Balancing TCP Buffer Size vs Parallel Streams in Application-Level Throughput Optimization

Esma Yildirim, Dengpan Yin, **Tevfik Kosar***

Center for Computation & Technology
Louisiana State University

June 9, 2009
DADC'09

CENTER FOR COMPUTATION & TECHNOLOGY

Motivation

- End-to-end data transfer performance is a major bottleneck for large-scale distributed applications
- TCP based solutions
  - Fast TCP, Scalable TCP etc
- UDP based solutions
  - RBUDP, UDT etc
- Most of these solutions require kernel level changes
- Not preferred by most domain scientists

Application-Level Solution

- Take an application-level transfer protocol (i.e. GridFTP) and tune-up for optimal performance:
  - Using Multiple (Parallel) streams
  - Tuning Buffer size

Roadmap

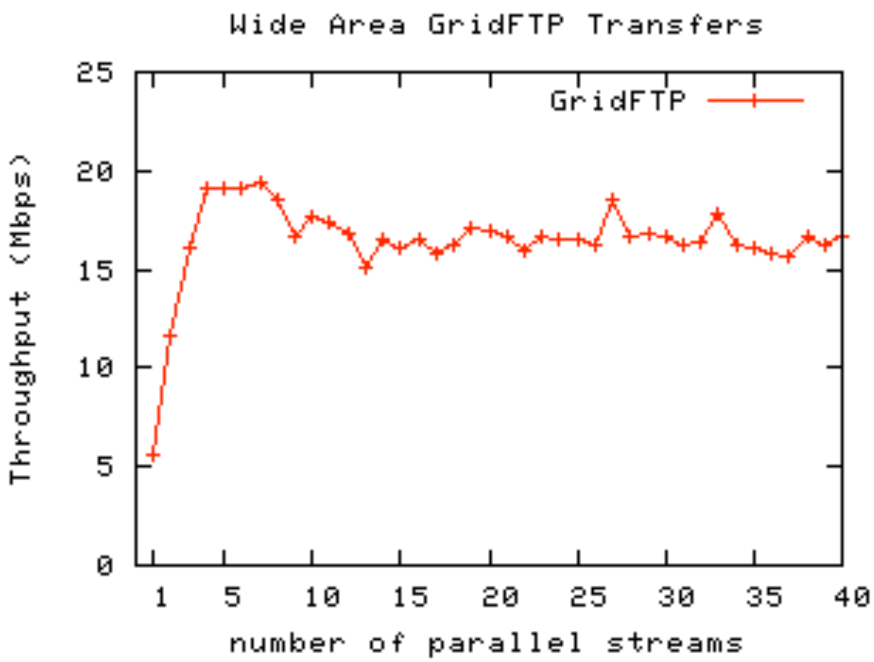
- Introduction
- Parallel Stream Optimization
- Buffer Size Optimization
- Combined Optimization of Buffer Size and Parallel Stream Number
- Conclusions

Parallel Stream Optimization

For a single stream, theoretical calculation of throughput based on MSS, RTT and packet loss rate:

$$Th <= \frac{MSS}{RTT} \frac{c}{\sqrt{p}}$$

For n streams?

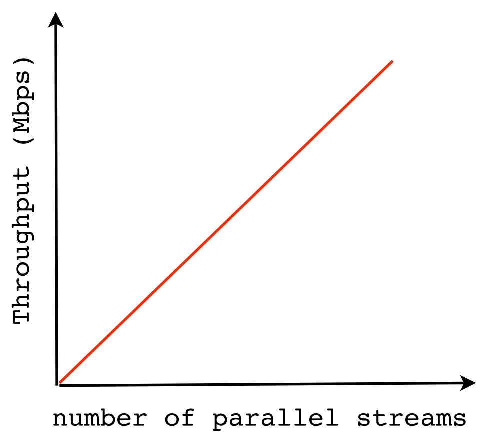


Previous Models

Hacker et al (2002)

An application opening *n* streams gains as much throughput as the total of *n* individual streams can get:

$$Th_n <= \frac{MSS \times c}{RTT}\left(\frac{n}{\sqrt{p}}\right)$$

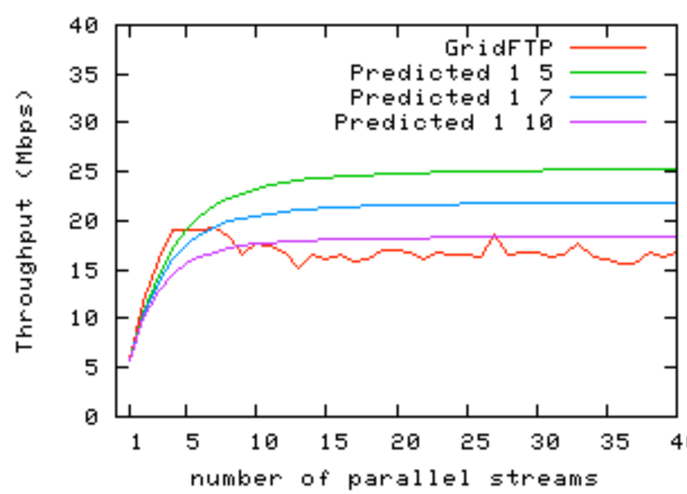


Dinda et al (2005)

A relation is established between *RTT*, *p* and the number of streams *n:*

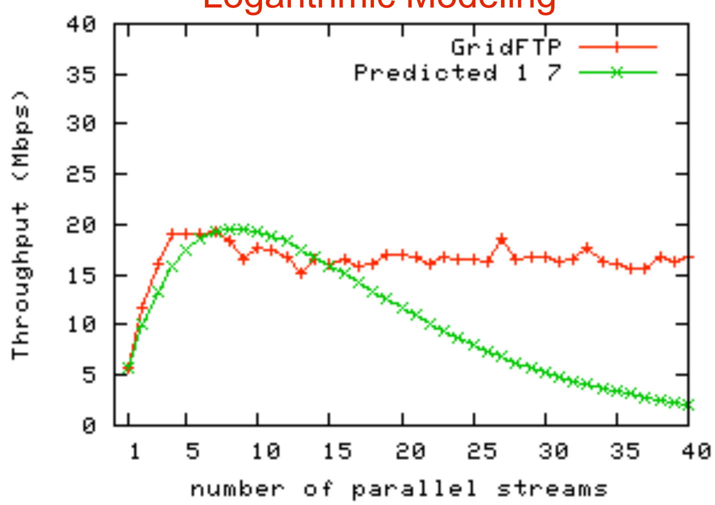
$$p'_n = p_n \frac{RTT_n^2}{c^2 MSS^2} = a'n^2 + b'$$

$$Th_n = \frac{n}{\sqrt{p'_n}} = \frac{n}{\sqrt{a'n^2 + b'}}$$

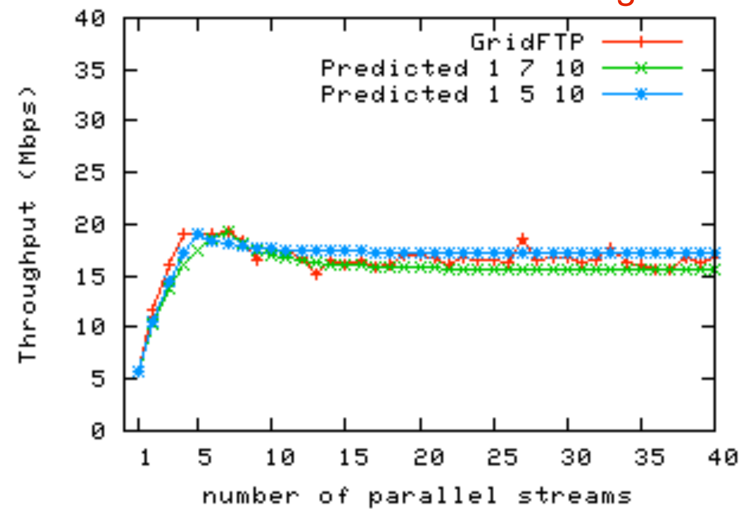


Kosar et al Models

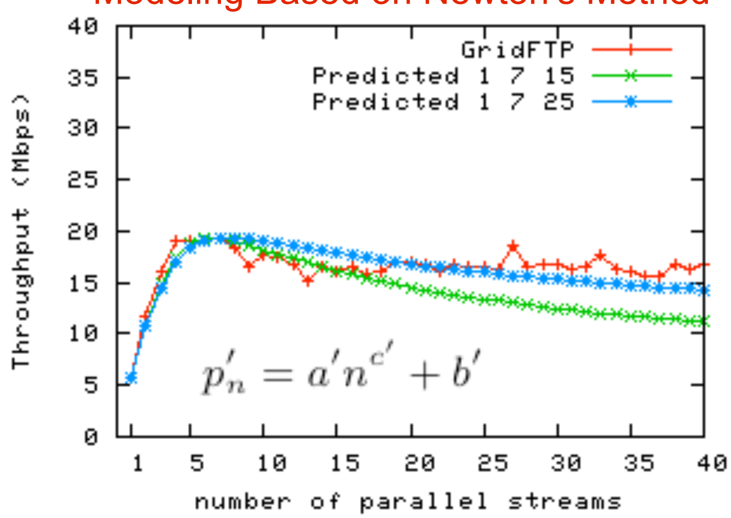
Logarithmic Modeling



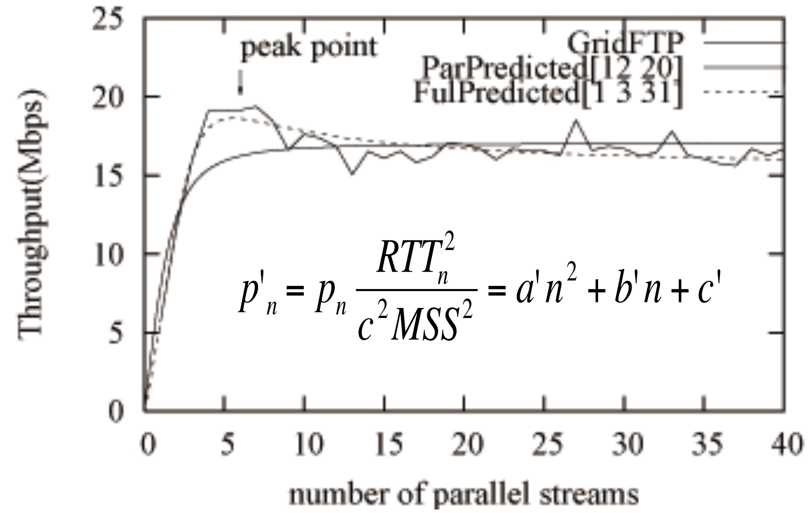
Break Function Modeling



Modeling Based on Newton's Method

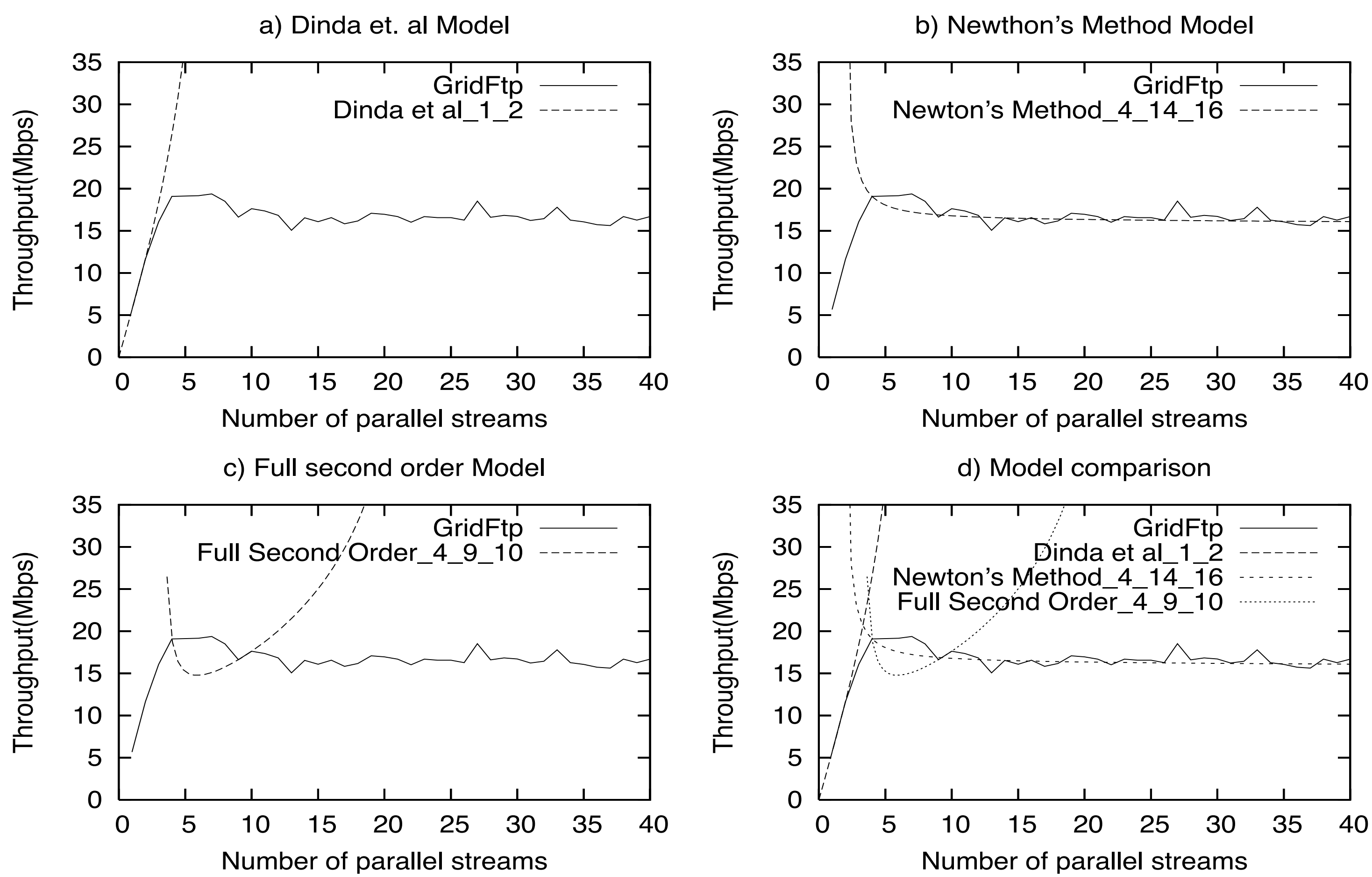


Modeling Based on Full Second Order



It is not a perfect World!

- The selection of point should be made intelligently otherwise it could result in mispredictions



Delimitation of Coefficients

- Pre-calculations of the coefficients of a’, b’ and c’ and checking their ranges could save us for elimination of error rate
- Ex: Full second order
  - a’ > 0
  - b’ < 0
  - c’ > 0
  - 2c’ + b’ > 1

$$p'_n = p_n \frac{RTT_n^2}{c^2 MSS^2} = a'n^2 + b'n + c'$$

Selection Algorithm

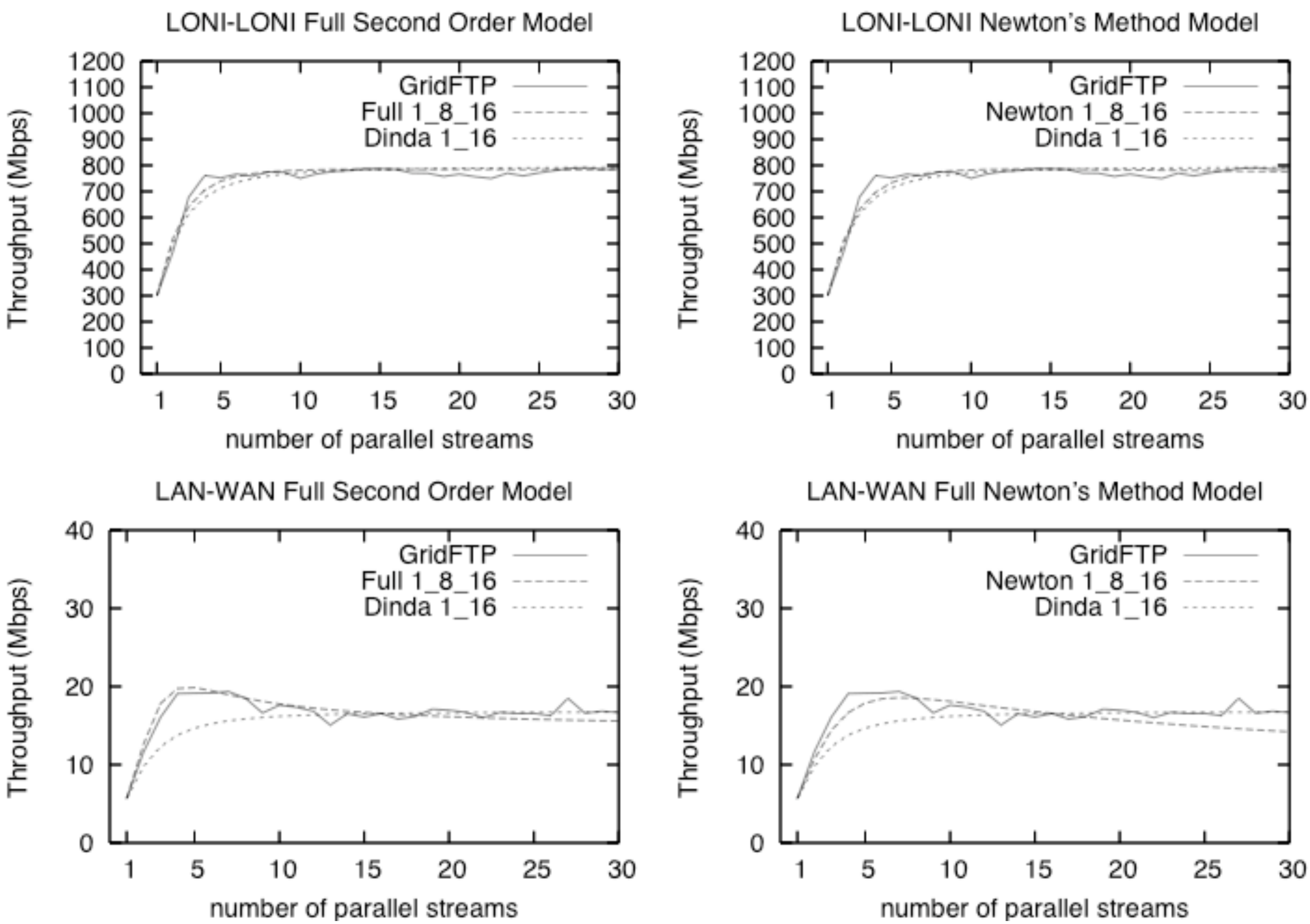
ExpSelection(T)

▷ **Input:** T
▷ **Output:** O[i][j]
Begin
  $accuracy \leftarrow \alpha$
  $i \leftarrow 1$
  $streamno1 \leftarrow 1$
  $throughput1 \leftarrow T_{streamno1}$
  $O[i][1] \leftarrow streamno1$
  $O[i][2] \leftarrow throughput1$
  do
    $streamno2 \leftarrow 2 * streamno1$
    $throughput2 \leftarrow T_{streamno2}$
    $slop \leftarrow \frac{throughput2 - throughput1}{streamno2 - streamno1}$
    $i \leftarrow i + 1$
    $O[i][1] \leftarrow streamno2$
    $O[i][2] \leftarrow throughput2$
    $streamno1 \leftarrow streamno2$
    $throughput1 \leftarrow throughput2$
  while $slop > accuracy$
End

BestCmb($O, n, model$)

▷ **Input:** O, n
▷ **Output:** $a, b, c, optnum$
Begin
  $\overline{err_m} \leftarrow init$
  for $i \leftarrow 1$ **to** $(n-2)$ **do**
    for $j \leftarrow (i+1)$ **to** $(n-1)$ **do**
      for $k \leftarrow (j+1)$ **to** n **do**
        $a', b', c' \leftarrow$ CalCoe($O, i, j, k, model$)
        if $a', b', c' are\ effective$ **then**
          $\overline{err} \leftarrow \frac{1}{n} \sum_{t=1}^{n} |O[t][2] - Th_{pre}(O[t][1])|$
          if $\overline{err_m} = init \,||\, err < err_m$ **then**
            $err_m \leftarrow err$
            $a \leftarrow a'$
            $b \leftarrow b'$
            $c \leftarrow c'$
          end if
        end if
      end for
    end for
  end for
  $optnum \leftarrow$ CalOptStreamNo($a, b, c, model$)
  return $optnum$
End

Points Chosen by the Algorithm



Buffer Size Optimization

- Buffer size affects the # of packets on the fly before an ack is received
- If undersized
  - The network can not be fully utilized
- If oversized
  - Throughput degradation due to packet losses which causes window reductions
- A common method is to set it to Bandwidth Delay Product = Bandwidth x RTT
- However there are differences in understanding the bandwidth and delay

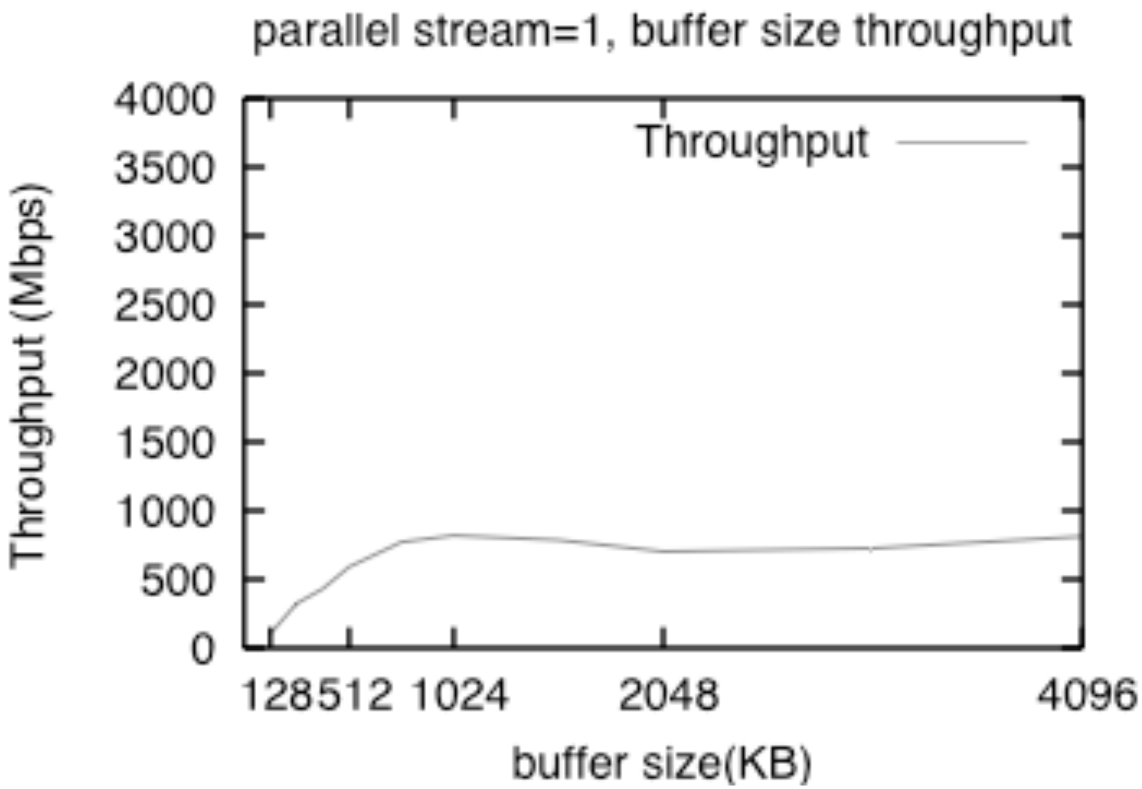
Bandwidth Delay Product

- BDP Types:
  - BDP1 = C x RTT_{max}
  - BDP2 = C x RTT_{min}
    - C -> Capacity
  - BDP3 = A x RTT_{max}
  - BDP4 = A x RTT_{min}
    - A -> Available bandwidth
  - BDP5 = BTC x RTT_{ave}
    - BTC -> Average throughput of a congestion limited transfer
  - BDP6 = B_{inf}
    - B_{inf} -> a large value that is always greater than window size

Existing Models

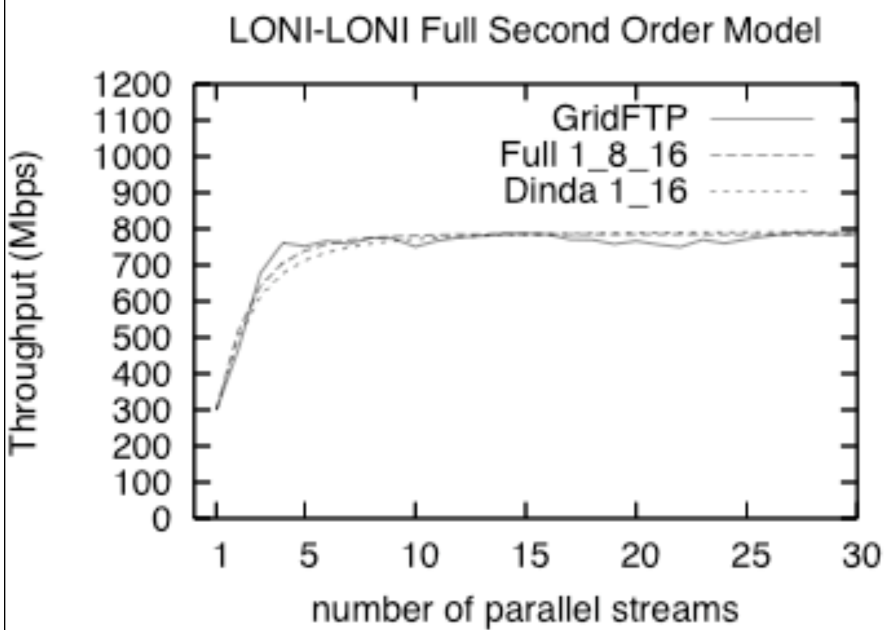
- Disadvantages of existing optimization techniques
  - Requires modification to the kernel
  - Rely on tools to take measurements of bandwidth and RTT
  - Do not consider the effect of cross traffic or congestion created by large buffer sizes
- Instead, can perform sampling and fit a curve to the buffer size graph

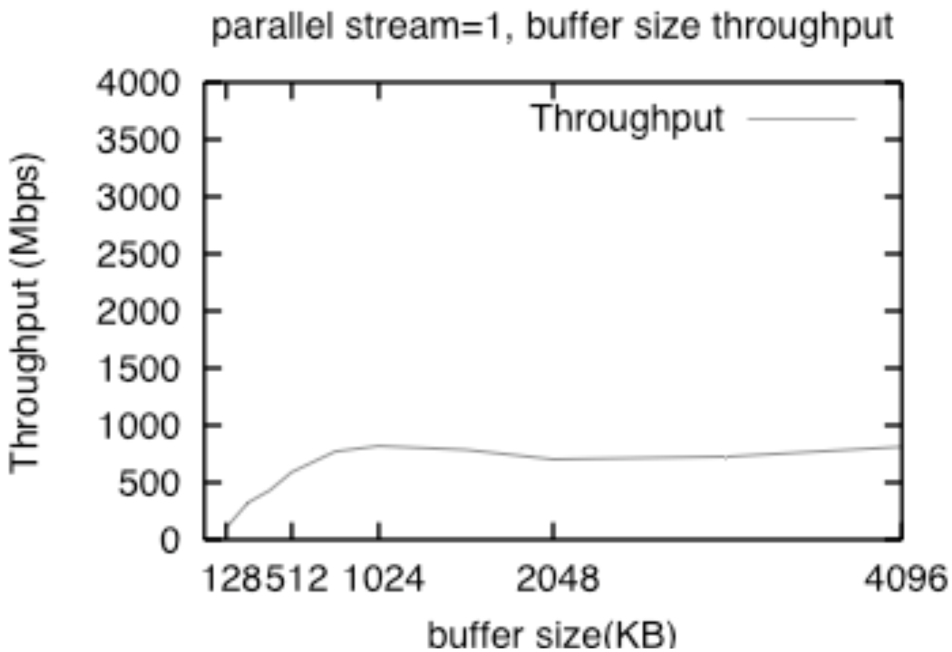
Buffer Size Optimization



- Throughput becomes stable around 1M buffer size

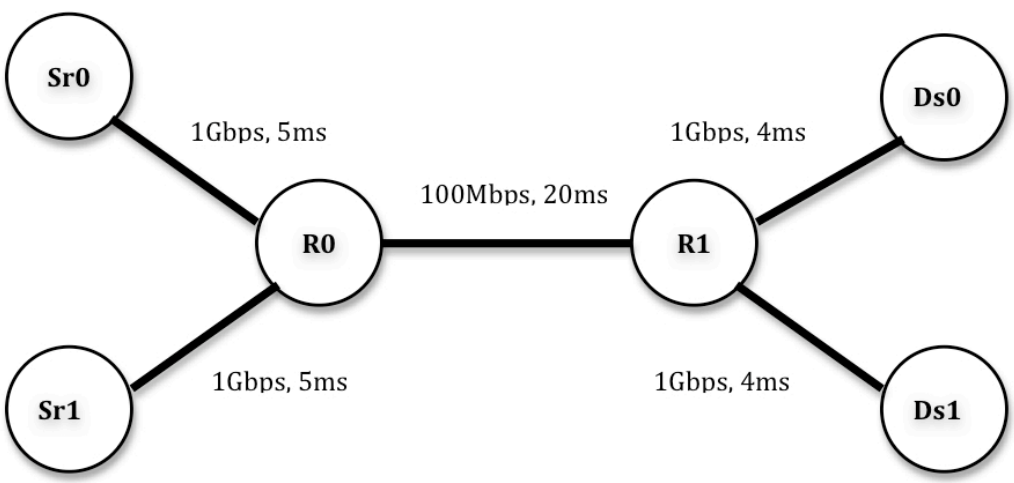
Combined Optimization





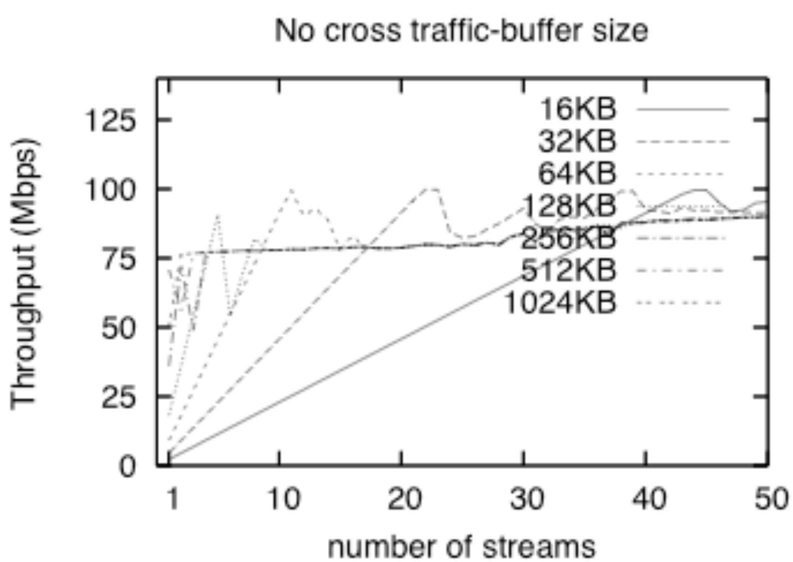
Balancing: Simulations

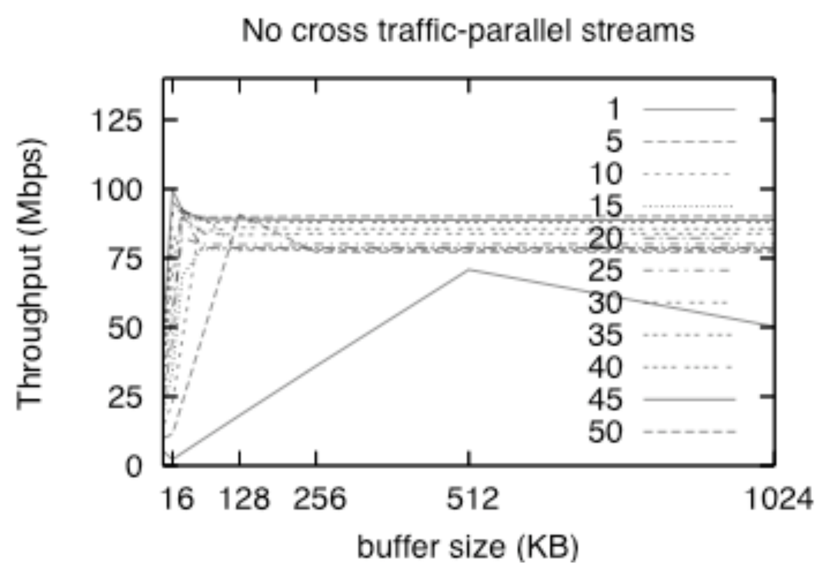
- Simulator: NS-2
- Range of different buffer sizes and parallel streams used
- Test flows are from Sr1 to Ds1 where cross traffic is from Sr0 to Ds0



1 - No Cross Traffic

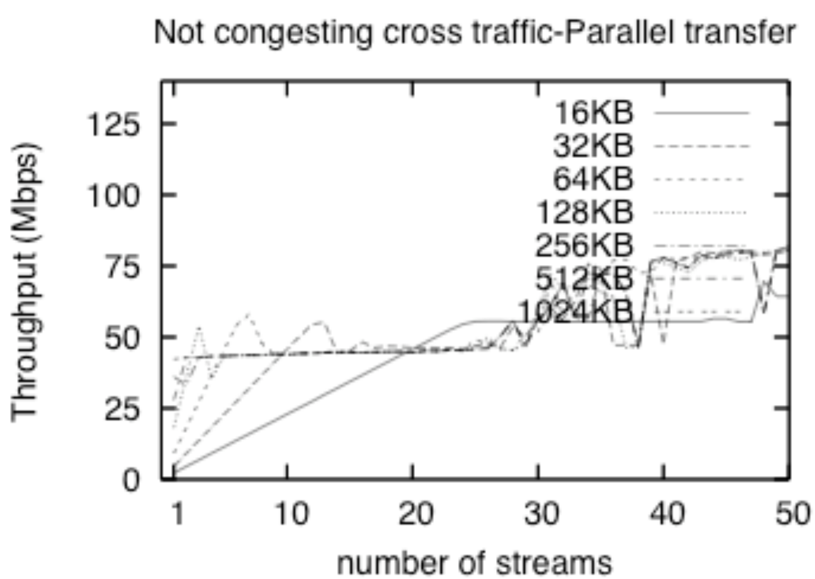
- Increasing the buffer size pulls back the parallel stream number to smaller values for peak throughput
- Further increasing the buffer size causes a drop in the peak throughput value

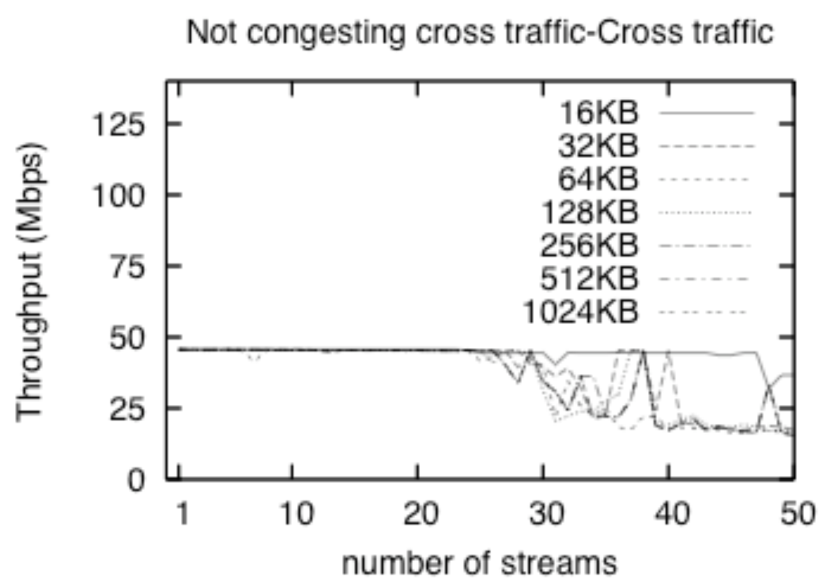




2 - Non-congesting Cross Traffic

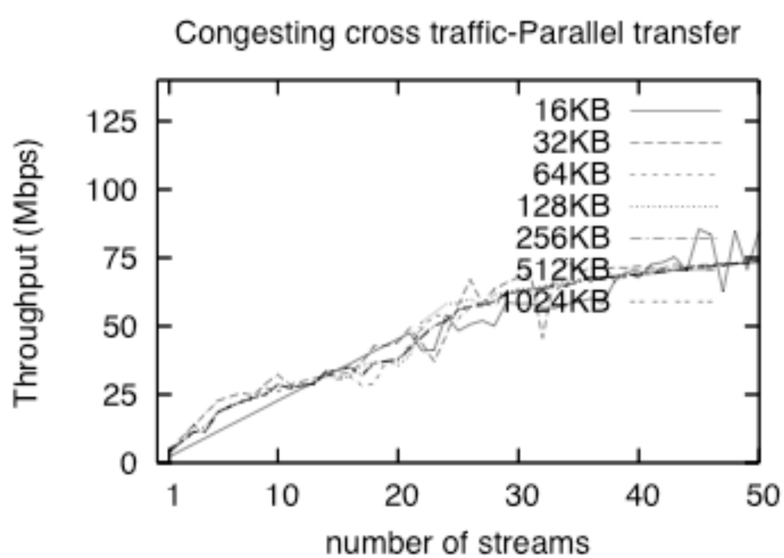
- 5 streams of 64KB buffer size as traffic
- Similar behavior as no traffic case until the capacity is reached
- After the congestion starts the fight is won by the parallel flows of which stream number keeps increasing

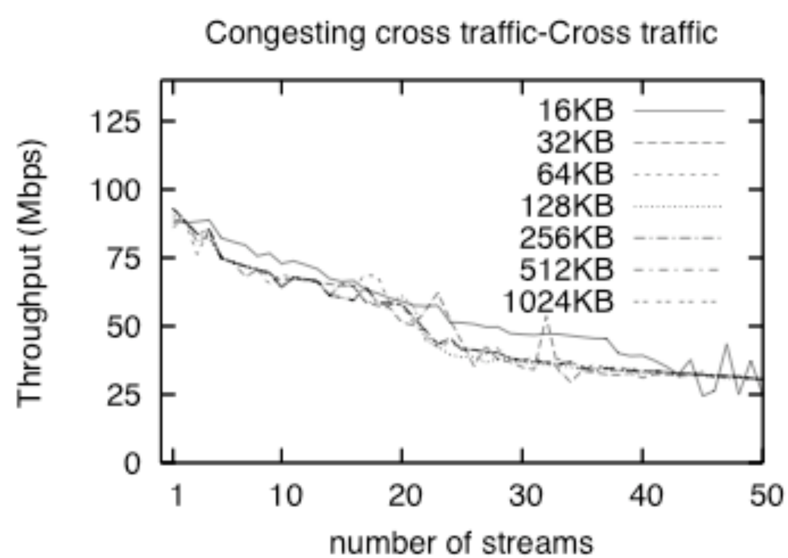




3 - Congesting Cross Traffic

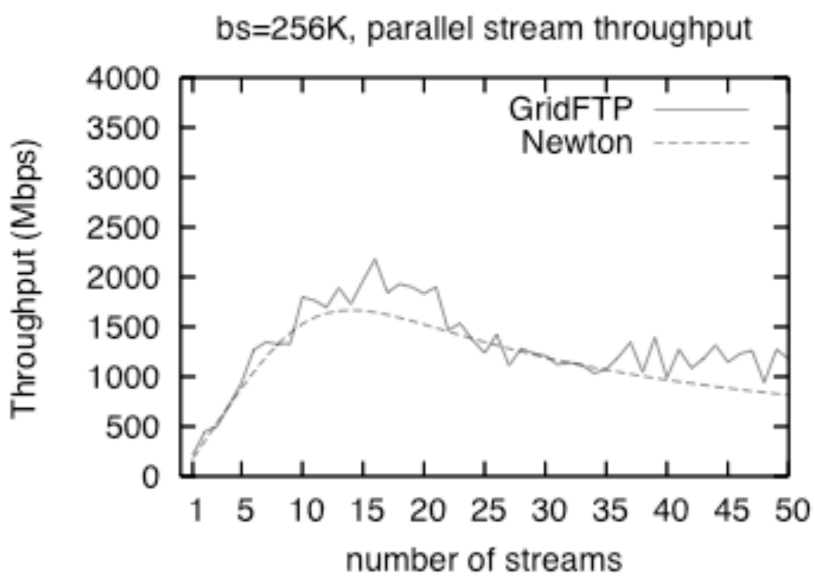
- 12 streams of 64KB buffer size traffic
- No significant effect of buffer size
- As the number of parallel streams increases the throughput increases and cross traffic throughput decreases

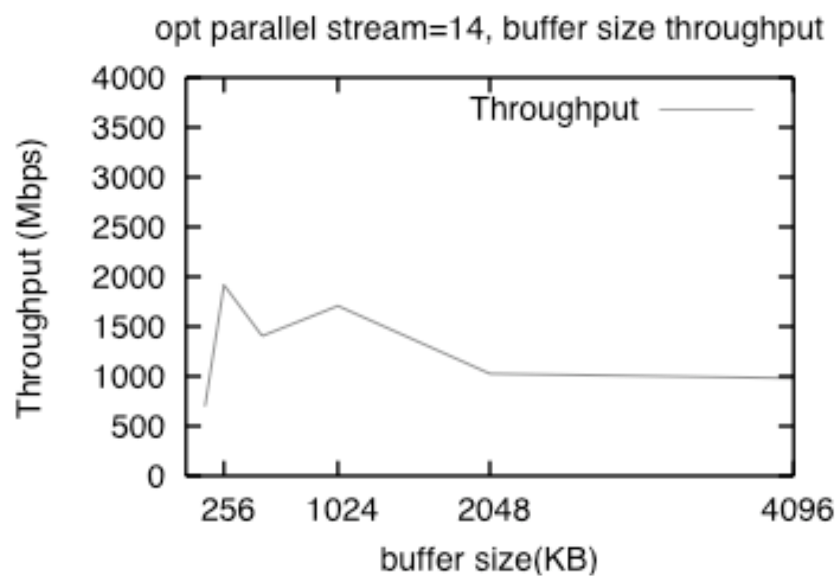




Experiments on 10Gbps Network

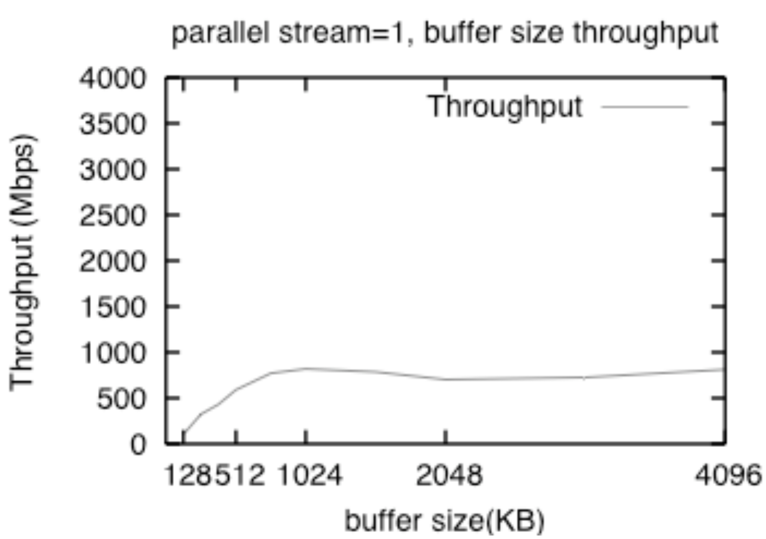
- Approach 1: Tune # of streams first, then buffer size
  - Optimal stream number is 14 and an average peak of 1.7 Gbps is gained
  - Optimal buffer size = 256

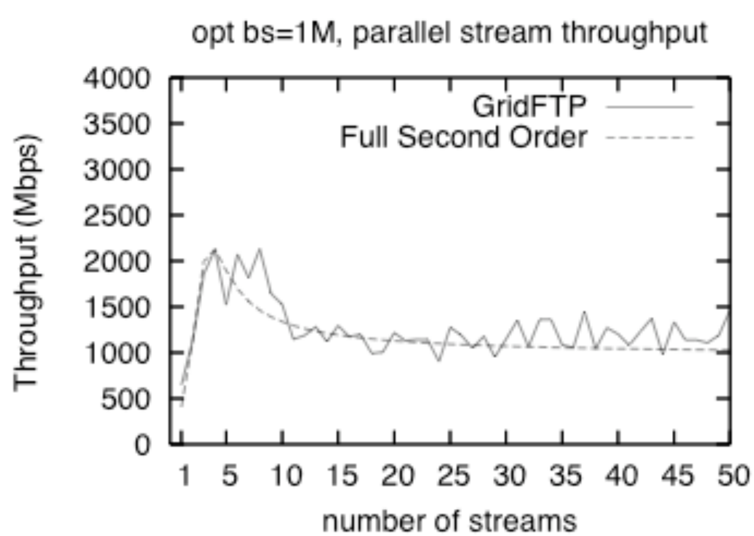


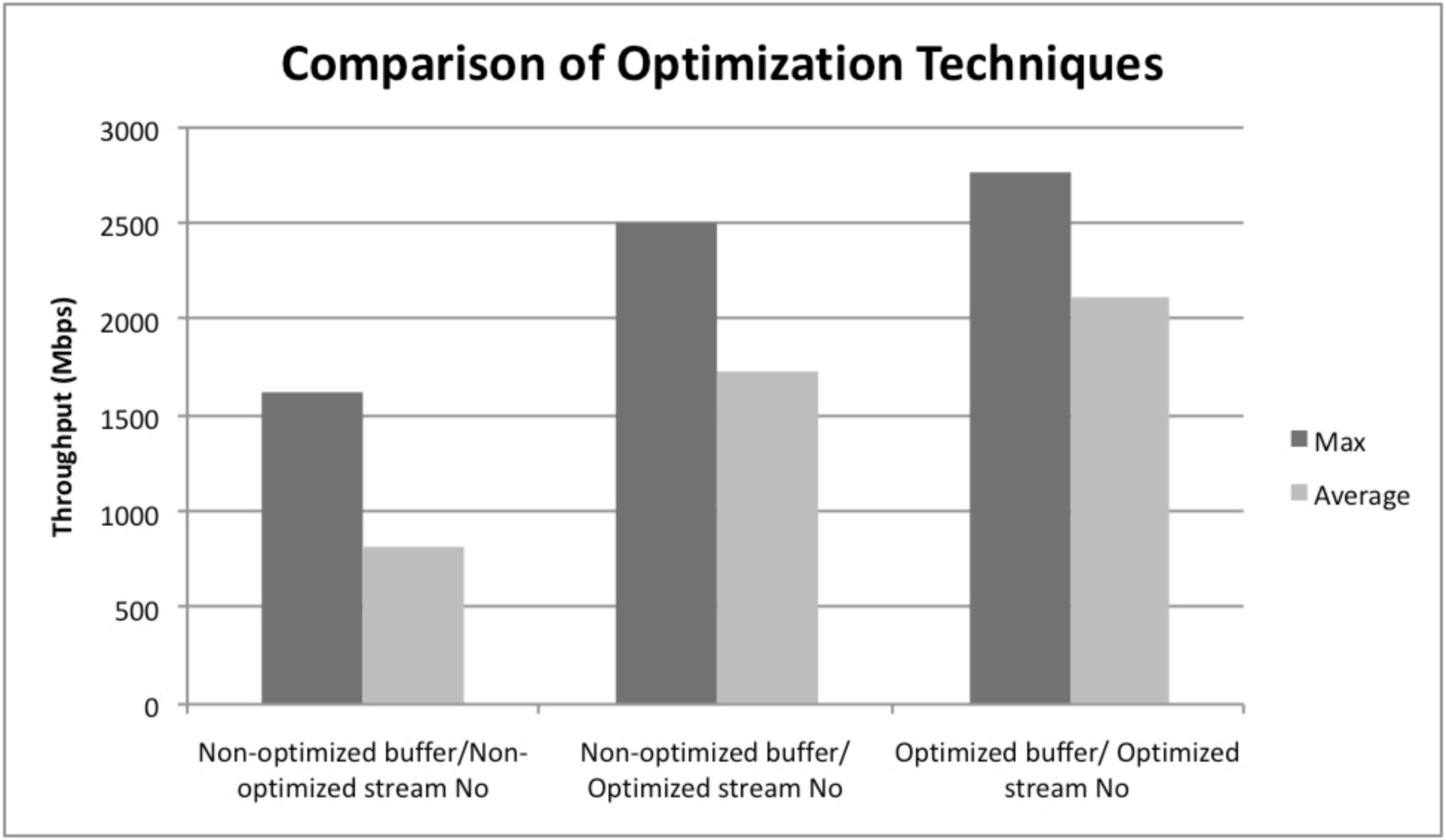


Experiments on 10Gbps Network

- Approach 2: Tune buffer size first, then # of streams
  - Tuned buffer size for single stream is 1M and a throughput of around 900 Mbps is gained
  - Applying the parallel stream model, the optimal stream number is 4 and an average of around 2Gbps throughput is gained



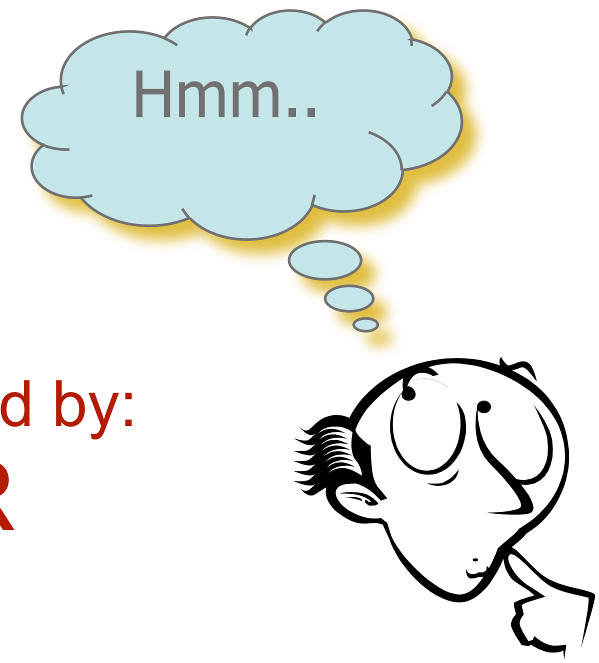




Comparison of Optimization Techniques
Throughput (Mbps)
3000
2500
2000
1500
1000
500
0
Non-optimized buffer/Non-optimized stream No
Non-optimized buffer/ Optimized stream No
Optimized buffer/ Optimized stream No
Max
Average

Conclusions and Future Work

- Tuning buffer size and using parallel streams allow improvement of TCP throughput at the application level
- Two mathematical models (Newtons & Full Second Order) give promising results in predicting optimal number of parallel streams
- Early results in combined optimization show that using parallel streams on tuned buffers result in significant increase in throughput



This work has been sponsored by:

NSF and LA BoR

For more information

Stork: http://www.storkproject.org

PetaShare:http://www.petashare.org