RAA 2021 Vol. **21** No. **3**, 54(9pp) doi: 10.1088/1674-4527/21/3/54
© 2021 National Astronomical Observatories, CAS and IOP Publishing Ltd.
http://www.raa-journal.org *http://iopscience.iop.org/raa*

Research in Astronomy and Astrophysics

NBFTP: a dedicated data transfer system for remote astronomical observation at Dome A [⋆]

Si-Yuan Huang (黄思远)[1], Ce Yu (于策)[1,2], Chao Sun (孙超)[1,2⋆⋆], Yi Hu (胡义)[3⋆⋆], Zhaohui Shang (商朝晖)[3,4⋆⋆⋆], Bin Ma (马斌)[3⋆⋆⋆], Ming Che (车明)[1] and Xiao-Xiao Lu (鹿孝孝)[1]

[1] College of Intelligence and Computing, Tianjin University, Tianjin 300350, China; *sch@tju.edu.cn*
[2] NAOC-TJU Joint Research Center in Astro-Informatic, Tianjin 300350, China
[3] National Astronomical Observatories, Chinese Academy of Science, Beijing 100101, China; *huyi.naoc@gmail.com*
[4] Tianjin Normal University, Tianjin 300387, China

Received 2020 April 29; accepted 2020 August 17

Abstract Dome A, Antarctica, has been thought to be one of the best astronomical sites on the Earth for decades. Since it was first visited by astronomers in 2008, dozens of facilities for astronomical observation and site testing were deployed. Due to its special geographical location, the data and message exchange between Dome A and the domestic control center could only depend on Iridium. Because the link bandwidth of Iridium is extremely limited, the network traffic cost is quite expensive and the network is rather unstable, the commonly used data transfer tools, such as rsync and scp, are not suitable in this case. In this paper, we design and implement a data transfer tool called NBFTP (narrow bandwidth file transfer protocol) for the astronomical observation of Dome A. NBFTP uses a uniform interface to arrange all types of data and matches specific transmission schemes for different data types according to rules. Break-point resuming and extensibility functions are also implemented. Our experimental results show that NBFTP consumes 60% less network traffic than rsync when detecting the data pending to be transferred. When transferring small files of 1KB, the network traffic consumption of NBFTP is 40% less than rsync. However, as the file size increases, the network traffic consumption of NBFTP tends to approach rsync, but it is still smaller than rsync.

Key words: Astronomical instrumentation, methods and techniques — techniques: miscellaneous — telescopes — site testing

1 INTRODUCTION

Astronomical observation always needs the most strict requirements of the experimental environment. Modern astronomical sites are usually selected as the most isolated places on the Earth, to get rid of the influence of human activities and obtain supreme observational conditions. For example, Dome A, the highest point on the Antarctic inland ice cap, is potentially one of the best astronomical observation sites on the planet (Saunders et al. 2009; Hu et al. 2014, 2019). With its extremely stable atmosphere and long polar nights, the site condition at Dome A could even be comparable to that in the space. The 21st Chinese National Antarctic Research Expedition (CHINARE) reached Dome A for the first time in 2005, and since the 24th CHINARE reached Dome A for the second time, dozens of facilities for astronomical observation and site testing were deployed (Yuan et al. 2008, 2010; Bonner et al. 2010; Hu et al. 2014; Shi et al. 2016; Ma et al. 2018; Shang et al. 2018).

However, because of the harsh environment and logistical difficulties, Dome A is still a completely unattended site, which means that there will be no human on-site to operate any facilities after the expedition team leaves at the end of every January (Hu et al. 2019). This situation will still be true for the foreseeable future. Although the facilities installed at Dome A were designed

[⋆] The source code has been uploaded on gitee (`https://gitee.com/AstroTJU/NBFTP`) and china-vo (`https://code.china-vo.org/ast3/nbftp`).
[⋆⋆] Corresponding authors
[⋆⋆⋆] The ORCIDs of Yi Hu, Zhaohui Shang and Bin Ma are `0000-0003-3317-4771`, `0000-0002-6796-124X` and `0000-0002-6077-6287`, respectively.

to operate as automatically as possible (Hu et al. 2016), it is still inevitable to transmit data back and forth between Dome A and the domestic control center (DCC). Thus, an effective method of data transfer is quite crucial for safely operating the remote astronomical observatory at Dome A (Zhang et al. 2016; Kubánek 2016).

As a fast and versatile incremental file transfer tool, rsync is probably the best choice for data transfer under various network conditions (Shial et al. 2015). Rsync is known for its delta transfer algorithm. It firstly compares the files in both source and target, and then only sends the differences between them so that it reduces the amount of data to transfer[1]. However, this tool does not achieve network traffic economization in all the cases, because it does not truly implement the network traffic control. When transferring small files, rsync performs unsatisfactorily due to the overhead of the file comparison and incremental calculations. It will also consume extra network traffic when checking the differences directories between the source and target.

Located in the innermost of the continent, Dome A has no other Internet access except Iridium (Lawrence et al. 2008; Jia et al. 2018). The power and Iridium connection for instruments are provided by an automated observatory platform PLATO-A (Ashley et al. 2010). Because the Iridium network communication is not only unstable but also very expensive (Shang et al. 2012), maximizing the utility of the Iridium channel is quite important for us to operate our observatory at Dome A. Most of the data transferred from Dome A to the DCC are system log files, stamp images, site-testing data, and alarm messages (Hu et al. 2016). The typical size of these data is around 10 KB. As we showed above, rsync will use the Iridium channel inefficiently because of its severe overhead. Hence, it is urgent to develop a new tool for transferring data, especially for files with a small size under Iridium network communication.

Therefore, we design NBFTP, which is an intelligent data transfer system that achieves fine-grained network traffic control. The NBFTP system is composed of a sender at Dome A and a receiver at the DCC. It can effectively transmit the data from the observatory at Dome A to the DCC with user-provided priority.

The rest of this article is organized as follows. We show the related works on data transfer in Section 2. We then describe the architecture of NBFTP and its detailed information on modules and algorithms in Section 3. We show the performance of NBFTP and compare NBFTP with other file transfer tools in Section 4. In the final section, we summarize and discuss future work.

[1] `https://linux.die.net/man/1/rsync`

2 RELATED WORK

Data transfer is one of the basic functions of computer networks. There are mature file transfer protocols such as File Transfer Protocol (FTP) and SSH File Transfer Protocol (SFTP), and most Linux distributions have integrated practical tools, such as rcp, scp, rsync, etc.

FTP is a standard network protocol for transfer files between clients and servers on a computer network[2]. The main advantage of FTP is that it comes with the most commonly used operating systems, including most Linux distributions and Microsoft Windows. Most FTP-based tools support resuming from break-points. SFTP is a component of Secure Shell (SSH), which supports encrypted file transfer and resuming[3]. The encryption and decryption technology of SFTP makes it more secure than FTP, but the process also creates another kind of overhead. Besides, file transfer based on FTP or SFTP requires interactive operations even when used in automated scripts. These interaction operations will cause extra network traffic.

As a remote file copy tool, rcp can transfer files or directories between different hosts[4]. Scp encrypted with SSH is a more secure version of rcp[5]. Both tools copy all the files in the source directory, including those that have been transferred. Thus, they may introduce unnecessary traffic consumption. Besides, neither tools support break-point resuming.

Rsync is designed to synchronize file directories between different hosts[6]. Its incremental transfer algorithm reduces the amount of data transmitted over the network by only sending the different parts between the source files and the existing files in the destination. This approach makes it consume less network traffic than rcp and scp do. Unlike rcp and scp, rsync supports break-point resuming. However, the incremental algorithm of rsync is not suitable for the situation where the number of files keeps increasing. Every time rsync is called, it will check all subdirectories and files in the directories at both ends, causing unnecessary overheads. Rsync supports resuming by storing extra information in a temporary file. Therefore, it needs to perform additional file operations and information calculations every time it transfers a file. Such a resuming procedure may decrease the transfer efficiency.

[2] `https://en.wikipedia.org/wiki/File_Transfer_Protocol`

[3] `https://en.wikipedia.org/wiki/SSH_File_Transfer_Protocol`

[4] `https://en.wikipedia.org/wiki/Berkeley_r-commands`

[5] `https://en.wikipedia.org/wiki/Secure_copy`

[6] `https://linux.die.net/man/1/rsync`

3 TRAFFIC-SAVING DATA TRANSFER SYSTEM

3.1 NBFTP Software Architecture

Our work focuses on transferring data from Dome A to the DCC stably and cost-effectively. As shown in Figure 1, NBFTP consists of a sender and a receiver. The sender of NBFTP is deployed at the Dome A. It includes a Monitor Module (MM), a Task Module (TM), and a Data Module (DM). The receiver is deployed at the DCC, which mainly includes a TM, a DM, and an Expansion Module (EM). It is loosely-coupled between the components which can effectively reduce the scale and complexity of the system. Although it is the sender that is transferring data to the receiver, the receiver can also transfer files to the sender.

Once the MM detects a file to transfer, it will add a transmission task to the task queue. The TM is responsible for managing the task queue, such as sending and receiving data, sorting the tasks in the queues according to priorities. We will interpret task types in Section 3.2. The DM provides a series of functions for packing data to or extracting from the buffer. The EM is used by the receiver to complete other functions using shell commands after receiving a file or message, such as sending mail and importing the received data to a database.

Overall, the main functions of NBFTP include:

– Optimizing data size/actual network traffic. In the TM, NBFTP intelligently matches the throttling transmission scheme according to the size of the data to be transmitted and assigns transmission tasks.
– Resuming break-points. In the TM, all tasks will be tracked to the block level. Only after each block is successfully transmitted can the next block be transferred. So that even after disconnection, the last position of the successful transmission can be found to start again.
– Extensibility functions. In the EM, we can start the automatic transaction processing, such as sending an email, by modifying the configuration file of the domestic receiver.

NBFTP is a tool for transferring data between Dome A and the DCC. It is an open-source project running on the Linux platform, coded in C++, with the following dependencies: GNU C Library, GNU GLIB 2.0, C Shell, Gamin. There are two programs of the nbftp and nbftp-server in our project. The nbftp-server is a daemon process, which starts immediately after the operating system is booted up. Given different configuration parameters, nbftp-server can act as either the sender or the receiver. Users can execute nbftp commands, which is the client of nbftp-server, instead of directly manipulating the file directory. The operations of sending data and receiving data are different. The specific instructions and their meanings are tabulated in Tables 1 and 2.

3.2 Transfer Protocol

The TM of NBFTP is carefully designed to use unified interfaces to arrange all types of data, corresponding to eight task types: SMALL FILE, LARGE FILE, MESSAGE, LASTPOS, COMPLETE SIG, CHECK PASS, CHECK FAILED and REPLY. The first three types are used to transfer small files, large files, and messages, while the latter five are used for transmitting feedback information to ensure the stability of the transmission. During a transmission process, each task of the sender will be packaged into one or more RawData segments, which are the smallest units of transmission. Then the receiver will resemble it. Different tasks follow different packing rules. The communication protocol of this transmission system is tabulated in Table 3.

For all the types of tasks, the first byte in NBFTP stores the type name, and the following four bytes store the ID of the task. The subsequent information will be different and correspond one-to-one in Table 3. For instance, when transferring a small size file, the TM of the sender uses only one RawData segment. It firstly sets the task type as SMALL FILE in the first byte and assigns a task ID. Then the TM calculates the MD5 checksum of the file and fills the value into the 16-byte MD5 field of the segment. Finally, it fills the length of the file name, the file name, the size of the file, and the file content into the segment. After these steps, the TM sends the RawData segment to the receiver. The TM of the receiver receives and extracts the segment, writes the content to a new file. Then it calculates the MD5 of the file and compares it with that in the MD5 field. If the file check is successful, the TM acknowledges a CHECK PASS segment to the sender to signify the success of transmission. Otherwise, it acknowledges a CHECK FAILED segment to the sender to ask for retransmitting.

It is worth mentioning that sending LARGE FILE tasks uses two types of RawData. The former is the LARGE FILE (begin), which contains metadata for large files, marking the beginning position of a large file chunked transfer task. Large files are divided into blocks, and each block uses the latter type of RawData named LARGE FILE (part), which is then transferred in turn. If the receiver TM receives a LARGE FILE (begin) or LARGE FILE (part) other than the last segment, it acknowledges a REPLY segment to the sender.

The five kinds of feedback information are sent by the receiver to indicate the current data reception status to the sender. Each of them uses 4 bytes to store the associated task ID. In particular, the LASTPOS uses 4

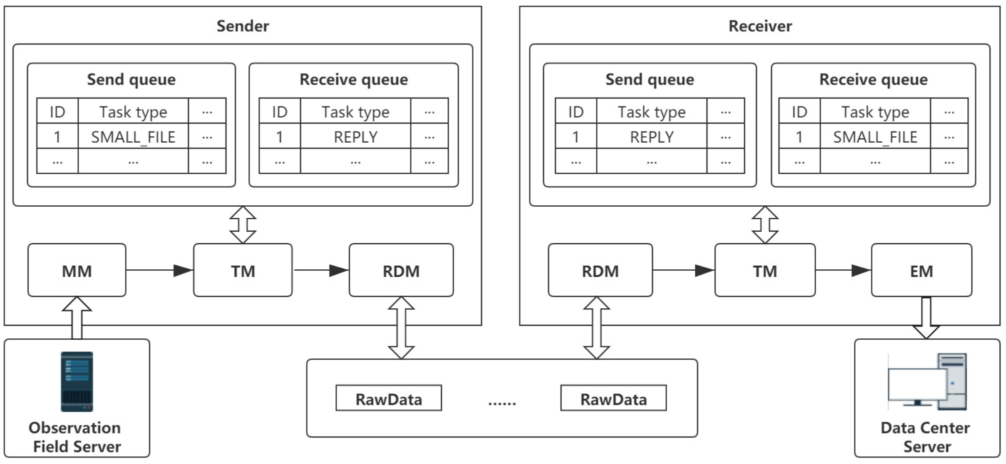


Fig. 1 The software architecture of NBFTP.

Table 1 Operations of Sending Data

Instruction	Meaning	Remark
nbftp push FILENAME [PRIORITY]	Submit a file to transfer	The default priority is 5
nbftp mail MESSAGE	Submit a message to transfer	The default priority is 3
nbftp pull TASKID	Cancel file transfer task	
nbftp pull all	Cancel all transfer tasks	
nbftp pending	List pending files	

Table 2 Operations of Receiving Data

Instruction	Meaning	Remark
nbftp list	List currently received files	
nbftp get TASKID	Save the file to the current directory or show message	
nbftp delete TASKID	Delete the reception record	
nbftp delete all	Delete all reception records	

bytes to show the current transmission position, which can help to implement the break-point resuming function.

3.3 Break-point Resuming Function

Since the Iridium network communication is extremely unstable, the packet loss must happen frequently. Therefore, any file transfer tools without supporting the break-point resuming function are not suitable for using at Dome A.

NBFTP implements break-point resuming function by dividing files into blocks. It then encapsulates the blocks into RawData segments, as tabulated in Table 3. RawData segment is the smallest unit to transmit in NBFTP, which means it will be either delivered successfully or discarded completely. As we demonstrated in Section 3.2, the sender of NBFTP issues only a single RawData segment for a small size file. On the contrary, for a large size file, the sender will firstly issue a LARGE FILE (begin) segment which contains only the metadata of that file; it then delivers the file body of the file by issuing LARGE FILE (part) segments. Each LARGE FILE (part) contains a “current position” field to record the position of the data block of the file. Whenever the sender delivers a segment, it will wait for a REPLY acknowledgment from the receiver. If the delivery fails or waiting for REPLY acknowledgment is timed out, the sender will suspend for a while and try to send the LARGE FILE (part) segment in the next transmission. In this case, the receiver will acknowledge a LASTPOS RawData segment rather than a REPLY segment to the sender. This segment also contains a “current position” field to signify to the sender to restart to transfer the file from that position. By applying the LASTPOS segment, NBFTP implements break-point resuming.

Obviously, the size of the block will significantly influence the transmission efficiency of NBFTP. When the loss rate is high in a bad network condition, adopting a large block size will more likely waste the cost of Iridium network communication. On the other hand, using a small

Table 3 NBFTP Transfer Protocol

RawData type	Data length (Byte)	Content
SMALL FILE	1	sign: SMALL FILE DATA
	4	task id
	16	MD5
	1 + name_length	file name
	4 + file_size	binary file content
LARGE FILE (begin)	1	sign: LARGE FILE BEGIN
	4	task ID
	16	MD5
	1 + name_length	file name
	4	file size
LARGE FILE (part)	1	sign: LARGE FILE PART
	4	task ID
	4	current position
	4 + file_block_size	binary file block content
MESSAGE	1	sign: MESSAGE DATA
	4	task ID
	1 + message_length	message
LASTPOS / COMPLETE SIG / CHECK PASS / CHECK FAILED / REPLY	1	sign: LASTPOS / COMPLETE SIG / CHECK PASS / CHECK FAILED / REPLY
	4	task ID
	4	associated task ID
	4	current position (LASTPOS only)

block size will introducing extra protocol overhead when encapsulating the RawData segments. Since this trade-off is strongly dependent on the network condition, we use an adaptive block size mechanism inspired by TCP congestion control[7]. The block size is set to 2^{13} bytes at the beginning. After the previous block is successfully transmitted, which means that the network may be in good condition, then the block size will be doubled, reducing the block prefix consumption. If it reaches a maximum of 2^{22} bytes, it will not increase. If a block loss occurs midway, it means that the network environment becomes poor and the packet loss rate is increased, then the block size will shrink to reduce extra network traffic. The block size will be adjusted according to the network conditions, which is more suitable for unstable network environments. By dynamical determining of the block size, NBFTP achieves a fine controlling network traffic.

3.4 Extensibility Function

In addition to transferring data with low network traffic consumption, NBFTP also provides an interface for users to implement extensibility functions, which are mainly used for subsequent processing after receiving data. After accepting a certain type of data, the receiver can automatically call other programs according to the configuration file, including but not limited to the following functions: proofreading data correctness, sending an short message service (SMS) or email to the specified location, importing the data in the file to the database, etc.

[7] http://www.hjp.at/doc/rfc/rfc5681.html

The implementation details of the extensibility functions are shown in Table 4.

3.5 Autonomous Data Transmission

In order to implement fully automatic data transmission, it is necessary to ensure that the system can continue to be stable under unmanned operation and poor network conditions to ensure the integrity of the data. At the same time, special events can be sent to the DCC as soon as possible, to achieve a certain degree of resilience.

NBFTP implements fully automatic data transmission through MM. The main functions are shown in Table 5.

4 EXPERIMENTS AND RESULTS

4.1 Experimental Settings

Astronomical observations at Dome A require NBFTP to support fully automatic continuous operation for a whole year. According to the actual needs of data transmission, we analyzed and evaluated the performance of NBFTP in the following aspects: single file transfer, periodic file transfer, directory synchronization overhead, and file block parameter settings.

Most of the files transferred from Dome A are system logs, site-testing data, and alarm messages. The system logs and site-testing data will be transferred periodically. The compressed file size of the system log and site-testing data is about 8 KB, and the size of the alarm message is less than 1 KB. The transmission of observation data will be carried out as needed, and the transmission frequency

Table 4 Functions of the Expansion Module

Basic features of the Expansion Module	Notes
Prerequisite	File reception completed
Configuration setting	Set the command with specific parameters in the configuration file
Supported program languages	C/C++/Shell/Java/Python/Perl/...
Supported function	Mail sending/Data cleaning/Data archiving/SMS/...

Table 5 Functions of the Monitor Module

Event	Response
Generate new observations	Start the data transfer process
Generate specific types of data	Set a specific action (e.g. send back to country with highest priority)
Tasks often terminate abnormally	Restart or terminate the task as needed

and file size cannot be determined in advance. In order to simulate Iridium network communication, we limit the network bandwidth to 128 Kbps.

As the main competitor, rsync is set to run in daemon mode, which requires less network transmission than the interactive shell mode.

4.2 Single File Transfer

Single file transfer is the basic function of NBFTP. We compared the network traffic of NBFTP, rsync, and scp when transferring single files of different sizes, as shown in Figure 2.

For the system log and site-testing data that need to be transferred periodically, a single file is usually less than 8 KB. When transmitting such data, NBFTP generates less network traffic than rsnyc and scp. The reason why the actual amount of data transferred by rsync is larger is that it requires additional operations to determine which files need to be transferred. The additional data transmission required by the scp method mainly comes from the encryption of the data transmission process. When transferring larger files, the amount of network traffic required by the three methods tends to be the same.

4.3 Directory Synchronization Overhead

During long-term synchronization, most files in the source directory are most likely the same as files in the remote destination directory. As mentioned earlier, rsync uses algorithms to determine the difference between the source and the target, thereby reducing network traffic. However, the frequent comparisons of the same files will consume unnecessary network traffic. The more files exist in both the source and destination directory, the more extra network traffic RSYNC consumes when detecting duplications.

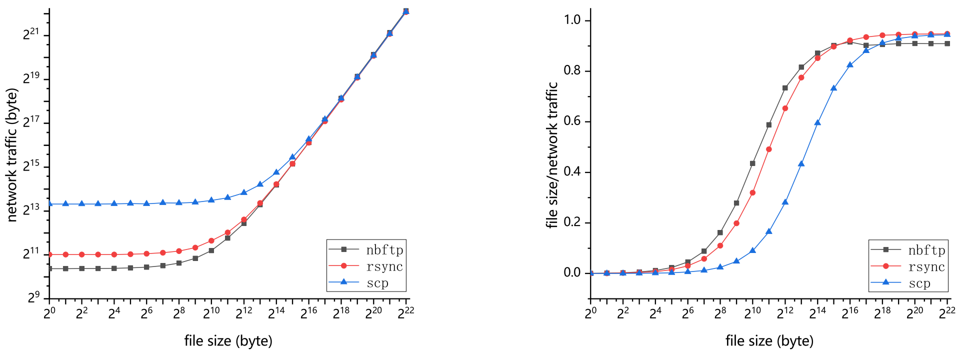
Figure 3 validates our conjecture. In each test case, the file sizes are also 1 KB, 8 KB, and 64 KB. In the beginning, the source and destination directories hold the same files, which means all files have been synchronized. Then we use NBFTP and rsync to transfer them again. The results show that the network traffic of rsync is about 2.4 times that of NBFTP on average, and increases when the size and number of synchronized files doubles, while the network traffic of NBFTP is almost unchanged. One method to reduce the overhead of using rsync is deleting the transferred files in the source directory, but it will cause potential data security issues. Overall, the directory synchronization overhead of NBFTP is less than rsync, so NBFTP is more suitable for long-term transmission between Dome A and DCC.

4.4 Periodically Generated Small Files

As mentioned earlier, the site-testing data will be transferred from Dome A periodically. In this section, we will simulate the transmission process of the actual situation. Small files are generated into the source directory at regular intervals other than immediately put into the queue to transmit. Just like the above experiments, files are of 1 KB, 8 KB, and 64 KB respectively. Figure 4 shows the network traffic when the number of files reached different levels.

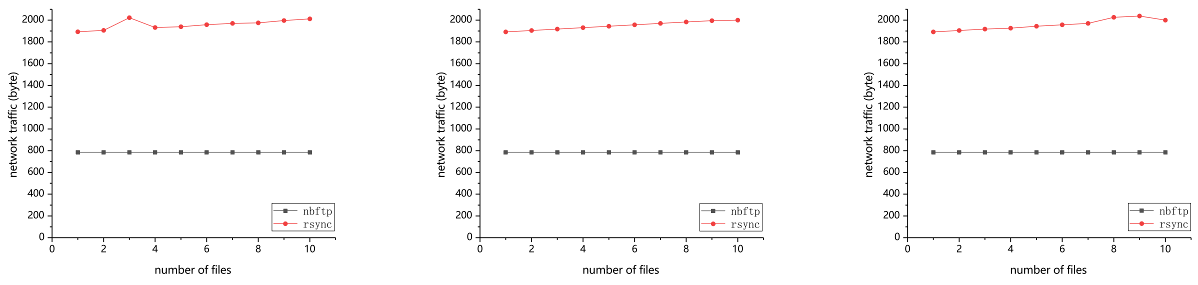
This experiment can be regarded as a combination of the above two experiments. On the one hand, the files are generated at regular times, and each generation will trigger a transmitting operation to transfer files one by one, corresponding to the first experiment. On the other hand, the number of transferred files doubles over time, and there can be a lot of duplicates in the source and target directories, which is like the second experiment.

From Figure 4, it can be seen that when the sizes of files to be transferred are 1KB and 8KB, the network traffic performance of NBFTP is better than rsync due to the same reason shown in Figure 2. As the number of files increases, NBFTP performs better than rsync because the latter need to check more and more same files, which



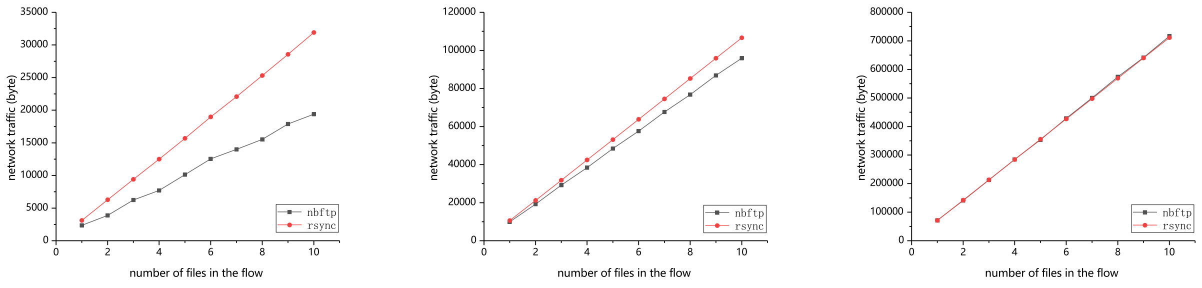
(a) network traffic (b) file size / network traffic

Fig. 2 The actual network traffic transmitting a single file of different sizes. The network traffic refers to the total size of the IP packets transmitted by the sender and receiver. When the size of the target file is less than 2^{15} bytes, NBFTP requires the least amount of network traffic. When the file size exceeds 2^{15} bytes, the efficiency of these three methods tends to be the same.



(a) 1 KB (b) 8 KB (c) 64 KB

Fig. 3 The actual network traffic transmitting multiple small files that have already been transferred.



(a) 1 KB (b) 8 KB (c) 64 KB

Fig. 4 The actual network traffic transmitting periodically generated small files of different sizes.

is shown in Figure 3. The network traffic performance of NBFTP improves as the number of files increases but decreases as the file size increases. In general, NBFTP can save more network traffic, because its performance is almost the same as rsync when transferring large files, and in the case of transferring smaller files, the efficiency of NBFTP is better than rsync.

4.5 Impact of Block Size in NBFTP

NBFTP divides large files into blocks for transmission. The file block size setting will affect the network traffic during transmission. When transferring files of the same size, the larger the block size, the smaller the number of blocks, the less space the header file takes up, and the less network traffic it consumes. Figure 5 shows the network

Table 6 Function Comparison of Several Data Transfer Tools

Function	scp	rsync	NBFTP
Large file transfer	Send the entire file and only receive a confirmation message	Compare the difference between the source and the target, and only transfer the difference	Split files and receive a confirmation message every time a block is sent
Break-point resuming	Unsupported	Supported	Supported
Network traffic control	Unsupported	Unsupported	Supported
Guarantee of file integrity	Unsupported	Unsupported	Supported
Encryption	SSH	Unsupported	OpenSSL

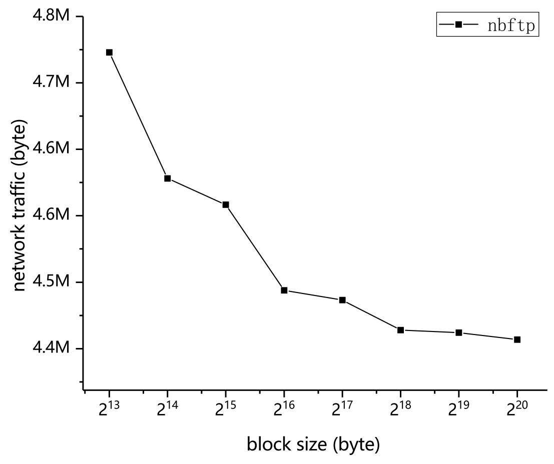


Fig. 5 Impact of block size in NBFTP.

traffic of transferring 4MB files with different block sizes using NBFTP. The experimental results are as expected.

However, large blocks cause high costs for re-transmissions. In order to save network traffic in the unstable Iridium network environment, NBFTP uses an adaptive block size mechanism to select the optimal block size according to the network situation. The specific introduction of the adaptive block size mechanism is in Section 3.

5 CONCLUSION

In this paper, we designed a dedicated data transfer system called NBFTP for remote astronomical observation at Dome A, Antarctic. NBFTP provides a highly stable, traffic-saving, and easy-to-use data channel by optimizing the ratio of file size to network traffic, resuming break-points, and implementing extensibility.

Iridium communication link is narrow-band and unstable, and the network traffic price is high. The retransmission mechanism of existing tools can cause extra network traffic. To this end, NBFTP controls the traffic at the block level, implements the break-point resuming function based on the adaptive block size mechanism, and effectively controls the overhead caused by retransmissions. Our experimental results show that NBFTP is superior to rsync and scp in data traffic. It consumes 60% less network traffic than rsync when detecting the data pending to be transferred. When transferring continuously generated 1KB files, NBFTP consumes 40% less traffic than rsync.

The reasons why NBFTP has transmission efficiency come from three levels of optimization. Firstly, NBFTP adopts a special transmission protocol that consider different file sizes to reduce extra data (see Sect. 3.2). Secondly, it uses a fine-grained data retransmission mechanism that reduces the overhead of data retransmission under unstable network conditions (see Sect. 3.3). Finally, when doing remote directory synchronization, NBFTP can avoid the overhead of full directory scanning required by rsync, based on local logs (see Sect. 4.3).

In addition, existing tools have only basic data transfer functions and cannot support automatic transaction processing. NBFTP implements an EM with triggering functions such as mail sending, data cleaning, and data archiving, which can provide more support for remote astronomical observation.

NBFTP and its data transfer protocol are specially designed for remote astronomical observations based on narrow bandwidth and an unstable Iridium communication network. Table 6 shows the comparison of related functions of NBFTP and other commonly used tools mentioned in Section 2.

A stable version of NBFTP has been serving the data transfer of KLAWS, KCLAM, KL-DIMM at Dome A. NBFTP can also be helpful for site testing and early-stage astronomical observation at other places where such narrow-band and unstable Iridium communication is the only choice. In future work, we will focus on more fine-grained network traffic control and try to expand it to support short message communication of BeiDou.

Acknowledgements This work is supported by the Joint Research Fund in Astronomy (U1931130) under the cooperative agreement between the National Natural Science Foundation of China (NSFC) and the Chinese Academy of Sciences (CAS). HU, SHANG, and MA acknowledge support from NFSC (Grant Nos. 11873010 and 117330037), and the Operation, Maintenance and

Upgrading Fund for Astronomical Telescopes, and Facility Instruments, budgeted from the Ministry of Finance of China (MOF) and administrated by the CAS.

References

Ashley, M. C. B., Bonner, C. S., Everett, J. R., et al. 2010, Proc. SPIE, 7735, 773540
Bonner, C. S., Ashley, M. C. B., Cui, X., et al. 2010, PASP, 122, 1122
Hu, Y., Shang, Z. H., Ma, B., & Hu, K. 2016, Proc. SPIE, 9913, 99130M
Hu, Y., Shang, Z. H., Ashley, M. C. B., et al. 2014, PASP, 126, 868
Hu, Y., Hu, K. L., Shang, Z. H., et al. 2019, PASP, 131, 015001
Jia, M. H., Chen, Y. Q., Zhang, G. Y., et al. 2018, Astronomy and Computing, 24, 17
Kubánek, P. 2016, in Software and Cyberinfrastructure for Astronomy IV, 9913, International Society for Optics and Photonics, 99132U
Lawrence, J. S., Allen, G. R., Ashley, M. C. B., et al. 2008, Proc. SPIE, 7012, 701227
Ma, B., Hu, K. L., Hu, Y., et al. 2018, Proc. SPIE, 10700, 1070052
Saunders, W., Lawrence, J. S., Storey, J. W. V., et al. 2009, PASP, 121, 976
Shang, Z. H., Hu, K. L., Yang, X., et al. 2018, Proc. SPIE, 10700, 1070057
Shang, Z. H., Hu, K. L., Hu, Y., et al. 2012, in Observatory Operations: Strategies, Processes, and Systems IV, 8448, International Society for Optics and Photonics, 844826
Shi, S. C., Paine, S., Yao, Q. J., et al. 2016, Nature Astronomy, 1, 0001
Shial, G., Majhi, S. K., & Phatak, D. B. 2015, Procedia Computer Science, 48, 133
Yuan, X. Y., Cui, X. Q., Liu, G. R., et al. 2008, Proc. SPIE, 7012, 70124G
Yuan, X. Y., Cui, X. Q., Gong, X. F., et al. 2010, Proc. SPIE, 7733, 77331V
Zhang, G. Y., Wang, J., Tang, P. Y., et al. 2016, MNRAS, 455, 1654