On ACK Filtering on a Slow Reverse Channel

Chadi Barakat & *Eitan Altman*

INRIA Sophia Antipolis - France

Workshop on Quality of Future Internet Services
September 25, 2000
Berlin, Germany

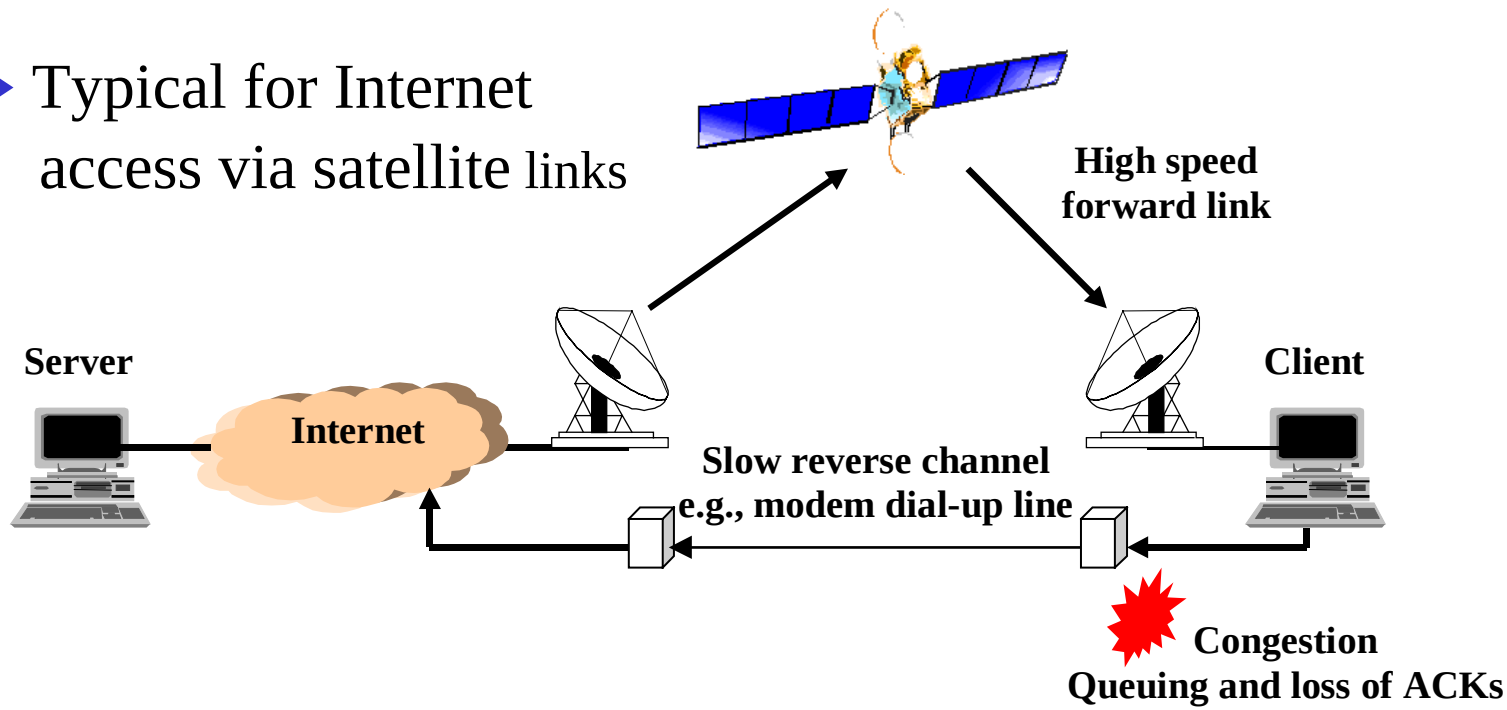
INRIA

Outline

- Introduction
  - Problem of bandwidth asymmetry and its impact on TCP
  - ACK filtering as a solution
  - Burstiness of TCP and ACK filtering tradeoff
- Static ACK filtering
  - How many ACKs to queue before filtering?
- Delayed or adaptive ACK filtering
  - Relate ACK filtering to the slow channel utilization
- Concluding remarks

Bandwidth asymmetry for TCP

- When the reverse channel (Client ⇒ Server) is not able to carry the flow of ACKs that results from a good utilization of the bandwidth in the forward direction
  - Typical for Internet access via satellite links



Congestion and ACK filtering

- Main problems with congestion on the reverse path
  - Increase in end-to-end delay, decrease in window growth rate, and thus throughput deterioration
  - Unfairness in the distribution of the slow channel bandwidth
    (unresponsiveness of the flow of ACKs to drops)
    - Blockage of a newly arriving connection

- ACK filtering as a proposed solution
  - Erase old ACKs from a connection when a new ACK arrives

H. Balakrishnan, V. Padmanabhan, and R. Katz,
"The effects of Asymmetry on TCP performance", ACM MOBICOM 1997

ACK filtering tradeoff

- ACK filtering has been proposed with the main objective to reduce the end-to-end delay and the buffer occupancy
- But, ACKs are also necessary for sender operation especially during slow start where they are used to increase the window
  - Slow start is already slow enough on satellite links
- **The tradeoff**

Reduce the length of the queue of ACKs to the minimum

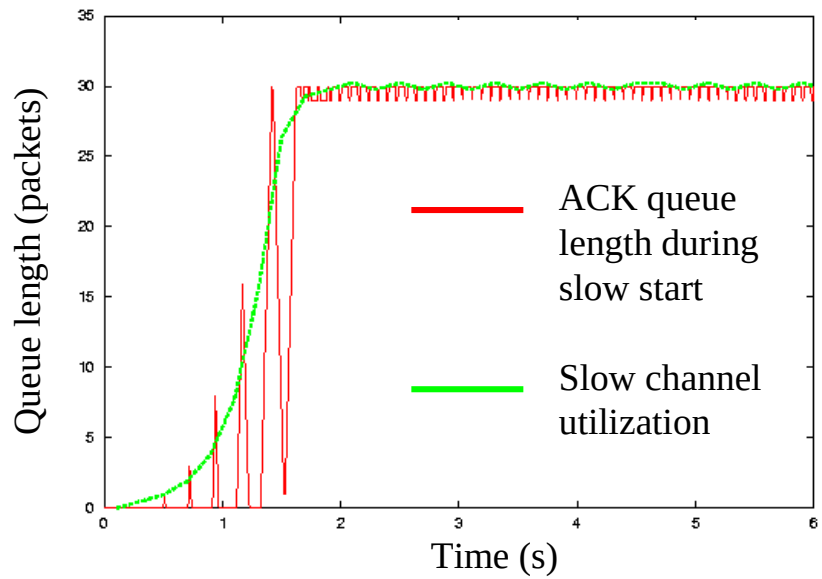
vs.

Pass the maximum possible number of ACKs to the source

Why there is a problem?

- There is no contradiction between the two objectives if ACKs arrive smoothly at the slow channel
- But, ACKs arrive in fact in bursts during slow start

→ Filtering bursts of ACKs before the full utilization of the reverse channel is not necessary since these bursts if absorbed, will not cause an increase in the round-trip time, but if filtered, they will slow considerably the window increase



- **Optimum behavior:**

  Filter ACKs just after the full utilization of the slow channel

Static ACK filtering

- **Idea:** Allow the building of a queue of δ ACKs at the input of the slow channel
  - δ = 1 is the case studied in the literature
- **Analysis:** The minimum δ to absorb ACK bursts

$$\delta_0 = \frac{1}{3}\mu_r T$$

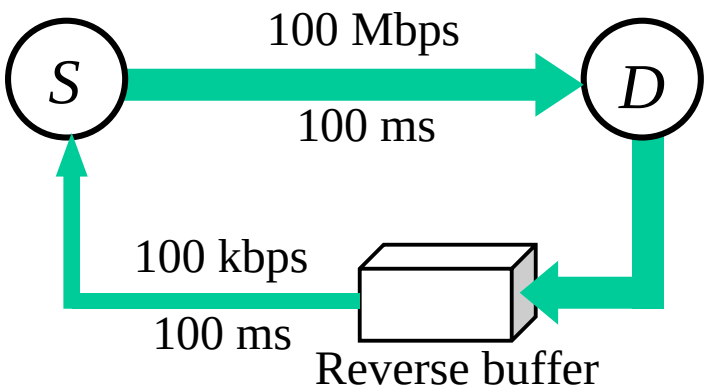
μ_r : bandwidth of the slow channel

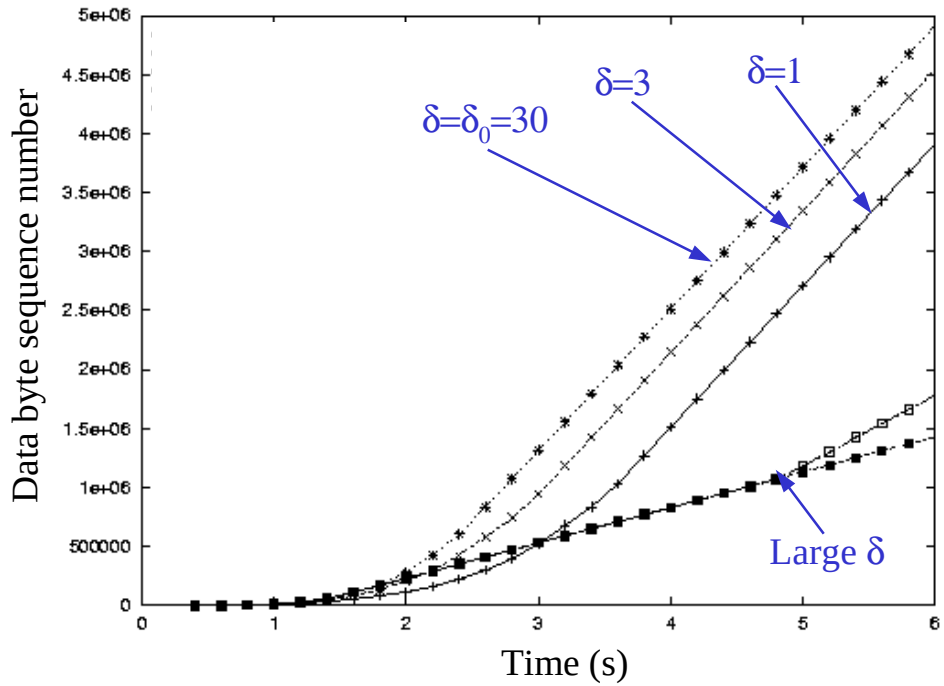
T : round-trip time

- **Tradeoff:**
  - A $\delta < \delta_0$ slows the window growth during slow start and makes it polynomial instead of exponential
  - A $\delta > \delta_0$ absorbs bursts of ACKs, but leads to an unnecessary increase in round-trip time after the full utilization of the slow channel

Simulation validation

Simulate with ns-2 :





- A short TCP transfer from S to D
- TCP version : Reno
- Packet size : 1000 bytes
- Very large reverse buffer

Delayed ACK filtering

- **Motivations:**
  - Difficulty of the calculation of the optimum queue length in practice
  - No need to queue ACKs once the slow channel is fully utilized
- **Algorithm:**
  - Measure directly the slow channel utilization
  - Halt the filtering when the utilization is less than a certain threshold
  - Filter ACKs with $\delta = 1$ if not
- **Utilization measurement:**
  - Measure the rate of ACKs at the output of the slow channel and compare it to the slow channel bandwidth
  - If the slow channel bandwidth is not known, measure how frequently ACKs are present in the buffer at the input of the slow channel

Time sliding window algorithm

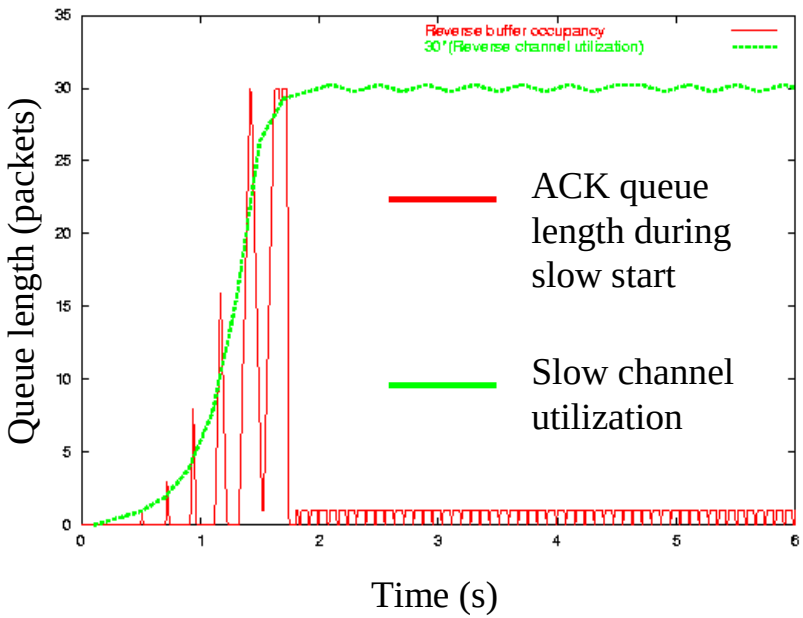
- An *Exponentially Weighted Moving Average* algorithm with the advantage that the convergence rate is a constant and not dependent on the frequency and the value of measurements. [Clark,Fang,1998]
  - The convergence rate is controlled by a *Time Window* variable
- For example, to measure the rate at the output of the slow channel:
  - Before transmitting an ACK on the slow channel:

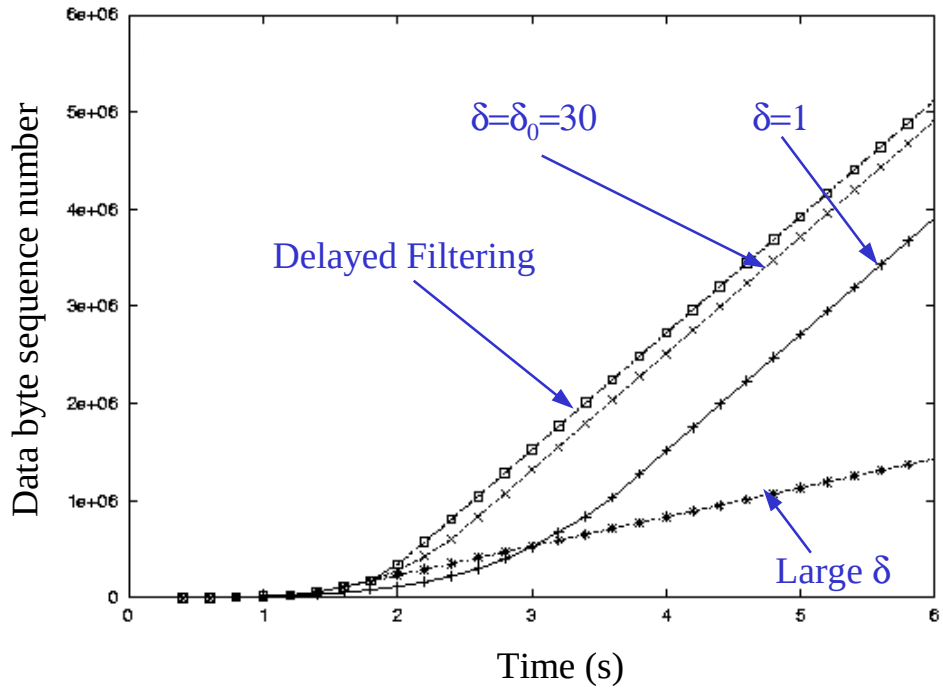
$$AvgRate \leftarrow \frac{AvgRate \times Window + ACKSize}{Window + (Now - LastTime)}$$

$$\approx AvgRate(t) = (Rate_0 - Rate_1)e^{-t/Window} + Rate_1$$

Validation: Case of 1 connection

For the same simulation scenario ...



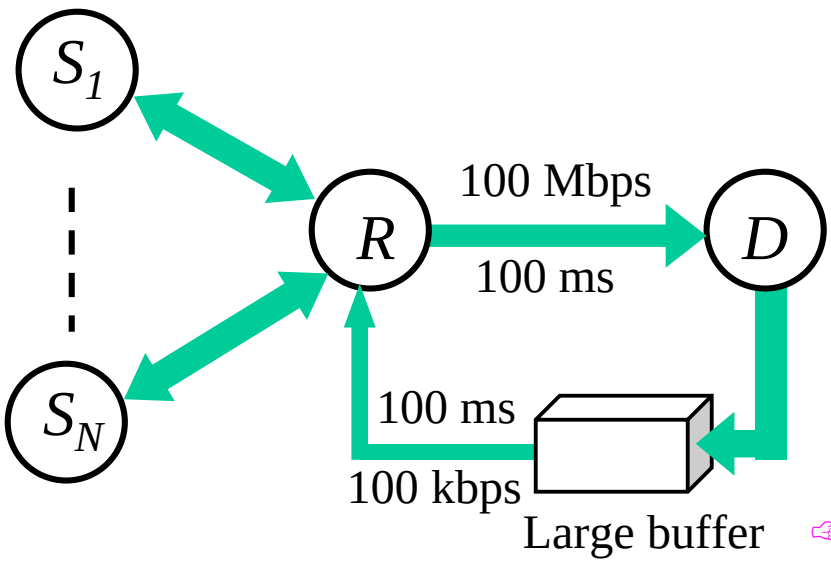


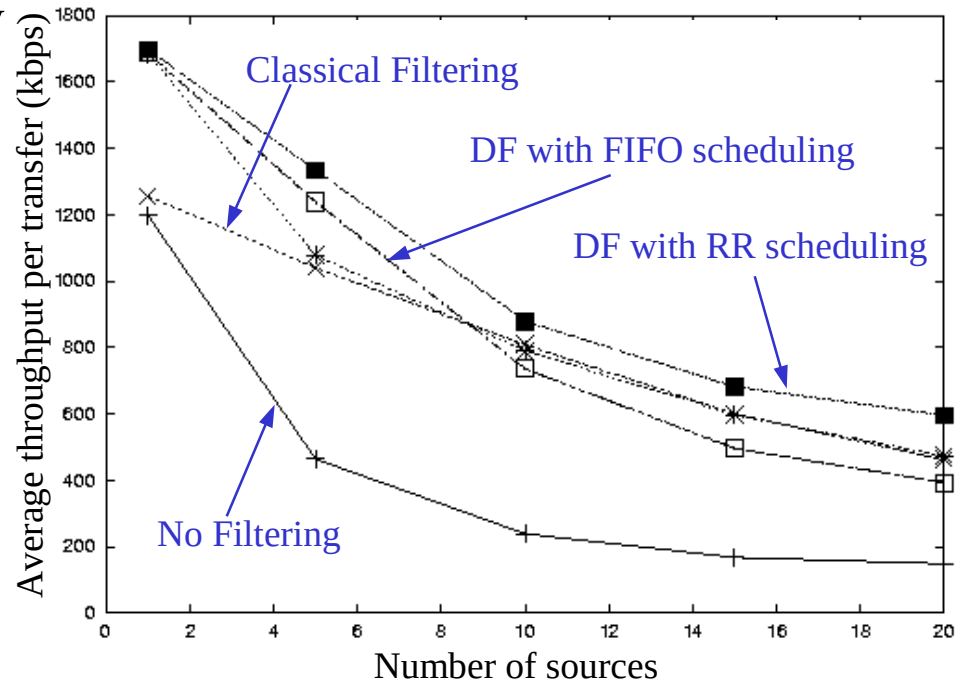
Case of multiple connections

- Assume that multiple connections share the reverse channel for their ACKs
- Need for a *per-connection* filtering decision
  - Using only the channel utilization to decide on ACK filtering is not enough to protect new connections from already running ones
  - Per-connection Delayed Filtering:
    - Measure the rate of ACKs for every running connection
    - Filter ACKs of a connection when their rate exceeds their fair share of the slow channel bandwidth
- Need for a per-connection scheduling to ensure a complete isolation
  - ACKs from filtered connections need to be protected from waiting behind ACKs from unfiltered ones (use of a Round Robin scheduler)

Simulation

- Every source transmits successively files of size chosen randomly between 10 kbytes and 10 Mbytes

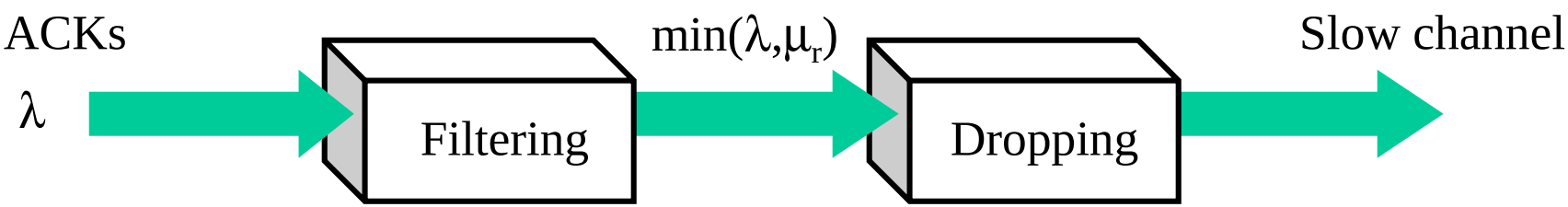




- Best performance for a Round Robin scheduling and per-connection delayed filtering
- No interest from FIFO scheduling when N is large

Case of small buffers

- A buffer management technique is required to ensure fairness in case the ACK filtering scheme is not enough to avoid buffer overflow



Reduce the rate of ACKs without the loss of information

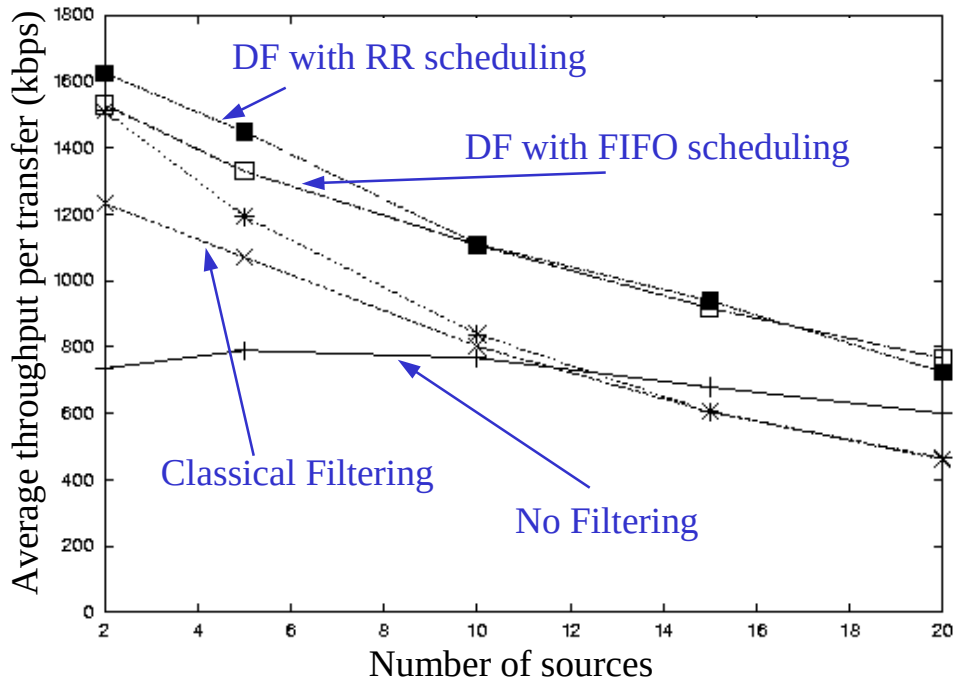
- Dropping techniques we used:
  - In case of no information on rates, use the Longest Queue Drop technique
  - In case of Per-connection Delayed Filtering, drop the oldest ACK from the connection with the highest rate

Simulation

- Take a small reverse buffer of 10 packets in the previous scenario

- When N is smaller than the buffer size, we get similar performance to that in case of a large buffer

- When N is larger than the buffer size, the drop policy becomes the most important factor.
  - Per-connection dropping gives the best performance.
  - Classical filtering ($\delta = 1$) behaves worse than the no filtering case.



Summary

- Delaying ACK filtering until the slow channel is fully utilized guarantees a small queuing time and the transmission of a large number of ACKs
- Delaying filtering until the rate of ACKs of a connection exceeds a certain threshold guarantees a certain fairness in slow channel bandwidth sharing
- Delayed filtering and the number of connections:
  - Convergence to classical filtering when the fair share of a connection from the slow channel bandwidth is about one ACK per round-trip time
  - No interest from FIFO scheduling when the number of connections is large
  - No interest from classical filtering when the number of connections is larger than the buffer size