End-to-End Network Performance Monitoring for Dispersed Computing

Quynh Nguyen, Pradipta Ghosh, and Bhaskar Krishnamachari
Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, USA
Email:{quynhngu, pradiptg, bkrishna}@usc.edu

***Abstract*—With a growing demand for computationally intensive applications with widely distributed data sources, there is an increased need for dispersed computing systems that can schedule computation on cloud nodes across a network. A key challenge in dispersed computing is the need to characterize the end to end network performance for data transfer between compute points and measure it at run-time so that optimized computation scheduling decisions can be made. To address this challenge, we empirically study file transfer times between geographically dispersed cloud computing points using SCP (secure copy). We show, to our knowledge for the first time, that the end to end file transfer latency experienced by this widely-used protocol is better modelled to have a quadratic dependency on the file size (instead of a simple linear dependency that would be expected if the network were treated as an bit-pipe with a deterministic average bandwidth). We incorporate this observation into the design of a real-time network profiler for dispersed computing that determines best fit quadratic regression parameters between each pair of nodes and reports them to the scheduler node. Our end to end network quadratic latency profiler has been released as a key part of an open source tool dispersed computing profiler called DRUPE, and also as part of a DAG-based dispersed computing scheduling tool called CIRCE.**

I. Introduction

Over last decade, we have witnessed a rapid growth in the demand of the computationally intensive applications such as image processing and voice recognition, in the end user devices such as a cellphone or a car dashboard. In the era of Internet of Things (IoT), with the availability of very low cost (as low as $\approx$ \$5) micro-computers such as Intel Edison and Raspberry Pi, there is a growing interest to perform computations on the edge devices (partly or fully) rather than sending the data to a central cloud for processing, to avoid unnecessary network overhead and delays. The cutting edge field of **Dispersed Computing** is focused on leveraging all the available computation resources in the communication path from an end device to a cloud (including processing capable routers) for timely and optimized processing of the data by jointly optimizing the computation costs and the communication overhead costs. To this end, our research is focused on developing a dispersed computing system that would real-time monitor the network traffics and available computation capable node resources (which we will refer to as a "compute node" or a "worker node" in this paper) to optimally distribute the execution of a networked set of tasks which can be represented as a Directed Acyclic Graph (DAG) task graph.

This material is based upon work supported by Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001117C0053. The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

As the core idea of dispersed computing involves dispersing a set of networked task into a set of geographically distributed compute nodes, one of the key requirements is a real-time profiling of the network characteristics such as file transfer delay along with a running database of available compute resources descriptions. For highly computation intensive tasks such as image processing, the amount of data required to be transferred between different compute nodes is significant (in the order of hundreds of Kilobytes) and therefore the data transfer time among different compute nodes can be a bottleneck in the processing of a DAG based Task graph. To this end, we are interested in a) understanding how network transfer latencies behave as a function of data/file sizes and b) building a tool that can monitor and tell us about the function over time.

The key contribution of this work is that we are the first (to our knowledge) to demonstrate via a measurement study of a widely used file transfer protocol (SCP) that expected file transfer latencies are well modeled as a quadratic function of the transferred file size instead of a simple linear function model. In our experiments, a quadratic curve-fit yields less than 5.4% mean absolute percentage error compared to 9.2% error with a linear curve-fit. The quadratic curve suggests that there is an additional penalty incurred by larger files - possibly because over longer time-spans the transport layer is more likely to see severe loss events resulting in lower average bandwidth. This latter conjecture remains to be further verified in future work. Secondly, we incorporate this observation into the design of a distributed network monitoring system where nodes across a computing network periodically take pairwise measurements, curve fit them using the quadratic regression, and send the parameters to a central/scheduler node. This network monitoring system is implemented as part of an open source profiler tool called DRUPE (DispeRsed compUting ProfilEr, https://github.com/ANRGUSC/DRUPE) as well as a DAG-based dispersed computing scheduler tool called CIRCE (CentralIzed Runtime sChedulEr, https://github.com/ANRGUSC/CIRCE).

Since the late 90's, investigating and predicting end-to-end latency has been an important topic for the network researchers. In 1999, Barford and Crovella [1] tried to measure world wide web performance by analyzing file transfer latency statistics (mean and standard deviations of packet delay, packet loss, etc.) over combinations of different server load, network load and file sizes (1KB, 20KB, 500KB) by traceroute network tool. In 2002, Sharma *et al.* [2] raised the question whether file size alone can help predict wide-area transfer time. While exploring HTTP traffic over the specific data set of web caching, the authors concluded that for small transfers of up to 30KB, there is virtually no correlation between file size and transfer time; and for larger files, file size and transfer time are increasingly well correlated, but using file size alone for prediction of transfer time is not highly accurate. In [3], the authors integrated multiple sources of data sets (GridFTP server data transfers log, Network Weather Service probe data for bandwidth estimation and iostat disk throughput data to measure disk behavior) as well as multiple kinds of predictors (mean-based, median-based, autoregressive techniques based, and regression models based) to obtain predictions of file transfer times between a storage system and a client. They concluded that multivariate predictors along with several combinations of data sets could help to improve the accuracy in predicting bulk data transfer time. In 2006, Guang Tan investigated the effects of bandwidth asymmetry on short-lived transfers by looking into different phases of TCP (Connection Establishment Phase, the Slow Start Phase and the Steady State Phase) and presented total transfer latency versus number of packets during a bulk transfer [4]. In 2009, Larsen *et al.* looked into the end-to-end latency between application the standard Gb Ethernet [5]. In 2013, Gangam has presented Approximate Latency Estimator to support analysis of real time TCP flow (only for TCP segments at transport layer) [6]. To the best of our knowledge we the first to identify that the end to end file transfer time over a network can be well-modelled as a quadratic function of the file size alone. Moreover, we use this finding toward developing a network profiling system (implemented as part of open source tools for dispersed computing: DRUPE and CIRCE) to provide an online estimation of the end-to-end file transfer latency by periodically and systematically probing the communication paths of a geographically distributed dispersed computing cluster and taking into account historical data of previous file transfer duration.

The rest of the paper is organized as follows. Section II details the empirical findings of our measurement experiments on a carefully chosen geographically distributed set of nodes hosted by DigitalOcean; section III introduces the system which we have designed for network monitoring in dispersed computing systems. And finally, we present a concluding discussion in section IV.

II. Real measurement based modeling of the end-to-end file transfer latencies

In the section, we detail our experiment setup and the results toward modeling the end to end file transfer latencies in a dispersed computing testbed setup.

A. Testbed Setup

In our dispersed computing related experiments, we leverage a set of carefully chosen geographically distributed machines (droplets) hosted by DigitalOcean. For the network profiling experiments we use three DigitalOcean droplets summarized in Table I. The droplets are carefully chosen to be located on three different continents to model the end of end latencies in a truly dispersed computing platform where the compute nodes are spread across the world.

TABLE I: List of droplets used for our experiment

Droplet's Host Name	**Region**
ubuntu-512mb-fra1-01	France (FRA)
ubuntu-2gb-blr1-01	Bangalore (BLR)
ubuntu-2gb-nyc2-01	New York (NYC)

B. File transfer time measurements

Method: We perform different sets of end to end file transfer time measurement experiments as follows:

- Experiment 1 (Single Time Slot) : From each node to every other node, we send 10 files of each of the sizes 1KB, 10KB, 100KB, 1MB, 10MB and log the time taken in each case. We carefully schedule this so that the node transmissions does not overlap with each other. We also interweave the file sizes to minimize chance of correlation bias. We perform this experiment multiple time during different randomly chosen time of the day to account for the temporal variability in the internet traffic with each experiment duration being 20 minutes.
- Experiment 2 (Multiple Slots): In this experiment, we repeat the experiment from experiment 1 once every 20 minutes for three 20 minute time slots, 4 hours apart from each other. We also repeat this experiment over different times of the day. This allows us to see the impact of temporal variability of the internet traffic over a day.

We use the well-known file transfer protocol called Secure Copy (SCP) for transferring the files. Next, we present and analyze our findings from these experiments.

1) Single Time Slot: In Figure 1, we present the results from Experiment 1 of pairwise file download latency measurements experiments for different file sizes over the six directional links among three servers located in France, New York, and Bangalore, respectively. The results are plotted as box whisker plots versus file size. Box whisker plot shows the lowest extreme (the smallest value), highest extreme (the largest value), the median (the middle value), the lower quartile (the median value of all data below the median), and the upper quartile (the median value of all data above the median) of the data. We can see that there are significant differences in the end to end performance of the communication paths between different nodes in the geographically dispersed experimental testbed. While the end-to-end file transfer time for FRA$\rightarrow$NYC (closest pair of nodes) ranges from 2s for 1 KB file to 4s for 10 MB file, the file transfer time from

BLR→NYC (farthest pair of nodes) is $\geq$ 3s for 1 KB file and $\approx$ 7s for 10 MB file. This suggests that the physical separation of the nodes significantly impacts the end to end file transfer time. Moreover, Figure 1 show that BLR→NYC communication has higher variability in end to end delay compared to the reversed link i.e., NYC→BLR. This suggests that the file transfer performance varies with the direction of communication between two compute nodes. Therefore, the direction of communication should be taken into account as well for optimal dispersed scheduling of a DAG based task graph.

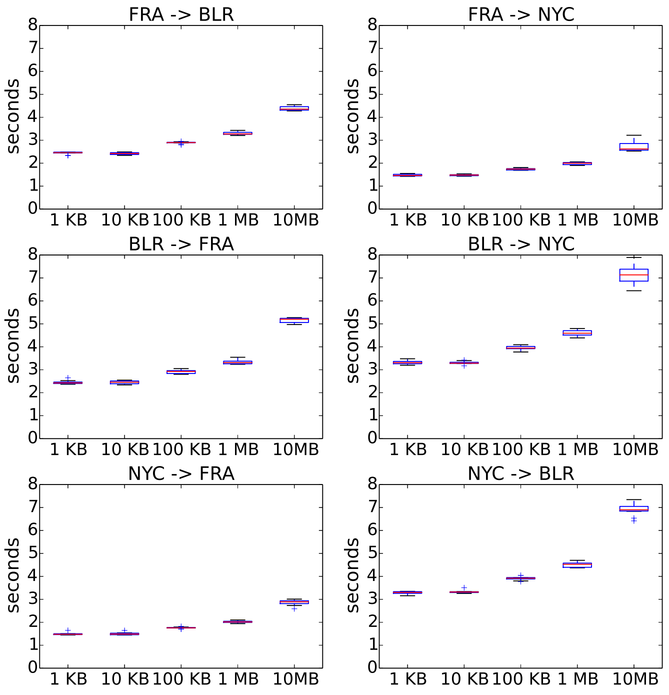


Fig. 1: File transfer time (single time slot)

2) Multiple 4 Hours Apart Time Slots: We obtain the aggregated average transfer time during different 20 minutes time slots of Experiment 2. Figure 2 presents the corresponding results for the first and third time slots. Yet again, the trend of significant difference in end to end file transfer delay performance due to geographical diversity and direction of communication among the testbed nodes is observed. And we observe that there is some temporal variation, particularly with respect to tail latencies.

All these experiments prove that for a truly geographically distributed set of nodes in a dispersed computing platform, the file transfer end-to-end delay performance varies significantly based on the relative geographic location of the pair of communicating nodes as well as the direction of the communication, and thus need to be accounted for optimal scheduling of pipelined/networked jobs in a dispersed computing system.

C. Other Network Characteristics

In Section II-B, we presented the end to end file transfer characteristics of a geographically distributed set of nodes. In this section, we present our results regarding other network characteristics such as round trip time (RTT) and maximum TCP bandwidth. To this end, we employ the following tools.

- IPERF: A tool to measure maximum TCP bandwidth, allowing the tuning of various parameters and UDP characteristics. Iperf reports bandwidth, delay jitter, datagram loss.
- PING: A tool to estimate Round Trip Time (RTT)
- OWAMP: A command line client application and a policy daemon used to determine one way latency between hosts.

For a concise presentation, we summarize the outcomes of these experiments in Table II. Table II clearly shows that the bandwidth (BW) mean, variance, and jitter are significantly different for different links in our test environments. The BW Mean is highest between FRA-NYC node pair and lowest between BLR-NYC node pair. This is justified as FRA-NYC node pair is the geographically closest pair of nodes and NYC-BLR is the geographically farthest pair of nodes in our experiment testbed. There is no loss of data in any of the tests which is justified by the use of TCP [7] as the transport layer protocol for the file transfers. The round-trip times between each pairs of nodes e.g., NYC→BLR and BLR→NYC, are identical for obvious reasons. However, one-way latency, bandwidth mean and jitter of a link is different from the reverse link (i.e, the link with switched origin and destination). This verifies that the direction of communication is indeed an important optimizing parameter in a dispersed computing setup where the a set of tasks are connected via a directional flow of data between them.

TABLE II: Network statistics results

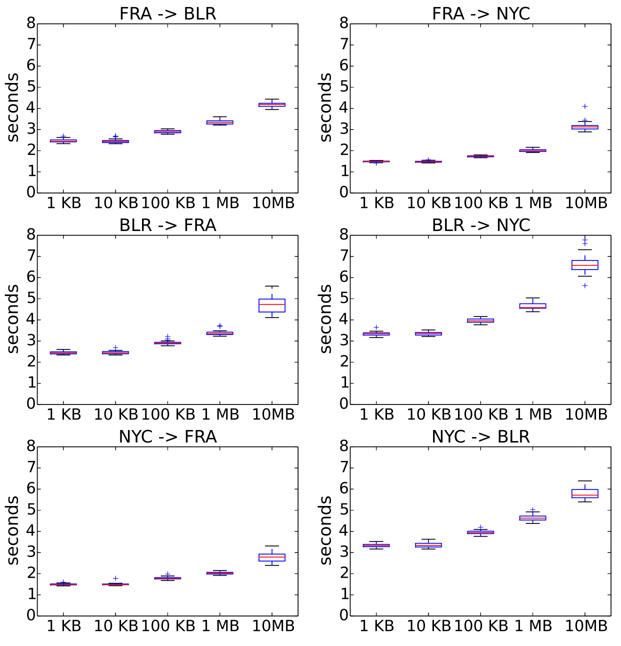
IPERF	BLR (Server)	NYC (Server)	FRA (Server)
BLR (Client)	N/A	BW Mean: 49.52 Mbps BW Std: 13.11 Jitter: 0.026 ms Loss: 0%	BW-Mean: 94.0 Mbps BW std: 25.068 Jitter: 0.915 ms Loss: : 0%
NYC (Client)	BW Mean: 59.68 Mbps BW Std: 14.44 Jitter: 0.033 ms Loss: 0%	N/A	BW-Mean: 108.18 Mbps BW std: 31.78 Jitter: 1.327 ms Loss: : 0%
FRA (Client)	BW Mean: 93.94 Mbps BW Std: 19.95 Jitter: 0.047 ms Loss: 0%	BW-Mean: 120.55 Mbps BW std: 26.65 Jitter: 0.053 ms Loss: : 0%	N/A

PING(RTT)	BLR (Server)	NYC (Server)	FRA (Server)
BLR (Client)	N/A	220.254 ms	145.819 ms
NYC (Client)	221.039 ms	N/A	85.544
FRA (Client)	145.869 ms	85.526 ms	N/A

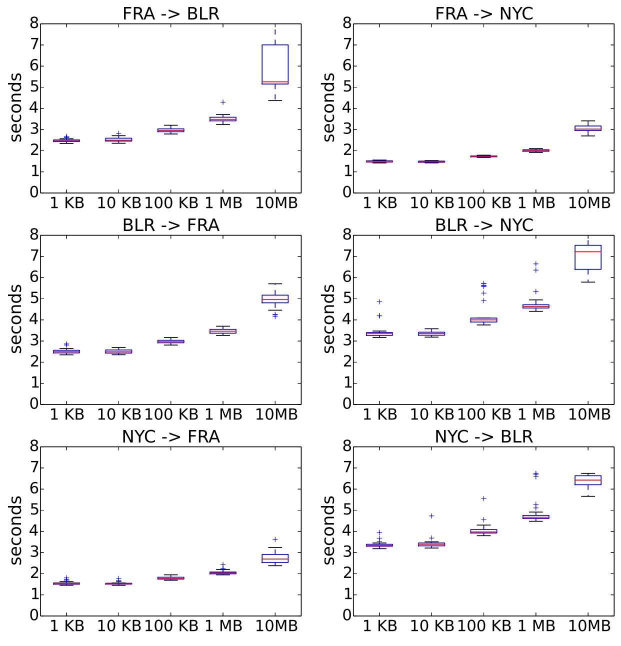
OWPING	BLR (Server)	NYC (Server)	FRA (Server)
BLR (Client)	N/A	Median delay = 112 ms Jitter = 0.2 ms	Median delay = 96.4ms Jitter = 1.1 ms
NYC (Client)	Median delay = 95.6 ms Jitter = 0.2 ms	N/A	Median delay = 38.2 ms Jitter = 2.2 ms
FRA (Client)	Median delay = 53.2 ms Jitter = 0.1 ms	Median delay = 44.9 ms Jitter = 0.4 ms	N/A

D. Quadratic regression in profiling network communication

In this section, we use the collected data to find a mapping function between the file size and the end to end file transfer delay. To this end, we employ three different classes of



(a) Time slot 1



(b) Time slot 3

Fig. 2: File transfer time over multiple 4 hours apart time slots

TABLE III: Regression error (Mean Square Error - MSE (in ms^2) & Mean Absolute Percentage Error - MAPE)

Error Type	**BLR → NYC**	**NYC → BLR**	**BLR → FRA**	**FRA → BLR**	**FRA → NYC**	**NYC → FRA**
MSE - Linear	178643.1	135661.1	70302.0	77093.8	27790.5	34521.1
MSE - Quadratic	85012.3	53971.8	27434.1	24075.2	11007.0	17850.1
MAPE - Linear	9.2 %	8.35 %	8.14 %	8.75 %	8.3 %	9.2 %
MAPE - Quadratic	5.4 %	4.7 %	4.79 %	4.6 %	4.97 %	5.4 %

regression lines. The first class of regression lines are linear with respect to the file size (f) and the bandwidth as follows: $t = x + y + f/z$ where t is the transfer time, x is the constant overhead, y represents the measured mean RTT, and z is the measured average end-to-end bandwidth. The second class also uses linear regression but with the file size (f) as the only variable: $t = a + b \cdot f$ where a and b are empirically determined constants. Lastly, we employ a quadratic regression with respect to the file size (f): $t = p + q \cdot f + r \cdot f^2$ where p, q, r are empirically determined constants. In Figure 3, we plot the empirical file transfer time versus file size and their corresponding linear and quadratic regression line for given file size, or combination of bandwidth and file size. Figure 3 clearly shows that the quadratic regression offers the best fit to approximate file transfer time for varying file sizes.

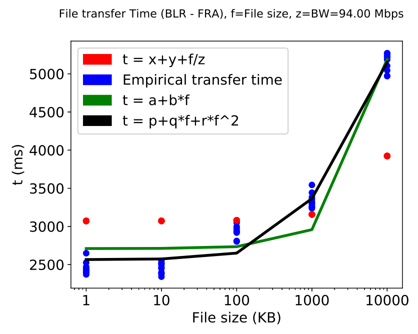
Table III summarizes the regression error for linear and quadratic cases. The results in Table III demonstrate clear improvement in error statistics when we use quadratic regression over the linear approach.

III. DRUPE Network monitoring tool

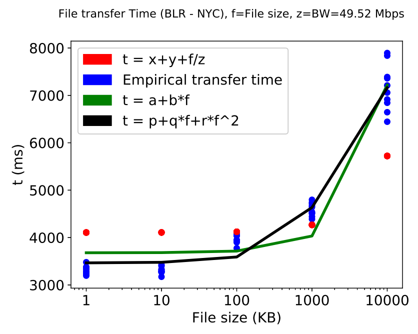
Based on our empirical findings on how to represent file transfer latency as a function of file size, we have designed a system for centrally collecting information about the pairwise end to end network performance for a set of dispersed computing points. A naive system may simply determine a single end to end latency metric that is independent of file size. A slightly more sophisticated system would implement a linear latency function by assuming a deterministic average bandwidth for the end to end link so that the file transfer latency would be modeled as being linearly proportional to the file size divided by that average bandwidth. However, as we have shown in the above, empirically a better approach is to model the end to end file transfer latency as a quadratic function of file size.

In the system, described below in more detail, each worker compute node of the dispersed computing system calculates the parameters of the best fit quadratic curve over a given window of measurements to each other node:

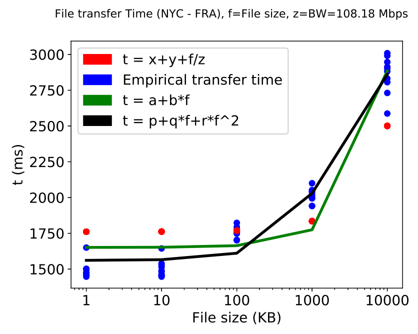
- Central node:
  - It performs profiler initialization and scheduling in the worker nodes
  - It periodically collects and stores all updated quadratic parameters information of all the nodes in the local database server in a circular buffer.
  - It has an on-demand function to query the local database server and calculate the expected latency for a given pair of worker/compute nodes and a file size.
- Worker/Compute Nodes:



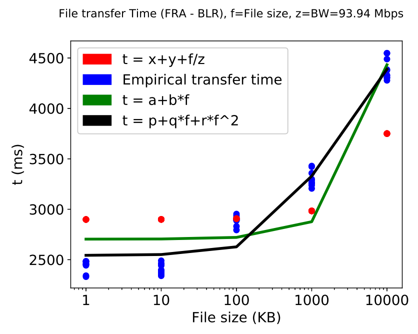
(a)



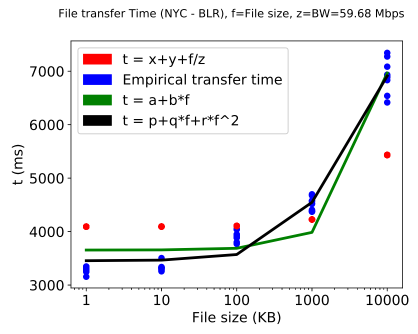
(b)



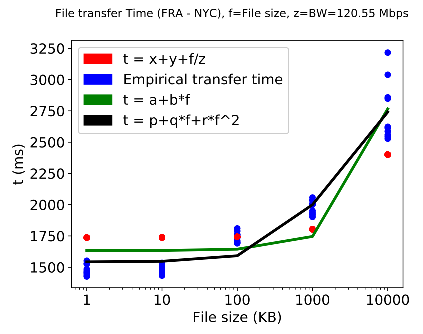
(c)



(d)



(e)



(f)

Fig. 3: File transfer time prediction

– Measurement process: Each worker node periodically (with a time period τ which we choose to be 1 minute) transfers a file of a random size from a range of $f_{min} - f_{max}$ (we choose f_{min} and f_{max} to be 1KB and 10MB, respectively) to every other worker node, and logs how long it takes to transfer that size file in a circular buffer of max size k.
– Regression process: A worker node looks at the last k measurements (or up to the last k measurements at the start when there are not enough measurements) and calculate the parameters of the best fit quadratic curve and sends those parameters to the central node.

We have implemented our network measurement system as the key part of an open source dispersed computing profiler called DRUPE as well as a dispersed computing scheduler called CIRCE.

IV. Conclusion

In this paper, we demonstrated via a set of real world measurement experiments that the end to end file transfer time in a dispersed computing environment can be modelled as a quadratic function of the file size. Based on this finding, we developed a network profiler (implemented as part of open source tools called DRUPE and CIRCE) that perform periodic profiling of the communication links in a dispersed computing cluster and performs a quadratic fit of the data and sends the information back to a scheduler. In future work, we would like to investigate and verify our conjecture that there is an additional penalty incurred by larger files because over longer time-spans the transport layer is more likely to see severe loss events resulting in lower average effective bandwidth. If that conjecture does not hold, we will consider alternative possibilities including implementation limitations particular to SCP. Further, we will perform a large scale experiments to substantiate our findings further.

References

[1] Paul Barford and Mark Crovella. Measuring web performance in the wide area. *SIGMETRICS Perform. Eval. Rev.*, 27(2):37–48, September 1999.
[2] Manish Sharma and John W. Byers. How Well Does File Size Predict Wide-Area Transfer Time? In *Proceedings of the 2002 Globecom Global Internet Symposium*, Taipei, Taiwan, October 2002.
[3] Sudharshan Vazhkudai and Jennifer M. Schopf. Using regression techniques to predict large data transfers. *Int. J. High Perform. Comput. Appl.*, 17(3):249–268, August 2003.
[4] Guang Tan and Stephen A. Jarvis. Prediction of short-lived tcp transfer latency on bandwidth asymmetric links. *Journal of Computer and System Sciences*, 72(7):1201 – 1210, 2006.
[5] S. Larsen, P. Sarangam, and R. Huggahalli. Architectural break-down of end-to-end latency in a tcp/ip network. In *Computer Architecture and High Performance Computing, 2007. SBAC-PAD 2007. 19th International Symposium on*, pages 195–202, Oct 2007.
[6] Sriharsha Gangam, Jaideep Chandrashekar, Ítalo Cunha, and Jim Kurose. *Estimating TCP Latency Approximately with Passive Measurements*, pages 83–93. Springer Berlin Heidelberg, 2013.
[7] Theodore S Rappaport. *Wireless communications: principles and practice*. prentice hall PTR New Jersey, 1996.