# 고속 네트워크에서 TCP 성능 개선 기법

양은호, 김종권
서울대학교 컴퓨터공학부
e-mail : ehyang@popeye.snu.ac.kr

# Performance Improvement of TCP Over High-speed Networks

Eun-ho Yang, Chong-kwon Kim
Dept. of Computer Science and Engineering, Seoul National University

**요 약**

Fast long-distance network 에서 기존 TCP 의 혼잡 제어 (congestion control) 알고리즘은 대역폭을 효과적 사용하지 못하는 문제점을 가지고 있다. 대역폭을 효과적으로 사용하기 위해서 TCP 혼잡 제어를 수정한 다양한 프로토콜들이 제안되었다. 이러한 프로토콜들은 디자인 시 주로 bandwidth scalability, TCP friendliness, 그리고 RTT fairness 와 같은 세 가지의 특성을 고려하고 있다. 하지만 제안된 프로토콜들은 어떤 것도 trade-off 관계로 있는 이 세 가지 특성을 동시에 만족시키지 못한다. 본 논문에서는 혼잡 제어 알고리즘의 증가 규칙 (increase rule)에 RTT 를 직접 반영함으로써 위 세가지 요구사항을 동시에 만족시키는 EIMD (Exponential Increase/ Multiplicative Decrease)라고 하는 새로운 TCP 혼잡 제어 알고리즘을 제안한다. EIMD 는 패킷 손실이 없는 한, 지수적으로 윈도우를 증가시켜 효과적으로 대역폭을 사용하면서도, 패킷손실 직전의 윈도우 크기, $w_{\max}$ 에 반비례하게 윈도우를 증가시킴으로써 fair share 에 빠르게 수렴할 수 있다는 특성을 갖는다. 모의실험을 통해 제안된 프로토콜이 fast long-distance network 에서 위 4 가지 특성들을 모두 만족하는지 검증한다

## 1. 서론

네트워크가 점점 고속화되는 동시에 세계화 되면서 Fast Long-Distance Networks 에 대한 연구가 점차 주목을 받고 있다. 대표적인 fast Long-Distance Networks 으로는 Grid Network 이나 Abilene, ESNet 등이 있다. 이러한 네트워크는 DataGrid 의 High Energy Physics 나 Medical Image Processing 과 같이 고성능 연산 능력뿐만 아니라 대용량의 파일 전송을 필요로 하는 응용분야들의 출연으로 중요도가 더욱 커지고 있다.

이런 네트워크에서는 현재 지배적인 전송 계층 프로토콜인 TCP (Transmission Control Protocol)의 성능이 현저하게 저하되는 문제가 발생한다. S. Floyd [1] 는 기존 TCP 의 혼잡 제어 (congestion control) 알고리즘이 혼잡이 발생하지 않은 상황에서는 하나의 RTT (Round Trip Time) 동안에 윈도우 크기를 1 씩만 증가시킬 뿐만 아니라 혼잡이 발생한 상황에서는 윈도우 크기를 절반으로 급격히 줄이기 때문에 fast long-distance network 의 대역폭을 효과적으로 사용하지 못함을 지적하고 있다. 예를 들어 패킷 (packet)의 크기가 1500 byte 이고 RTT 가 100ms 인 TCP 플로우가 있다고 했을 때, 10Gbps 의 처리율을 얻기 위해서는 평균 윈도우 크기가 83,333 이 되어야 한다. 이런 큰 평균 윈도우 크기를 유지하기 위해서는 대략 1 시간 40 분마다 하나 정도의 패킷만이 손실 되어야 한다. 이 수치는 네트워크의 물리적인 BER 한계를 초과하기 때문에 기존 TCP 를 이용해서는 병목 링크 (bottleneck link)의 대역폭이 아무리 크다고 해도 이를 충분히 활용할 수 없다 [1]. 따라서 TCP 가 여전히 지배적인 전송계층 프로토콜임에도 불구하고, fast long-distance network 에서는 새로운 high-speed 프로토콜이 필요하다.

Fast Long-Distance Networks 에서 TCP 의 문제를 해결하기 위해서 다양한 프로토콜들이 제안되었다. 이 중에서 우리는 윈도우-기반 (window-based) 혼잡 제어 프로토콜에 초점을 맞추어 연구를 하였다. 이러한 윈도우-기반 프로토콜은 라우터의 수정을 필요로 하지 않고, 점차적 배치(incremental deployment)에 더욱 유리

하다는 장점이 있기 때문이다 [2]. 하지만 기존의 윈도우 기반 프로토콜들은 bandwidth scalability, TCP-friendliness 및 RTT-fairness 를 동시에 만족시키지 못한다. 뿐만 아니라 자신의 fair-share 에 빠르게 수렴하지 못한다는 단점이 있다. 이러한 문제는 대역폭-지연시간 곱이 커짐에 따라서 더욱 심각해질 수 있다.

본 논문에서 우리는 bandwidth scalability 와 TCP-friendliness 를 유지하면서도 RTT-fairness 문제를 근본적으로 해결할 수 있는 새로운 전송 계층 프로토콜인 EIMD (Exponential Increase/ Multiplicative Decrease)를 제안하고자 한다. 먼저 우리는 trade-off 관계에 있는 세 가지 특성(scalability 와 TCP-friendliness, 그리고 RTT-fairness)을 동시에 만족시킬 수 없는 기존 프로토콜의 반응함수 (response function)의 본질적인 문제를 해결하기 위하여 scalability 와 TCP-friendliness 를 유지하면서도 RTT-unfairness 문제를 해결할 수 있는 새로운 반응함수를 제안하였다. 그리고 EIMD 가 제안한 반응함수를 따르도록 혼잡 제어 알고리즘의 증가/감소 규칙 (increase/decrease rules)을 구하였다. 기존의 윈도우-기반 프로토콜과 비교하여 볼 때, EIMD 는 다음과 같은 네 가지의 특성을 갖는다.

-RTT-fairness: EIMD 는 처리율이 $RTT^{-k}$ 에 비례한다. 본 논문에서는 k 를 1 로 하여 RTT 에 반비례하는 처리율을 갖는다.

-Scalability: RTT 가 100ms 인 EIMD 플로우가 $10e^{-7}$ 의 손실률에서 10Gbps 의 처리율을 얻는다. 이 때 다른 RTT 를 갖는 플로우들은 100ms 플로우를 기준으로 RTT 에 반비례하는 처리율을 얻는다.

-TCP-friendliness: EIMD 는 패킷 손실률에 관계없이 기존 high-speed 프로토콜보다 보다 뛰어난 TCP-friendliness 를 제공한다.

-Fair share convergence: 패킷손실 직전의 윈도우 크기, $w_{\max}$ 에 반비례하게 윈도우를 증가시킴으로써 fair share 에 빠르게 수렴할 수 있다는 특성을 갖는다.

본 논문의 구성은 다음과 같다. 2 장에서는 기존에 제안된 윈도우 기반 high-speed 프로토콜들을 살펴본다. 3 장에서는 EIMD 의 반응함수와 이에 맞는 윈도우 혼잡제어 알고리즘의 윈도우 증가/감소 규칙을 어떻게 결정하는지 자세히 설명하고 4 장에서는 모의실험을 통해 EIMD 가 위에서 언급한 네 가지의 특성들이 만족함을 보인다. 마지막으로 5 장에서 본 논문의 결론을 맺는다.

## 2. 관련 연구

Scalable TCP [3]와 HSTCP [1]는 고정된 윈도우 크기의 증가량 대신에 현재의 윈도우 크기를 바탕으로 윈도우 증가량을 매 RTT 마다 새롭게 계산한다. 즉, 혼잡이 발생하지 않은 경우에 윈도우 증가량을 크게 하여 대역폭 이용효율 (bandwidth utilization)을 좋게 하고 반대로 혼잡 상황이 발생했을 때는 상대적으로 윈도우 증가량을 작게 하여 네트워크의 혼잡을 경감시킴은 물론 보다 작은 RTT 를 가지는 기존 TCP 플로우 (현재 인터넷 플로우의 대다수)에 피해를 최소화 한다. 그러나 이런 방법들은 Xu and et al. [4]가 지적한 것처럼 극심한 RTT-unfairness 를 겪게 된다. 특히 라우터가 드롭테일 큐 (drop-tail queue)를 사용하고 대역폭이 커질수록 더 많은 동시손실 (synchronized loss)이 발생하고, 이로 인해 RTT-unfairness 문제가 더욱 심각해진다. 이러한 문제를 경감하기 위해서는 반응함수의 기울기를 줄여야 하는데 그럴 경우 scalability 나 TCP-friendliness 가 나빠지게 된다. BIC [4]는 부분적으로 반응함수의 기울기를 줄임으로써 RTT-unfairness 를 제공하지만 Scalable TCP 보다 TCP-friendliness 가 나쁘고 HSTCP 보다도 scalability 가 나쁘다는 단점이 여전히 존재한다.

## 3. Exponential Increase/Multiple Decrease (EIMD)

이번 장에서는 bandwidth scalability, TCP-friendliness, RTT-fairness 및 빠른 fair-share convergence 을 모두 만족시키는 Exponential Increase/Multiple Decrease (EIMD) 라고 하는 프로토콜을 제안한다.

### 3.1 Response function of EIMD

Scalability, TCP-friendliness 그리고 RTT-fairness 를 동시에 만족시키기 위하여 EIMD 는 RTT 를 고려한 새로운 반응함수를 따른다.

$$w_i = \frac{A}{p^s} RTT^r \qquad (1)$$

$s$ 는 EIMD 의 반응함수의 기울기를 결정하는 파라미터이고, $A$ 와 $r$ 은 각각 aggressiveness 와 RTT-fairness 의 정도를 결정하는 파라미터 값이다. $A$ 가 1.75 이고 $r$ 과 $s$ 가 0.82 일 때, 40ms, 240ms 인 두 플로우의 반응함수는 그림 1 와 같다.

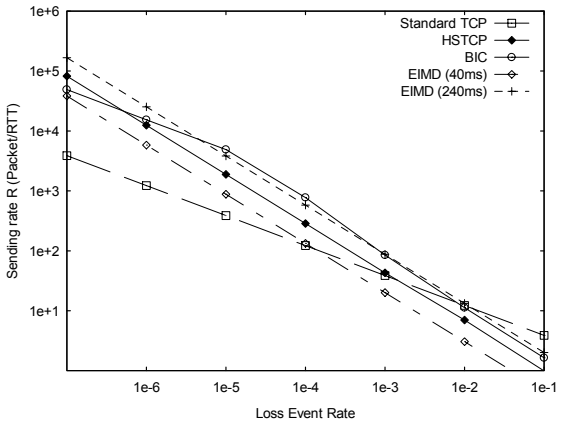


그림 1. Response function of EIMD

완벽한 동시손실 모델에서 $s$ 와 $r$ 에 따른 EIMD 의 RTT-fairness 를 분석해 보도록 하겠다. $w_i$ 와 $RTT_i$ 를 각각 플로우 $i$ 의 윈도우 크기와 RTT 값을 나타낸다고 하고 $t$ 는 평형 상태 (steady-state)에서 두 개의 연속적인 패킷 손실 사이의 시간 (second 단위)이라고 하자. 플로우 $i$ 의 손실이 $p_i$ 의 확률로 균일하게 분포되어있다고 한다면 두 개의 연속적인 패킷 손실 동

안 보낸 총 packet 의 수는 $1/p_i$ 가 된다. 이때 총 필요한 RTT 의 개수가 $t/RTT_i$ 번 이므로 다음 식을 얻을 수 있다.

$$w_i = \frac{1/p_i}{t/RTT_i} = RTT_i/(t \cdot p_i) \qquad (2)$$

식(1), (2)를 이용하면, 두 플로우의 처리율의 비는

$$\frac{(w_1/RTT_1)/(1-p_1)}{(w_2/RTT_2)/(1-p_2)} \approx \frac{w_1/RTT_1}{w_2/RTT_2} = \left(\frac{RTT_2}{RTT_1}\right)^{\frac{1-r}{1-s}} \quad (3)$$

이 된다. 이 때 패킷 손실률이 거의 0 에 가깝기 때문에 $(1-p_i)$ 는 근사적으로 1 이라고 할 수 있다. RTT-fairness 가 RTT 에 반비례하기 위해서는 $r$ 과 $s$ 가 같은 값을 가지면 된다. 결과적으로 $r$ 값을 조절함으로써 기울기 $s$ 를 줄이지 않고 RTT-fairness 를 향상시켰다.

다음은 EIMD 의 TCP-friendliness 에 대해서 살펴보자. 상대적으로 작은 RTT 를 가지는 high-speed 플로우가 TCP-friendliness 를 더 많이 해치게 된다. 그러므로 상대적으로 작은 RTT 를 가지는 high-speed 플로우를 덜 공격적 (aggressive)으로 만들고 상대적으로 큰 RTT 를 가지는 플로우를 더 공격적으로 만드는 것은 RTT-fairness 뿐만 아니라 TCP-friendliness 도 좋게 할 수 있다. 즉, RTT-fairness 와 TCP-friendliness 를 동시에 증진시키기 위해서는 서로 다른 RTT 를 가지는 플로우들의 처리율 간의 간격을 줄여야 한다. 그림 2 에서 EIMD 가 HSTCP 에 비해서 처리율 간의 간격이 더 좁다는 것을 보여준다.

마지막으로 EIMD 의 scalability 에 대해서 살펴보자. 100ms RTT 를 갖는 flow 가 $10^{-7}$ 의 loss rate 에서 10Gbps 이상을 제공하므로 다른 프로토콜에 비견될 수 있다.

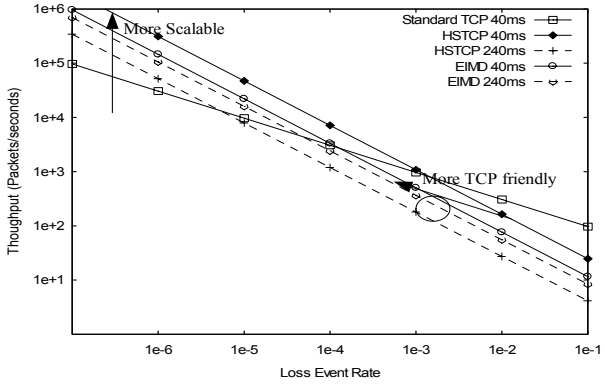


그림 2. Throughputs of EIMD

### 3.2 EIMD 의 증가/감소 규칙

3.1 장의 반응함수를 바탕으로 증가/감소 규칙을 유도해보자. EIMD 는 패킷 손실이 일어난 이후 시간에 지수적으로 비례하여 윈도우를 증가시킨다. 그러므로 w(T)를 시간 T (RTT 단위)에서 윈도우의 크기라고 할 때, EIMD 의 윈도우는 $w(T) = w(0) + cT^u$ 와 같이 표현될 수 있다. 이를 각 패킷에 대한 Ack 레벨의 규칙으로 바꿔보면 다음과 같다.

증가 규칙: $w_{T+1} \leftarrow w_T + \alpha(w_T - w_0)^{1-1/u}$, $\alpha = u \cdot c^{1/u}$

감소 규칙: $w_{T+\delta} \leftarrow w_T - \beta w_T$

[5]에서 사용한 방법을 이용하여 EIMD 의 반응함수에 맞는 $\alpha$ 와 $u$ 값을 구해보자. 평형 상태 (steady-state)를 가정하면 윈도우의 증가량과 감소량이 같아야 하므로 $cT^u = \beta w_{\max}$ 가 성립한다. 이 때 전송된 패킷의 수는 $\int_0^T ((w_{\max} - \beta w_{\max}) + cT^u)dT$ 과 같이 구할 수 있다. 그러므로 다음과 같은 식이 성립한다.

$$T = \left(\frac{\beta w_{\max}}{c}\right)^{1/u} \qquad (4)$$

$$Total\ \ number\ \ of\ \ packets = \frac{1}{p}$$

$$= (w_{\max} - \beta w_{\max})T + \frac{c}{u+1}T^{u+1} \qquad (5)$$

식 (2)와 $T = t/RTT$ 라는 사실로부터 평균 윈도우의 크기는 $w = 1/(p \cdot T)$ 이 됨을 알 수 있고, 이 값이 식 (1)과 일치해야 하므로

$$\frac{1}{p \cdot T} = \frac{A}{p^s} RTT^r \qquad (6)$$

을 만족해야 한다. 결과적으로 식 (4), (5), (6)으로부터 $c$ 값과 그에 따른 $\alpha$ 값을 구할 수 있다.

$$c = \left(\frac{(A \cdot RTT^r)^{1/(1-s)}}{1-(u/(u+1))\beta}\right)^{u(1-s)/s} \cdot \frac{\beta}{\Gamma(u+1)} w_{\max}^{1-u(1-s)/s} \qquad (7)$$

$$\alpha = u\left(\frac{(A \cdot RTT^r)^{1/(1-s)}}{1-(u/(u+1))\beta}\right)^{(1-s)/s} \left(\frac{\beta}{\Gamma(u+1)}\right)^{1/u} w_{\max}^{\frac{1}{u}\frac{1-s}{s}} \qquad (8)$$

식 (7)로부터, 윈도우의 크기가 $w_{\max}^{1-u(1-s)/s}$ 에 비례함을 알 수 있다. $-1 = 1 - u(1-s)/s$ 을 만족시키는 $u$ 값을 선택하여 윈도우의 증가가 $1/w_{\max}$ 에 비례하도록 하였다. 즉, $c \propto 1/w_{\max}$ 이 성립하도록 하여, fair-share 에 빠르게 수렴할 수 있도록 하였다.

## 4. 모의 실험

EIMD 의 성능을 평가하기 위하여 덤벨 토폴로지 (dumbbell topology)에서 HSTCP, BIC 와 비교 실험하였다. 일반적인 인터넷에서의 병목 링크와는 다르게 high-speed 환경에서는 high-speed 링크가 만나는 곳에서 주로 병목 현상이 생기므로 덤벨 토폴로지가 실제 high-speed network 의 환경과 흡사하다고 할 수 있다. Phase effect [6]를 줄이기 위해서, 각 플로우들은 독립적으로 시작하고 종료하도록 하였다. Background traffic 으로는 양쪽 방향으로 윈도우의 크기가 64 이하로 제한된 small TCP 와 web traffic 을 사용하였다.

표 1 과 2 는 하나의 플로우는 RTT 가 40ms 로 고정되어 있고, 다른 하나는 40ms, 120ms, 240ms 로 바뀌게 되는 경우, 병목 링크가 200Mbps 와 2.5Gbps 에서 RTT 의 비율에 따른 처리율 (throughput)의 비를 나타낸 것이다. EIMD 가 HSTCP 와 BIC 보다 더 RTT-fair 함을 알 수 있다.

<표 1> Inverse throughput ratios under 200Mbps

| RTT ratio | 1 | 3 | 6 |
|---|---|---|---|
| EIMD | 1.04 | 2.27 | 2.93 |
| BIC | 0.98 | 18.54 | 43.49 |
| HSTCP | 0.96 | 10.45 | 26.51 |

<표 2> Inverse throughputs ratios under 2.5Gbps

| RTT ratio | 1 | 3 | 6 |
|---|---|---|---|
| EIMD | 1.03 | 3.2 | 4.91 |
| BIC | 0.87 | 10.63 | 35.08 |
| HSTCP | 1.02 | 28.03 | 108.08 |

그림 3 은 각 프로토콜을 사용하였을 때, 플로우에 할당된 대역폭의 비율을 나타낸 그래프이다. 여기서 long-lived TCP 플로우와 background traffic 의 합이 각 프로토콜의 TCP-friendliness 를 나타내게 된다. BIC 는 사용 가능한 대역폭을 효과적으로 사용한 반면 윈도우의 크기에 상관없이 long-lived TCP 플로우들로부터 많은 양의 대역폭을 뺏어 오는 단점이 있다. 패킷 손실이 높은 경우에도 하나의 BIC 플로우가 10 개 정도의 long-lived TCP 플로우들이 사용하는 대역폭을 사용하게 되어 TCP-friendliness 가 많이 떨어진다는 것을 알 수 있다. 반면에 EIMD 는 패킷 손실에 상관없이 가장 TCP-friendliness 가 가장 뛰어남을 알 수 있다. EIMD 의 경우 미사용 대역폭이 다른 두 프로토콜에 비해서 상대적으로 약간 높은 것을 볼 수 있지만, 이는 가장 TCP-friendly 하기 때문이다. 또한 표 3 에서 보듯이, 병목링크의 대역폭이 커질수록 EIMD 의 이용효율 (utilization)이 점점 높아짐을 알 수 있다.

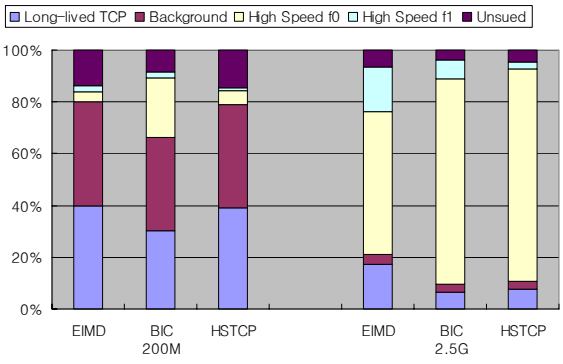


그림 3. Percentage of bandwidth share for various protocols under 200Mbps and 2.5Gbps

<표 3> Utilizations of EIMD under various bandwidths

| Bottleneck(bps) | 200M | 2.5G | 5G |
|---|---|---|---|
| Utilization(%) | 86.22 | 93.34 | 96.95 |

마지막으로 EIMD 의 fair-share 수렴에 대해서 살펴보자. 100ms 의 RTT 를 가지는 하나의 high-speed 플로우와 background traffic 이 0 초에서 20 초 사이에 랜덤하게 전송을 시작하게 하였다. 80 초부터 같은 RTT 를 갖는 다른 high-speed 플로우가 새로이 전송을 시작하여 병목 링크를 공유하도록 하였다. 80 초 이후부터 5 초 간격으로 각 플로우의 누적 처리율을 계산하였다. 그림 4 은 이 누적 처리율을 바탕으로 fair share index [7] 값을 구한 것이다. 다른 프로토콜보다 EIMD 가 훨씬 빠르고 안정적으로 fair-share 에 접근함을 알 수 있다.

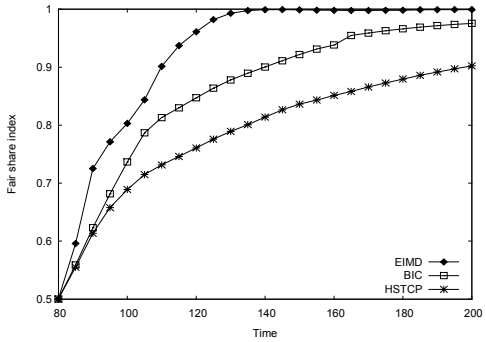


그림 4. Fair-share convergence of high-speed protocols

## 5. 결론

본 논문에서는 bandwidth scalability 와 TCP-friendliness 를 만족하면서도 RTT-unfairness 문제를 근본적으로 해결할 수 있는 EIMD 라고 하는 새로운 전송계층 프로토콜을 제안하였다. 이를 위해 EIMD 는 RTT 를 반응함수에 고려함으로써 trade-off 관계에 있는 위 세가지 특성들을 동시에 만족 시킬 수가 있었다. 뿐만 아니라 $w_{\max}$ 에 반비례하게 윈도우 크기를 증가시킴으로써 fair share convergence 가 매우 좋다. 모의실험을 통하여 EIMD 의 성능을 증명하였다. 앞으로 EIMD 를 실제 구현하여 성능을 평가해보고자 한다.

## 참고문헌

[1] S. Floyd. “HighSpeed TCP for Large Congestion Windows”, RFC 3649, December 2003

[2] D. Bansal, H. Balakrishnan, S. Floyd, and S. Shenker. “Dynamic Behavior of Slowly-Responsive Congestion Control Algorithms”, SIGCOMM ’01, August 2001

[3] T. Kelly. “Scalable TCP: Improving performance in highspeed wide area networks”, ACM SIGCOMM Computer Communication Review, Vol. 33, Issue 2, pp. 83-91, April 2003

[4] L. Xu, K. Harfoush, I. Rhee. “Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks”, IEEE INFOCOM ’04, March 2004

[5] S. Jin, L. Guo, I. Matta, and A. Bestavros. “A Spectrum of TCP-friendly Window-based Congestion Control Algorithms”, IEEE/ACM Transactions on Networking, vol. 11 no 3, pp. 341-355, Jun 2003

[6] S. Floyd, E. Kohler. “Internet research needs for better models”, http://www.icir.org/models/bettermodels.html, October 2002

[7] D.-M. Chiu, R. Jain. “Analysis of the increase and decrease algorithms for congestion avoidance in computer networks”, Computer Networks and ISDN systems, 17:1-14, 1989