변형된 퍼지 논리 기반의 DASH 적응 알고리즘

*김현준 **손예슬 ***김준태
건국대학교
*khkim38@konkuk.ac.kr

A Modified Fuzzy logic Based DASH Adaptation Algorithm

*Kim, Hyun-Jun **Son, Ye-Seul ***Kim, Jun-Tae
Konkuk University

요약

퍼지 논리를 기반으로 한 적응형 스트리밍 기법인 FDASH 적응 알고리즘은 빠르게 변하는 네트워크 상황에서 우수한 콘텐츠의 화질을 보장하면서 끊김 없는 서비스를 제공하는 특성을 보이지만 비디오의 화질이 자주 변하기 때문에 최고의 사용자 체감 품질 (QoE: Quality of Experience)을 제공하지 못 할 수도 있다.

본 논문에서는 제한된 버퍼 크기를 가지고 동일한 콘텐츠의 화질을 보장하면서도 비디오 화질의 변화 횟수를 줄여서 최적의 QoE를 제공할 수 있도록 하는 변환된 퍼지 논리 기반의 DASH 적응 알고리즘을 제안하고자 한다. 제안된 방식은 우선 퍼지 논리 제어부(FLC : Fuzzy Logic Controller)의 수정을 통하여 다음 세그먼트의 비트율에 대해 최적의 판단을 하도록 하였고, 세그먼트 비트율 필터링 모듈 (SBFM: Segment Bitrate Filtering Module)을 추가하여 비디오 화질의 변화 횟수가 최소화 될 수 있도록 하였으며, 스트리밍 서비스 시작 시 SBFM에 의해 일정시간 저화질의 비디오를 시청해야 하는 상황을 막기 위한 Start Mechanism을 추가하였고, 마지막으로 버퍼의 오버플로우를 방지하기 위해 Sleeping Mechanism을 추가하였다.

NS-3를 이용한 네트워크 모의실험 결과를 통해 제안된 방식이 FDASH 방식에 비하여 제한된 버퍼크기 상황 하에서도 오버플로우가 발생하지 않으며 점대점(Point to Point) 상황에서는 거의 동일 화질 성능을 보이면서도 비디오 화질 변화 횟수를 50% 이상 줄일 수 있음과 일반 Wifi환경에서는 오히려 17.8%정도 더 뛰어난 비디오 화질 성능을 보이면서 비디오 화질 변화 횟수 측면에서는 53.1%정도 줄일 수 있음을 보여준다.

1. 서론

DASH[1]는 가변적인 네트워크 상황과 서로 다른 단말기의 특성을 고려하여 끊김 없는 미디어 서비스를 제공하기 위한 적응형 스트리밍 표준이다. 또한 HTTP/TCP를 기반으로 한 스트리밍 프로토콜로, 방화벽에 차단 될 수 있다는 문제점을 해결 할 수 있고, 이미 존재하는 HTTP 서버를 사용 할 수 있어 비용 측면에서도 효율적이다. DASH에서는 미디어 콘텐츠를 여러 가지 비트율 버전으로 인코딩 한 뒤, 각각의 버전을 수 초에서 수 십초 단위의 세그먼트로 분할하여 HTTP 서버에 저장한다. DASH 서비스 사용자는 가변적인 네트워크 상황에 적응적으로 알맞은 버전의 세그먼트를 HTTP 서버에 요청을 해서 서비스를 받는다.

DASH를 이용해서 최적의 적응형 스트리밍을 제공하기 위해 지금까지 많은 연구가 진행이 되었다. 현재 버퍼의 상태와 사용가능한 대역폭의 예측 값과 실제 값을 사용하는 적응형 스트리밍 기법인 Agile and Smooth Video Adaptation Algorithm(SVAA)[2]가 소개 되어있고, 세그먼트 수신하는데 소비되는 시간과 그 세그먼트가 재생이 되는 시간의 차이를 이용해서 네트워크의 대역폭을 측정하는 Rate Adaptation for Adaptive HTTP Streaming(RAAHS)[3] 알고리즘이 소개 되어있다. 또한, 퍼지 논리를 이용한 적응형 스트리밍 기법인 FDASH[4]가 소개 되어있다. FDASH는 평균 비디오 화질, 미디어 재생의 끊김 현상에 대해서는 다른 알고리즘에 비해 좋은 특성을 보인다. 하지만 FDASH는 버퍼의 오버플로우에 대한 고려를 하지 않으며, 비디오 화질 변화가 많이 발생하는 문제점을 보인다. [5]에 정의되어 있는 내용에 의하면, QoE는 비디오 화질 변화 빈도, 버퍼 언더플로우/오버플로우 등의 요인에 영향을 받기 때문에 버퍼 오버플로우의 발생과 비디오 화질 변화가 빈번히 발생하는 것은 비디오 품질에 부정적인 영향을 미친다. 따라서 DASH 서비스 사용자에게 최적의 QoE를 제공하기 위해서 비디오 화질 변화 횟수는 적어야 하고, 버퍼 오버플로우는 발생하지 않아야 한다.

본 논문에서 소개할 mFDASH는 퍼지 논리를 기반으로 한 알고리즘으로, FDASH의 장점을 가져가면서, 단점은 보완한 알고리즘이다.

FLC[6]가 최적의 출력을 낼 수 있도록 수정하였고, FLC의 출력이 적합한 지를 판단하는 SBFM을 추가하여 비디오 화질 변화 횟수를 줄였다. 또한, SBFM을 에서 알고리즘이 동작할 때 사용 가능한 대역폭의 측정값이 사용되며, mFDASH에서는 FDASH에서 사용한 대역폭 측정 기법보다 더 정확하게 사용 가능한 대역폭을 예측하면서, 평활도는 적절히 유지하는 기법을 사용한다. 또한, 멀티미디어 스트리밍을 시작할 때 버퍼가 언더플로우가 나지 않게 하면서 최적의 QoE를 제공할 수 있게 Start Mechanism을 추가 하였다. 마지막으로 오버플로우가 발생하는 것을 방지하기 위한 Sleeping Mechanism도 추가 하였다.

NS-3를 이용한 네트워크 모의실험 결과를 통해 제안된 방식이 FDASH 방식에 비하여 제한된 버퍼크기 상황 하에서도 오버플로우가 발생하지 않으며 점대점 상황에서는 거의 동일 화질 성능을 보이면서도 비디오 화질 변화 횟수를 50% 이상 단축시킬 수 있음과 일반 Wifi 환경에서는 오히려 17.8%정도 더 뛰어난 비디오 화질 성능을 보이면서 비디오 화질 변화 횟수 측면에서는 53.1%정도 줄일 수 있음을 보여준다.

2. mFDASH 알고리즘

A. Fuzzy logic Controller

FLC는 퍼지 논리[7]를 이용한 제어 시스템이다. FLC는 일반적인 입력 값(Crisp Input)을 퍼지 규칙과 소속 함수들을 통하여 일반적인 출력 값(Crisp Output)을 만들어낸다. mFDASH에서 사용하는 FLC는 버퍼 점유율, 버퍼 점유율의 차이 두 가지 입력을 받고, 하나의 출력값 f를 가진다. f는 다음 세그먼트의 비트율을 현재 측정한 사용 가능한 대역폭보다 얼마나 증가/감소시킬 것인지를 나타낸다. 입력, 출력의 소속 함수들은 그림 1에 나타나 있다. 버퍼 점유율은 가장 최근에 요청한 k번째 세그먼트가 완전히 수신 되었을 때 버퍼가 가지고 있는 데이터를 초(sec)단위로 나타낸 값으로, $(t^{(e)})$라고 표기하고, 그 소속 함수는 그림 1의 (a)에 나타나 있다. 버퍼 점유율의 차이는 k번째 세그먼트를 완전히 수신 했을 때의 버퍼 점유율과 k-1번째 세그먼트를 완전히 수신했을 때 버퍼 점유율의 차이로, $\Delta q(t_k^{(e)}) = q(t_k^{(e)}) - q(t_{k-1}^{(e)})$라고 표기하고, 소속 함수는 그림 1의 (b)에 나타나있다.

버퍼 점유율의 소속 함수에는 Short(S), Close(C), Long(L)이 있다. 버퍼 점유율 차이의 소속 함수에는 Falling(F), Steady(S), Rising(R)이 있다. 출력의 소속 함수에는 Reduce(R), No change(NC), Increase(I)가 있다. 각 입력의 소속 함수들의 값은 퍼지 규칙에 의해 출력 소속 함수 값을 연산하는데 사용된다.

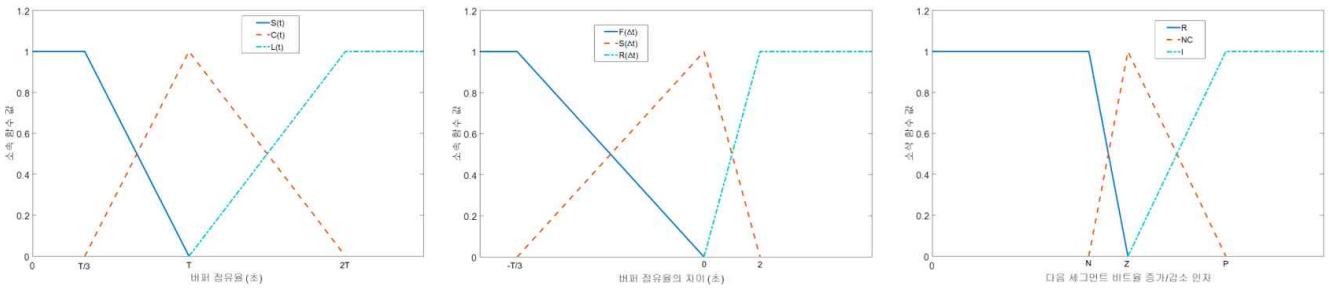
퍼지 규칙은 아래와 같이 정의 하였다.

규칙 1(r_1) : if (short) 이고 (falling) 이면 R
규칙 2(r_2) : if (close) 이고 (falling) 이면 R
규칙 3(r_3) : if (long) 이고 (falling) 이면 NC
규칙 4(r_4) : if (short) 이고 (steady) 이면 R
규칙 5(r_5) : if (close) 이고 (steady) 이면 NC
규칙 6(r_6) : if (long) 이고 (steady) 이면 I
규칙 7(r_7) : if (short) 이고 (rising) 이면 NC
규칙 8(r_8) : if (close) 이고 (rising) 이면 I
규칙 9(r_9) : if (long) 이고 (rising) 이면 I

FDASH에서 버퍼 점유율의 소속 함수 Short가 1이 되는 범위는 $0 \sim \frac{2}{3}$ 이다. 여기서 T는 이상적인 버퍼 점유율 값이다. 이 범위로 인해 버퍼가 최대한으로 활용되기 전에 퍼지 규칙에 의해 다음 세그먼트의 비트율이 낮아 질 수 있다. 따라서 mFDASH에서는 해당 범위를 $0 \sim \frac{T}{3}$ 으로 수정 하여 버퍼 점유율을 보다 효율적으로 사용 할 수 있도록 하였다. 또한, 소속 함수 Long이 1이 되는 범위를 $4T$이상에서 $2T$이상으로 수정하면서 버퍼 오버플로우가 발생하기 전에 세그먼트 비트율이 증가하게 하였다.

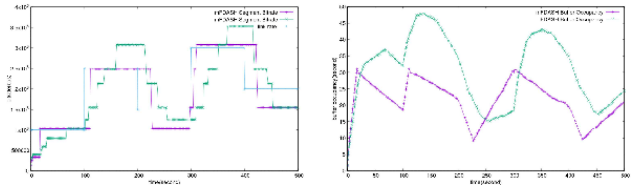
버퍼 점유율 차이의 소속 함수 Falling이 1이 되는 범위를 $-\frac{2T}{3}$ 이하에서 $-\frac{T}{3}$ 이하로 수정하였다. 모의실험을 진행하면서 결과를 분석해봤을 때, 네트워크 상황이 아무리 혼잡하여도 $\Delta q(t_k^{(e)})$ 값은 -5초에서 5초 사이를 벗어나지 않는다. 따라서 해당 범위를 $-\frac{T}{3}$ 이하로 수정하여 현재 버퍼 점유율의 변화 상태가 잘 반영되도록 하였다. 또한, 소속 함수 Rising이 1이 되는 범위를 $4T$이상에서 세그먼트 재생 시간(τ)이상으로 수정하였다. 버퍼 점유율의 차이는 네트워크 상황이 아무리 좋아도 세그먼트의 재생 시간 보다 짧을 수밖에 없다. 따라서 해당 범위의 수정을 통해서 네트워크 상황이 좋을 때 $\Delta q(t_k^{(e)})$이 f에 적절히 반영되도록 하였다.

출력 함수의 범위는 그림 1의 (c)에서 보듯이 N, Z, P로 표현 되어 있다. N, Z, P의 값을 적절히 선택해야 네트워크 상황과 버퍼의 상태를



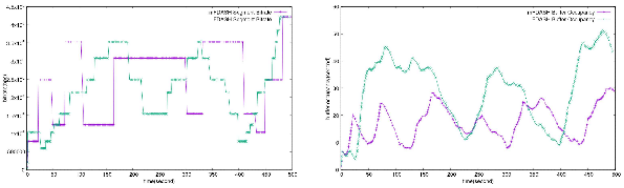
(a)버퍼 점유율 소속 함수 (b)버퍼 점유율 차이 소속 함수 (c) 출력 값 소속 함수

그림 1. 입력 출력 소속 함수



(a) 세그먼트 비트율 (b) 버퍼 점유율

그림 2. 점대점 네트워크 환경 시뮬레이션 결과



(a) 세그먼트 비트율 (b) 버퍼 점유율

그림 3. Wifi 네트워크 환경 시뮬레이션 결과

고려한 알맞은 다음 세그먼트의 비트율을 선택 할 수 있다. N은 세그먼트의 비트율을 줄이는 인자로, 1이하 이고, Z는 세그먼트 비트율을 유지시키는 인자로 1이 된다. 또한 P는 세그먼트 비트율을 높이는 인자로 1이상 이어야 한다.

출력 소속 함수의 값 R, NC, I 는 다음과 같이 구할 수 있다.

$$R = r_1^2 + r_2^2 + r_4^2 \quad (1)$$

$$NC = \sqrt{r_3^2 + r_5^2 + r_7^2} \quad (2)$$

$$I = \sqrt{r_6^2 + r_8^2 + r_9^2} \quad (3)$$

출력 값 f는 아래와 같은 식으로 나타난다.

$$f = \frac{N \times R + Z \times NC + P \times I}{R + NC + I} \quad (4)$$

B. 대역폭 측정 기법

FDASH에서는 과거 S_{window} 초안에 포함되는 세그먼트의 비트율 값들을 평균을 취한 값을 대역폭 측정값으로 사용해서 다음 세그먼트 비트율을 판단 한다. 이 방법은 사용 가능한 대역폭을 평활하게 만들지만, 대역폭의 level shift, outlier를 탐지하지 못해서 정확한 대역폭 예측이 불가능 하다는 단점이 있다. 따라서 level shift, outlier를 감지한 뒤 대역폭을 측정하는 History-Based Tcp Throughput Predictor(HBTTP)[8]를 사용한다.

C. Segment Bitrate Filtering Module

FLC에서 출력한 f는 버퍼 점유율과 버퍼 점유율의 차이에 의해 결정된다. 그래서 비디오 화질이 변하게 되면, 버퍼 점유율과 버퍼 점유율이 순간적으로 변하게 되고, 그 때문에 f값은 다시 바뀐 입력에 대한 값을 출력하게 된다. 이러한 현상 때문에 f만으로 다음 세그먼트의 비트율을 결정하게 되면 비트율이 오실레이션(Oscillation) 현상을 격게된다. 따라서 mFDASH에서는 SBFM를 추가적으로 적용하여 현재 세그먼트 비트율을 바꿔도 되는지를 판단한다.

알고리즘1은 SBFM 알고리즘의 의사코드이다. q_{high} 는 버퍼의 최대 점유율이고, q_{low}, $q_{\min}$ 은 세그먼트 비트율의 급격한 저하와 버퍼의 언더플로우를 방지하기 위한 2개의 한계점(threshold)이다. FLC의 출력 값인 f는 HBTTP의 값인 T_k 와 곱해져서 다음 세그먼트의 비트율이 된다. $Q(\bullet)$ 는 양자화 함수로, $\hat{v}$ 보다 작은 가장 큰 비트율이 $v(k+1)$ 이 된다. $v(k+1)$ 이 $v(k)$ 보다 큰 경우, T_k 와 $v(k+1)$ 의 비율이 a 이상이거나, $q(t_k^{(e)})$ 이 q_{high} 이상이면 다음 세그먼트의 비트율을 바꾸고, 그 외에는 비트율을 바꾸지 않는다. $v(k+1)$ 이 $v(k)$ 보다 작은 경우, T_k 와 $v(k+1)$ 의 비율이 b 이하거나, $q(t_k^{(e)})$ 이 q_{low} 보다 작은 경우 다음 세그먼트의 비트율을 바꾼다. $q_{low-flag}$ 는 버퍼가 언더플로우에 가까워질 때, 세그먼트의 비트율이 너무 낮게 바뀌는 것을 방지하는 플래그이다. $q(t_k^{(e)})$ 이 q_{low} 보다 작고 $q_{\min}$ 보다 클 때에는 한번만 세그먼트의 비트율을 바꾸고, $q(t_k^{(e)})$ 이 $q_{\min}$ 보다 떨어지지 않는 이상 세그먼트 비트율의 변화는 발생하지 않는다. $q(t_k^{(e)})$ 가 $q_{\min}$ 이하로 낮아지면, $q_{\min}$ 이상이 될 때까지 비디오 화질 변화가 발생한다.

알고리즘1 Segment Bitrate Filtering Module

```
1:    = f ×    ;
2:  v(k+1) = Q(v̂)
3:  if v(k+1) > v(k) then
4:    if T_k / v(k+1) > a and q(t_k^(e)) < q_high then
5:      v(k+1) = v(k);
6:    end if
7:  else if v(k+1) < v(k) then
8:    if T_k/v(k+1) < b and q(t_k^(e)) > q_low then
9:      if q_low−flag ≡ true then
10:       q_low−flag ≡ false;
11:     v(k+1) = v(k);
12:     end if
13:   end if
14:   if q(t_k^(e)) < q_low and q(t_k^(e)) > q_min and
15: q_low−flag ≡ false then
16:       q_low−flag = true;
17:   else if q(t_k^(e)) < q_low and q(t_k^(e)) > q_min and
18: q_low−flag ≡ true then
19:       v(k+1) = v(k);
20:   end if
21: end if
```

D. Start Mechanism

멀티미디어 스트리밍 서비스를 시작할 때 SBFM로 인해 수십 초간 저화질의 비디오를 시청해야 하는 현상이 발생한다. 따라서 mFDASH에서는 Start Mechanism을 추가하여 스트리밍 서비스를 시작할 때 언더플로우가 나지 않으면서 더 높은 화질로 서비스를 받을 수 있게 하였다.

알고리즘2 Start Mechanism

```
1:   EGIN =  (T/c);
2:  if Begin−flag ≡ false then
3:    if v(k+1) > v(k) then
4:      if T_k > T_{k−1} then
5:        v(k+1) = v_BEGIN;
6:      else
7:        Begin−flag = true;
8:      end if
9:    end if
10:  end if
```

알고리즘2는 Start Mechanism에 대한 의사 코드이다. 스트리밍 서비스를 시작하면 시작 비트율인 v_{BEGIN}을 $\hat{Q}(\frac{T_k}{c})$로 설정한다. 이때 $\hat{Q}(\bullet)$는 양자화 함수로, $\frac{T_k}{c}$보다 큰 가장 작은 비트율이 v_{BEGIN}이 된다. 또한, c는 시작 비트율을 조절하는 계수로, 너무 작게 하면 언더플로우가 발생 할 수 있다. Start Mechanism은 T_k가 처음으로 T_{k-1}보다 작은 값을 가지기 전까지 $v(k+1)$을 변화 시킨다. 그리고 T_k가 T_{k-1}보다 작아지는 순간 Start Mechanism이 끝난다.

E. Sleeping Mechanism

보통 모바일 기기의 버퍼 사이즈는 한계가 정해져있다. 따라서 버퍼의 최대 사이즈를 넘어가면 Sleeping Mechanism이 동작하여 버퍼의 오버플로우 발생을 방지한다. DASH 서비스 사용자는 $q(t_k^{(e)}) > q_{high}$일 때, 다음 세그먼트 요청까지 $d = q(t_k^{(e)}) - q_{high}$ 만큼 기다린다.

3. 모의실험 환경 설정 및 결과

NS-3를 이용하여 두 가지 환경에서 모의실험을 진행하였다. 첫 번째 환경에서는 서버와 DASH 서비스 사용자가 점대점 링크로 연결되어있다. 두 번째 환경에서는 Wifi에 하나의 서버와 하나의 DASH 서비스 사용자, 그리고 5개의 Background traffic이 존재한다.

세그먼트 재생시간(τ)는 2초로 설정하였다. 또한, 현실적인 버퍼 사이즈는 초 단위로 30초 이하이므로 이상적인 버퍼 점유율(T)은 20초로 설정하였다. q_{high}는 30초, $q_{low}, q_{\min}$은 각각 10초, 7초로 설정하였다. SBFM에서 사용된 계수 a, b는 각각 0.8, 1.5로 설정하였고, Start Mechanism에서 사용한 계수 c는 3으로 정하였다.

점대점 환경에서의 시뮬레이션 결과는 그림2에 나타나있고, 성능지표는 표 1에 표기되어있다. mFDASH가 FDASH에 비해 평균 세그먼트 비트율은 0.7%정도 낮지만, 비트율 변화 횟수 면에서는 54%정도 좋은 성능을 내는 것을 확인 할 수 있다. 재생 끊김 횟수는 두 스트리밍 기법 모두 0번으로 좋은 성능을 보인다. 또한, 그림2의 (b)를 보면 FDASH는 버퍼 점유율이 50초 근처까지 올라가서 오버플로우가 발생할 위험이 있지만, mFDASH는 q_{high}이상으로 올라가지 않기 때문에 오버플로우 발생 위험이 없다.

Wifi 환경에서의 시뮬레이션 결과는 그림 3에 나타나있으며, 성능지표는 표2에 표기되어있다. Wifi 환경에서는 오히려 mFDASH가 FDASH에 비해 17.8%정도 높은 평균 세그먼트 비트율을 가지는 것을 확인 할 수 있다. 또한, 비트율 변화 횟수 면에서도 mFDASH가 53.1% 정도 좋은 성능을 내는것을 볼 수 있으며, 재생 끊김은 두 기법 모두 발생하지 않는 것을 볼 수 있다. 그림 3의 (b)를 보면 점대점 환경에서의 버퍼 점유율과 마찬가지로 FDASH는 버퍼 점유율이 50초 이상 올라가면서 오버플로우가 발생 할 수 있는 가능성이 높고, mFDASH는 q_{high}이상 올라가지 않는 안정적인 모습을 보인다.

적응적 스트리밍 기법	평균 세그먼트 비트율	비트율 변화 횟수	재생 끊김 횟수
FDASH	1.721Mbps	24	0
mFDASH	1.708Mbps	11	0

표 1. 점대점 환경 성능 지표

적응적 스트리밍 기법	평균 세그먼트 비트율	비트율 변화 횟수	재생 끊김 횟수
FDASH	1.841Mbps	32	0
mFDASH	2.169Mbps	15	0

표 2. Wifi 환경 성능 지표

ACKNOWLEDGMENT

본 연구는 미래창조과학부 및 정보통신기술연구진흥센터의 정보통신·방송 연구개발사업의 일환으로 수행하였음. [R0101-16-0189, 네트워크가 결합된 매체 독립형 차세대 융합방송 시스템 및 모니터링 시스템 개발]

참고문헌

[1] Sodagar, Iraj. "The mpeg-dash standard for multimedia streaming over the internet." *IEEE MultiMedia* 18.4 (2011): 62-67.

[2] Tian, Guibin, and Yong Liu. "Towards agile and smooth video adaptation in dynamic HTTP streaming." *Proceedings of the 8th international conference on Emerging networking experiments and technologies*. ACM, 2012.

[3] Liu, Chenghao, Imed Bouazizi, and Moncef Gabbouj. "Rate adaptation for adaptive HTTP streaming." *Proceedings of the second annual ACM conference on Multimedia systems*. ACM, 2011.

[4] Vergados, Dimitrios J., et al. "FDASH: A Fuzzy-Based MPEG/DASH Adaptation Algorithm." *IEEE Systems Journal* 10.2 (2016): 859-868.

[5] SG12, I. T. U. T. "Definition of quality of experience." *TD 109rev2 (PLEN/12), Geneva, Switzerland* (2007): 16-25.

[6] Lee, Chuen-Chien. "Fuzzy logic in control systems: fuzzy logic controller. I." IEEE Transactions on systems, man, and cybernetics 20.2 (1990): 404-418.

[7] Klir, George, and Bo Yuan. *Fuzzy sets and fuzzy logic*. Vol. 4. New Jersey: Prentice hall, 1995.

[8] He, Qi, Constantine Dovrolis, and Mostafa Ammar. "On the predictability of large transfer TCP throughput." *ACM SIGCOMM Computer Communication Review*. Vol. 35. No. 4. ACM, 2005.