기대비용기반 캐쉬교체 알고리즘

이정준*°, 황규영**
한국과학기술원 전산학과, 첨단정보기술 연구센터

Expected-Cost-based Cache Replacement Algorithm

Jeong-Joon Lee and Kyu-Young Whang***
Department of Computer Science and
Advanced Information Technology Research Center(AITrc)
Korea Advanced Institute of Science and Technology(KAIST)

요 약

웹 데이타는 기존의 페이지를 기반으로 한 교체 알고리즘이 고려하지 않은 다양한 데이타 아이템의 크기, 네트워크 밴드위쓰 등으로 인한 다양한 참조 비용과 데이타의 만기시간(expiration time)을 갖는다. 그러나, 기존의 연구에서는 만기시간이 미치는 영향에 대한 연구가 초보적인 수준이다. 본 논문에서는 만기시간이 참조비용에 미치는 영향을 반영한 기대비용기반 캐쉬교체 알고리즘을 제안한다. 제안한 알고리즘은 만기시간내에 참조되어 캐쉬효과를 얻을 확률을 이용하여 참조비용의 기대값을 구하고, 이 값을 비교하여 교체대상을 선정한다. 제안한 알고리즘은 데이타의 크기, 참조비용 뿐만 아니라 만기시간의 영향을 확률적으로 정확히 반영하므로, 기존의 교체 알고리즘보다 우수한 성능을 보인다.

1. 서론

최근 들어, 웹의 폭발적인 사용 증가에 따라, 네트워크 지연(network delay)으로 인한 성능 저하를 최소화하기 위한 방법으로 웹 캐쉬 기술에 관한 연구가 활발하게 이루어지고 있다[1, 11, 12]. 웹 데이타를 위한 캐쉬는 크게 프락시 서버(proxy server)의 캐쉬와 웹 서버(web server)의 캐쉬, 클라이언트의 캐쉬로 나누어 진다. 특히, 클라이언트와 가까운 서버에 데이타를 캐쉬하는 프락시 서버의 캐쉬는 서비스 시간을 줄이는 것은 물론, 네트워크 밴드위쓰 소모를 줄이는 장점이 있다[11].

효율적인 캐쉬 관리의 핵심은 캐쉬가 가득 찼을 때, 어느 데이타 아이템을 교체할 것인가를 결정하는 교체 알고리즘이다. 기존에 개발된 페이지 기반 교체 알고리즘은 동일한 크기의 페이지를 참조(reference)하는 경우만을 고려하므로, 데이타 아이템의 크기와 참조 비용을 각각 동일하게 간주하여 각 페이지의 참조율만을 교체 알고리즘에 사용한다[4, 7]. 그러나, 웹 데이타 아이템은 크기가 다르고, 동일한 크기의 데이타 아이템이더라도, 네트워크 상에서의 위치와 밴드위쓰에 따라 전송속도가 다르므로, 참조 비용이 서로 다르다[1, 11, 12]. 또한, 캐쉬된 웹 데이타에는 유효성(validity)을 확인하기 위해 만기시간(expiration time)이 부여되고 있다[5, 7, 6]. 만기시간이란, 서버의 원본 데이타가 갱신되어 캐쉬된 데이타와 원본 데이타가 달라져 캐쉬된 데이타가 유효하지 않게되는 시점을 의미한다. 만기시간이 주어지는 경우에는 캐쉬된 데이타를 참조할 확률이 아닌, 캐쉬된 데이타 아이템을 만기시간 이전에 참조할 확률을 이용해야 캐쉬 효과를 얻을 정확한 확률이 된다. 따라서, 웹 데이타를 캐쉬하기 위해서는 데이타 아이템의 크기, 참조율, 참조 비용, 만기시간을 모두 고려한 교체 알고리즘의 개발이 요구된다.

본 논문에서는 만기시간이 캐쉬에 미치는 영향을 이론적으로 반영한 참조율을 유도하여 데이타 아이템의 크기, 참조 비용을 모두 캐쉬 효과에 반영한 비용기반 캐쉬 교체 알고리즘을 제안한다. 제안된 알고리즘은 웹 데이타와 같이 만기시간이 있는 데이타를 캐쉬하는 프락시 서버, 웹 서버에 사용될 수 있을 것이다.

본 논문의 구성은 다음과 같다. 제 2. 장에서는 교체 알고리즘의 비용과 최적 알고리즘 대해서 설명하고, 웹 데이타를 위해 제안된 기존의 교체 알고리즘에 대해서 살펴본다. 제 3. 장에서는 제안하려는 알고리즘에 대해서 설명하고, 알고리즘의 유효성을 직관적으로 살펴보고, 제 4. 장에서 결론을 맺는다.

*본 연구는 첨단정보기술 연구센터를 통하여 과학재단의 지원을 받았음

2. 관련연구

본 장에서는 교체 알고리즘에 관한 연구 초기부터 사용되던 교체 알고리즘의 비용에 대한 정의와 최적 알고리즘의 정의를 설명하고, 기존 교체 알고리즘의 기본 개념에 대해서 살펴본다.

2.1 알고리즘의 비용

본 절에서는 교체 알고리즘 비용에 관해서 설명한다. 데이타 아이템들의 집합을 $N = \{1, \ldots, n\}$ 이라 할 때, 데이타 아이템들의 읽기 요청들의 순열(sequence)을 참조스트링(reference string)이라 하고, $\omega = r_1 r_2 \ldots r_t \ldots$, $r_t \in N$의 같이 표현한다[4]. 여기에서 $r_t = x$는 t번째 요청에서 데이타 아이템 x를 참조함을 의미한다. t는 순열상의 위치를 나타내지만 이산 시간(discrete time) 축에서의 하나의 상대적 시점으로 해석할 수 있다. 따라서, 연속 시간상의 절대적 시간 위치와는 관계 없이 참조된 순번을 의미하므로, 절대 시간으로 주어지는 만기 시간의 영향을 반영하기 어려운 단점이 있다.

[4]에서는 캐쉬 크기 m에서, 참조 스트링 $\omega = r_1 r_2 \ldots r_T$을 처리하는 데 소모되는 비용을 $C(A, m, \omega)$으로 정의한다. 또한, 캐쉬되지 않은 데이타 아이템 $i \in N$를 참조하는데 소모되는 비용을 c_i로 표현하고, 그 비용은 알려져 있다고 가정한다. 그러면, $C(A, m, \omega)$를 다음과 같이 정의할 수 있다.

정의 1

$$C(A, m, \omega) = \sum_{t=1}^{T} ref\_cost(r_t).$$

$$where\ ref\_cost(r_t) = \begin{cases} 0 & if\ r_t \in Cache \\ c_{r_t} & otherwise \end{cases}$$

알고리즘들의 성능 비교를 위해서는 가능한 모든 ω에 대한 $C(A, m, \omega)$의 기대값을 비교해야 한다. 참조 스트링의 확률 분포, $p(\omega)$ 가 정의되어 있다면, $C(A, m, \omega)$의 기대값 $C(A, m)$을 다음과 같이 정의한다.

정의 2

$$C'(A,m) = \sum_{\omega \in N^+} p(\omega)C(A.m,\omega),$$
$$where\ N^+ = postive\ closure\ of\ N$$
$$p(\omega) = \ the\ probability\ of\ \omega$$

2.2 최적 알고리즘

본 절에서는 최적 알고리즘을 정의를 설명하고, 독립참조 모델 가정하에 A_0에 대해서 설명하고, A_0 알고리즘의 원리를 이용한 LRU-K 알고리즘의 개념에 대해 설명한다. 최적 알고리즘이란, 주어진 $p(\omega)$에 대해서 $C(A,m)$이 최소인 알고리즘 A를 말한다. 그리고, 독립 참조 모델이란 참조스트링 $r_1 r_2 \cdots r_t$ 가 독립적인 확률변수(random variable)들의 순열이고, 이들은 모두 정상성(stationary property)[9]을 갖는 확률 분포 $p[r_t = i] = \beta_i$인 확률분포를 갖는다는 가정이다[4]

A_0 알고리즘은 데이타 아이템의 크기와 참조 비용이 동일할 때, 캐쉬에서 β_i 값이 가장 작은 데이타 아이템을 교체하는 알고리즘이다[2, 4]. 독립 참조 모델의 정상성으로 인해, 다음 참조 시점, $t+1$에서의 참조 확률도 t에서의 참조 확률과 동일한 점을 이용하여, A_0 알고리즘이 최적 알고리즘임이 [4]에서 증명 되었다. 그러나, 현실적으로는 β_i를 알 수 없기 때문에, LRU-K 알고리즘에서는 각 데이터 아이템들에 대한 최근 K개의 참조 시점을 이용하여, β_i를 추정한 값인 λ_i을 이용하여 교체 대상을 선정한다[8].

2.3 웹 캐쉬 교체 알고리즘

본 절에서는 웹 데이타의 특성을 반영한 웹 캐쉬 교체 알고리즘을 소개한다. 웹 데이타는 기존의 페이지 기반 교체 알고리즘과는 달리, 데이타 아이템마다 참조비용과 크기가 다르고, 만기시간이 주어진다. 이런 요소를 교체 전략에 반영하여 개발된 알고리즘들을 크게 키기반(key-based) 교체전략과 함수 기반 교체전략으로 분류할 수 있다[1, 12].

2.3.1 키기반 교체 전략

키기반 교체전략은 키의 우선순위를 정하고, 우선 순위대로 키값을 비교하여 교체 대상을 선정하는 전략이다. 다시 설명하면, 먼저 첫번째 키값에 따라 캐쉬된 데이타 아이템을 정렬한다. 만일, 동일한 값의 데이타 아이템이 여러개 있어서 교체대상을 선정할 수 없으면, 두번째, 세번째 키 값 순서대로 교체 대상의 우선 순위를 정하는 방법이다[1] SIZE 알고리즘은 데이타 아이템의 크기,마지막 참조시간을 순서대로 키로 사용하는 알고리즘이고[1], Hyper-G 알고리즘은 참조된 회수, 마지막 참조시간, 데이타 아이템의 크기를 순서대로 키로 사용하는 알고리즘이다[3]. 키기반 교체전략은 우선되는 키값이 동일하지 않으면, 그 이후의 키값은 교체 전략에 전혀 반영되지 않는 단점을 갖고 있다

2.3.2 함수기반 교체 전략

함수 기반 교체 전략은 마지막 참조시간, 캐쉬 진입시간, 참조율, 크기 등 여러가지 값들을 파라미터로 한 함수를 정의하고, 그 값을 이용하여 교체 대상을 선정하는 전략이다. SLRU[1]는 참조율/크기가 작은 데이타 아이템을 교체하려는 전략으로서, 데이타 아이템의 참조비용과 만기시간을 반영하지 않아 웹 데이타의 특성을 충분히 반영하지 못했다. [1]에서 제시한, Aggawal의 전략은 비용/크기, 참조율에 만기시간의 영향을 *refresh-overhead factor*라는 근사값을 이용하여 반영하려 하였다. 그러나, 이 알고리즘은 교체시점부터 만기시간까지의 시간적 거리를 고려하지 않았고, 이 산시간을 단위로 한 값과 연속시간을 단위로 한 값을 함께 함수의 파라미터로 사용하여 이산시간 일이기 같이도, 연속시간으로 표현된 기간은 큰 차이가 발생할 수 있어서 연속시간으로 주어지는 만기시간의 영향을 정확히 반영하지 못한다

3. 기대비용기반 교체알고리즘

본 장에서는 기대비용과 유효참조율을 정의하고, 데이타 아이템의 참조비용과 크기의 영향을 반영한 기대비용기반 교체 알고리즘의 전략을 소개한다. 마지막으로 알고리즘의 유효성을 살펴본다

3.1 기대비용

본 절에서는 기대비용에 관하여 설명한다 비용기반 교체 알고리즘의 시각에서 캐쉬를 하는 이유는 다음 참조시점에서, 캐쉬하지 않았을 경우 소모되는 참조비용을 소모하지 않기 위해서다. 다시 말하면, 다음번에 참조될 데이타 아이템을 미리 예측하여 캐쉬함으로써 다음번 참조 때, 소모하지 않아도 되는 비용을 크게 하려는 것이 캐쉬의 목적이다 따라서 교체 대상을 선정할 때, 캐쉬를 한다면, 다음 참조시점에서 소모하지 않아도 되는, 확률적 비용을 예측하는 것이 중요하다

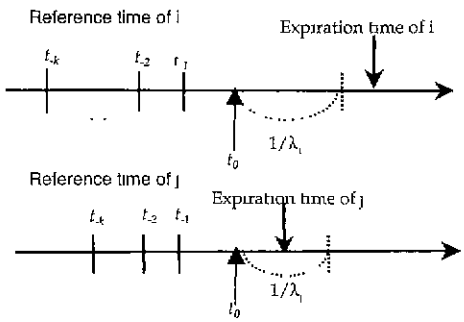
본 논문에서는 이 값을 데이타 아이템의 기대비용(*expected cost*)이라고 정의한다. 즉, 기대비용은 어떤 데이타 아이템을 캐쉬했을 경우, 다음번 참조시 캐쉬한 효과로 소모하지 않아도 되는 비용의 기대값이다 데이타 아이템 i의 참조율 β_i를 기존 논문과 표기 일관성을 지키기 위해서, λ_i로 표기하고, 캐쉬되지 않았을 경우에 소모되는 참조비용을을 c_i라하면, i의 기대비용 C_i는

$$C_i = c_i \times \lambda_i \quad (1)$$

이다 그러나, 식 (1)의 참조율 λ_i는 만기시간을 고려하지 않은 값으로, 데이타 아이템의 만기시간이 주어지면 캐쉬한 효과를 얻을 확률은 변하게 된다. 왜냐하면, λ_i는 교체시점 이후의 처음으로 참조한 데이타 아이템의 i의 만기여부에 관계없는 참조율이므로, 캐쉬효과를 얻을 확률과 오차가 발생한다 그림 1에서 i, j의 참조율을 각각 λ_i, λ_j라고 할 때, $\lambda_i > \lambda_j$이면, 기존의 만기시간을 고려하지 않은 교체방법은 다음 참조시점에서 캐쉬효과를 얻을 확률이 작은 j를 교체하는 전략을 사용한다 그러나, 그림과 같이 i의 만기시간이 얼마 남지 않은 상태라면, i가 캐쉬효과를 얻을 확률이 작으므로, i를 교체하는 전략이 효율적이다. 왜냐하면, 데이타 아이템 i를 캐쉬한 효과를 얻을 확률은 교체시점부터 i의 만기시간이내에 첫번째 참조가 발생하고, 그 참조 데이타 아이템이 i일 확률이므로. 기존의 교체 알고리즘과는 달리 i를 교체하여 얻는 캐쉬효과가 j를 교체하여 얻는 캐쉬효과보다 짧은 만기시간까지의 거리로 인하여 확률적으로 더 크기 때문이다 즉, $\lambda_i > \lambda_j$ 이지만, $\lambda'_i < \lambda'_j$ 이기 때문이다 따라서, 식 (1)의 λ_i는 만기시간 이전에 첫 참조가 일어날 확률로 바뀌어야 한다. 이 확률을 유효참조율이라고 정의하고, λ'_i로 표기하면, 기대비용은

$$C_i = c_i \times \lambda'_i \quad (2)$$

이 된다.



t_{-k}: 교체시점 이전의 마지막 k번째 참조시간
t_0 : 교체시점

그림 1 만기시간을 고려하지 않은 기존 교체 알고리즘의 문제점

λ_i에는 교체 시점 이후의 첫 참조 시점과 각 데이타 아이템의 만기시간까지의 거리가 반영되지 않았다. 즉, λ_i는 참조가 일어났다는 가정하에서 참조 시점을 고정하고, 다른 데이타 아이템과 참조 가능성을 비교하기 위한 값으로, 단위시간당 참조회수를 의미한다 따라서, 단기시간까지 남은 시간적 여유의 크기가 반영된 λ'_i을 구해야 한다

본 논문에서는 교체후 첫 참조시점을 임의변수(random variable)로 하여 데이타 아이템이 만기시간 이전에 참조될 확률인 유효참조율 λ'_i-캐쉬효과를 얻을 확률-을 수학적인 방법으로 유도한다 λ'_i을 이용한 기대비용기

반 교체 알고리즘은 교체시점에서 $c_i \times \lambda'_i$ 값이 적은 순서로 필요한 공간을 확보할 수 있는 최소한의 데이타 아이템들을 교체하므로, 교체시점 다음의 참조비용의 기대값을 최소함을 3.5에서 설명한다.

3.2 유효 참조율

본 절에서는 유효참조율을 수학적으로 유도하는 과정을 설명한다. λ'_i는 t_0이후의 첫 참조가 i이고, 첫 참조가 i의 만기시간 이전에 발생할 확률이므로, λ'_i은 다음과 같다.

$$\begin{aligned}\lambda'_i &= P[r_{T_1} = i, T_1 < t_i^e] \\ &\text{,where } t_o = \text{교체시점}, \\ &T_1 = t_0\text{이후의 첫 참조시점을 나타내는 변수} \\ &t_i^e = i\text{의 만기시간}\end{aligned} \tag{3}$$

T_1의 분포를 유도하기 위해, 독립참조 모델(independent reference model)같이 참조 스트링 $\omega = r_1 r_2 \ldots r_n$이 서로 독립적인 확률 변수(random variable)의 순열이고, 이들은 모두 정상성(stationary property)를 갖는 참조율 분포 $\lambda_1, \lambda_2, \ldots, \lambda_n$을 따른다는 가정을 한다. 단, λ_i는 참조회수가 아닌 연속 시간상에서의 참조율이다. 그러면, 데이타 아이템 i의 참조회수가 평균 $\lambda_i t$를 갖는 포아송 과정임을 유도할 수 있다. 그리고, 복합 포아송 과정[9]의 정리를 이용하면, 모든 데이타에 대한 참조회수는 평균 참조율이 $\lambda(=\sum_i \lambda_i)$인 포아송 과정이 되고, 도착간격(interarrival time)에 관한 정리를 이용하면, 평균 λt인 포아송 과정의 첫번째 도착시간의 확률분포는 평균 $1/\lambda$인 지수분포가 된다. 즉, $T_1 \sim exp(\lambda)$가 된다[9].

데이타 아이템 i의 첫 참조시점을 나타내는 확률변수를 $T_{1,i}$라 하면, 식 (3)는 다음과 같이 전개된다

$$\begin{aligned}\lambda'_i &= P[r_{T_1} = i, T_1 < t_i^e] \\ &= P[T_1 < t_i^e] P[r_{T_1} = i | T_1 < t_i^e] \\ &\quad (\because \text{조건부 확률의 정의}) \\ &= P[T_1 < t_i^e] P[T_1 = T_{1,i} | T_1 < t_i^e] \\ &\quad (\because r_{T_1} = i \Rightarrow T_1 = T_{1,i})\end{aligned} \tag{4}$$

$T_1 \sim exp(\lambda)$이므로, 식 (4)의 $P[T_1 < t_i^e]$은 지수 분포의 누적확률식을 이용하면, 다음과 같다

$$P[T_1 < t_i^e] = 1 - e^{-\lambda(t_i^e - t_0)} \tag{5}$$

식 (4)의 $P[T_1 = T_{1,i} | T_1 < t_i^e]$은 지수분포의 정상성에 따라, $T_1 < t_i^e$의 조건에 영향을 받지 않으므로, $P[T_1 = T_{1,i} | T_1 < t_i^e] = P[T_1 = T_{1,i}]$이고, $P[T_1 = T_{1,i}] = \lambda_i / \lambda$을 유도할 수 있다. 따라서, 다음 식을 얻을 수 있다.

$$\lambda'_i = (1 - e^{-\lambda(t_i^e - t_0)}) \frac{\lambda_i}{\lambda} \tag{6}$$

3.3 데이타 아이템의 참조 비용과 크기의 반영

본절에서는 각 데이타 아이템의 참조비용과 크기를 어떻게 기대비용에 반영하는가에 대해서 기술한다. 데이타 아이템의 참조비용은 각 응용에서 최소화하려는 요소로 정할 수 있다. 예를 들면, 프락시 서버와 같은 환경에서는 네트워크 지연으로 정의할 수 있고, 단일서버의 데이타베이스에서는 페이지의 참조 회수로 정의할 있을 것이다 본 논문에서는 데이타아이템의 참조비용은 주어진다고 가정한다.

캐쉬의 크기는 한정되어 있으므로, 캐쉬의 공간을 최대한 효율적으로 사용해야 한다 객체의 크기가 달라 캐쉬했을 경우 차지하는 공간이 다르다면, 캐쉬효과를 최대화하기 위해서는 캐쉬의 단위공간당 비용인 C_i/S_i값이 큰 데이타 아이템을 캐쉬해야 한다[1, 10, 10, 11]. 따라서, 식 (2)은 캐쉬의 단위공간당 참조비용의 기대값, C'_i은 다음과 같다.

$$C'_i = \frac{C_i}{S_i} = \frac{c_i}{S_i}(1 - e^{-\lambda(t_i^e - t_0)}) \frac{\lambda_i}{\lambda} \tag{7}$$

3.4 기대비용기반 교체 알고리즘의 전략

기대비용기반 교체 알고리즘은 교체시점에서 먼저 만기된 데이타를 캐쉬에서 제거한다 그 이후에도 참조하려는 데이타 아이템을 캐쉬할 공간이 부족하면, 단위공간당 기대비용 C'_i가 작은 순서대로 캐쉬에서 제거하여 빈 공간을 확보하는 전략을 사용한다.

3.5 기대비용기반 교체 알고리즘의 유효성

크기가 다른 데이타 아이템을 교체하는 알고리즘은 배낭 문제(knapsack problem)의 일종이고, 배낭 문제는 NP-hard임이 알려져 있다[1]. 따라서, 크기가 다른 데이타 아이템을 다루는 교체 알고리즘은 그리디 정책(greedy policy)과 같은 휴리스틱(heuristic)을 대부분 사용하게 된다. 즉, 교체 시점이후의 첫 참조에서 드는 비용이 가장 작도록 캐쉬를 유지하는 방법을 사용한다. 이 방법은 캐쉬교체 시점에 단위공간당 참조비용의 기대값이 가장 작은 데이타 아이템을 교체하는 것으로 달성할 수있다 제안한 방법은 교체시점에서 만기시간을 고려하여, 단위공간당 참조비용의 기대값이 가장 작은 데이타를 교체하므로 기존 방법보다, 우수한 성능을 보이게 된다.

4. 결론

본 논문에서는 만기시간의 영향을 반영한 캐쉬교체 알고리즘에 관하여 논의하였다 만기시간이 캐쉬효과에 미치는 영향을 수학적으로 반영한 유효 참조율을 정의하고, 이값을 이용한 기대비용기반 교체 알고리즘을 제시했다.

제안한 기대비용기반 교체 알고리즘은 만기시간이 주어지는 웹 데이타를 다루는 프락시 서버(proxy server)와 웹 서버의 캐쉬 교체 알고리즘으로 사용되어 사용되어 빠른 응답시간을 제공하고, 네트워크 부하를 감소시킬 것이다.

Reference

[1] Aggarwal, C , Wolf, J. L , and Yu, Phillip S., "Caching on the World Wide Web," *IEEE Transaction on Knowledge and Data Engineering*, Vol. 11, No 1, Feb. 1999

[2] Aho, A. V., et al , "Principles of Optimal Page Replacement," *Journal of ACM*, Vol 18, No. 1, pp. 80–83, Jan. 1971.

[3] Andrews, A., et al , "On Second Generation hypermedia Systems," *Proc. of ED-Media 95, World Conf. on Educational Multimedia and Hypermedia*, Jun. 1995.

[4] Coffman , Edward G. Jr., and Denning, Peter J., *Operating Systems Theory*,Prentice Hall, 1975.

[5] Fielding, R., Gettys, J., Mogul, J C., Frystyk, H., and Berners-Lee., T., "Hypertext Transfer Protocol – HTTP/1 1 RFC 2068," *http://nic ddn.mil/ftp/rfc/rfc2068.txt*, Jan 1997.

[6] Gwertzman, J. and Seltzer, M., "World-Wide Web Cache Consistency," *Proc. of Usenix Technical Conference*, Jan., 1996.

[7] Liu, G., and Maguire, G. Q., "A Survey of Caching and Prefetching Techniques in Distributed Systems." *Technical Report TRITA-IT R 94-40*, Oct. 1994.

[8] O'Neil, E. J., O'Neil, P. E., and Weikum, G., "The LRU-K Page Replacement Algorithm For Database," *Proc. of the 1993 ACM SIGMOD Conf. on Management of Data*. pp 297–306, May 1993.

[9] Ross, S. M., *Introduction to Probability Models*, Academic Press, 1993.

[10] Sinnwell, M., and Weikum, G., "A Cost-Model-Based Online Method for Distributed Caching," *Proc. of the 13th Int'l Conf. on Data Engineering*, pp. 532–541, 1997

[11] Scheuermann, P., Shim, J H., and Vingralek, R , "A Case for Delay-Conscious Caching of Web Documents," *Proc. of 6th Int'l WWW Conference*, Feb. 1997

[12] Williams, S., Abrams, M., Standridge, C. R , Abdulla, G., and Fox, E A., "Removal Policies in Network Caches for World-Wide Web Documents," *Proc of the ACM SIG COMM'96 Conference*, Aug., 1996.