社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

並列TCP方式のスループット解析とパラメータ設定に関する考察

張　　宗升† 　長谷川　剛† 　村田　正幸†

† 大阪大学大学院情報科学研究科
E-mail: †{zhang, hasegawa, murata}@ist.osaka-u.ac.jp

あらまし　高速高遅延ネットワークにおいてTCPがその帯域を十分使うことができない問題に対して、複数のTCPコネクションを送受信端末間に設定してデータ転送を行なう並列TCP方式が存在し、GridFTPなどのアプリケーションにおいて用いられている。シミュレーションや実ネットワークでの実験などから、並列TCP方式はデータ転送スループットを簡易な方法で向上することができる一方、TCPコネクション数の決定が難しいことが知られている。本稿では、並列TCP方式を用いた場合のデータ転送スループットを数学的解析により導出することによって、並列TCP方式の本質的な性質を明らかにする。その結果、理想的なTCPコネクション数はネットワークパラメータなどによって大きく変化し、その設定が困難であることを示す。また、高速TCP方式との比較を行い、ネットワーク環境の変動に対する性質などの点で、並列TCP方式が劣っていることを明らかにする。

キーワード　parallel TCP, 有効性

Reasons not to Parallelize TCP Connections for Fast Long-Distance Networks

Zongsheng ZHANG†, Go HASEGAWA†, and Masayuki MURATA†

† Graduate School of Information Science and Technology, Osaka University
E-mail: †{zhang, hasegawa, murata}@ist.osaka-u.ac.jp

Abstract　For utilizing fast long-distance networks effectively, parallel TCP was proposed, and has been employing. However, as high-speed transport-layer protocols appear, it is necessary to reinvestigate the performance of parallel TCP. In this paper, we use mathematical analysis to explore its performance. Analysis results show that the open issue of choosing the number of TCP connections is difficult to be solved in practice. Despite the mechanism that adjusts the number of TCP connections during data transfer is proposed, some potential problems still remain. In contrast, it is a better choice to use high-speed TCP in your future applications.

Key words　parallel TCP, effectiveness

1　Introduction

Currently, Transmission Control Protocol (TCP) [1] is the most widely used transport-layer protocol in the Internet. TCP was designed to provide a reliable end-to-end byte stream over an unreliable IP network, while attempting to maintain high utilization of the network link, avoid overloading the bottleneck and provide fair sharing among competing flows. When TCP was designed in the 1960–70s, the T1 link (1.544 Mbps) was a fast network. The link bandwidth of the network has grown quickly since that time, and link bandwidths of 100 Mbps, 1,000 Mbps, or even higher, are common. In such long fat networks (LFNs), which are high-bandwidth and large-delay networks, TCP can not fully utilize the link bandwidth. This is primarily because of the principle of Additive Increase Multiplicative Decrease (AIMD) in the congestion control mechanism of TCP. In the congestion avoidance phase, TCP increases its congestion window linearly by one packet per Round Trip Time (RTT), and sharply decreases its congestion window by half, once packet loss is detected. TCP then requires a long time to increase its congestion window size for fully utilizing LFNs. For example, when a TCP Reno connection fills a 10 Gbps link, and RTT is 100 ms, a congestion window of 83,333 packets is needed. A time of 4,000 sec are required to recover throughput when packets are lost in the network [2].

Addressing the problem of TCP used in LFNs, several high-speed protocols, such as HSTCP [2] and XCP [3], have been proposed in recent years. These protocols modify the TCP algorithm, and their capability to utilize LFNs and their performance have been evaluated [4, 5].

Prior to these high-speed protocols, parallel TCP was proposed as a method of dealing with the problem of TCP in LFNs and was implemented through a number of applications, e.g., BBCP [6] and

GridFTP [7]. In parallel TCP, instead of using one TCP connection, multiple TCP connections are utilized between two end-hosts for one data transmission task. The implementation of parallel TCP is relative simple compared with the TCP modification mentioned above, because parallel TCP can be implemented in the application layer.

The mechanism of parallel TCP can be viewed from different aspects. When the parallel TCP mechanism is used for bulk data transfer, the data file is divided into a number of small chunks, and each chunk is transmitted by one TCP connection. Since each TCP connection uses the AIMD algorithm, the aggregate of congestion window is increased by N (N: number of TCP connections) packets per RTT when there is no packet loss. Thus, parallel TCP uses a larger additive increasing parameter for the congestion window than that used by the normal TCP connection. With respect to the network, the link bandwidth is shared by concurrent parallel TCP connections. Intuitively, each TCP connection passes a "stripped" network link [8]. "stripped" network link can be considered as a "tight" network link, but it has a smaller BDP value. Compared with the case of only one TCP connection, the bandwidth-delay product (BDP) becomes small for each TCP connection. Thus, each TCP connection requires less time to recover its congestion window to utilize the "stripped" link after packet loss occurs, and so parallel TCP can boost the throughput of TCP in LFNs.

Although increasing throughput is the primary purpose of parallel TCP, fairness of parallel TCP should be taken into account when it traverses the public network. H. Hacker et al [9] discuss this issue and propose a solution which uses a long "virtual round trip time" in combination with parallel TCP to prioritize fairness at the expense of effectiveness when the network is fully utilized. However, in this paper, we focus on whether parallel TCP is effective even when the fairness is not taken into consideration. In this sense, we believe high-speed TCPs are more effective. In addition, high-speed TCPs are suitable when fairness is taken into account, for fairness is one of requirements when these high-speed TCPs are designed.

Based upon its mechanism, the throughput of parallel TCP is generally increased as the number of TCP connections is increased. However, the overhead of end-hosts is also increased. Consequently, using twice the number of parallel connections does not necessarily mean doubling the performance. Because the issue of overhead on end-hosts is out of the scope of this paper, it is assumed that the bottleneck is not the end-hosts but the link bandwidth. In addition, although increasing throughput is the primary purpose of parallel TCP, the fairness of parallel TCP should also be taken into account when traversing a public network. In this paper, we focus on the issue of whether parallel TCP can really achieve high throughput, even when fairness is not taken into account. In particular, we are interested in the comparing parallel TCP with high-speed protocols, such as HSTCP. Moreover, we intend to assess the impact of the network on parallel TCP, and we do not give a great deal of consideration to the tuning parameters of TCP.

In the present paper, the performance of parallel TCP is evaluated by mathematical analysis. In the analysis of DropTail deployment, not only is the number of TCP connections taken into account, but "global synchronization" is also investigated. When the impact of global synchronization is considered, two extreme cases, the synchronization case and the non-synchronization case, are evaluated. In the synchronization case, all TCP connections are synchronized, and the throughput for this case is regarded as the lower limit. In the non-synchronization case, TCP connections are not synchronized to any degree, and this case gives the upper limit. The results reveal the difficulty involved in using parallel TCP in practice. Even in the non-synchronization case, which benefits the throughput of parallel TCP, the results show that choosing the number of TCP connections also depends on the network conditions.

2 Analysis of throughput

It is impossible to obtain a uniform expression that can be used to evaluate the performance of parallel TCP for all cases. As mentioned above, two extreme cases – synchronization and non-synchronization cases – are analyzed, and the results are regarded as the lower and upper limits of its throughput.

2.1 Network topology and metrics

A dumbbell topology is used in the analysis. DropTail management is deployed, and the buffer size of the routers is B packets. The bottleneck link bandwidth between the routers is C bps, the minimum RTT is RTT_{min}. The value of BDP is D ($D = RTT_{min} \times C$). There are N TCP connections with the same access link bandwidth and propagation delay competing for a fixed bottleneck link. These connections use the same AIMD algorithm as TCP Reno. The access link bandwidth of each connection is larger than C bps.

We focus on the aggregate behavior of parallel TCP. Packet drop rate (p) and goodput are used as metrics. Here, goodput is the amount of data received by the receiver in unit time, and is not the same as useful throughput, for duplicated packets may be received. It is calculated as:

$$goodput = throughput \times (1-p) \tag{1}$$

2.2 Synchronization case

Under synchronization, we assume that each of the parallel TCP connections fairly shares the bottleneck link, the buffer of the routers, and their behaviors are identical. So the aggregate congestion window ($cwnd$) of parallel TCP with N connections can be considered as follows. In response to a single acknowledgment, parallel TCP increases the number of segments in its congestion window as:

$$cwnd \leftarrow cwnd + \frac{a(cwnd)}{cwnd}.$$

In response to a congestion event, parallel TCP decreases the number of segments in its congestion window as:

$$cwnd \leftarrow (1 - b(cwnd)) \times cwnd.$$

Here, $a(cwnd) = N$, and $b(cwnd) = 1/2$. Figure 1 shows a sketch of the aggregate congestion window of parallel TCP. We use N_{pkts} to denote the total number of packets transmitted in one cycle ($t1+t2$).

Time-out is another factor that can affect the performance of TCP and can occur with reasonable certainty. Upon time-out, the congestion window is set to one packet, then the lost packet is retransmitted. We use p_{to} to denote the probability of packet loss resulting

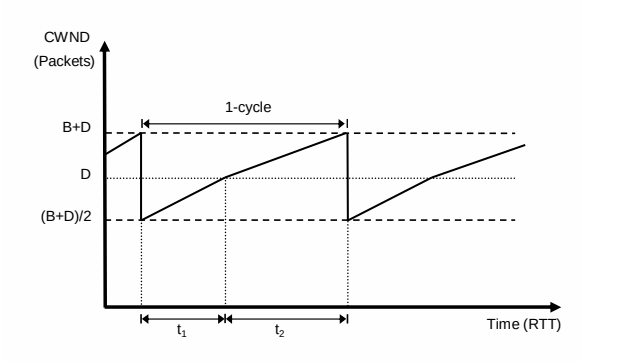


Fig. 1 Congestion window in the steady state

in a time-out. $E(t)$ is the mean time, and $E(n)$ is the mean number of packets sent during the time-out period. Moreover, we assume that the limitation on the congestion window size, the maximum congestion window size, is W_{max}. If the number of parallel TCP connections is less than a certain value, there is no congestion on the bottleneck link. When these components are considered, the packet drop rate and throughput are determined by different equations according to the aggregate value of the congestion window:

$$p = \begin{cases} 0 & \text{if } N \times W_{max} \leqq B+D \\ \dfrac{8N^2}{3(B+D)(B+D+2N)} & \text{if } N \times W_{max} > B+D \end{cases} \quad (2)$$

$$throughput = \begin{cases} N \times \dfrac{W_{max}}{RTT} & \text{if } N \times W_{max} < D \\ BW & \text{if } D \leqq N \times W_{max} \leqq B+D \\ \dfrac{N_{pkts} + N \cdot p_{to} \cdot E(n)}{t1 + t2 + p_{to} \cdot E(t)} & \text{if } N \times W_{max} > B+D \end{cases} \quad (3)$$

2.3 Non-synchronization case

When there are many TCP connections sharing a bottleneck link, each TCP connection obeys the AIMD algorithm. Its throughput ($b(p)$) can be calculated according to the *square root p* formula [10] if the packet drop rate is known.

$$b(p) \approx \frac{1}{RTT\sqrt{\frac{2bp}{3}} + T_0 min\left(1, 3\sqrt{\frac{3bp}{8}}\right) p(1+32p^2)} \quad (4)$$

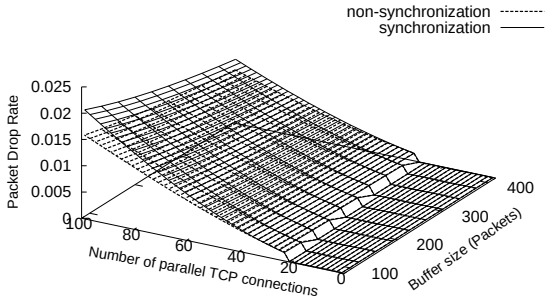
where RTT is the average round trip time, T_0 is the time-out, b is the number of packets that are acknowledged by a received ACK, and p is the packet drop rate.

From the point of view of all TCP connections, the distribution of the aggregate window size is a normal distribution based on the central limit theorem, if TCP connections are not synchronized [11]:

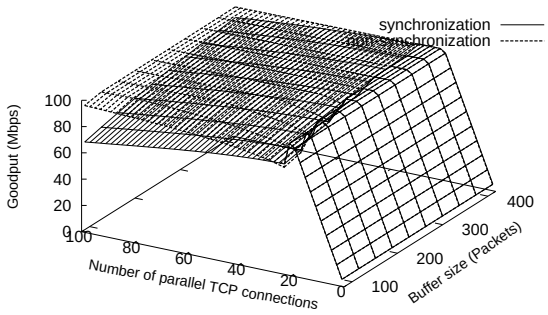
$$W(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (5)$$

Here, W is the aggregate of congestion window size. μ is the mean of the aggregate congestion window size, and σ is its standard deviation. The packet drop rate can be obtained from this distribution.

Based on Eqs. (4) and (5), the packet drop rate and throughput of parallel TCP can be evaluated by the fixed point method.



(a) Packet drop rate



(b) Goodput

Fig. 2 Numerical results (Bandwidth = 100 Mbps)

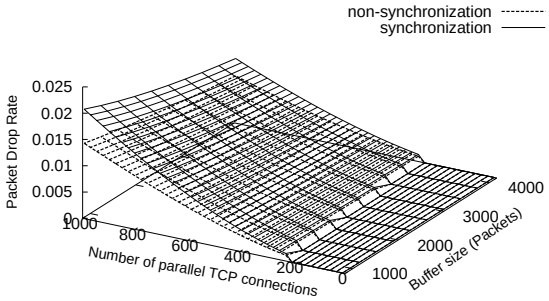
2.4 Numerical results and discussion

In this subsection, the performance of parallel TCP is shown visually by numerical results. There are N TCP connections that compete for a bottleneck link. The parameters are set as follows, and the numerical results are shown in Figs. 2 and 3.

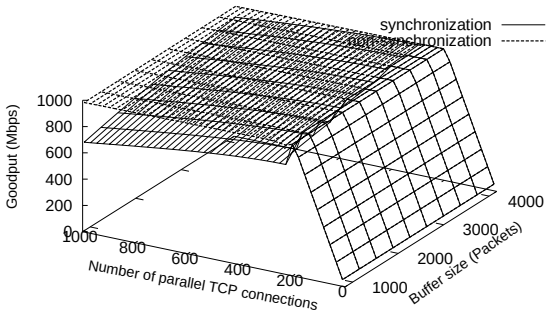
```
Example-1:
  BW = 100 Mbps | 1 Gbps, RTT = 100 ms,
  Packet size = 1,500 Bytes,
  Buffer size = (0.1--0.5)BDP,
  Wmax = 64 KBytes.
```

Note that the maximum value of router buffer size is set to BDP/2. The reason is building a router with a buffer size of BDP is very difficult as the link bandwidth is increased further because of limitations of the commercial memory devices used by routers [11].

As the throughput equation (Eq. (3)) suggests, there exist three regions based on the aggregate congestion window size W_{sum}. In the first region, W_{sum} is less than the value of BDP. Because of the limitation on the congestion window, the bottleneck link cannot be fully utilized if the number of TCP connections is less than a certain value. In this region, throughput and goodput are identical, because there is no congestion on the bottleneck link. They increase linearly as the number of TCP connections increases. However, the utilization is very low if there are a small number of TCP connections. The buffer size of the routers has no effect in this region, and there



(a) Packet drop rate



(b) Goodput

Fig. 3 Numerical results (Bandwidth = 1 Gbps)

is no difference between synchronization and non-synchronization. In the second region, W_{sum} lies between BDP and BDP+buffersize. This is the best region, because parallel TCP achieves its maximum throughput and goodput is equal to throughput. However, it is hard to find a condition that fulfills W_{sum} within this region, for this condition varies significantly with several parameters, such as W_{max}, the value of BDP, and the buffer size of the router. This is illustrated by Figs. 2 and 3. Usually, the value of BDP and the buffer size of the routers are unknown to the end-hosts. On the other hand, if the network link is shared by several users, the valid values of these parameters for a pair of end-hosts varies with time. These make finding the optimal number of TCP connections in practice even more difficult. Of course, there are some users they maybe not expect the optimal performance when they employ parallel TCP. Their purpose is not completely consistent with the object of parallel TCP. For these users, we think their purpose of using parallel TCP should be achieving the expected throughput. They also have to face the problem of choosing the number of TCP connections as well because of dynamics of network. When W_{sum} is larger than BDP+buffersize, network congestion appears, and the throughput of parallel TCP is located in the third region because the number of TCP connections is too large. In this region, the packet drop rate becomes larger and the goodput is degraded as the number of TCP connections increases. The difference between synchronization and

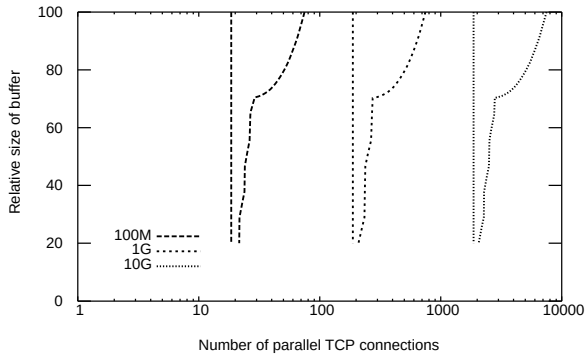


Fig. 4 Contour of utilization (RTT = 100 ms, BW = 100 M/1 G/10 G)

non-synchronization is noticeable and becomes greater as the number of TCP connections increases.

Despite the higher throughput in the non-synchronization case, synchronization is common when DropTail is deployed, and it easily occurs if TCP connections have the same RTT [12, 13]. Parallel TCP possesses the exact properties that induce synchronization. In the synchronization case, whereas the router has a small buffer size, the performance of parallel TCP deteriorates significantly as the number of TCP connections is increased. Therefore, a great deal of attention should be paid to synchronization. In synchronization case and router with small buffer size (less than BDP), parallel TCP can not achieve higher throughput as high speed TCPs once congestion occurs, because the decreasing parameter of parallel TCP is larger than high speed TCP (e.g., HSTCP).

In order to clarify the difficulty in choosing the number of TCP connections in the synchronization case, the contours of the expected throughput are plotted in Fig. 4. We assume that the expected throughput of parallel TCP is 95% of the bottleneck link bandwidth. Note that the Y-axis is the relative value of the router's buffer size, which denotes the percentage of buffer size as BDP/2. Here, BDP/2 is used as a normalization constant because the maximum buffer size is set to BDP/2 in Example-1. The parameters for this graph are as follows. The bottleneck link bandwidth is set to 100 Mbps, 1 Gbps, and 10 Gbps, respectively. Other parameters are as described in Example-1.

In the graph, the areas bounded by the same type of lines are expected in each case. The positions of the areas are different in each case. That is, in order to achieve the expected throughput, the number of TCP connections must be changed according to the different network conditions. In particular, the range of the number of TCP connections for the expected throughput is narrow if the buffer size of the routers is small. This makes it more difficult to find the optimal number of TCP connections. In practice, this range is significant because building a router with a large buffer size is difficult, as mentioned above.

Although the non-synchronization case may benefit parallel TCP, an extra mechanism is necessary. In order to break synchronization, the approach of adding random packet-processing time is required, or Active Queue Management (AQM), such as RED, is deployed at routers [12, 13]. If the approach of adding random packet-processing time is employed, then an extra mechanism at the end-

hosts is required, and moreover, this approach increases RTT. This is contrary to the purpose of alleviating, by parallel TCP, the problem of using TCP in LFNs. We consider that adding random packet-processing time is not a good approach. When RED is deployed, it must be deployed at the router of the bottleneck link. In practice, there are several hops between two endhosts. The location of the bottleneck link is usually unknown. This means that all routers along the path must use the RED mechanism. It is not an actual require.

3 Supplemental discussion

To some extent, the analyses of the present paper indicate how to choose the optimal number of TCP connections for different network conditions based on the premise that the parameters of the network are known by the end-hosts. The results also show that the optimal number is sensitive to network parameters. However, if this premise does not hold, then obtaining an optimal value is difficult.

In addition, the number of TCP connections is unchangeable during data transmission in the above analyses. When the number of TCP connections (N) is determined, parallel TCP can be considered as a high-speed protocol with an increase parameter of N packets per RTT. This is not appropriate if the variability of network conditions is taken into account, because the increase parameter of the high-speed protocol varies with the network conditions. For example, the increase parameter of HSTCP becomes larger and the decrease parameter becomes smaller as the size of the congestion window increases. This may benefit parallel TCP if the number of TCP connections is alterable during data transfer. Such a mechanism has been proposed, e.g., dynamic network resources allocation of GridFTP v2 [14], in which an active peer can open/close one or more additional TCP connections dynamically during data transfer. However, this mechanism may lead to a number of problems.

- Determination of the granularity of changing the number of TCP connections is required. If the granularity is large, tracing the change in link bandwidth is not effective. However, small granularity may lead to overhead in handling TCP connections.
- It is difficult to manage opening/closing of TCP connections and control data channels dynamically. In order to increase the number of TCP connections and attain a steady state, a few dozen RTT are required each time a new connection is created due to the effects of three-way handshake and slow-start phase.
- This mechanism determines the number of TCP connections based on measurement of network conditions. Parallel TCP uses several TCP connections, and the interaction among these TCP connections may affect the accuracy of measurement. Therefore, the performance of parallel TCP may be influenced
- Because the number of TCP connections is changed dynamically, setting up the chunk size is not easy. In addition, chunk management is difficult when the number of TCP connections decreases, because a TCP connection may be closed during the transmission of a chunk.

In contrast, high-speed protocols can offer more flexibility to dynamic networks. Since there is only one high-speed TCP connection when a high-speed protocol is employed, the above-mentioned problems do not affect high-speed protocols. We believe that high-speed protocols can work well and are more efficient than parallel TCP, even for a scenario in which the performance of parallel TCP is not very sensitive to the number of TCP connections. Although high speed TCPs are not widely available in production OS's, e.g., Solaris and Windows, this is likley to change shortly.

4 Conclusions

In the present paper, the characteristics of parallel TCP, which have been obtained by simulations and experiments in past, are clarified by mathematical analysis. When DropTail is deployed, both the number of TCP connections and global synchronization are investigated. The analysis results show that, in practice, parallel TCP does not effectively improve throughput. Future works will include further investigation of the performance of parallel TCP by simulations, comparison with high speed TCPs, and validation in the Internet.

References

[1] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," RFC 2581, April 1999.

[2] S. Floyd, "HighSpeed TCP for large congestion windows," RFC 3649, December 2003.

[3] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," Proc. of SIGCOMM 2002, August 2002.

[4] K. Kumazoe, K. Kouyama, Y. Hori, M. Tsuru, and Y. Oie, "Transport protocol for fast long-distance networks: Evaluation of their penetration and robustness on JGNII," Proc. of SIGCOMM 2005, Feburary 2005.

[5] R. Gupta, S. Ansari, R.L. Cottrell, and R. Hughes-Jones, "Characterization and evaluation of TCP and UDP-based transport on real networks," Proc. of SIGCOMM 2005, Feburary 2005.

[6] A. Hanushevsky, A. Trunov, and L. Cottrell, "Peer-to-Peer computing for secure high performance data copying," Proc. of CHEP'01, September 2001.

[7] W. Allcock, "GridFTP: Protocol extensions to FTP for the Grid," Available as: `http://www.ggf.org/documents/GFD.20.pdf`, April 2003.

[8] H. Sivakumar, S. Bailey, and R.L. Grossman, "PSockets: The case for application-level network striping for data intensive applications using high speed wide area networks," Proc. of the IEEE/ACM SC2000, November 2000.

[9] T. Hacker, B. Noble, and B. Athey, "Improving throughput and maintaining fairness using parallel TCP," Proc. of IEEE INFOCOM 2004, March 2004.

[10] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," Proc. of ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication, September 1998.

[11] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," Proc. of ACM SIGCOMM 2004, September 2004.

[12] S. Floyd and V. Jacobson, "Traffic phase effects in packet-switched gateways," Journal of Internetworking: Practice and Experience, September 1992.

[13] L. Qiu, Y. Zhang, and S. Keshav, "Understanding the performance of many TCP flows," Computer Networks, November 2001.

[14] I. Mandrichenko, W. Allcock, and T. Perelmutov, "GridFTP v2 protocol description," Available as: `http://www.ggf.org/documents/GFD.47.pdf`, May 2005.