEG standard [Huang97]. A highly scalable rate control allows the thin-client system to offer graceful degradation of QoS, to support dynamic tradeoff of different performance parameters. In particular, wavelet-based coding enables trade-offs between reconstructed image quality, encoded image size, and the decoder's computational complexity.

Additionally, Figure 3 shows the context discovery module (CDM) as part of the adaptation proxy. The CDM discovers the context in which the thin client and proxy operate, where the context can include attributes of the wireless link and thin client. Device attributes include but are not limited to processing power, memory and display size, color depth, and battery power, while link attributes include bandwidth, latency, and error rate. CDM feeds context information to the fuzzy inference engine, which processes these data then decides how to control the thin-client system to meet user-specified QoS constraints.

Our implementation of this framework targets the case where the thin client presents a continuously updated active media object. Examples include an animated JPEG image file on the Web, Flash animation, or a Java applet.

TABLE I
FUZZY RULE-BASE FOR THE ADAPTATION MECHANISM

		$B_{virtual}$		
		<i>Neg_Small</i>	<i>Near_Zero</i>	<i>Pos_Small</i>
$B_{virtual}$	<i>Neg_Med</i>	Pos_Med	Pos_Med	Pos_Small
	<i>Neg_Small</i>	Pos_Small	Pos_Small	Near_Zero
	<i>Near_Zero</i>	Pos_Small	Near_Zero	Neg_Small
	<i>Pos_Small</i>	Near_Zero	Neg_Small	Neg_Small
	<i>Pos_Med</i>	Neg_Small	Neg_Med	Neg_Med

An 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 by measuring 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, per Equation (4).

A. Fuzzy Rule-based Controller

Fuzzy logic uses imprecise empirical or expert knowledge to describe dynamic systems, instead of more analytically precise formalisms such as differential equations. An important application of fuzzy logic is the control of complex, nonlinear systems. The reason for using a fuzzy controller in our system is to avoid the problem of directly measuring available wireless bandwidth. Also, the fuzzy controller supports high adaptability with minimum overhead.

Let us examine the fuzzy rule base used to control the latency of our wireless thin-client system. For example, the fuzzy rule that corresponds to the cell located at the first row and first column of Table I 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 contribute to the decision process depend on input values fed to the fuzzy control engine. Fuzzy *if-then* rules, such as shown above, produce results that overlap to yield a fuzzy output set. To get a crisp value that represents the output more precisely, a *defuzzification* process is applied [Yager93]. The resulting crisp value is input recursively to the system, to further modify its behavior.

V. EXPERIMENTAL RESULTS

A. Experimental Set-up

Fig. 3 shows the basic experimental test bed, which we used to test and evaluate our wireless adaptation framework. The VNC server runs on Red Hat Linux version 7 operating system on a Dell workstation with a Pentium 3, 450 MHz processor

having 256 MB RAM memory. The adaptation proxy runs on an identical machine, and both machines are connected to each other through a 100 Mbps LAN. Also, a Cisco Wireless LAN access point is connected to the local area network and support wireless communication speeds up to 11 Mbps.

The thin client device is an HP IPAQ hx4705 PDA that has an XScale processor running at 624 MHz. The IPAQ has 64 MB RAM memory and a 64k-color display of size 480 x 640 pixels. It has a built-in Wireless LAN card that supports throughput up to 11 Mbps with Bluetooth technology. The thin client runs on the IPAQ PDA as a Java application, and communicates with the adaptation proxy using the Wireless LAN access point.

Emulation of bandwidth variability is achieved by setting the link bandwidth between the thin client and the proxy using a CBQ-based traffic control script. This Linux script is flexible and effective, allowing the user-specified link bandwidth to range from 10 kbps to 10 Mbps, which is not feasible with a stock wireless access point.

Variable processing power constraints enable realistic performance test and evaluation for the thin client. A commercial utility, XCPUScalar 2004, sets the XScale processor frequency to 104, 208, 312, 416, 520 or MHz.

As an example of experimental results, Fig. 3 shows decoding time for the GWIC wavelet-based decoder as a function of bit rate in bits per pixel (bpp). These thin-client performance measurements were made on a Dell Pentium 4, 1.8 GHz with 512 MB RAM, screen size of 800x600 pixels and color depth of 24 bit/pixel. Decoding time information supports characterization of the decoding rate function's behavior for the wavelet-based decoder. Fig. 3 shows that wavelet-based coding is computationally expensive, and that the GWIC decoder has limited complexity scalability.

Inferences drawn from Fig. 3 supported design of the adaptation framework. For example, we inferred that decoding rate mainly depends on the wavelet decoder's complexity. We thus assumed that decoding time is a linear function of bit rate over the range of compression ratios from 1:1 to 150:1.

Additionally, we estimated the thin client's decoding rate D_0 by computing virtual bandwidth from measured total system latency. This was done with $\alpha \approx 0$, to effectively eliminate the effect of transmission latency. This capability was implemented by sending the first two screen updates with very small compression ratio (e.g. $\alpha = 1/126$). In this case, the decoding rate of $D_0 \approx 1/(\mu T_{total})$ determines the fuzzy controller's target virtual bandwidth $Q \cdot D_0$.

Fig. 4 illustrates the performance of the adaptation mechanism for two different machines. The square symbols represent a Dell Pentium 4, 1.8 GHz with 512 MB RAM while triangles represent a Dell Pentium 3, 450 MHz with 256 MB RAM. The screen rectangle size is 800x600 pixels and color depth is 24 bpp (true color). Fig. 4 demonstrates two adaptation behaviors of interest to this study. Firstly, it shows how our system adapts to changes in link bandwidth by

controlling compression, to maintain a prespecified target value for total latency.

As mentioned previously, a sudden decrease in the link bandwidth causes the fuzzy controller to output data at a higher compression level. For example, the target latency for the Pentium 4 machine is 1.7 seconds, and for the Pentium 3 machine is 3.36 seconds. Therefore, the fuzzy engine increases the compression level for the Pentium 3, to adapt to its increased latency. Secondly, Figure 4 shows how our system adapts and responds to different thin-client processing speeds. For the fast machine (Pentium 4), the fuzzy controller must increase compression, thereby reducing transmission latency, in order to meet the thin client's fast decoding rate. We thus circumvent transmission latency as a performance bottleneck.

An important advantage of our adaptation mechanism is that we avoid measuring the actual link bandwidth. Bandwidth measurement is costly and difficult to implement, and introduces unacceptable overhead. Instead, we rely on total system latency to estimate virtual bandwidth.

Link bandwidth variation is simulated via CBQ-based traffic control [Kuz03] available on the Linux operating system. We first set the link bandwidth between the thin client and the proxy to a specific value, then observe the compression ratio output by the fuzzy controller at the steady-state condition.

Fig. 4 illustrates adaptation using bit rate notation to express the amount of compression applied to screen update rectangles, where bit rate = $24 \cdot \alpha$. This figure portrays relationships similar to Figure 3, but also depicts a linear relationship between bit rate and link bandwidth.

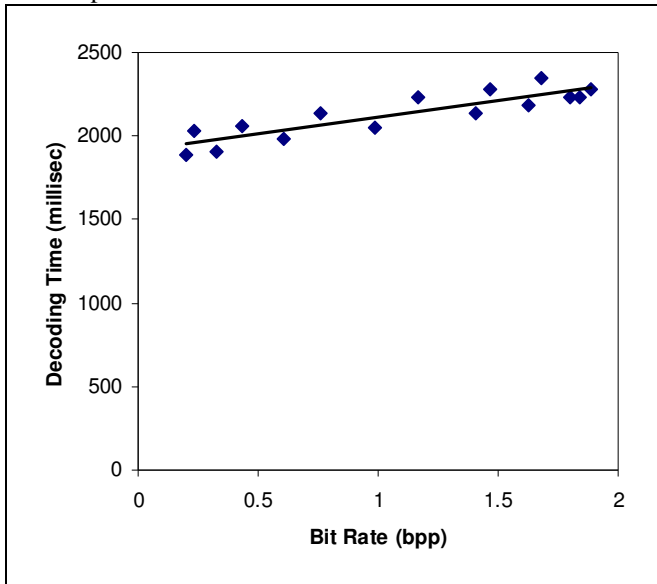


Fig. 3. GWIC decoder's decoding time.

B. Fuzzy Controller Tuning

Tuning the fuzzy controller optimizes controller performance by adjusting fuzzy membership function parameters for each fuzzy variable. This is achieved by designing the shape and the number of fuzzy sets for each fuzzy variable. The tuned membership functions are shown in Fig. 5. To better understand how the controller's output gain

factor K_α significantly affects compression control performance, we experimentally determined that higher values of K_α result in better latency control, but lead to more controller output fluctuations, per Fig. 5.

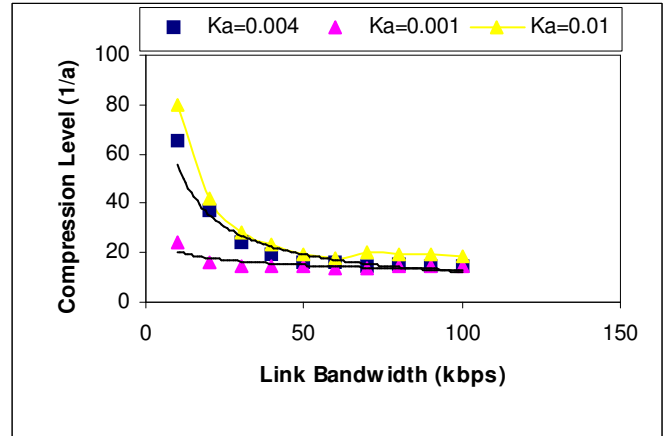
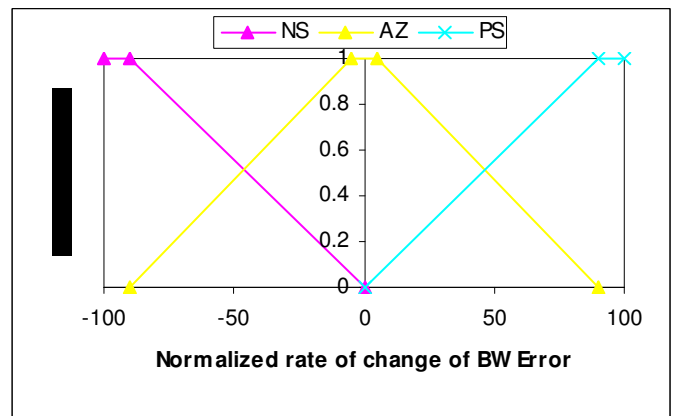
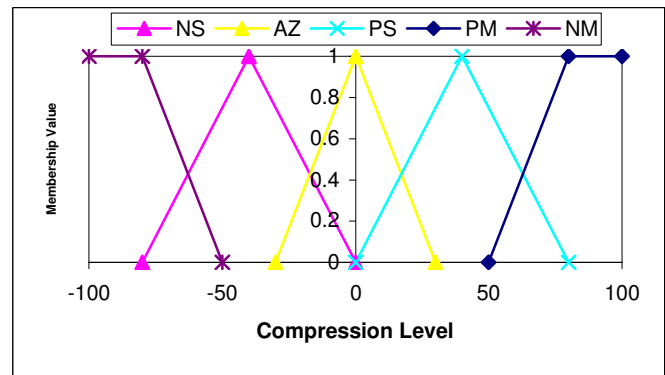


Fig. 4. The effect of output gain factor.



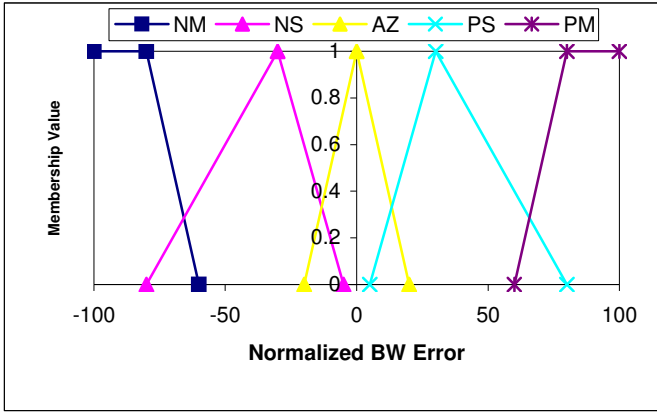


Fig. 5. Tuned membership functions for the fuzzy variables.

C. Fuzzy Fluctuation Effect

The fuzzy controller output can be affected by sudden changes in available link bandwidth or battery level, which can cause the thin client's processor frequency to change. For example, Fig. 6 shows the response of the controller to an abrupt decrease in bandwidth from 100 kbit/s to 30 kbit/s.

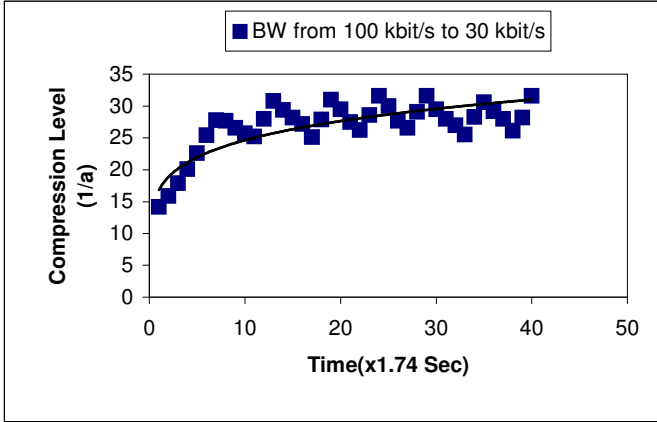


Fig. 6. Fuzzy controller's response to bandwidth decrease.

D. Fuzzy Rule Base Reduction

We evaluated the effect of tuning the fuzzy rule base on the adaptive performance of the wireless thin-client system. This involves finding a set of fuzzy rules that satisfy acceptable performance criteria, then adjusting the control surface. Fig. 7 shows that, although the controller with 15 rules has lower total latency, the controller with seven rules has similar performance. The ability to maintain controller behavior while reducing the number of rules is desirable when processing power is limited. In particular, having fewer rules reduces processing delays at the adaptation proxy. Table II lists the fuzzy inference rule base for the controller with seven rules.

TABLE II
REDUCED FUZZY RULE BASE

$B_{virtual} \backslash \hat{B}_{virtual}$	<i>Neg_Small</i>	<i>Near_Zero</i>	<i>Pos_Small</i>
<i>Neg_Med</i>		<i>Pos_Med</i>	
<i>Neg_Small</i>	<i>Pos_Small</i>		<i>Near_Zero</i>
<i>Near_Zero</i>		<i>Near_Zero</i>	
<i>Pos_Small</i>	<i>Near_Zero</i>		<i>Neg_Small</i>

<i>Pos_Med</i>		<i>Neg_Med</i>	
----------------	--	----------------	--

E. Adaptive Performance under Variable Processing Speed

We experimentally evaluated the performance of our adaptation framework under variable processor speed, on an HP IPAQ PDA. The nominal frequency for the IPAQ is 624 MHz. We used the Xscale CPU frequency Scalar utility software to vary the frequency to the desired value. Figure 8 illustrates the experimental setup.

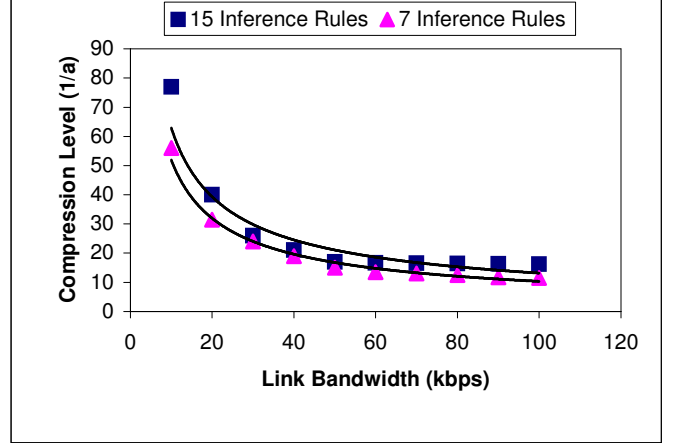


Fig. 7. Effect of reducing the number of fuzzy inference rules.

Fig. 9 depicts the result predicted by theory, namely, that the fuzzy controller responds to decreased CPU frequency by increasing the compression level, which leads to higher decoding rate. This control action maintains the target value for total latency by compensating for the effect of decreased processor frequency. When the quality factor increases, corresponding to lower total latency, the controller increases the compression ratio, to maintain the target value for total virtual bandwidth and, therefore, total system latency. Fig. 10 shows the improved compression level control for our highly tuned fuzzy controller.

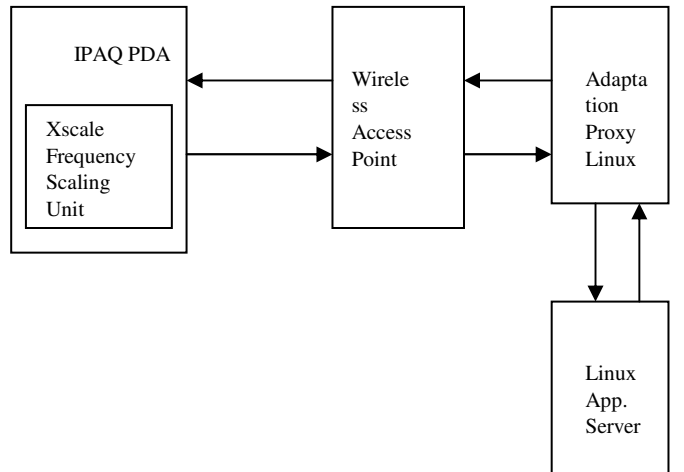


Fig. 8. Setup for performance evaluation under variable processor speed.

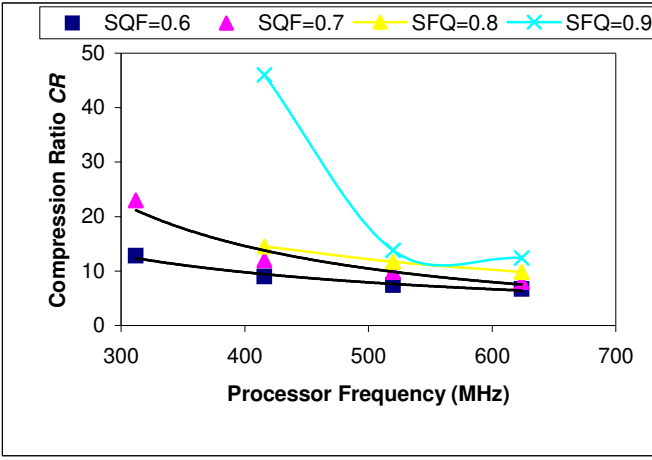


Fig. 9. Adaptive performance under variable CPU frequency.

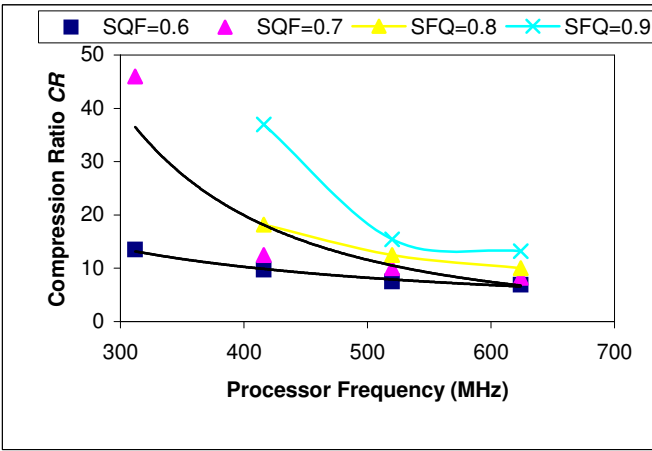


Fig. 10. Adaptive performance of tuned controller.

F. Tradeoffs between Screen Quality and Latency

Fig. 11 shows the effect of quality factor on the performance of the adaptation framework for thin clients. Generally, increasing the quality factor results in reduced system latency, but increased distortion of screen updates. Therefore, the fuzzy controller needs to increase compression ratio with quality factor. In practice, increasing the quality factor Q reduces the quality of screen updates, as shown by the region of the curve in Fig. 11 that has shifted upward.

Fig. 12 illustrates two observations about the experimental (linear) relationship between total system latency and quality factor. Firstly, when quality factor increases, adaptive control causes total latency to decrease. Secondly, processing and transmission latencies are handled identically by the adaptive system, because the control system is sensitive to the component latencies, versus the *source* of a given latency.

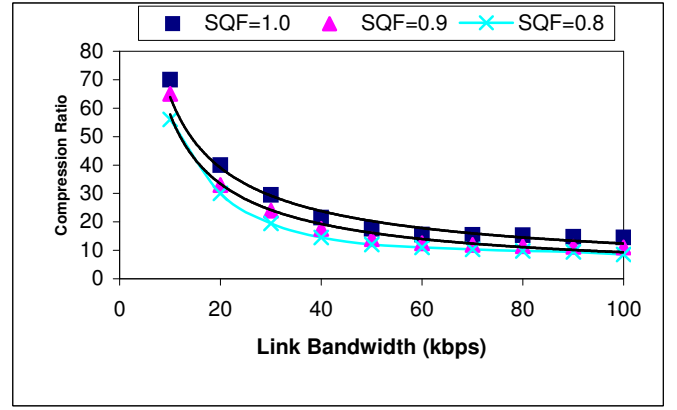


Fig. 11. Effect of quality factor on compression level control.

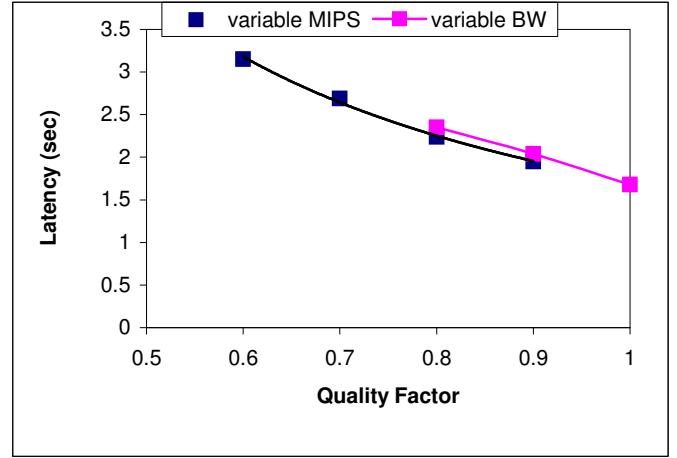


Fig. 12. Experimental relationship between latency and quality factor.

One of the essential characteristics of our approach is the tradeoff between total latency and screen rectangles quality (also expressed as distortion). Thus, it is also useful to consider image distortion metrics. Although perceptual quality metrics have been extensively reported, in this study, heuristics decide the value of Q . For instance, the ratio λ/μ represents the activity characteristic of each application, as well as the traffic rate generated by the application. In particular, we suggest assigning higher Q values for active applications, where

$$Q \propto k \cdot \frac{\lambda}{\mu}, \quad (10)$$

with k representing the distortion tolerance of a given application. Also, λ and $1/\mu$ should be estimated for different levels of screen update activity. As before, higher Q results in lower total latency, at the cost of increased distortion in screen updates sent to the remote thin client.

G. Size-Based Distortion Optimization

For small size RGB screen update rectangles, a high compression level may be undesirable. For example, wavelet encoding produces distortions that are more severe for small screen rectangles than for large rectangles. Therefore, we

recommend adjusting the compression level α in terms of screen rectangle size. This additional basis for optimization improves the perceived quality of client-side applications. One possible approach is given by

$$\alpha = \alpha \cdot \frac{A_{rect}}{A_{full}} + c, \quad (11)$$

which means that, within the paraxial viewing assumption, the compression level is proportional to the ratio between rectangle size A_{rect} and a device's full display size A_{full} , subject to an empirical constant c . The quasi-linear relationship shown in Fig. 13 suggests that the decoding rate does not change with screen update size. This makes adaptive control more robust by implementing spatial scaling in terms of area ratio.

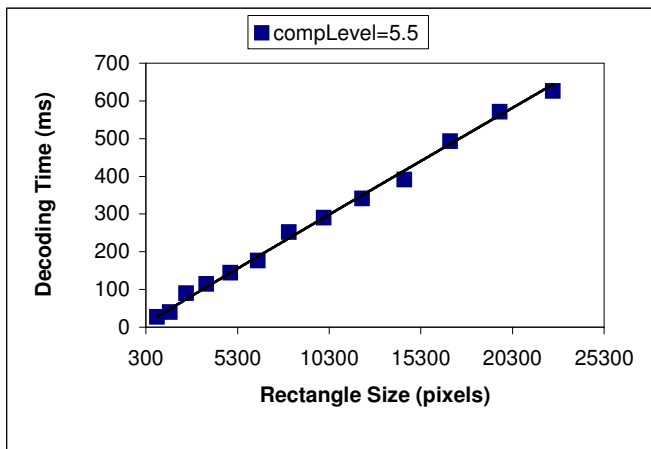


Fig. 13. Decoding time and rectangle size relationship.

VI. RELATED WORK

The Transend system at UC Berkeley [Fox98] is a proxy-based adaptation system that uses dynamic transcoding of multimedia web objects. Transend employs lossy compression of images to reduce bandwidth usage and to enable low-latency web surfing on handheld devices such as Palm devices. This demonstrates that on-demand adaptation using transformational (transcoding) proxies are practicable and economical.

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 researchers observe that their system potentially realizes best performance when utilizing a network connection monitor. Fox et al. suggest an automatic adaptation mode, where a network monitor can discover effective bandwidth and roundtrip latency. However, [Fox98] does not present performance results for the automatic adaptation mode.

By comparison, our system enables user-transparent adaptation of active media presentation, such as active Flash or a Java applet. We employ a virtual bandwidth concept, which represents the combined effect of wireless link bandwidth and the client's computational resources, expressed as processing speed or throughput. Based on this information, the fuzzy

controller chooses a compression level that achieves user-specified latency and quality of screen updates. Our system does not need to measure directly the available link bandwidth or the client processing speed.

The Network Computing Laboratory (NCL) of Columbia University [21], evaluated the performance of several thin-client systems and demonstrated experimentally that attempts at improving bandwidth, such as screen updates compression, may degrade overall system performance in wide area networks (WANs). This effect is due to the computational overhead required for decoding screen updates at a resource-limited thin client. The NCL group reported quantitative measurements that illustrate the impact of WAN latency on thin-client computing performance, demonstrating the feasibility of thin-client computing in a WAN (Internet 2) environment. Their experimental results suggest that optimizing for network latency is more important than optimizing the bandwidth usage of a thin-client system in a WAN environment.

Additionally, the NCL group demonstrated experimentally [20] that thin-client systems could provide acceptable performance for web and multimedia applications in local area network (LAN) environments. Performance evaluation shows that an eager server-push update policy, like the one adopted by X protocol, results in better overall performance than lazy update policy (such as that adopted by AT&T's VNC system for multimedia video applications).

In the lazy update model, a server saves bandwidth by discarding intermediate display updates. Thus, optimizing for bandwidth efficiency degrades the performance of multimedia applications in LAN environment. This implies the need for adaptive usage of available bandwidth, to balance the computational overhead of decoding against possible bandwidth savings. This indicates that client processing time is more important in high-bandwidth environments, while bandwidth efficiency is more important in low-bandwidth situations.

The NCL group evaluated the performance of thin-client web browsing in a wireless LAN environment [22], investigating web-browsing performance of a thin-client computing model under high packet loss rates. Experimental evaluation shows that wireless thin-client based web browsing under low network quality conditions (i.e. high packet loss rate) has faster response time (e.g., less web page download times) than a local web browser running on fat client.

Packet loss rate increases as the wireless network quality deteriorates. The error correction mechanism, which is handled by a TCP protocol in most thin-client systems, has great impact on web browsing performance. Recall that a thin client maintains only one connection to the server, while a local web browser on a fat client often uses many TCP connections. Setting up and maintaining such TCP connections in the presence of packet loss errors introduces excessive overhead and latency compared to thin-client browsing. These experimental results show that thin clients perform better in

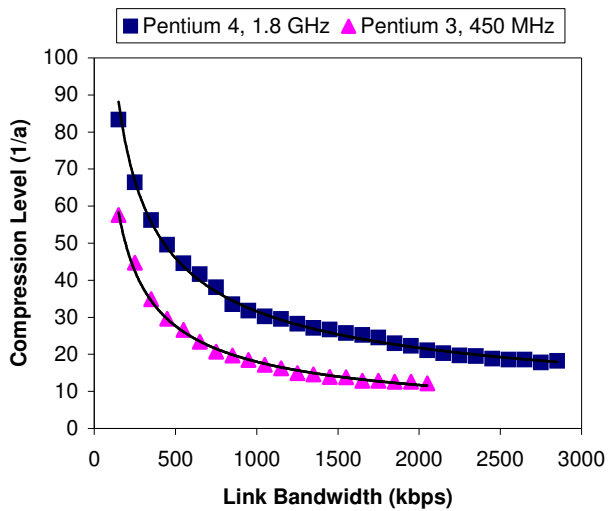


Fig. 14. Compression level control.

wireless web browsing compared to a local browser on wireless PDA clients, because thin-client browsing exhibits lower response time than a local browser.

VII. CONCLUSIONS

A proxy-based adaptation framework for wireless thin-client systems was presented, which dynamically adapts the performance of a wireless thin client using dynamic context discovery. This context information is input to a fuzzy rule-based inference engine, to optimize usage of wireless resources by tradeoffs among different quality of service parameters offered to end users. Our approach uses highly scalable wavelet-based image coding to support high spatial and temporal scalability of quality of service, to support graceful degradation. This framework shields the user from the ill effects of abrupt variations in structure or functionality of a wireless network and its mobile device resources.

ACKNOWLEDGMENT

The authors thank the developers of the following open-source projects which we used in the implementation and testing of our proposed adaptation framework; Virtual Network Computing system of AT&T Labs [Att03], GWIC-GNU Wavelet Image Codec [Leh98], and RFB Proxy[Wau02].

REFERENCES

- [Att03] AT&T Laboratories Cambridge, Virtual Network Computing, <http://www.uk.research.att.com/vnc>, 2003.
- [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,

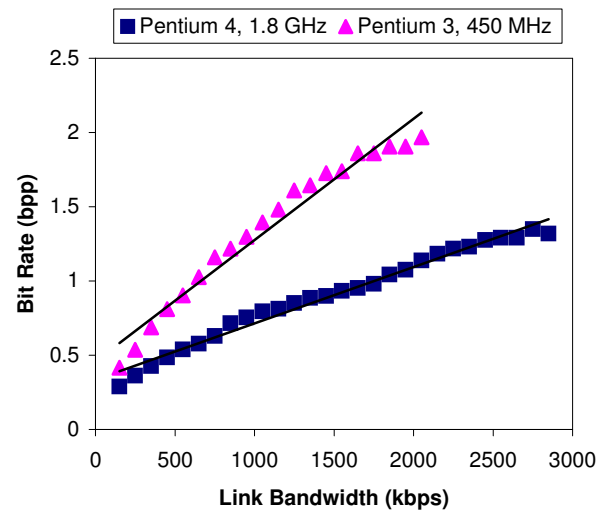


Fig. 15. Bit Rate control.

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.

[Huang97] Huang, J., Wang, Y., and Wong, E.K. Check Image Compression: A Comparison of JPEG, Wavelet and Layered Coding Methods, *IEEE International Conference on Image Processing*, 1997.

[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.

[Kuz03] Kuznetsov, A., CBQ.init Traffic Management Script, <https://sourceforge.net/projects/cbqinit>, 2003.

[Leh98] Lehtinen, J., GWIC- GNU Wavelet Image Codec, <http://jole.fi/research/gwic>, 1998.

[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.

[Wau02] Waugh, T., RFB proxy, <http://cyberelk.net/tim/rfbproxy>, 2002.

[Yager93] Yager, R.R. and F. Dimitar. On the Issue of Defuzzification and Selection based on a Fuzzy Set. *Fuzzy Sets and Systems* 55:255-271, 1993.