CESNET Technical Report 1/2013

Choice of Data Transfer Protocols in Remote Storage Applications

Lukáš Hejtmánek, David Antoš, and Luboš Kopecký

Received 25. 6. 2013

Abstract

This study is devoted to analysis of latency impact of various network data transfer protocols to performance. The data transfers protocols were selected to match the protocols that CESNET Data Storage department uses for communication with clients, these protocols include FTP and FTP with SSL, SCP, rsync and rsync over ssh, NFSv4 with and without Kerberos authentication and encryption, and CIFS. The protocols were evaluated in two ways: 1. on dedicated hardware and dedicated local and wan area network connection to analyse latency impact without bias, 2. in a real-world setup using the production national backbone network. We used various types of load, e.g., number of small files, several large files, reads, writes to simulate typical user access scenarios. We conclude with recommendation of appropriate protocol for particular work load types.

Keywords: RTT latency, NFSv4, FTP, rsync, SCP, performance

1 Introduction

For data transfers, pure network bandwidth is not the only limiting factor from performance point of view. Even if one has powerful clients and servers as well as super-fast network connection, it may not be possible to achieve good data throughput. For many state-of-the-art protocols, latency is an extremely significant factor of data throughput via network. Latency is usually measured as round trip time. While pure network bandwidth and power of clients and servers is still increasing, the latency doesn't allow much space for improvement, beeing limited by the speed of light.

Latency became a problem at different layers of network ISO/OSI model. For example, TCP, as the base for many data transfer protocols, starts slowly on high latency connections. There exist attempts to deal with this kind of slowdown such as High Speed TCP [3]. Other problems occur at the presentation layer which includes protocols like ssh/SCP [6], NFS [2], CIFS [1]. These protocols often use synchronous request/reply paradigm which are directly affected by latency. Even if we wanted to change synchronous request/reply paradigm to asynchronous, we still have problem with synchronous POSIX API, e.g., open/create a file, such an operation is synchronous per se, one cannot create a file before checking if a file of the same name exists. Level of performance degradation with respect to latency depends on number of required synchronous request/reply calls, e.g., CIFS uses more request/reply calls compared to NFS, and we will observe the outcome of the fact later in the report. Choosing a proper protocol for a particular workload can improve overal time of data transfer.

CESNET establishes three large data centers in the Czech Republic. They are distributed in three cities across the country: Brno, Jihlava, and Pilsen. It is obvious that there will be a number of clients connecting over WAN with various latencies. It is desired to provide the best possible quality of service to all the clients. In this report, we present an overview of supported protocols and how they are influenced with latency.

We conducted two sets of tests. The first set was focused on so called laboratory testing, i.e., dedicated server, dedicated client and dedicated network line. The second set was focused on real life testing, i.e., production data server from Data Storage department, production server acting as a client for data transfers, and public wide area network. The first set of tests represents the essence of protocol dependency on latency, the second set of tests includes influence of other factors such as load on server or client, load on network. While the results of the first sets should be fully reproducible, the results of the second sets depend on environment conditions and may vary in time, on the other hand, they illustrate what can be expected.

2 Testing Environment

Both sets of tests were split into two parts. One part consisted of tests of protocols on a server and a client connected through a local network, i.e., low latency client. The other part consisted of test of protocols on a server and a client connected through a wide area network, i.e., high latency client.

2.1 Laboratory Environment

Both clients and server used the same Linux distribution—Debian version 6.0.3 with standard Debian kernel 2.6.32-5. On the server side, we used standard in-kernel NFS version 4 server both with and without Kerberos (RPCSEC), we used SAMBA version 3.5.6, rsync 3.0.7, ssh/SCP version 5.5p1, and vsftpd 2.3.2-3. On the client side, we used standard in-kernel NFS version client with and without Kerberos (RPCSEC), we used CIFS version 2:4.5-2, rsync 3.0.7, ssh/SCP version 5.5p1, and lftp client version 4.0.6.

The server was equipped with two Intel X5650 CPUs running at 2.67GHz with total number of 24 cores including hyperthreaded cores, with 96GB RAM, and with Intel 598EB 10GE NIC. Both clients were identical, equipped with two Intel 5160 CPUs running at

3.0GHz with total number of 4 cores without hyperthreading, with 16GB RAM and with Myricom 10GE NIC.

In the laboratory environment, we did not use disk-to-disk nor memory-to-disk data transfers, our tests deployed memory-to-memory transfers only. We used standard Linux RAM disk block device (brd module) formated to XFS. RAM disk writes on the server were at speed 1162±27MB/s, reads on the server were at speed 1996±52MB/s, measured with dd utility[1] and they are averages of 10 repeats.

Latency measurements were taken by ping utility version 3:20100418-3. Network performance was measured by iperf utility version 2.0.4-5. We used non-standard settings of TCP stack as follows. Limit of network core read and write memory buffers were raised to 200MB from usual 128kB. TCP scaling window was increased to 50MB from standard 16kB. These tweaks were experimentally evaluated for maximum performance.

We used 10Gbps network connection in both cases—local and wide area network. In case of local area, we achieved 9.43Gbps speed over TCP connection in both directions, round trip time was 0.3 ± 0.018 ms. In case of wide area, we achieved 8.15Gbps speed over TCP connection in both directions, round trip time was 3.4 ± 0.065 ms.

2.2 Real-life Environment

Server used SLES 11sp2 Linux distribution with 3.0.74-0.6.8.1 kernel with SGI's enhanced NFSv4 1.2.3-18.17.2 with and/or without Kerberos (RPCSEC), SAMBA version 3.6.3, rsync 3.0.4, ssh/SCP version 5.1p1, and vsftpd 2.3.5. Client used RHEL 6.3 Linux distribution with 2.6.32-385.6.2 kernel with standard in-kernel NFS version 4 client with and without Kerberos (RPCSEC), we used CIFS version 4.8.1-18, rsync 3.0.6, ssh/SCP version 5.3p1, and lftp client version 4.0.9-1.

The server was equipped with two Intel X5650 CPUs running at 2.67GHz with total number of 24 cores including hyperthreaded cores, with 48GB RAM, and with Intel 599EB 10GE NIC. Local area client was equipped with two Intel E5506 CPUs running at 2.13GHz with total number of 8 cores without hyperthreading, with 32GB RAM and with Emulex OneConnect 10Gb NIC. Wide area client was equipped with two Intel E5-2609 CPUs running at 2.4GHz with total number of 8 cores without hyperthreading, with 96GB RAM and with Mellanox MT26428 Infiniband/10Gb NIC.

Server was part of SGI's HSM[2] system that consisted of Tier 0 disks (600GB Fibre Channel disks in HW RAID 6 array organized as 8+2—8 data disks and 2 parity disks), Tier 1 disks (2TB SATA disks in HW RAID 6 array organized as 16+2—16 data disks and 2 parity disks), and Tier 2 tape library with capacity of 3.3PB. We used 4.4TB volume located on Tier 0 formated as CXFS version 6.6.0. DMF version 5.6.0 for data migration was enabled but no migration was triggered by tests. The server acted as a client of the CXFS file system, i.e., it was not a metadata server (running on the metadata server, it would result in faster small file operations). Disk writes on the server were at speed

[1]`dd if=/dev/zero of=test-file bs=1M count=10240` and vice versa.

[2]Hierarchical Storage Management

870±119MB/s, reads on the server were at speed 1434±302MB/s, measured with dd utility[3] and are averaged of 10 repeats.

Measurement tools for latency and network performance are exactly the same as for the laboratory environment. We also used identical fine-tuning parameters of the TCP stack.

We used 10Gbps network connection in both cases—local and wan area network. In case of local area, we achieved 8.77Gbps speed over TCP connection from client to server and 7.11Gbps from server to client, round trip time was 0.2 ± 0.04ms. In case of wide area, we achieved 6.91Gbps speed over TCP connection from the client to the server and 5.14Gbps in the opposite direction, round trip time was 5.36 ± 0.05ms.

3 Tests Structure

We used two different test sets. The first set was used for non-file system protocols such as ssh/SCP/SFTP, rsync [5], FTP/FTPS [4]. The second set was used for file system protocols—NFSv4 in all variants and CIFS. In case of real life environment, we evaluated only protocols that Data Storage offers, so we omitted CIFS, NFSv4 without Kerberos and rsync in daemon mode from testing.

3.1 Non-file System Tests

For non-file system tests, the testing pattern was as follows.

1. We created 10GB file on a client's source disk (RAM disk or /tmp directory depending on laboratory tests or real life tests).
2. We transferred the file to the server's destination disk (RAM disk or disk array depending on laboratory tests or real life tests) ten times. This step is marked as *Put file.*
3. We transferred the file from the server's destination disk to the client's source disk ten times. This step is marked as *Get file.*
4. We unzipped Linux kernel sources 2.6.39 into a client's source disk.
5. We recursively transfered all the source files to the server (i.e., 36723 files in 2256 directories, average file size was 13.7kB) ten times. This step is marked as *Put linux.*
6. We recursively transfered all the source files from the server to the client ten times. This step is marked as *Get linux.*
7. We zipped (without compression) whole Linux kernel source tree on the client, transfered the zipped file to the server and unzipped it on the server. This whole step was done ten times. This step is marked as *Put tar.*

[3] `dd if=/dev/zero of=test-file bs=1M count=10240` and vice versa.

8. We zipped (without compression) whole Linux kernel source tree on the server, transfered the zipped file to the client and unzipped it on the client. This whole step was done ten times. This step is marked as *Get tar*.

We evaluated duration of transfers in the steps 2, 3, 5, 6, 7, and 8.

Transfers were done through: (1) SCP, (2), rsync over ssh, (3) rsync in daemon mode, (4) standard FTP, (5) FTP over SSL. The results can be found in the following section.

3.2 File System Tests

For file system tests, the testing pattern was as follows.

1. We unzipped Linux kernel source into remote file system. This step is marked as *untar*.
2. We run `make defconfig` in unzipped sources on the remote file system. This step is marked as *def config*.
3. We zipped the unzipped and configured sources from the remote file system. This step is marked as *tar*.
4. We recursively copied the unzipped and configured sources from the remote file system to local disk. This step is marked as *cp down*.
5. We recursively deleted the unzipped and configured sources from the remote file system. This step is marked as *rm*.
6. We recursively copied the unzipped and configured sources from the local disk back to the remote file system. This step is marked as *cp up*.
7. We stored a 10GB file into the remote file system using *dd* tool (from `/dev/zero`). This step is marked as *dd up*.
8. We transfered the 10GB file from the remote file system using *dd* tool into `/dev/null`. This step is marked as *dd down*.

We evaluated duration of all the steps. The whole sequence of the steps was repeated 10 times. The remote file system was: (1) NFSv4 without Kerberos—NFS-SYS, (2) NFSv4 with Kerberos (authentication only)—NFS-KRB5, (3), NFSv4 with Kerberos and message integrity checking—NFS-KRB5i, (4) NFSv4 with Kerberos and full encryption—NFS-KRB5p, and (5) CIFS—CIFS. The results can be found in the following section.

4 Results

In order to make the results mutually comparable, we use duration of each test in seconds as the measure.

4.1 Laboratory Results

Results of non-file system tests over the local area network can be found in Figure 1. As expected, the fastest protocol for large files transfer is FTP without SSL, i.e., without expensive encryption. Performance of ssh/SCP and rsync over ssh are comparable in this case, as both use the same encryption in the same way—both SCP and rsync over ssh are based on ssh. Implementation of rsync in daemon mode seems to be worse compared to FTP without encryption for large files.

Results show that rsync (either in daemon mode or over ssh) outperforms the other protocols when one needs to transfers many small files. This is caused by the way rsync handles the transfer, creating a list of files to be transferred in advance and performing a bulk transfer. In comparison, SCP transfers files one by one synchronously. The worst performance is in case of FTP (no matter if plain or encrypted) as FTP creates new connections often. It is especially significant in case of SSL encryption, causing an SSL renegotiation for each file which is quite expensive.

The results also show that instead of transferring huge number of small files, it is faster to zip them (without compression), transfer single zip file, and unzip them on target if not using rsyncd or rsync over ssh.

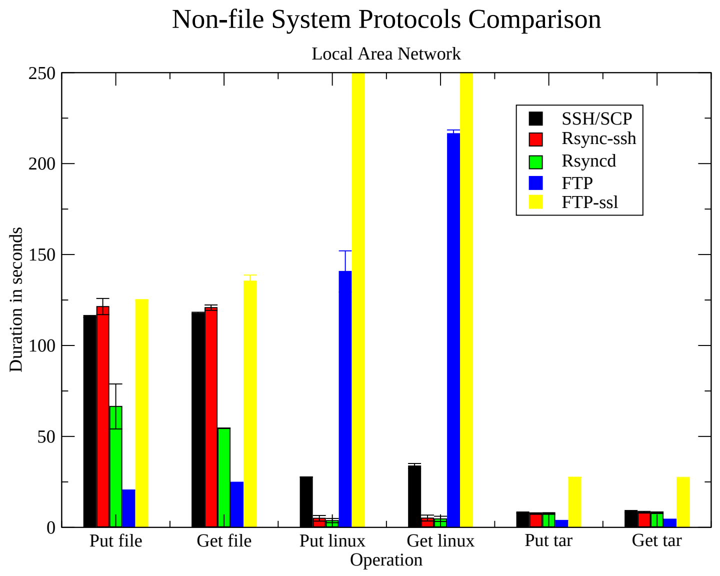

Figure 1: Duration of operations of non-file system tests over the local area network in the laboratory setup.

Results of non-file system tests over the wide area network can be found in Figure 2. Similarly to the local area network, the fastest protocol for large files transfer is FTP without SSL, and ssh/SCP and rsync over ssh are comparable. Also for wide area network, rsync in daemon mode performs worse compared to FTP without encryption for large files.

In case of small files, i.e., transferring Linux kernel sources, it can be seen that ssh/SCP and FTP are very slow over the wide area as synchronous operations take more time to finish due to latency. Using tar for bulk transfers performs by an order of magnitude better than just transferring file by file using these protocols.

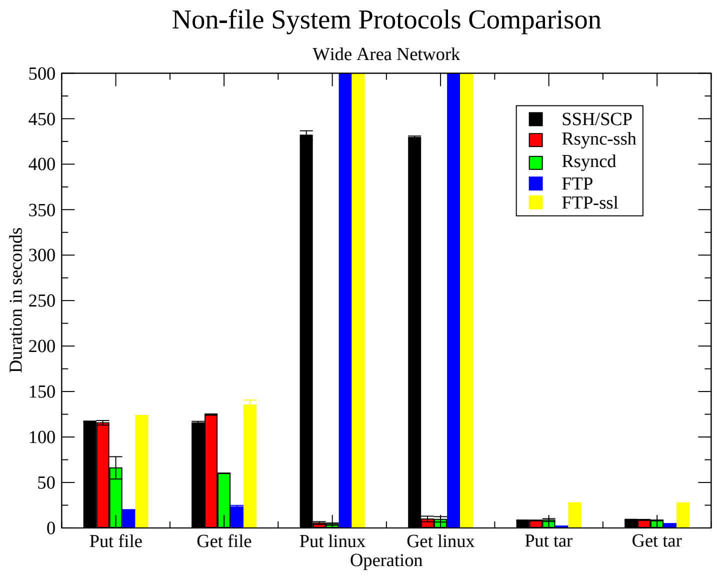

Figure 2: Duration of operations of non-file system tests over the wide area network in the laboratory setup.

Table 1 shows exact duration times for each data transfer and allows comparison of local area and wide area transfers. As we can see, there is no significant difference between local and wide area transfers for large files. The same holds for transferring tar files, i.e., zipped Linux kernel sources, it is also a single large file even if the time to create the archives and to split them again is included in the comparison. In case of small files, wide area transfers perform significantly worse compared to local area transfers for ssh/SCP and both FTP and FTP with SSL protocols.

Results of file system tests over the local area network can be found in Figure 3. Performance of NFS with SYS or Kerberos authentication is comparable to CIFS in case of metadata intensive operations (untar, def config, tar, cp down, rm, cp up). Differences among NFS with SYS, Kerberos, and Kerberos + integrity checking are insignificant. NFS with Kerberos + full encryption performs worse as it consumes some time encrypting which is CPU intensive.

Operation	**Protocol**	**LAN**	**WAN**
Put file	SCP	115.9 ± 0.52sec	117.1 ± 0.37sec
	rsync-ssh	121.4 ± 4.43sec	115.6 ± 2.55sec
	rsyncd	66.4 ± 12.36sec	66.0 ± 12.27sec
	FTP	20.5 ± 0.06sec	19.9 ± 0.06sec
	FTP-ssl	124.9 ± 0.29sec	123.5 ± 0.14sec
Get file	SCP	117.7 ± 0.52sec	115.5 ± 1.63sec
	rsync-ssh	120.8 ± 1.46sec	124.6 ± 0.71sec
	rsyncd	54.5 ± 0.14sec	60.0 ± 0.29sec
	FTP	24.5 ± 0.29sec	23.1 ± 1.57sec
	FTP-ssl	135.3 ± 3.33sec	135.1 ± 5.52sec
Put Linux	SCP	27.6 ± 0.08sec	431.8 ± 4.80sec
	rsync-ssh	4.9 ± 1.55sec	5.4 ± 1.55sec
	rsyncd	3.7 ± 1.15sec	4.2 ± 1.28sec
	FTP	140.7 ± 11.32sec	988.7 ± 0.62sec
	FTP-ssl	3671.6 ± 7.02sec	5048.5 ± 8.81sec
Get Linux	SCP	33.7 ± 1.30sec	429.3 ± 1.53sec
	rsync-ssh	5.1 ± 1.63sec	9.8 ± 3.07sec
	rsyncd	4.6 ± 1.42sec	9.4 ± 2.96sec
	FTP	216.4 ± 2.02sec	873.3 ± 0.53sec
	FTP-ssl	3492.1 ± 5.57sec	4772.6 ± 5.59sec
Put tar	SCP	8.1 ± 0.30sec	8.2 ± 0.43sec
	rsync-ssh	7.5 ± 0.40sec	8.2 ± 0.41sec
	rsyncd	7.6 ± 0.43sec	8.6 ± 1.56sec
	FTP	3.7 ± 0.02sec	2.0 ± 0.01sec
	FTP-ssl	27.4 ± 0.02sec	27.5 ± 0.03sec
Get tar	SCP	8.7 ± 0.50sec	8.9 ± 0.53sec
	rsync-ssh	8.3 ± 0.45sec	8.8 ± 0.47sec
	rsyncd	8.0 ± 0.45sec	8.2 ± 0.46sec
	FTP	4.4 ± 0.02sec	4.5 ± 0.16sec
	FTP-ssl	26.8 ± 0.60sec	26.5 ± 1.11sec

Table 1: Duration of operations of non-file system tests in laboratory setup.

In case of large files, there is significant difference between NFS (SYS, Kerberos, Kerberos + integrity checking) and CIFS, CIFS performance is much worse because of missing optimizations (such as compound operations) for large data transfers.

Results of file system tests over the wide area network are presented in Figure 4. Performance of NFS with SYS or Kerberos authentication is comparable to CIFS in case of metadata intensive operations (untar, def config, tar, cp down, rm, cp up) similarly to local area network. In case of the wide area network, full encryption of data transfers using NFS

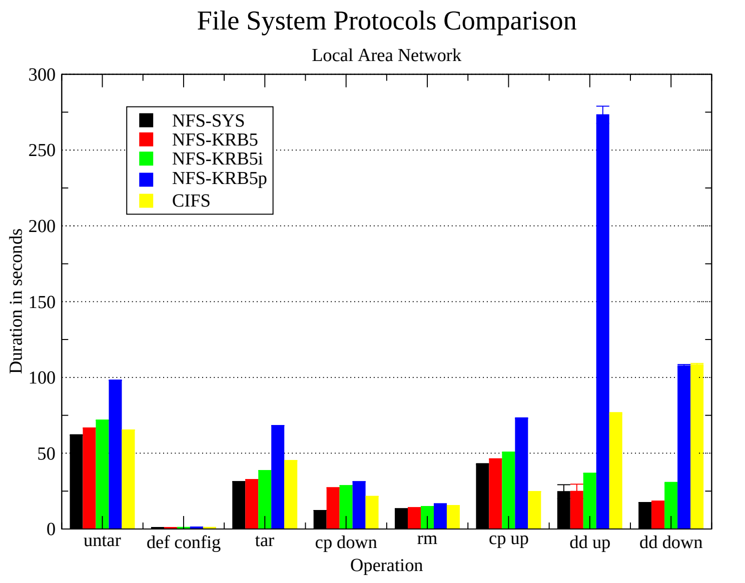

Figure 3: Duration of operations of file system tests over local area network in laboratory setup.

does not decrease performance, i.e., CPU has enough time to encrypt data due to high latency of replies.

In case of large files, there is a significant difference among NFS (SYS, Kerberos, Kerberos + integrity checking) and CIFS or NFS (Kerberos + full encryption), CIFS performance is much worse—missing optimizations for large data transfers while NFS has such optimizations thus full encryption becomes a significant bottleneck.

Table 2 shows exact duration times of each file system data transfer and allows comparison of local area and wide area transfers. For large files, there is a significant difference only for CIFS; NFS in all variants is quite comparable over local or wide area network. Other types of operations are metadata-intensive, so wide area network transfers have significant negative impact on performance.

Figures 5 and 6 show comparison for the same type of operations over file system and non-file system tests. There are two types of common operations for both file system and non-file system tests—put/get a large file, put/get many small files. It can be seen that for a large file, the most appropriate protocols are FTP without SSL and NFS with SYS or Kerberos authentication (authentication with integrity checking is a bit worse). The least appropriate protocols are the ones having full encryption, i.e., ssh/SCP, rsync over ssh, FTP with SSL, NFS with Kerberos (authentication and full encryptyion) and also CIFS for its ineffectiveness. For many small files, the best protocols are rsync (either in rsyncd mode or over ssh) followed by ssh/SCP and NFS. The least appropriate is FTP in any variant. This analysis holds for both local area and wide area networks.

Operation	Protocol	LAN	WAN
untar	NFS-SYS	61.9 ± 0.22sec	1546.3 ± 154.22sec
	NFS-KRB5	66.5 ± 0.11sec	1558.6 ± 155.41sec
	NFS-KRB5i	71.7 ± 0.13sec	1622.9 ± 1.38sec
	NFS-KRB5p	98.1 ± 0.06sec	1600.5 ± 157.53sec
	CIFS	65.3 ± 0.03sec	1425.8 ± 119.00sec
defconfig	NFS-SYS	0.9 ± 0.00sec	13.2 ± 1.33sec
	NFS-KRB5	0.9 ± 0.00sec	13.3 ± 1.30sec
	NFS-KRB5i	1.0 ± 0.00sec	12.9 ± 0.03sec
	NFS-KRB5p	1.2 ± 0.00sec	13.6 ± 1.30sec
	CIFS	1.0 ± 0.00sec	15.8 ± 1.16sec
tar	NFS-SYS	31.2 ± 0.06sec	812.8 ± 37.13sec
	NFS-KRB5	32.5 ± 0.05sec	707.2 ± 0.21sec
	NFS-KRB5i	38.4 ± 0.05sec	714.4 ± 0.16sec
	NFS-KRB5p	68.1 ± 0.08sec	743.0 ± 0.25sec
	CIFS	45.0 ± 0.18sec	1091.8 ± 0.13sec
cp down	NFS-SYS	12.2 ± 0.01sec	555.1 ± 4.34sec
	NFS-KRB5	27.1 ± 0.08sec	549.6 ± 0.19sec
	NFS-KRB5i	28.5 ± 0.07sec	553.1 ± 0.17sec
	NFS-KRB5p	31.1 ± 0.07sec	557.5 ± 0.70sec
	CIFS	21.4 ± 0.07sec	415.9 ± 0.90sec
rm	NFS-SYS	13.3 ± 0.01sec	414.1 ± 0.40sec
	NFS-KRB5	14.0 ± 0.01sec	417.4 ± 0.38sec
	NFS-KRB5i	14.8 ± 0.01sec	419.9 ± 0.17sec
	NFS-KRB5p	16.5 ± 0.09sec	422.9 ± 0.47sec
	CIFS	15.4 ± 0.00sec	297.1 ± 0.04sec
cp up	NFS-SYS	42.7 ± 0.32sec	930.0 ± 0.33sec
	NFS-KRB5	46.1 ± 0.16sec	937.6 ± 0.20sec
	NFS-KRB5i	50.5 ± 0.17sec	945.0 ± 0.25sec
	NFS-KRB5p	73.0 ± 0.23sec	971.1 ± 0.49sec
	CIFS	24.6 ± 0.02sec	497.2 ± 0.04sec
dd up	NFS-SYS	24.6 ± 4.55sec	33.0 ± 1.04sec
	NFS-KRB5	24.8 ± 4.73sec	33.9 ± 0.25sec
	NFS-KRB5i	36.6 ± 0.13sec	43.9 ± 2.06sec
	NFS-KRB5p	273.2 ± 5.67sec	274.2 ± 0.55sec
	CIFS	76.2 ± 0.53sec	705.6 ± 0.58sec
dd down	NFS-SYS	17.4 ± 0.06sec	26.4 ± 0.52sec
	NFS-KRB5	18.3 ± 0.06sec	26.6 ± 0.50sec
	NFS-KRB5i	30.3 ± 0.44sec	29.7 ± 1.32sec
	NFS-KRB5p	107.6 ± 0.88sec	159.0 ± 3.64sec
	CIFS	108.3 ± 0.86sec	2277.5 ± 0.27sec

Table 2: Duration of operations of file system tests in the laboratory setup.

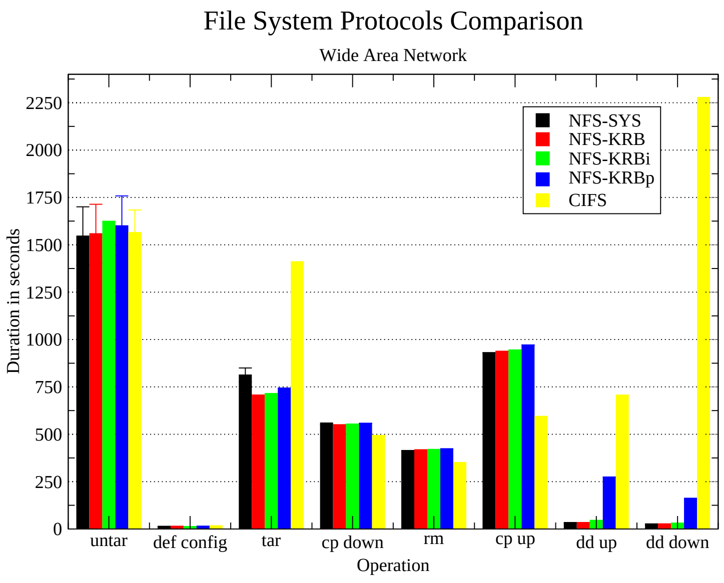

Figure 4: Duration of operations of file system tests over the wide area network in the laboratory setup.

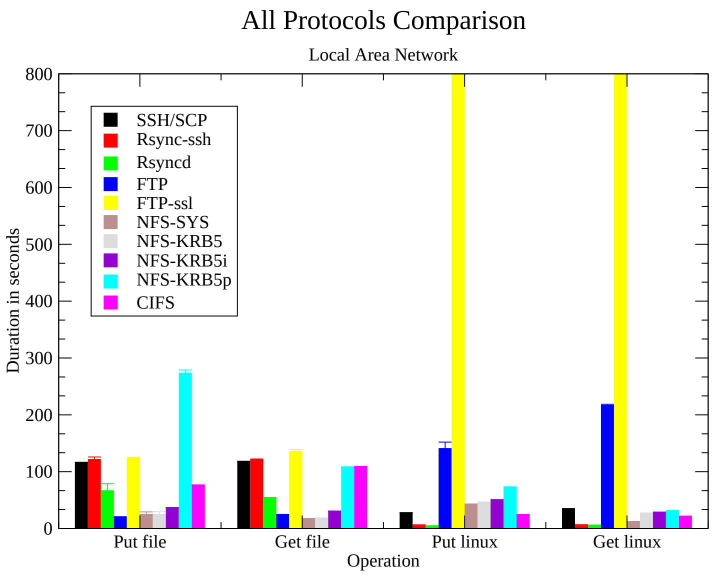

Figure 5: Duration of operations of all tests over the local area network in the laboratory setup.

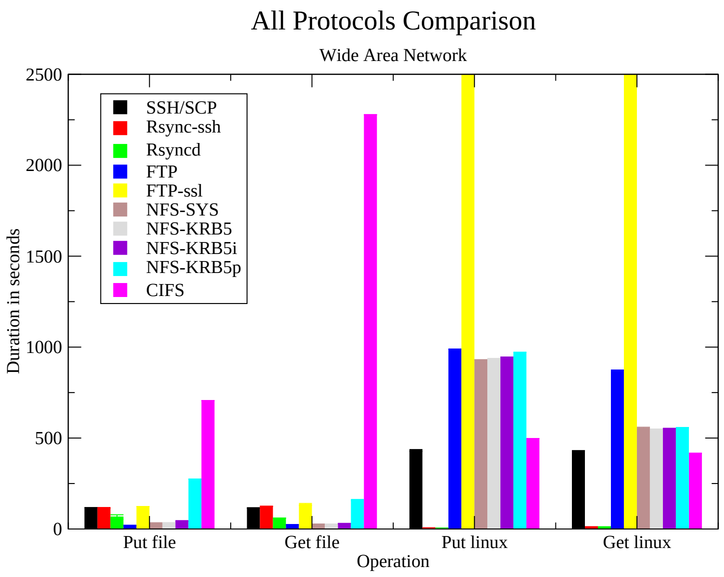

Figure 6: Duration of operations of all tests over the wide area network in the laboratory setup.

Table 3 shows detailed values and direct comparison of local and wide area networks. For a large file, there are no significant differences between local and wide area networks except CIFS that is ineffective over wide area. As for rsync in both variants, there is basically no difference between local and wide area networks in case of transferring many small files. NFS and CIFS in any variant suffer over 20 times slow down. In case of FTP, there is not such a big slow down but it is the slowest protocol of all the testing set for small files.

4.2 Real-life Results

Results of non-file system tests over the local area network can be found in Figure 7. In case of a large file, results are consistent with laboratory tests, i.e., FTP without data encryption is the fastest protocol, followed by SCP and rsync over ssh. In case of many small files, performance depends on direction, performance is worse if we store files to the server because CXFS does not handle large numbers of small files well. Due to this fact, SCP, rsync and FTP without data encryption perform the same. When retrieving data, FTP without data encryption and rsync perform best followed by SCP. FTP with data encryption performs poorly for all kinds of data transfers. Zipped transfers show a surprising result in the real-life environment: zipping/unzipping basically does not help on local area network. It seems that overhead caused by zipping/unzipping is higher than

Operation	**Protocol**	**LAN**	**WAN**
Put file	SCP	115.9 ± 0.52sec	117.1 ± 0.37sec
	rsync-ssh	121.4 ± 4.43sec	115.6 ± 2.55sec
	rsyncd	66.4 ± 12.36sec	66.0 ± 12.27sec
	FTP	20.5 ± 0.06sec	19.9 ± 0.06sec
	FTP-ssl	124.9 ± 0.29sec	123.5 ± 0.14sec
	NFS-SYS	24.6 ± 4.55sec	33.0 ± 1.04sec
	NFS-KRB5	24.8 ± 4.73sec	33.9 ± 0.25sec
	NFS-KRB5i	36.6 ± 0.13sec	43.9 ± 2.06sec
	NFS-KRB5p	273.2 ± 5.67sec	274.2 ± 0.55sec
	CIFS	76.2 ± 0.53sec	705.6 ± 0.58sec
Get file	SCP	117.7 ± 0.52sec	115.5 ± 1.63sec
	rsync-ssh	120.8 ± 1.46sec	124.6 ± 0.71sec
	rsyncd	54.5 ± 0.14sec	60.0 ± 0.29sec
	FTP	24.5 ± 0.29sec	23.1 ± 1.57sec
	FTP-ssl	135.3 ± 3.33sec	135.1 ± 5.52sec
	NFS-SYS	17.4 ± 0.06sec	26.4 ± 0.52sec
	NFS-KRB5	18.3 ± 0.06sec	26.6 ± 0.50sec
	NFS-KRB5i	30.3 ± 0.44sec	29.7 ± 1.32sec
	NFS-KRB5p	107.6 ± 0.88sec	159.0 ± 3.64sec
	CIFS	108.3 ± 0.86sec	2277.5 ± 0.27sec
Put Linux	SCP	27.6 ± 0.08sec	431.8 ± 4.80sec
	rsync-ssh	4.9 ± 1.55sec	5.4 ± 1.55sec
	rsyncd	3.7 ± 1.15sec	4.2 ± 1.28sec
	FTP	140.7 ± 11.32sec	988.7 ± 0.62sec
	FTP-ssl	3671.6 ± 7.02sec	5048.54 ± 8.81sec
	NFS-SYS	42.7 ± 0.32sec	930.0 ± 0.33sec
	NFS-KRB5	46.1 ± 0.16sec	937.6 ± 0.20sec
	NFS-KRB5i	50.5 ± 0.17sec	945.0 ± 0.25sec
	NFS-KRB5p	73.0 ± 0.23sec	971.1 ± 0.49sec
	CIFS	24.6 ± 0.02sec	497.2 ± 0.04sec
Get Linux	SCP	33.7 ± 1.30sec	429.3 ± 1.53sec
	rsync-ssh	5.1 ± 1.63sec	9.8 ± 3.07sec
	rsyncd	4.6 ± 1.42sec	9.4 ± 2.96sec
	FTP	216.4 ± 2.02sec	873.3 ± 0.53sec
	FTP-ssl	3492.1 ± 5.57sec	4772.6 ± 5.59sec
	NFS-SYS	12.2 ± 0.01sec	555.1 ± 4.34sec
	NFS-KRB5	27.1 ± 0.08sec	549.6 ± 0.19sec
	NFS-KRB5i	28.5 ± 0.07sec	553.1 ± 0.17sec
	NFS-KRB5p	31.1 ± 0.07sec	557.5 ± 0.70sec
	CIFS	21.4 ± 0.07sec	415.9 ± 0.90sec

Table 3: Duration of test operations in the laboratory setup.

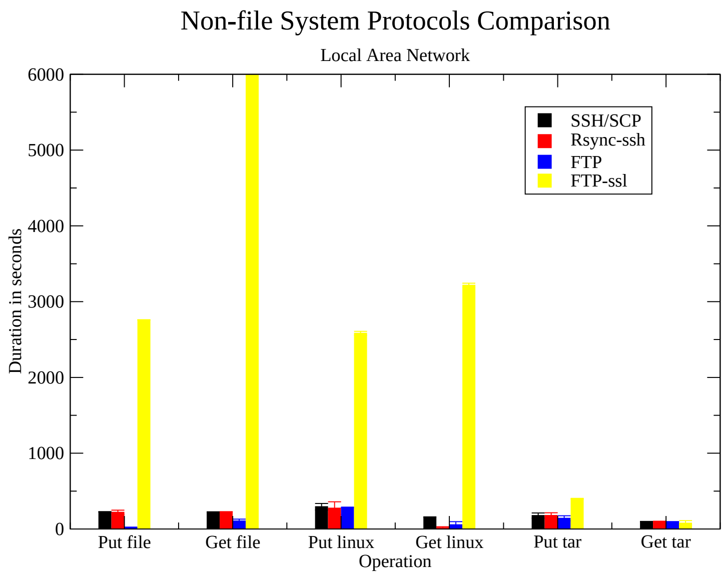

Figure 7: Duration of operations of non-file system tests over the local area network in the real-life setup.

slowdown due to network latency with an exception of FTP with data encryption, SSL renegotiation is quite expensive compared to zipping/unzipping.

Figure 8 shows results of tests over the wide area network. In case of a large file, results differ from laboratory tests mainly because the client had worse implementation of SSL for FTP, while not for ssh, therefore ssh does not overload the client's CPU. In case of small files, rsync over ssh followed by SCP are the fastest protocols which is consistent with the laboratory testing. In contrast to the local area network, zipping/unzipping brings performance benefits in case of ssh/SCP and both FTP and FTP with SSL.

Table 4 summarizes exact values for all non-file system tests over both local and wide area networks. In case of a large file, all protocols are comparable except FTP with data encryption. Differences are caused only by varying utilization of the shared network. In case of small files, rsync over ssh performs well in both cases—local and wide area network, FTP without data encryption suffers the most. Performance in case of zipping/unzipping is mostly independent of the scope of the network.

In Figures 9 and 10, results are shown for NFSv4 transfers in three supported variants. In case of metadata intensive operations (i.e., all except dd up and dd down), there are no significant differences among the variants (authentication only, authentication and integrity checking, full encryption) for both local and wide area networks. In case of a large file, slower CPU (local area network) imposes huge performance drop for full encryption (NFS-KRB5p) in both transfer directions.

Table 5 summarizes exact values for all file system tests over both local and wide area

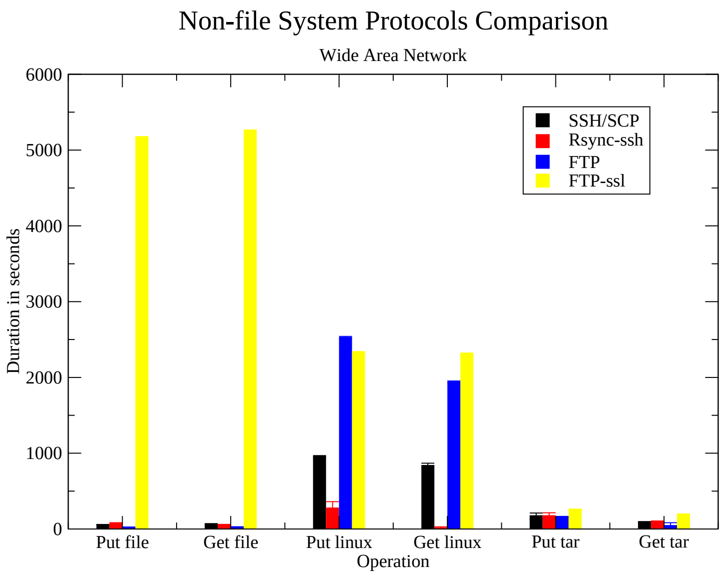

Figure 8: Duration of operations of non-file system tests over the wide area network in the real-life setup.

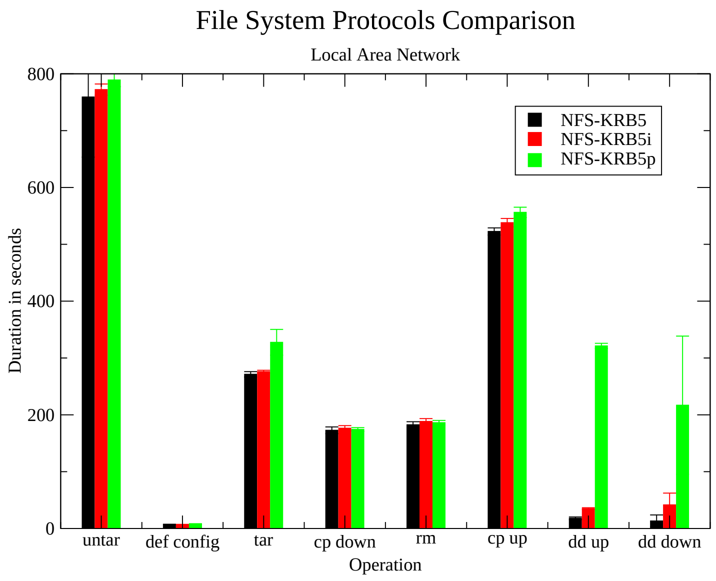

Figure 9: Duration of operations of file system tests over the local area network in the real-life setup.

Operation	**Protocol**	**LAN**	**WAN**
Put file	SCP	226.6 ± 1.97sec	58.3 ± 2.02sec
	rsync-ssh	220.3 ± 28.43sec	71.5 ± 11.79sec
	FTP	22.0 ± 1.57sec	22.1 ± 4.10sec
	FTP-ssl	2759.5 ± 1.04sec	5166.4 ± 10.40sec
Get file	SCP	224.5 ± 2.82sec	60.2 ± 11.14sec
	rsync-ssh	226.2 ± 2.15sec	57.7 ± 2.95sec
	FTP	106.2 ± 23.64sec	26.2 ± 3.46sec
	FTP-ssl	9804.5 ± 351.46sec	5259.2 ± 6.18sec
Put Linux	SCP	293.6 ± 43.34sec	956.2 ± 12.19sec
	rsync-ssh	275.4 ± 83.71sec	276.6 ± 84.44sec
	FTP	284.0 ± 4.46sec	2526.4 ± 14.59sec
	FTP-ssl	2582.9 ± 25.01sec	2335.7 ± 4.96sec
Get Linux	SCP	157.5 ± 2.48sec	838.2 ± 30.06sec
	rsync-ssh	17.1 ± 12.68sec	14.7 ± 13.99sec
	FTP	55.4 ± 41.70sec	1939.8 ± 13.86sec
	FTP-ssl	3216.1 ± 27.74sec	2312.4 ± 9.21sec
Put tar	SCP	177.1 ± 34.41sec	174.7 ± 36.51sec
	rsync-ssh	177.9 ± 36.24sec	175.1 ± 39.00sec
	FTP	141.629 ± 33.6sec	162.3 ± 4.02sec
	FTP-ssl	399.0 ± 5.67sec	258.5 ± 3.71sec
Get tar	SCP	98.6 ± 1.90sec	94.5 ± 3.52sec
	rsync-ssh	99.3 ± 3.65sec	102.7 ± 3.49sec
	FTP	95.2 ± 1.52sec	45.7 ± 37.65sec
	FTP-ssl	77.5 ± 32.87sec	190.5 ± 8.06sec

Table 4: Duration of operations of non-file system tests in the real-life setup.

networks. For metadata intensive operations, one can observe a big performance drop between local and wide area networks. In case of large file transfer, the difference of results of local vs. wide area operation is dependent more on various load of interconnecting shared network than on the latency difference.

Figure 11 shows all comparable operations using all tested protocols over the local area network. In this case, FTP without data encryption and NFS without full encryption perform the best for a large file. FTP without data encryption and rsync over ssh perform the best for small files. These values are still a bit slower than in the laboratory testing which can be attributed to the shared network infrastructure.

Figure 12 shows all comparable operations using all tested protocols over the wide area network. In this case, all protocols except FTP with data encryption and NFS with full encryption are quite comparable for a large file. For small files, the best protocol is rsync

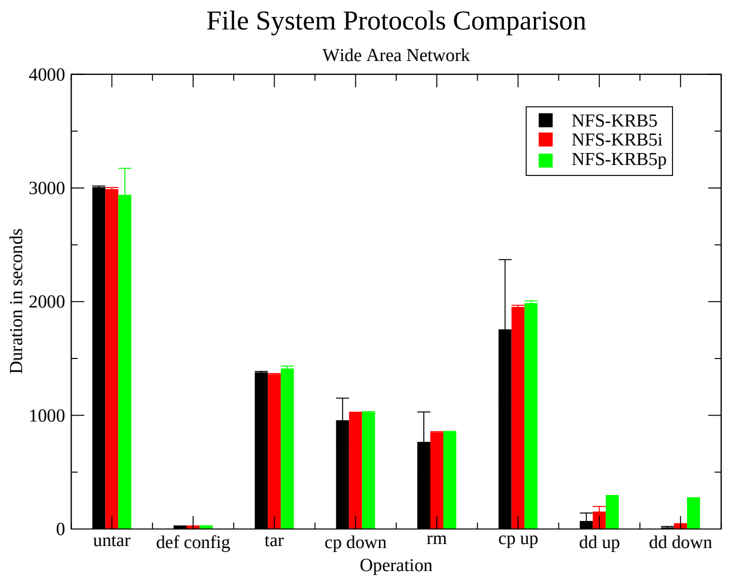

Figure 10: Duration of operations of file system tests over the wide area network in the real-life setup.

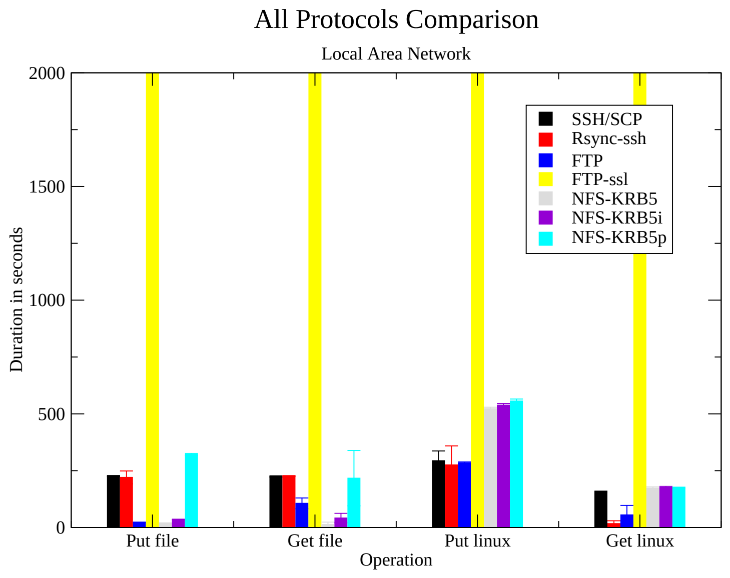

Figure 11: Duration of operations of all tests over the local area network in the real-life setup.

Operation	Protocol	LAN	WAN
untar	NFS-KRB5	759.1 ± 105.84sec	3003.2 ± 14.04sec
	NFS-KRB5i	772.1 ± 9.84sec	2985.8 ± 18.27sec
	NFS-KRB5p	789.1 ± 11.85sec	2935.5 ± 237.34sec
defconfig	NFS-KRB5	7.1 ± 0.39sec	26.6 ± 0.51sec
	NFS-KRB5i	7.0 ± 0.24sec	26.4 ± 0.32sec
	NFS-KRB5p	8.0 ± 0.44sec	27.3 ± 0.86sec
tar	NFS-KRB5	271.2 ± 4.72sec	1374.2 ± 12.18sec
	NFS-KRB5i	275.9 ± 2.49sec	1356.9 ± 10.35sec
	NFS-KRB5p	327.5 ± 22.68sec	1408.1 ± 24.73sec
cp down	NFS-KRB5	172.7 ± 5.94sec	952.3 ± 198.52sec
	NFS-KRB5i	176.4 ± 4.48sec	1020.4 ± 7.08sec
	NFS-KRB5p	174.2 ± 3.29sec	1023.0 ± 9.43sec
rm	NFS-KRB5	182.5 ± 5.31sec	762.1 ± 267.79sec
	NFS-KRB5i	188.3 ± 4.98sec	846.9 ± 9.33sec
	NFS-KRB5p	186.1 ± 4.01sec	850.6 ± 8.37sec
cp up	NFS-KRB5	522.6 ± 6.33sec	1751.7 ± 617.76sec
	NFS-KRB5i	538.0 ± 7.40sec	1949.4 ± 18.61sec
	NFS-KRB5p	556.1 ± 9.09sec	1982.7 ± 23.26sec
dd up	NFS-KRB5	18.0 ± 2.36sec	66.3 ± 73.91sec
	NFS-KRB5i	35.0 ± 1.51sec	149.0 ± 49.09sec
	NFS-KRB5p	321.2 ± 4.54sec	293.4 ± 1.30sec
dd down	NFS-KRB5	13.2 ± 10.55sec	9.9 ± 12.09sec
	NFS-KRB5i	41.6 ± 20.65sec	41.4 ± 5.40sec
	NFS-KRB5p	216.8 ± 121.59sec	273.9 ± 0.07sec

Table 5: Duration of operations of file system tests in the real-life setup.

over ssh followed by SCP. All variants of NFS are comparable for small files over the wide area network.

Table 6 summarizes comparison of all protocols over both local and wide area networks. Differences between local and wide area networks are significant only in the FTP with data encryption protocol for a large file. In case of many small files, significant differences are with FTP protocol without data encryption, NFS in all variants and SCP; rsync over ssh seems to be the only appropriate protocol for small files.

5 Conclusions

In this report, we present an overview of various data transfer protocols and their sensitivity to network latency. We evaluated a set of protocols that were expected to be available on CESNET data storage infrastructure. We created a test laboratory to evaluate unbiased

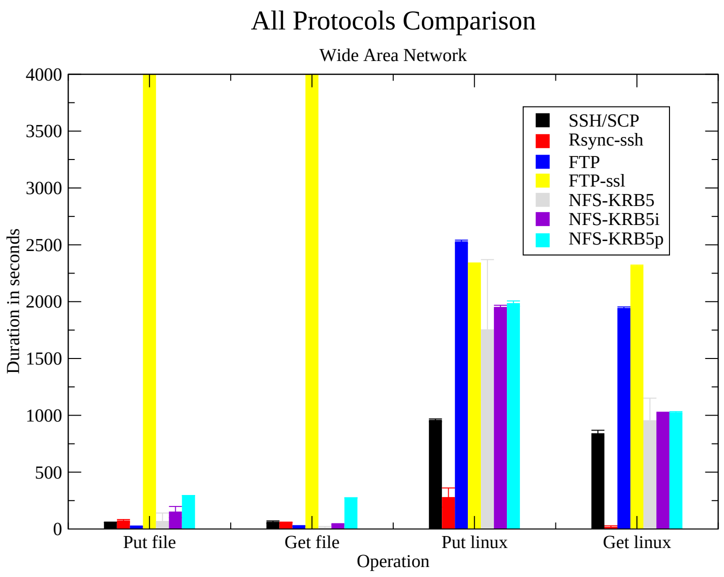

Figure 12: Duration of operations of all tests over the wide area network in the real-life setup.

behavior of protocols, then we restricted the tests to the protocols that were actually available and evaluated them in similar manner in real-world conditions. All the measurements meet the expected result: synchronous operations are highly sensitive to network latency. Consequently, file system operations with file metadata (size, attributes, creating file, etc.) are very slow over wide area networks, while pure data transfers (i.e., large files) are not significantly slowed down by high latency when appropriate protocol is chosen. Problems with synchronous operations over wide area network can be mitigated using proper protocol such as rsync.

We also observed that even powerful CPUs have still problems with on-the-fly data encryption, mainly when using non-trivial encryption such as AES or 3DES. With such an encryption, we measured significant throughput drop.

We expect that vast majority of clients of the CESNET storage systems will be connected through wide area network. For such clients, we generally recommend to modify their workflow in such a way that smaller numbers of large files can be transferred. Throughput of such data is nearly independent on the protocol in use. If no other way than storing large quantities of small files is possible, then rsync should be used, outperforming other available protocols significantly.

Operation	Protocol	LAN	WAN
Put file	SCP	226.6 ± 1.97sec	58.3 ± 2.02sec
	rsync-ssh	220.3 ± 28.43sec	71.5 ± 11.79sec
	FTP	22.0 ± 1.57sec	22.1 ± 4.10sec
	FTP-ssl	2759.5 ± 1.04sec	5166.4 ± 10.40sec
	NFS-KRB5	18.0 ± 2.36sec	66.3 ± 73.91sec
	NFS-KRB5i	35.0 ± 1.51sec	149.0 ± 49.09sec
	NFS-KRB5p	321.2 ± 4.54sec	293.4 ± 1.30sec
Get file	SCP	224.5 ± 2.82sec	60.2 ± 11.14sec
	rsync-ssh	226.2 ± 2.15sec	57.7 ± 2.95sec
	FTP	106.2 ± 23.64sec	26.2 ± 3.46sec
	FTP-ssl	9804.5 ± 351.46sec	5259.2 ± 6.18sec
	NFS-KRB5	13.2 ± 10.55sec	9.9 ± 12.09sec
	NFS-KRB5i	41.6 ± 20.65sec	41.4 ± 5.40sec
	NFS-KRB5p	216.8 ± 121.59sec	273.9 ± 0.07sec
Put Linux	SCP	293.6 ± 43.34sec	956.2 ± 12.19sec
	rsync-ssh	275.4 ± 83.71sec	276.6 ± 84.44sec
	FTP	284.0 ± 4.46sec	2526.4 ± 14.59sec
	FTP-ssl	2582.9 ± 25.01sec	2335.7 ± 4.96sec
	NFS-KRB5	522.6 ± 6.33sec	1751.7 ± 617.76sec
	NFS-KRB5i	538.0 ± 7.40sec	1949.4 ± 18.61sec
	NFS-KRB5p	556.1 ± 9.09sec	1982.7 ± 23.26sec
Get Linux	SCP	157.5 ± 2.48sec	838.2 ± 30.06sec
	rsync-ssh	17.1 ± 12.68sec	14.7 ± 13.99sec
	FTP	55.4 ± 41.70sec	1939.8 ± 13.86sec
	FTP-ssl	3216.1 ± 27.74sec	2312.4 ± 9.21sec
	NFS-KRB5	172.7 ± 5.94sec	952.3 ± 198.52sec
	NFS-KRB5i	176.4 ± 4.48sec	1020.4 ± 7.08sec
	NFS-KRB5p	174.2 ± 3.29sec	1023.0 ± 9.43sec

Table 6: Duration of operations of all tests in the real-life setup.

References

[1] CIFS: A common internet file system. P. Leach, D. Perry. `http://www.microsoft.com/mind/1196/cifs.asp`. 1996.

[2] RFC 3530: Network File System (NFS) version 4 Protocol. S. Shepler et al. RFC 3530, April 2003.

[3] RFC 3649: HighSpeed TCP for Large Congestion Windows. Sally Floyd. RFC 3649, Experimental, December 2003.

[4] RFC 4217: Securing FTP with TLS. P. Ford-Hutchinson. RFC 4217. October 2005.

[5] http://rsync.samba.org.

[6] http://en.wikipedia.org/wiki/Secure_copy.