Advanced Engineering Forum *Online: 2012-06-04*
ISSN: 2234-991X, Vol. 4, pp 3-8
doi:10.4028/www.scientific.net/AEF.4.3

An Expansion Method of Reliable Data Transmission of the Short-message Communication on Satellite Network

Jie Huang[1, a], Gang Li[2,b]

[1] School of Computer, National University of Defense Technology, Chang Sha, China

[2] School of Computer, National University of Defense Technology, Chang Sha, China

[a]huangjie@nudt.edu.cn, [b] agnes_nudt@hotmail.com

Keywords: Reliable Data Transmission, Short Message, Network performance, Satellite Network

Abstract. Short-message communication mechanism is one of the most frequently methods used on the field of satellite data transmission. Under the existing satellite network, we provide an expansion method of reliable data transmission of short message communication named EMRDT-SMC, which solves negative compression problem of transmission capacity expansion when the amount of short message is below normal, and also solves reliable data transmission problem while satellite microwave transmission causes short message loss. Based on the measurement over a small test-bed, it shows that the method EMRDT-SMC can obtain the better network performance.

Introduction

Currently GEO satellites have short-message communication capabilities, which have two kinds of modes: one is short-message communication between the satellite terminals; the other is communication between satellite terminals and the short message central station. Short message communication system based GEO satellite has a wide range of applicability and strong adaptability. However, the ability and capacity of satellite communication is limited, such as each short message is limited in several 100 bytes, and each user is allowed to send short message in a limited frequency which is in the range of a few seconds per time to tens of seconds per time. The limitation on GEO satellite based short message communication has restricted its development.

GEO satellite based short message communication can not only apply to transmit text message, but also apply to transmit binary user data. However, binary data transmission demands a large bandwidth, so capacity expansion scheme for data transmission under the existing satellite condition is needed. Because the size of single short message is small, the simple data compression technique cannot solve capacity expansion problem on data transmission. In practical application, the compression data, which is produced by data compression on each short message, usually has bigger size than original data. This is so called negative compression which cannot save the bandwidth, nor can save computing resources.

Moreover, short message communication sends data using wireless microwave, which is subjected to various environmental impacts including weather condition and ionosphere condition and so on. Short message is prone to transmission loss and transmission error because of transmission interference. However, transmission error, which can be detected by cyclic redundancy check (CRC) code, will be discarded in receivers. Therefore, the transmission loss problem of short message will have an adverse effect on message transmission.

The rest of the paper is organized as follows. We briefly review the related works in Section II and present an expansion method of reliable data transmission of short message communication on the satellite network named EMRDT-SMC in Section III. In Section IV, we show the scheme to achieve fairness, and we present the evaluation results. Finally, we conclude the paper and point out potential future works in Section V.

Related Work

In modern and future satellite network, messages from one or more senders are delivered to a large number of receivers. Moreover, a major fraction of communication patterns in these domains are small messages suffering relatively high overhead and poor performance during transmission[1]. For example, radar signal processing algorithms work on relatively short vectors and small matrices in general, but at a fast pace and with large sets of vectors and matrices. Even if incoming data from the antenna contains data is viewed in three dimensions (channel, pulse and distance), it is quite easy to divide the data into small tasks that each can be executed on a separate processing element in a parallel computer[2]. By reducing the combined overhead of protocols like MAC (18 bytes), LLC (4 bytes), IP (20 bytes) and UDP (8 bytes), the performance can be significantly improved. Another critical item arises from the Quality of Service (QoS) requirement of these small-messages, whose correct performance is specified in terms of time constrained message transmissions[3]. In reference [4], the application example mentioned above, the control traffic is periodic and must have a deterministically guaranteed delay-bound, while the data traffic is periodic and should have a probabilistically guaranteed delay bound. These problems have received attention in networking and parallel and distributed computing research communities in recent years.

There have been several attempts to achieve lower latency, better bandwidth and utilization, for example, reducing the network interface access time [5,6]. However, the high performance network interface design only alleviate, not overcome the unsatisfying bandwidth characteristics of the small messages. In addition, disadvantages in higher costs are introduced. Moreover, some results have been published on a pipelined fiber-ribbon ring network that supports several simultaneous transmissions in non-overlap segments to achieve higher throughput [7,8]. Offered services in this network include a guaranteed real-time service and other services for parallel and distributed processing such as barrier synchronization and global reduction. Numbers of protocols and schemes have been proposed to improve real-time characteristics of switched Ethernet networks. Ref. [9,10] includes a bandwidth reservation scheme inside the switches without any hardware or operating system modification. Support for periodic real-time traffic on an extended switched Ethernet network, using earliest deadline first (EDF) scheduling, have been reported in [11]. A thin software layer is added between the Ethernet protocols and the TCP/IP suite in the end stations. The switch is responsible for admission control where the feasibility analysis is made for each link and direction between the end nodes and the switch. More results on establishing real-time channels in packet switched networks and using deadline sorting in the switch to gain real-time support have been presented in [12].

An Expansion Method of Reliable Data Transmission of Short Message Communication

In data sending stage, the process caches sending data, and waits until caching data exceed a certain amount or reach a certain long time, and then carries on integration on the caching data. After integration, we carry on compression on sending data, and check the compressing effect. Comparing the size of uncompressed and compressed data, we use the smaller one for subsequent process. We partition sending data according to the size of short message, and number all blocks, from which we calculate a redundancy block using XOR operation. All sending blocks will be received and forwarded by the satellite, and the redundancy block will be sent first, then the rest of blocks are sent out in order.

In data receiving stage, the process caches data blocks received from satellite in buffer queue, and checks if there is transmission loss using marking number of blocks. If more than two data blocks are lost, all data blocks will be discarded, but if only one block is lost and the lost block is not the redundancy block, we can calculate the missing block by carrying XOR operation on redundancy block and rest blocks. The process carries on integration on caching data and checks if caching data has been compressed. We uncompress caching data which have been compressed, and recover data packs which belong to different applications from receiving data, and then put these data packs which wait for receiving applications in buffer queue.

Based on the above analysis, we present the method EMRDT-SMC, and the detail is as follow.

Phase 1. Transmission stage

(1) Get data from applications, and save the data in buffer queue. When receiving the data from applications, we add a data header which shows the size of the data, and cache the data with header to buffer queue, and waits for the sending pre-treatment.

(2) Check if the total size of data in buffer queue exceeds the preset threshold value and if the waiting sending time reaches the preset threshold time. We calculate the total size of data in buffer queue, and see if the total size exceeds preset threshold value. At the same time, we see if the time interval from last sending time reaches the preset low threshold time. If one of these two checking is satisfied, the pre-treatment process gets into step (3), or else gets into step (1) and waiting for new data.

(3) Carry on integration and compression to the data in buffer queue. We splice all data in buffer queue together to form a contiguous data block, and compress the data using compression algorithm in the base of integration.

(4) Check if the compression reaches a certain compression ratio. Comparing the size of uncompressed and compressed data, if the size of original uncompressed data is smaller, the process gets into step 6), or else gets into step (5) which processes sending pre-treatment.

(5) Partition sending data into blocks according to the largest size of short message, and fill the last block with 0 to reach the largest size of short message. We partition the compressed data into a number of blocks, and all blocks except the last one are exactly the largest size of short message. If the last block shorter than the largest size of short message, we fill the last block with 0 to reach the largest size of short message. The process gets into step (10) when partition is finished.

(6) Continue to receive data from applications, check if the total size of data in buffer queue exceeds the preset threshold value and if the waiting sending time reaches the preset threshold time. If the sending data is lack for transmission optimization, more sending data from applications are needed, and the process waits until that caching data in buffer queue is more enough to process compression, or that the time interval from last sending time reaches the preset low threshold time, then the process gets into integration and compression steps.

(7) Integration and compression on sending data in buffer queue from several applications. We splice all data in buffer queue together to form a contiguous data block, and compress the data using compression algorithm in the base of integration.

(8) Check if the compression reaches a certain compression ratio by seeing if the size of compressed is smaller than uncompressed data. If the compression reaches a certain compression ratio, the process gets into step (5), but if the size of uncompressed is smaller than compressed data, we use the uncompressed data for subsequent process.

(9) Partition sending data into blocks according to the largest size of short message, and fill the last block with 0 to reach the largest size of short message. We partition the compressed data into a number of blocks, and all blocks except the last one are exactly the largest size of short message. If the last block shorter than the largest size of short message, we fill the last block with 0 to reach the largest size of short message. The process gets into step (10) when partition is finished.

(10) Number and mark the existing data blocks. We add a header which contains sequence number from 1 to n on each data block. The highest digit of header in last block is set to 1, which identify the end of data pack.

(11) Calculate a new block by carrying bitwise exclusive XOR operation on data blocks of which the marking number are in the range of 1 to n, and then we mark this new block as Num.0 block.

(12) Send the block using short message communication to satellite in the order of 0 to n. We send the Num.0 block first, and then send the rest of blocks to satellite in the order of 1 to n.

Phase 2. Receiving stage

(1) Receive a short message block from satellite, and save the block in buffer queue. The receiving devices get short message blocks from satellite, and put receiving blocks in received buffer queue, and wait for the rest of blocks.

(2) Check if the marking number of receiving block is 0. The process check the header of receiving block, see if the receiving block is redundancy block which takes 0 as marking number. If not, the process gets into step (5), but if so, the process continues the following steps.

(3) Check if the receiving block is the ending block of previous data pack.

Check if the highest digit of previous block in buffer queue is 1, and if so, the process gets into step (5), but if not, the process continues the following steps.

(4) Recover the last lost block by carrying bitwise exclusive XOR operation on all data block of last data pack. We carry on bitwise exclusive XOR operation on all data block of last data pack to recover the last lost block. The process gets into step (10) when the recovery is finished.

(5) Continue to receive the rest of blocks until the last block of this data pack has been received. The process continues to receive the short message block from satellite, and check if the highest digit of receiving block is 1. The process will not stop receiving block until the highest digit of receiving block is 1.

(6) Check if there is transmission loss using marking number of blocks. The process checks if marking numbers of blocks in buffer queue are continuous. If so, the process gets into step (10), and if not, the process continues the following steps.

(7) Check if there are more than one data blocks are lost. The process sees if there are more than one data blocks are lost. If so, the process gets into step (15), and if not, the process continues the following steps.

(8) Check if the marking number of lost block is 0. The process sees if the block which takes 0 as marking number is lost. If not, the process gets into step (10), and if so, the process continues the following steps.

(9) Recover the lost block by carrying bitwise exclusive XOR operation on all data blocks of current data pack. The process recover the lost block by carrying bitwise exclusive XOR operation on the rest of data blocks, and put the recovery block in buffer queue in the marking number order.

(10) Combine data blocks into an integrated data pack.

The process removes all the header of blocks, and combines all the blocks in the order of marking number from 1 to n. The integrated package is recovered.

(11) Check if the data pack has been compressed.

The process check if the data pack has been compressed. If not, the process gets into step (13), and if so, the process continues the following steps.

(12) Decompress the data pack. The process decompresses the data pack, and recovers the original data pack.

(13) Recover several original data packs. The process recovers original data packs which belong to different applications according to header information of original data packs.

(14) Put the original application data packs in buffer queue and wait for receiving applications, and then the process gets into step (1), and continues to receive blocks of next data pack.

(15) Discard all the blocks from the current data pack.

When more than one block are lost, the process can’t carry on block recovery. The process discards all the blocks from the current data pack, gets into step (1), and continues to receive blocks of next data pack.

Performance Analysis

In order to verify the feasibility and performance of the EMRDT-SMC, we make a set of test with the different parameters.

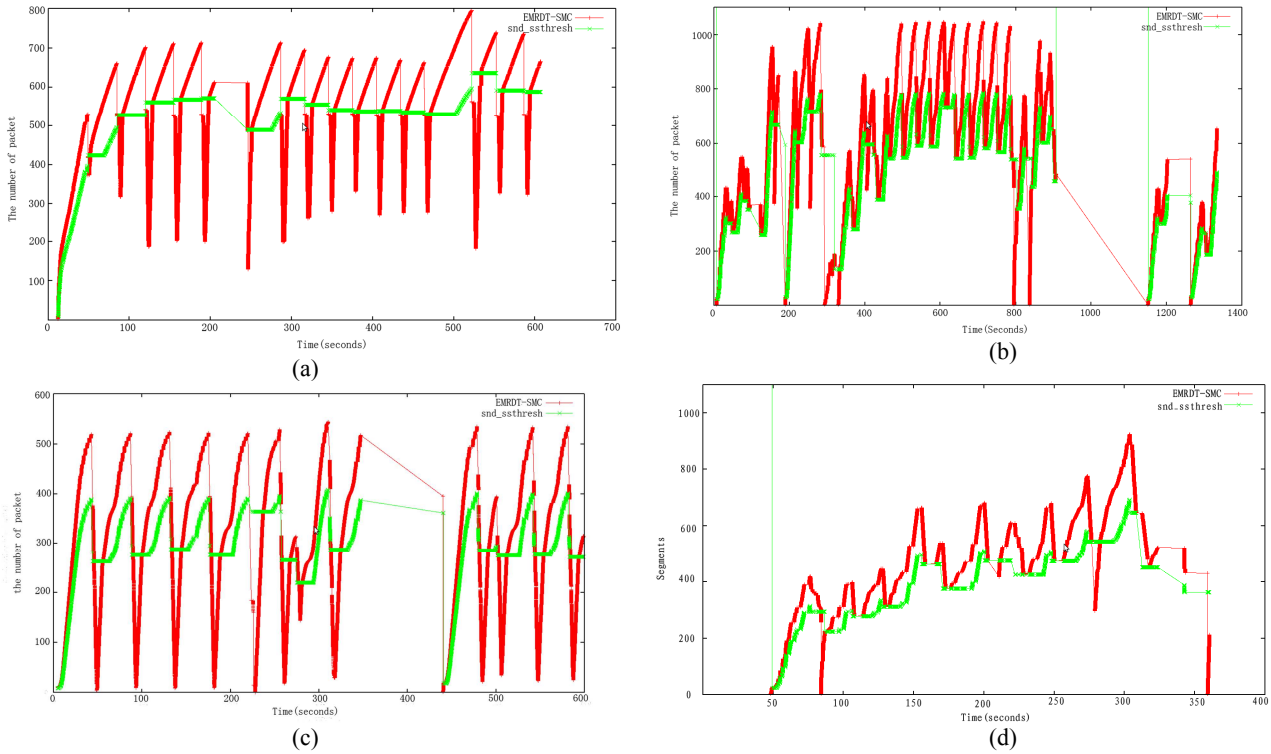

Fig. 1. Transmission quality: Comparisons of EMRDT-SMC and snd_ssthresh

Based on the EMRDT-SMC, we can transmit application data in an efficient and flexible way, and carry on capacity expansion on data transmission ability, and ensure the transmission after expansion has certain reliability. The method can carry on data recovery automatic when no more than one block is lost in every transmission, which guarantees the reliability of the transmission after expansion.

Based on the EMRDT-SMC, we can design a reliable data transmission and capacity expansion device based on short message satellite communication mechanism.

Summary

The paper presents the EMRDT-SMC method which solves negative compression problem of transmission capacity expansion when the amount of short message is below normal, and also solves reliable data transmission problem while satellite microwave transmission causes short message loss. This solution can be used for many network applications that require small messages with high bandwidth, low latency and bounded delay. A similar method can be used for analyzing other related network standards.

Acknowledgment

My work is supported by the National Natural Science Foundation of China under Grant No.61070200 and No.61003303, the National Science and Technology Support Program of China under Grant No.2008BAH37B03, the National High-Tech Research and Development Plan of China under Grant No. 2009AA01Z432. Jie Huang is corresponding author.

References

[1] Draves, R.; Padhye, J.; Zill, B. Comparison of Routing Metrics for Static Multi-hop Wireless Networks. In Proceedings of ACM International Conference of the Special Interest Group on Data Communication, Portland, OR, USA, August, 2004; pp. 133–144.

[2] Bergenhem, C., Jonsson, M, Gördén, B., Åhlander, A.: Heterogeneous real-time services in high-performance system area networks - application demands and case study definitions. Technical Report IDE - 0254, School of Information Science, Computer and Electrical Engineering (IDE), Halmstad University (2002).

[3] Weber, R., Santos, D., Bianchini. R., Amorim, C. L.: A survey of messaging software issues and systems for Myrinet-based cluster, Parallel and Distributed Computing Practices. In: Special Issue on High-Performance Computing on Clusters, Vol. 2, No. 2 (2001).

[4] Cerpa, A.; Wong, J.L.; Kuang, L.; Potkonjak, M.; Estrin, D. Statistical Model of Lossy Links in Wireless Sensor Networks. In Proceedings of ACM/IEEE International Symposium on Information Processing in Sensor Networks, Los Angeles, CA, USA, April, 2005; pp. 81–88.

[5] Gu, Y.; He, T. Data Forwarding in Extremely Low Duty-cycle Sensor Networks with Unreliable Communication Links. In Proceedings of ACM International Conference on Embedded Networked Sensor Systems, Sydney, Australia, November, 2007; pp. 321–334.

[6] Bergenhem, C., Jonsson, M.: Fibre-ribbon ring network with inherent support for earliest deadline first message scheduling. In: Proceedings of Workshop on Parallel and Distributed Real-Time Systems (WPDRTS'2002) in conjunction with International Parallel and Distributed Processing Symposium (IPDPS'02), Fort Lauderdale, FL, USA (2002).

[7] Jonsson, M., Bergenhem, C., Olsson, J.: Fiber-ribbon ring network with services for parallel processing and distributed real-time systems. In: Proceedings of ISCA 12th International Conference on Parallel and Distributed Computing Systems (PDCS-99), Fort Lauderdale, FL, USA (2003) 94-101.

[8] Zamalloa, M.Z.; Krishnamachari, B. An Analysis of Unreliability and Asymmetry in Low-power Wireless Links. ACM Trans. Sens. Netw. 2007, 3, 7.

[9] Varadarajan, S.: Experiences with Ethereal: a fault tolerant real-time Ethernet switch. In: Proceedings of 8th IEEE International Conference on Emerging Technologies and Factory Automation, (2001) 183-194.

[10] Hoang, H., Jonsson, M., Hagström, U., Kallerdahl, A., Switched real-time Ethernet with earliest deadline first scheduling − protocols and traffic handling. In: Proceedings of Workshop on Parallel and Distributed Real-Time Systems (WPDRTS'2002) in conjunction with International Parallel and Distributed Processing Symposium (IPDPS'02), Fort Lauderdale, FL, USA (2002).

[11] Xing Fan and Magnus Jonsson. Efficient Support for High Traffic-Volumes of Short-Message Real-Time Communication Using an Active Ethernet Switch. Proc. 10th International Conference on Real-time and Embedded Computing System and Application(RTCSA'04),Sweden, Aug.25-27, 2004,pp.517-533.

[12] Fan, X., Jonsson, M., Hoang. H.: Efficient many-to-many real-time communication using an intelligent Ethernet switch. In: Proceedings Of International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN'2004), Hong Kong, China (2004) 280-287.