Dynamically Tuning Level of Parallelism in Wide Area Data Transfers*

Esma Yildirim†
Department of Computer Science & CCT
Louisiana State University
Baton Rouge, LA 70803
USA
esma@cct.lsu.edu

Mehmet Balman
Department of Computer Science & CCT
Louisiana State University
Baton Rouge, LA 70803
USA
balman@cct.lsu.edu

Tevfik Kosar‡
Department of Computer Science & CCT
Louisiana State University
Baton Rouge, LA 70803
USA
kosar@cct.lsu.edu

ABSTRACT

Using multiple parallel streams for wide area data transfers may yield much better performance than using a single stream, but overwhelming the network by opening too many streams may have an inverse effect. The congestion created by excess number of streams may cause a drop down in the throughput achieved. Hence, it is important to decide on the optimal number of streams without congesting the network. Predicting this 'magic' number is not straightforward, since it depends on many parameters specific to each individual transfer. Generic models that try to predict this number either rely too much on historical information or fail to achieve accurate predictions. In this paper, we present a set of new models which aim to approximate the optimal number with least history information and lowest prediction overhead. We measure the feasibility and accuracy of these models by comparing to actual GridFTP data transfers. We also discuss how these models can be used by a data scheduler to increase the overall performance of the incoming transfer requests.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Modeling Techniques, Measurement Techniques, Performance Attributes
; C.2.2 [**Network Protocols**]: Protocol Verification, Applications; C.2.3 [**Network Operations**]: Network Monitoring

General Terms

Design, Performance, Measurement

---

*

†

‡

DADC'08, June 24, 2008, Boston, Massachusetts, USA.

Keywords

Parallelism, Parallel TCP, Data transfer, GridFTP, File Transfer, Measurement

1. INTRODUCTION

The effective throughput that is achievable by an application given the capacity of a network and current load is a study area in which many different methods have been developed either in high or low-level. As an example of low level methods, many different transport protocols have been developed [5, 8, 4, 10] and also tuning the existing protocol parameters [12, 11, 9] gave promising results. Among those protocols, TCP is the most widely used one and different versions of TCP are implemented on this purpose for increasing efficiency in achievable transfer rate. On the high level, other techniques are found just by using the existing underlying protocol. Opening parallel streams is one way of doing that and is widely used in many application areas from data-intensive scientific computing to live multimedia, and peer-to-peer paradigms.

It is shown that parallel streams achieve high throughput by mimicking the behavior of individual streams and get an unfair share of the available bandwidth [15, 12, 2, 7, 5, 8, 13]. On the other hand, using too many simultaneous connections reaches the network on a congestion point and after that threshold, the achievable throughput starts to drop down. Unfortunately it is difficult to predict the point of congestion and is variable over some parameters which are unique in both time and domain. The prediction of the optimal number of streams is hence very challenging and almost impossible without some parameters of current network conditions such as available bandwidth, RTT, packet loss rate, bottleneck link capacity and data size.

It is easier to optimize the network parameters in the case of bulk data transfers. After some point in time, enough historical information is gathered and the parameters are optimized according to the condition of the network. This kind of optimization has been already used with Stork data scheduler [11] before. However for individual data transfers the situation becomes harder. Instead of relying on historical information, the transfer should be optimized based on instant feedback. In our case, this optimization is achieving optimal number of parallel streams. However, an optimization technique not relying on historical data in this case must not cause overhead of gathering instant data that is larger than the speed up gained with multiple streams for a

particular data size.

The studies that try to find the optimal number of streams are so few and they are mostly based on approximate theoretical models [6, 14, 1, 4, 10]. They all have specific constraints and assumptions. Also the correctness of the model is proved with simulation results mostly. Hacker et al. find that the total number of streams behaves like one giant stream that transfers in capacity of total of each streams' achievable throughput [6]. However, this model only works for uncongested networks. Thus it does not give an answer to at what point the network will be congested. Another study [4] declares the same theory but develops a protocol which at the same time provides fairness. Dinda et al. models the bandwidth of multiple streams as a partial order two equation and needs two different throughput measurement of different stream numbers to predict the others [14]. Yet in another model the total throughput always shows the same characteristics [1] depending on the capacity of the connection as the number of streams increases and 3 streams are sufficient to get a 90% utilization. A new protocol study [10] that adjusts sending rate according to calculated backlog presents a model to predict the current number of flows which could be useful to predict the future number of flows.

In this study, we present theoretical models that are used to predict the behavior of parallel streams and discuss their assumptions in application. We also applied those models and measured their accuracy against actual GridFTP transfers with the improvements we have made. It has been observed that GridFTP transfers show different characteristics in existence or absence of congestion due to opening too many streams and none of the existing models could predict this behavior. With the improvements we have made on the existing models, with little historical or instant information we were able to predict this behavior and our tests have proven the correctness of our approach. With this information at hand any data scheduler may optimize its each individual transfer with little information to be gathered.

2. EXISTING MODELS

In this section we present the existing models for predicting the behavior of parallel TCP and discuss their advantages, shortcomings and ease of applicability.

2.1 Hacker et al Model

In this model, the achievable throughput depends on three parameters: Round trip time, packet loss rate and maximum segment size. The maximum segment size is in general IP maximum transmission unit (MTU) size + TCP header. Round trip time is the time it takes for the segment to reach the receiver and for a segment carrying the generated acknowledgment to return to the sender. The packet loss rate is the ratio of missing packets over total number of packets. The following formula represents an upper bound on the achievable throughput [6]:

$$Th <= \frac{MSS}{RTT}\frac{c}{\sqrt{p}} \tag{1}$$

Th represents the achievable throughput by an application opening single stream, MSS represents the maximum segment size, RTT is the round trip time, p is the packet loss rate and c is a constant. Of MSS, RTT, and p variables, packet loss is the most dynamic one while MSS is the most static one. According to TCP's AIMD congestion avoidance algorithm, packet loss is considered as an indication of congestion and the window size of TCP is halved immediately. Then it increases linearly until another packet loss event occurs. This gives a saw tooth behavior to the TCP congestion window [6]. Equation 2 gives the total packets sent between two packet loss events where W represents the window size. According to this relation, W and accordingly the throughput is inversely related to the square root of p.

$$\frac{1}{p} = \frac{W}{2} + (\frac{W}{2} + 1) + ... + W = \frac{3}{8}W^2 \tag{2}$$

An application opening n connections actually gains n times the throughput of a single connection, assuming all connections experiencing equal packet losses. Also the RTTs of all connections are equivalent since they most likely follow the same path. In that case, Equation 1 is rearranged for n streams as:

$$Th_n <= \frac{MSS \times c}{RTT}\left(\frac{n}{\sqrt{p}}\right) \tag{3}$$

However this equation accepts that packet loss is stable and does not increase as the number n increases. At the point the network gets congested, the packet loss rate starts to increase dramatically and the achievable throughput starts to decrease. So it is important to find that point of knee in packet loss rate. One possible way is to gather statistical data to apply a time series prediction of p, MSS and RTT. In addition to that, information regarding the number of parallel streams necessary to maximize throughput can be stored and this information can be used for a search algorithm. The requirement of that model is to gather data regarding RTT, MSS, and p. There are separate tools that give that information however in that study Web100 is used to collect it.

2.2 Dinda et al Model

The model presented in the previous section is only valid for uncongested networks and does not give us a relation between packetloss rates and the number of streams. Considering both p_n and RTT_n all depend on the number of streams, if a model can be presented that computes this relationship with only small number of measurements, then throughput of n streams can be predicted. The model proposed by Dinda et al. tries to solve Equation 3 by computing the relationship between p, RTT, and n. MSS and c are considered as constants. The relationship is represented by a new variable p'_n which is equalized to a partial second order polynomial which is believed to be best suited [14]:

$$p'_n = p_n\frac{RTT_n^2}{c^2 MSS^2} = a'n^2 + b' \tag{4}$$

After placing p'_n in Equation 3, total throughput of n streams is calculated as follows:

$$Th_n = \frac{n}{\sqrt{p'_n}} = \frac{n}{\sqrt{a'n^2 + b'}} \tag{5}$$

a' and b' are parameters to be found by measurements. To be able to solve this equation, two achievable throughput measurements for two different parallelism levels are needed. The only possible way to find those throughput values is either use a tool that has the capability to do parallel transfers or to use information about past actual transfers.

$$Th_{n_1} = \frac{n_1}{\sqrt{a' n_1^2 + b'}} \tag{6}$$

$$Th_{n_2} = \frac{n_2}{\sqrt{a' n_2^2 + b'}} \tag{7}$$

By using the two equations above the values of a' and b' are calculated and placing them to Equation 5, the aggregate throughput for any number of streams can be calculated by the following formula:

$$Th_n = \frac{n}{\sqrt{\frac{\frac{n_2^2}{Th_{n_2}^2} - \frac{n_1^2}{Th_{n_1}^2}}{n_2^2 - n_1^2} \times n^2 + \frac{n_1^2}{Th_{n_1}^2} \times \frac{\frac{n_2^2}{Th_{n_2}^2} - \frac{n_1^2}{Th_{n_1}^2}}{n_2^2 - n_1^2} \times n_1^2}} \tag{8}$$

n_1 and n_2 are two parallelism levels of which achievable throughput needs to be measured and n is the parallelism level of which achievable throughput will be predicted. Besides using partial second order equations, they also use linear and full second order equations and based on experimental results partial second order equation is chosen. The experimental results claim that the parallelism level correctly can be predicted, however in those results the aggregated throughput of connections increases and then becomes stable but never falls down. So it will be interesting to analyze the behavior of that function when the throughput starts to decrease.

By predicting the achievable throughput of n streams it should be decided which stream number will be used. For this, a percentage is given as input for cross traffic effect. In that case, we may find the optimal number of streams that will effect the cross traffic by x%. For measurement of sample throughput values for two different parallelism levels that will be fed into the function, they use Iperf which has the capability to open parallel connections. The predicted results however again compared to actual Iperf measurements and the behavior of actual protocol transfers are not considered.

2.3 Altman et al Model

The third model bases its correctness on the fact that opening too many connections imposes a processing overhead and leaves smaller bandwidth to other flows. Only 3 streams are enough to get a 90% link utilization. It is noted that for single stream situation, a TCP connection may utilize only 75% of the bandwidth because of its AIMD (additive increase and multiplicative decrease) property. Only with parallel streams we could increase this ratio. Because only a small subset of the connections undergoes multiplicative decrease during congestion, this helps parallel streams to recover quickly. Assuming only one of the connections undergoes a multiplicative decrease at any time, the following formula can be used to predict the throughput [1]:

$$Th_n = C\left(1 - \frac{1}{1 + \frac{1+B}{1-B}n}\right) \tag{9}$$

B is 1/2 for TCP connections as it decreases its window by half for the multiplicative decrease and C is the capacity of the link. There are two issues that need to be solved for that approach to be applied. First of all, we need to know the bottleneck link capacity for the overall connection to be able to produce correct results since this formula gives the throughput over a single link. Second, this formula gives the total number of streams that survive to get this aggregate throughput however if we need to know the additional number of streams to open then we must have an idea about how many other streams are using the link as cross traffic. We must find a way to determine the existing flow number.

2.4 Crowcroft&Oechslin Model

The model presented by Crowcroft and Oechslin[4] advocates the same findings with Hacker et al Model. However a protocol MulTCP is implemented which mimics the behavior of parallel TCP connections and a methodology is given for maintaining fairness by limiting the buffer sizes of connections. The throughput of n connections is represented by the following formula:

$$Th_n = \frac{\sqrt{2}\sqrt{n(n-1/4)}MSS}{RTT\sqrt{p}} \approx \frac{\sqrt{2}nMSS}{RTT\sqrt{p}} \tag{10}$$

As it can be seen, this formula is quite similar to the Hacker et al Model, except that it also gives an approximation of c constant.

2.5 Kola&Vernon Model

A new protocol implementation is given with the purpose of high bottleneck link utilization and low average backlog at the bottleneck link as well as maintaining fairness among all flows utilizing the bandwidth [10]. A target backlog is computed from the current measured bottleneck to achieve the stated goals. Four input parameters are needed for calculations: Capacity of the bottleneck link, Average and minimum round trip times and packet loss rate. With these inputs it is possible to derive the cross-traffic throughput and average number of flows sharing the bottleneck link.

Step 1 : Calculate backlog of a single flow. $b = S \times d$ where S is the sending rate and $d = RTT_{avg} - RTT_{min}$.

Step 2 : Calculate total backlog . $B = d \times 12/C$ in terms of packets of MTU size 1500 bytes. In this equation C is the capacity of the bottleneck link.

Step 3: Calculate link utilization U.

$$U \approx 2B/(2B+1) \tag{11}$$

Step 4: Calculate link utilization of a flow U_{flow} sending at rate S .

$$U_{flow} = S \times 12/C \tag{12}$$

Step 5: Calculate cross traffic throughput Th_{xt}.

$$Th_{xt} = (U - U_{flow}) \times C/12) \tag{13}$$

Step 6: Calculate average number of equivalent flows n sharing the bottleneck link.

$$n = B/b = (C \times U)/12S \tag{14}$$

If we know the average number of flows sharing the bottleneck link we can combine this with Altman et al Model. Say that we want a 98% link utilization and we know the capacity of the bottleneck link. We can calculate the optimal n and subtract the average number of flows sharing the link. So we find the number of additional streams to open.

$$n_{add} = n_{opt} - n \tag{15}$$

However the information for those calculations can be gained in the low level protocol layer. For example the sending rate can only be decided by the underlying protocol.

3. PROPOSED MODELS

In this section, we propose several model improvements both based on Hacker et al and Dinda et al Models and discuss their derivations and implications.

3.1 Modeling Packet Loss Rate

The shortage of Hacker et al Model is that there is no information on when the point of congestion will occur as a result of opening multiple streams. The reason of that conclusion is that the behavior of the packet loss rate as the number of streams increase is unpredictable. However, if we can find a model to characterize the packet loss rate, then we can use Equation 3 to calculate the throughput gained by n streams.

To achieve this, a methodology similar to the one in Dinda et al Model can be used. However this time we can not use a partial second order polynomial to model the packet loss rate, since it increases exponentially as the throughput increases logarithmically. By changing the places of Th_n and p_n in Equation 3 we get the following equation:

$$p_n = \frac{MSS^2 c^2}{RTT^2 Th_n^2} \tag{16}$$

In this case, we define a new variable Th'_n and correlate it to RTT, MSS, n and Th_n. Hacker et al. proves that throughput increases linearly in uncongested networks as the number of streams increases however this is not true for congested networks. The throughput achieved by n streams increases logarithmically so we related Th'_n to the following equation:

$$Th'_n = \frac{RTT_n^2 Th_n^2}{MSS^2 c^2} = a' n^{1/x} + b' \tag{17}$$

Ranging x between 2 and a greater number, we play how sharp the increase in packet loss will be after the point of congestion. By placing Th'_n in Equation 15 we get the following equation for the packet loss of n streams:

$$p_n = \frac{n^2}{Th'_n} \tag{18}$$

Considering we can gather the packet loss rates of transfers with two different stream numbers p_{n_1} and p_{n_2} we could find the values of a' and b'.

$$a' = \frac{\frac{n_2^2}{p_{n_2}} - \frac{n_1^2}{p_{n_1}}}{n_2^{1/x} - n_1^{1/x}} \tag{19}$$

$$b' = \frac{n_1^2}{p_{n_1}} - a n_1^{1/x} \tag{20}$$

After finding the value of p_n we could easily calculate the throughput value of n streams by using equation 2 in Hacker et al Model. However this value represents an upper bound on the throughput achieved. In section 5, we show that this calculation gives a better approximation to the throughput curve.

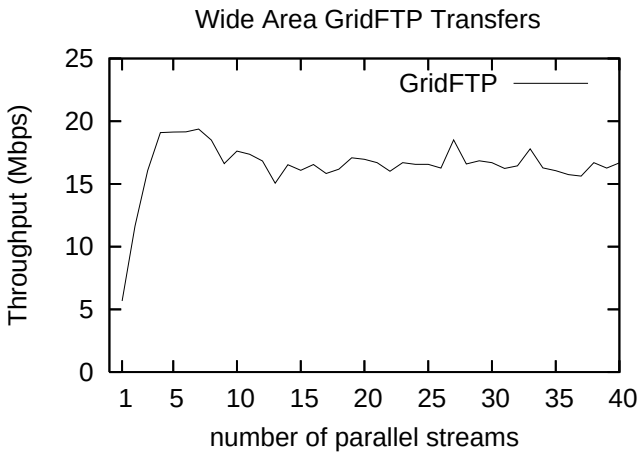

Figure 1: The aggregate throughput results of GridFTP transfers of a 512MB file over 155ms latency wide area network

3.2 Increasing Curve Fitting with More Information

The study in [14] shows that the characteristic of a throughput curve increases sharply then becomes stable until it reaches the capacity of the link therefore the model represented fits to the data presented. However in real experimental environment by using actual file transfer protocols(e.g. GridFTP) the results could be different from the proposed situation. Figure 1 presents a file transfer of 512MB with GridFTP over a wide area network using up to 40 parallel streams. As the number of streams increases , the throughput achieved also increases. However after some point the created congestion by opening too many streams causes a decrease. The peek point in this case gives us the optimum number of streams.

Instead of using two throughput measurements for two different parallelism level we plan to increase this information level into three measurement values. However the overhead of the gathering extra information must not surpass the actual speed up gained by opening parallel streams. In this case we may apply two different methodology to make use of the extra information. First the calculated a' and b' for using parallelism levels n_{12} and n_{13} are averaged either by using arithmetic, geometric or quadratic averaging methods. Then throughput can be calculated with the averaged values of a' and b'. Second, as we can see from Figure 1 the throughput curve acts as two different functions. Until reaching the peek point it acts as a certain function. However when falling down it shows a different characteristics. So instead of using a single function we could break the function into two. By using parallelism level n_1 and n_2 we could model a certain function and by using n_2 and n_3 we could model the second part. Passing between two functions can be a little sharp however this indicates that the optimal number of streams is certainly between n_2 and n_3. The result of experiments are further analyzed in section 5. In the next section, we present a model that smooths out this sharp transition between two functions.

3.3 Logarithmic Modeling of Throughput Curve

The relationship between p, RTT and n is modeled with a partial second order polynomial equation in[14]. However the proof of their correct type of equation is based on the experiments done. Hence in some of the cases a linear or full second order polynomial equations may give good result. But none of them could predict the knee point in the throughput curve as the number of streams increases.

We know that the packet loss rate increases exponentially however, we may not know the order of the equation to use. In this case instead of using a partial second order polynomial, we use an exponential equation whose order may change depending on the a' and b' parameters. The following equation is used to define the variable p'_n we have mentioned before:

$$p'_n = a' e^{b' n} \tag{21}$$

and

$$Th_n = \frac{n}{\sqrt{a' e^{b' n}}} \tag{22}$$

By using two throughput measurements Th_1 and Th_2 with different parallelism levels n_1 and n_2 the following values are calculated for a' and b':

$$a' = \frac{n_1^2}{Th_{n_1}^2 \times e^{b n_1}} \tag{23}$$

$$b' = \log_{e^{n_1 - n_2}} \frac{Th_{n_2}^2}{Th_{n_1}^2} \times \frac{n_1^2}{n_2^2} \tag{24}$$

3.4 Predicting Model Equation Order by Newton's Method

The discussion about the type of equation models led us to the conclusion that the order of the equation should be extracted dynamically. The logarithmic modeling of throughput that is explained in the previous section is able to make a smooth transition from the increasing to the decreasing part of the prediction curve. However as the stream number increases the prediction curve approaches to 0 and may not give an approximate prediction result to the actual throughput in the decreasing part of the curve. To be able to make a good prediction we have formulized the p'_n by addition of a new variable to predict the order as follows:

$$p'_n = a' n^{c'} + b' \tag{25}$$

In this case c' is the unknown order of the equation additional to a' and b'. Thus our throughput formulation becomes as:

$$Th_n = \frac{n}{\sqrt{a' n^{c'} + b'}} \tag{26}$$

Notice that to be able to solve this equation we need three measurements Th_{n_1}, Th_{n_2} and Th_{n_3} on the throughput curve for stream values n_1, n_2 and n_3. Also c' being to the power n makes the solving of the equation much harder. After several substitutions we come up with the following equations for a', b' and c':

$$\frac{n_3^{c'} - n_1^{c'}}{n_2^{c'} - n_1^{c'}} = \frac{\frac{n_3^2}{Th_{n_3}^2} - \frac{n_1^2}{Th_{n_1}^2}}{\frac{n_2^2}{Th_{n_2}^2} - \frac{n_1^2}{Th_{n_1}^2}} \tag{27}$$

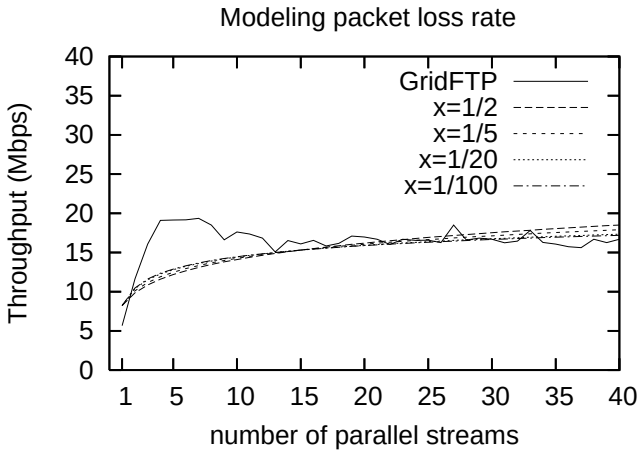

Figure 2: The throughput results of prediction based on packet loss modeling

$$a' = \frac{\frac{n_2^2}{Th_{n_2}^2} - \frac{n_1^2}{Th_{n_1}^2}}{n_2^{c'} - n_1^{c'}} \tag{28}$$

$$b' = \frac{n_1^2}{Th_{n_1}^2} - a n_1^{c'} \tag{29}$$

The derivation of a' and b' all depends on c'. To be able to solve the first equation we applied a mathematical root finding method called Newton's method. We revised the method to be suitable to our own problem:

$$c'_{x+1} = c'_x - \frac{f(c)}{f'(c)} \tag{30}$$

According to that method after $x+1$ iterations we are able to find a very close approximation to c'. Starting with a small number for c'_0 we continued to calculate through c'_{x+1}. The value of the most approximate c' depends on only $f(c)$, in this case the first equation above, and its derivative. After calculating a most approximate c' which is possible with only a few iterations, the value of a' and b' can easily be calculated.

4. COLLECTION OF INSTANT MEASUREMENT INFORMATION

The prediction models presented in the previous sections can be applied with the existence of little information on previous transfers or parameters regarding network such as RTT, MSS, packet loss rate and capacity. However, in absence of that information we may need to use a network measurement tool for predicting the achievable transfer rate of the network.

We have compared two popular network measurement tools, Iperf and NWS, in terms of many aspects including their accuracy in predicting certain protocol transfer rates for different data sizes and the overhead of making a good approximation. The overhead parameter is so important in the sense that the usage of such a tool for a prediction model

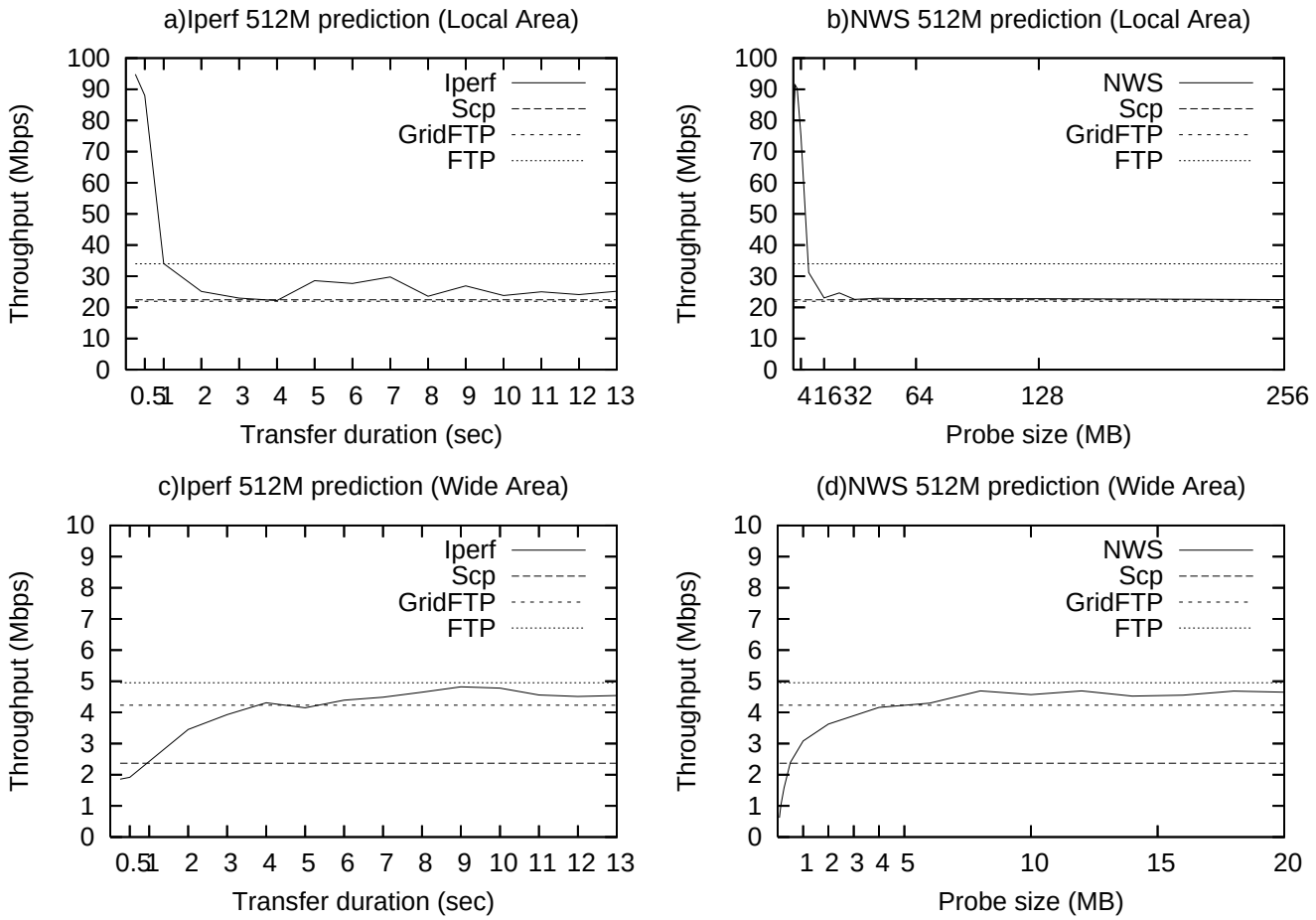

Figure 3: Accuracy results of Iperf and NWS for a 512MB file size over local and wide area networks

should not surpass the speed up we gained in opening an optimal number of parallel streams. Unfortunately there is a trade-off between the accuracy of a measurement and the overhead caused.

The experimental results showed that the most important parameter that affects the throughput accuracy with a trade-off of intrusiveness is the probe size. Figure 3.a and b show the accuracy of Iperf and NWS with respect to SCP, GridFTP and FTP in local area network. For both of the tools a certain amount of data is needed to be sent to merge into the actual protocol sending rate. For Iperf this size is 4-5 seconds of transfer duration and for NWS it is about 16MB of probe size to reach the transfer rate of GridFTP. Interestingly, the wide area measurement characteristics of both tools are quite different from the local area. In Figure 3.c and d, the transfer rate measured by both tools increase through transfer duration and probe size while in local area this behavior is vice versa. Again for a steady measurement, Iperf needs 5-6 seconds while NWS needs 5-6 MB probe size in the wide area. Those amounts are quite reasonable if we would like to predict a large file's transfer rate(e.g. 512MB), however it is not wise to predict the transfer rate of 16MB file by using a tool which needs to send quite the same amount.

The models presented in the previous sections needs 2-3 measurements of throughput of different parallelism levels. In this case, our goal is that the total overhead of those measurements will be negligible comparing the gain regarding to the parallelim which is quite possible for medium and large data files.

5. EXPERIMENTAL RESULTS

In this section, we present our wide area test results regarding GridFTP transfers and give prediction results of several methods used. The wide area test bed consists of two Linux machines, one is a Redhat workstation in the Center for Computation and Technology at Lousiana State University, the other belongs to a Suse cluster in University of Trento in Italy. Both machines have installed Globus Toolkit. Also for packet behavior measurements, we used a Linux network analyzer tool called Wireshark.

We have conducted GridFTP transfers with a file size of 512MB. The number of streams ranged between 1 and 40. The results presented in Figure 1 are the average of multi-iteration of tests. The actual file transfers by means of a protocol, in our case GridFTP, follows a different characteristics than presented in study [14]. In this case, the aggregate throughput achieved by parallel streams increases as the number of streams increases, after reaching its peek

it presents a wavy behavior, however further increasing of stream number results in a descent in throughput. The current models can not predict this unstable behavior.

Starting with the shortcomings of Hacker et al Model, the idea of the total aggregate throughput of parallel streams being equal to the throughput of a single stream times the number of streams seems to be proven in uncongested networks. Figure 1 shows that up to 3 streams this equivalence is true more or less, however after that point the linear ascent of throughput curve turns into a logarithmic one indicating the existence of congestion in the network. The primary reason of this behavior is the change in packet loss rate due to congestion. By modeling packet loss according to some measured packet loss rates of different parallelism levels taken from Wireshark with our presented model and applying Equation **??** we get a predicted packet loss rate and replace it in Equation 3 to calculate throughput. As a result, we get the following results presented in Figure 2. The difficulty of applying this model is that we need a lot of information like MSS, RTT and packet loss rate. Also the resultant value gives us an upper bound on throughput. As we can see from the figure instead of a linear increase which is not suitable with behavior of GridFTP results we can get a logarithmic increase which better suits assuming that packet loss rate increases exponentially as we have proposed in our model. By increasing the x value we could get a sharper knee and a flatter continuous behavior for the rest of the stream numbers.

Figure 6.a represents the prediction results of Dinda et al Model for different parallelism levels. The prediction curve calculated with throughput results of 1 and 5 parallel streams presents higher results than the ones calculated with 1 and 7, and also 1 and 10. The closest results given for low number of streams are 1 and 5, and also for high number of streams 1 and 10. All of the prediction curves have a logarithmic ascent behavior and then becomes stable for high number of streams but never fall down. The study in [14] mentions that best parallelism values to use are either 1-5 or 1-10. However we could only tell this after some experiences with the transfers. Instead of that we could try to increase our information level by using 3 parallelism levels. Another important point is that no matter which parallelism values we have used to predict the throughput we must get close results to the optimal number of streams. A more detailed analysis is given later in this section. According to our approach, we may detect the behavior change in the function by calculating average a' and b' values. The results of this model is given in Figure 6.b. The averaging results give that best stream number values to predict this curve is between 7 and 10. However we do not need to know exactly this number by using our averaging approach. The averaged curve gives a closer result to both the increasing and decreasing part of the GridFTP throughput curve.

In this case, we know that the throughput curve of GridFTP behaves like two different functions for ascending and descending behaviors. Hence we could use the information regarding three parallelism levels and model the curve by breaking into pieces. Figure 6.c shows the prediction results taken by 1-5-10 and 1-7-10 parallelism levels. We could see that the prediction curve almost perfectly suits the GridFTP curve. Although the transitions are a little sharp between the pieces of the curve that gives us an exact idea where the optimal parallel stream number lies. In another model improvement, we have modeled packet loss rate as an exponential function and hence throughput as a logarithmic function (shown in Figure6.d). The transition between the ascending and descending part of the throughput curve is smoothed out. However, the descending part of the throughput curve can not be predicted well as the curve approaches to 0 as the number of streams increases further. In our last approach we have solved the problems of both the 'break function' and 'logarithmic prediction' methods. Figure 6.e shows the results of the predicted throughput by using Newton's Method mentioned in 3.4. The best results are taken with 1,7 and 25 parallelism levels although 1,7 and 15 give pretty close results. We are able to predict the actual GridFTP curve with a smooth transition between the increasing and decresing part of the curve and with a good approximation overall. With this method the peek point of the curve will give us the optimal stream number to open for maximum throughput and no matter which parallelism values we have used we were still be able to get optimal or near optimal results.

We have compared several models presented with a distance metric between actual and predicted throughput values as follows :

$$Distance_n = \sum_{i=1}^{n} (p_i - a_i)^2 \tag{31}$$

where $Distance_n$ presents the total distance up to n parallelism level between predicted and actual throughput values, p_i presents the predicted throughput value for parallelism level i and a_i presents the actual throughput value. Figure 4 presents the results of distance values for several methods presented up to a total of 20 streams. According to that the best values are given with the "Break function" and "Newton's Method" models, "Averaging" and "Logarthmic" models follows them and Dinda et al model performs the worst.

Another comparison metric which is so important for a data scheduler to optimize its transfers is the difference between the predicted optimal number of streams and the actual optimal number of streams which is 7 in our case according to figure 1. Moreover this difference should be minimum even the proposed models are applied with measurements of random stream numbers. In Figure 5, we applied Dinda et al Model and the other improved methods by keeping n_1 and n_3 fixed ,however by changing n_2 in the range of 4 to 12 streams to be able to measure the power of the model with arbitrary measurement information. The predicted optimal stream number is the peek point of the curve for "break function", "logarithmic" and "Newton's method" models. For Dinda et al and "Averaging" models we have taken the first stream number which gives the 95% of the maximum throughput. According to that, Newton's method performed almost perfect, "Break function", "Logarithmic" and "Averaging" methods folllow it and Dinda et al Model again performs the worst.

6. EXTENSION TO MULTI-PARAMETER OPTIMIZATION

In previous sections, we have discussed how to set the optimum number of parallel streams for a single data transfer operation in order to increase the throughput. Of course,

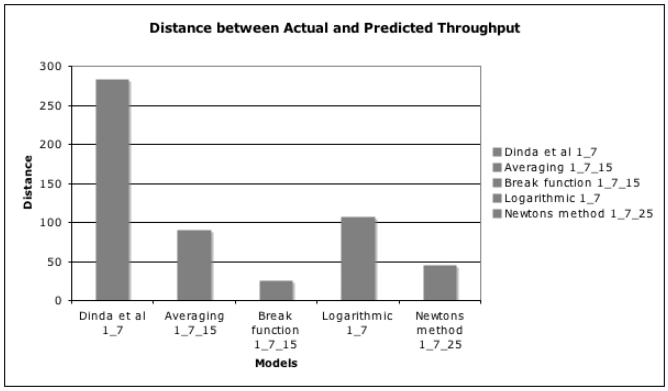

Figure 4: Distance between actual and predicted throughput values up to 20 streams

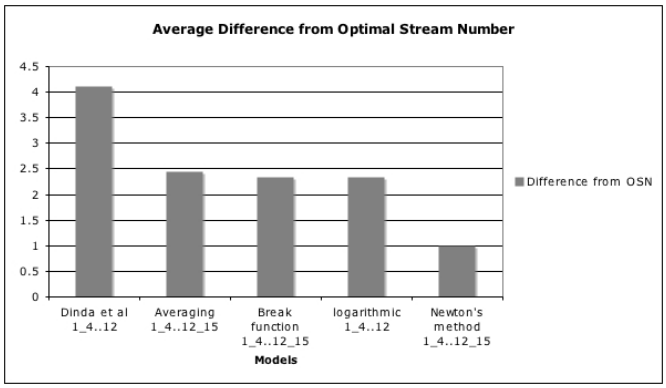

Figure 5: Average distance between actual and predicted optimal number of streams

the number of parallel streams is not the only parameter to be tuned up for increasing the throughput. There are other parameters such as TCP buffer size and I/O block size which can be tuned up as well to get even better performance. On the other hand, trying to optimize all of these parameters at once is a complex problem. In a case where there are multiple ongoing transfers to be optimized at the same, the problem gets even more complex.

In an environment where the goal is to optimize the throughput of multiple transfers, which may have initiated by the same or different users, and which may use the same resources (i.e. network link, source and destination hosts), performing optimizations only considering individual transfers may harm the performance of the rest (if not all) of the ongoing transfers. As a simple example, although n parallel streams may give optimal throughput for a single transfer, using n streams for m different transfers at the same time may result in very poor performance for all of the ongoing transfers of interest.

The first requirement for such a complex optimization would be to have a component which could oversee all of these transfers. In our model, we envision that a data scheduler, such as Stork [11], to take this role. The data scheduler can collect information from multiple sites and links at the same time and use global knowledge to perform multi-

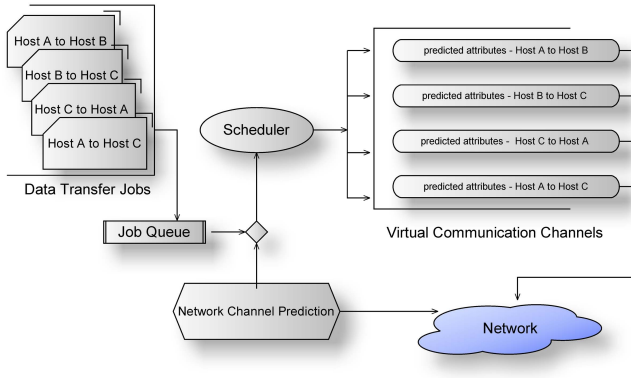

Figure 7: Virtual communications channels for multi-paramater optimization

parameter optimization for multiple transfers. Without a central scheduler and global knowledge of network conditions, overcommitting of network links and in turn decreased performance for all transfers would be inevitable.

Therefore, data scheduler plays an important role in order to enhance the overall performance. It needs to have control on tuning, scheduling, and executing data transfer operations not only to estimate optimal network parameters but also to order data transfers to prevent starvation and contention[3]. Since several data transfer tasks can share the common links and resources, a virtual connection layer inside the data scheduler would help to keep track of network statistics for each data transfer. The virtual communication channel enables the scheduler to apply network predictions to all transfers with a global knowledge of the system. Figure7 represents the overall structure of a data scheduler using virtual channels to predict network parameters for individual transfers in a multi-transfer environment.

7. CONCLUSIONS

The parallel transfer behavior of GridFTP over wide area networks is analyzed and several prediction models are presented and improved according to the characteristics of the transfers. It has been observed that the aggregate throughput starts to fall down in existence of congestion and none of the models could mimic this behavior. By using minimum information on historical results, we have improved the current models to predict the throughput curve of GridFTP and we have observed promising results. As our future work, we consider to gather this information to predict the throughput of parallel GridFTP transfers, instantly by using network prediction tools like Iperf and Nuttcp and extend our test results with different file sizes and also local area measurements. We also plan to implement this capability to a current data scheduler, Stork, and improve the intelligence of the tool by instant decision making for individual transfers.

8. ACKNOWLEDGMENTS

This project is in part sponsored by the National Science Foundation (NSF) under Award Numbers CNS-0619843 (PetaShare), and EPS-0701491(CyberTools), and by the Board

of Regents, State of Louisiana, under Contract Numbers LEQSF (2006-09)-RD-A-06 and NSF/LEQSF (2007-10)-CyberRII-01. We would like to thank Fatih Turkmen for letting us use his testbed at University of Trento, and Dr. Suat Namli for his feedback in developing some of the mathematical models in this paper.

9. REFERENCES

[1] E. Altman, D. Barman, B. Tuffin, and M. Vojnovic. Parallel tcp sockets: Simple model, throughput and validation. In *Proceedings of INFOCOM '06*, pages 1–12. IEEE, April 2006.

[2] H. Balakrishman, V. N. Padmanabhan, S. Seshan, M. Stemm, and R. H. Katz. Tcp behavior of a busy internet server: Analysis and improvements. In *Proceedings of INFOCOM '98*, pages 252–262. IEEE, March 1998.

[3] M. Balman and T. Kosar. Data scheduling for large scale distributed applications. In *Proceedings of the 5th ICEIS Doctoral Consortium, In conjunction with the International Conference on Enterprise Information Systems (ICEIS'07)*, June 2007.

[4] J. Crowcroft and P. Oechslin. Differentiated end-to-end internet services using a weighted proportional fair sharing tcp. *ACM SIGCOMM Computer Communication Review*, 28(3):53–69, July 1998.

[5] L. Eggert, J. Heidemann, and J. Touch. Effects of ensemble-tcp. *ACM SIGCOMM Computer Communication Review*, 30(1):15–29, January 2000.

[6] T. J. Hacker, B. D. Noble, and B. D. Atley. The end-to-end performance effects of parallel tcp sockets on a lossy wide area network. In *Proceedings of IPDPS '02*, page 314. IEEE, April 2002.

[7] T. J. Hacker, B. D. Noble, and B. D. Atley. Adaptive data block scheduling for parallel streams. In *Proceedings of HPDC '05*, pages 265–275. ACM/IEEE, July 2005.

[8] R. P. Karrer, J. Park, and J. Kim. Tcp-rome:performance and fairness in parallel downloads for web and real time multimedia streaming applications. In *Technical Report*. Deutsche Telekom Laboratories, September 2006.

[9] G. Kola, T. Kosar, and M. Livny. Run-time adaptation of grid data-placement jobs. *Scalable Computing: Practice and Experience*, 6(3):33–43, September 2005.

[10] G. Kola and M. K. Vernon. Target bandwidth sharing using endhost measures. *Performance Evaluation*, 64(9-12):948–964, October 2007.

[11] T. Kosar and M. Livny. Stork: Making data placement a first class citizen in the grid. In *Proceedings of ICDCS '04*, pages 342–349. IEEE, March 2004.

[12] J. Lee, D. Gunter, B. Tierney, B. Allcock, J. Bester, J. Bresnahan, and S. Tuecke. Applied techniques for high bandwidth data transfers across wide area networks. In *International Conference on Computing in High Energy and Nuclear Physics*, April 2001.

[13] D. Lu, Y. Qiao, and P. A. Dinda. Characterizing and predicting tcp throughput on the wide area network. In *Proceedings of ICDCS '05*, pages 414–424. IEEE, June 2005.

[14] D. Lu, Y. Qiao, P. A. Dinda, and F. E. Bustamante. Modeling and taming parallel tcp on the wide area network. In *Proceedings of IPDPS '05*, page 68.2. IEEE, April 2005.

[15] H. Sivakumar, S. Bailey, and R. L. Grossman. Psockets: The case for application-level network striping fpr data intensive applications using high speed wide area networks. In *Proceedings of SC'00 ACM/IEEE conference on Supercomputing*, pages 37–es. ACM/IEEE, September 2001.

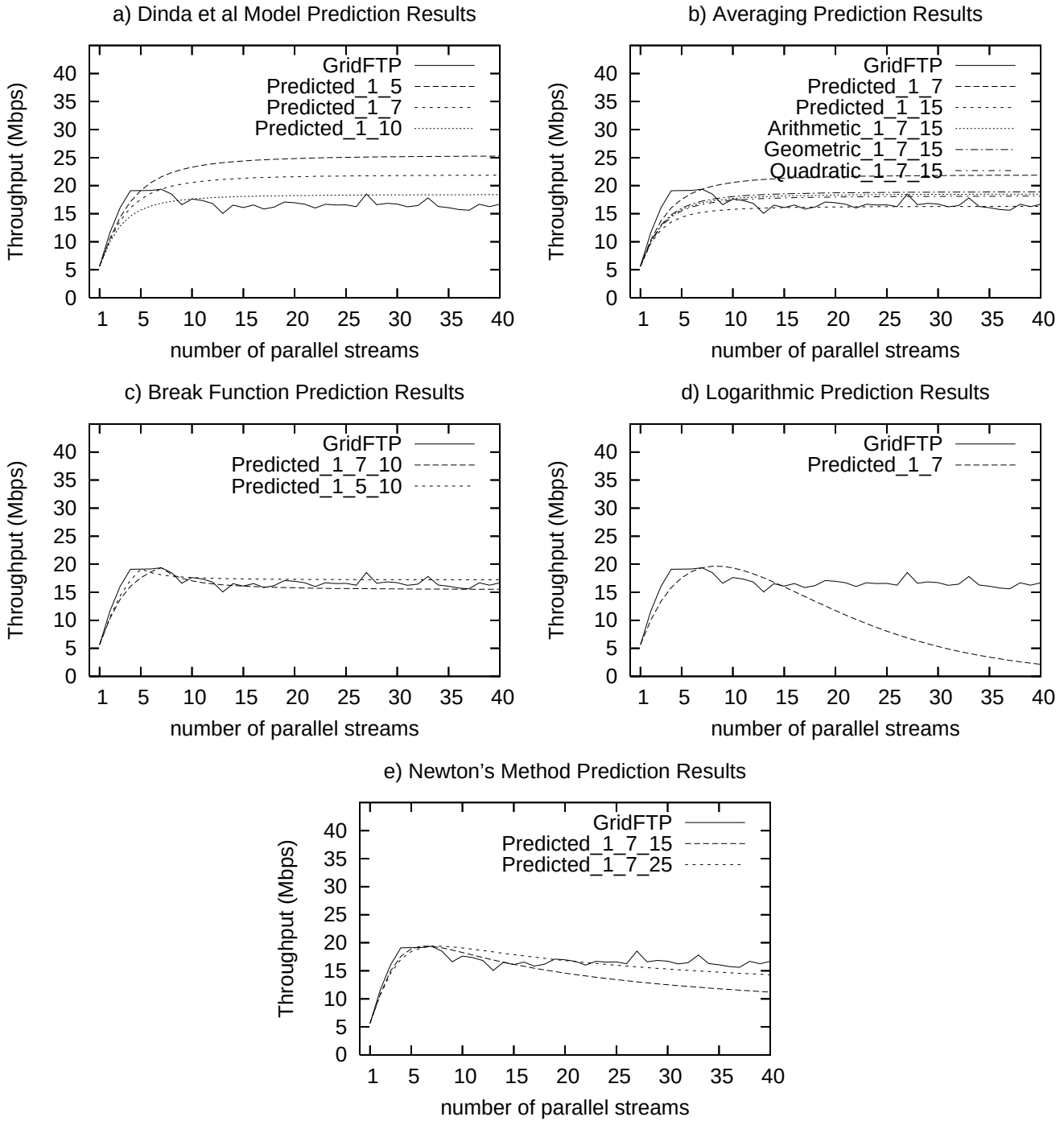

Figure 6: The prediction results of GridFTP transfers by applying Dinda et al Model and its improvements