A Protocol for Disaster Data Evacuation

Tilmann Rabl, Florian Stegmaier, Mario Döller, and The Tong Vang
Chair of Distributed Information Systems, University of Passau
Passau, Germany
{rabl,stegmai,doeller,vang}@fim.uni-passau.de

ABSTRACT

Data is the basis of the modern information society. However, recent natural catastrophes have shown that it is not possible to definitively secure a data storage location. Even if the storage location is not destroyed itself the access may quickly become impossible, due to the breakdown of connections or power supply. However, this rarely happens without any warning. While floods have hours or days of warning time, tsunamis usually leave only minutes for reaction and for earthquakes there are only seconds. In such situations, timely evacuation of important data is the key challenge. Consequently, the focus lies on minimizing the time to move away all data from the storage location whereas the actual time to arrival remains less (but still) important. This demonstration presents the dynamic fast send protocol (DFSP), a new bulk data transfer protocol. It employs striping to dynamic intermediate nodes in order to minimize sending time and to utilize the sender's resources to a high extent.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Protocols

General Terms

Reliability, Security

1. INTRODUCTION

To enable a fast data evacuation solution, we developed the Fast Send Protocol (FSP) which aims at overcoming the weakest link by introducing intermediate nodes [1]. The central idea of the protocol is to partition the data into smaller blocks and start a striped transfer to intermediate nodes. As soon as all blocks are distributed, the server can retreat from the transfer, while the receiver collects the blocks from the intermediate nodes. Although the protocol shows excellent results in terms of data distribution speed, there are limitations related to security, reliability and the choice of the intermediate nodes (which are fixed beforehand). Especially, the selection process of intermediate nodes is important. Therefore, an enhanced version, the *Dynamic Fast Send Protocol* (DFSP), has been implemented which overcomes the identified weaknesses. DFSP comprises a novel approach for selecting the number and set of intermediate nodes. Reliability is guaranteed in the enhanced protocol by a concept similar to RAID-1 which ensures *k-safety*. Security has been enhanced by integrating a challenge response approach based on an asymmetric cryptosystem.

The demonstration of the *Dynamic Fast Send Protocol* will present the benefits of DFSP for large data transfers in contrast to the predecessor protocol FSP.

SIGCOMM'11, August 15–19, 2011, Toronto, Ontario, Canada.
ACM 978-1-4503-0797-0/11/08.

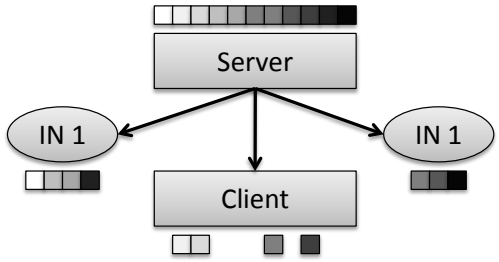

Figure 1: Striping Phase of the DFSP Protocol

2. DYNAMIC FAST SEND PROTOCOL

DFSP is an application layer protocol. It extends the well-known File Transfer Protocol (FTP) and is fully compatible with it. DFSP is designed to fully exploit available bandwidth at a server to evacuate its data. To maximize the throughput the following issues have been addressed in DFSP: (i) slow network link between a sender and a receiver, (ii) slow receivers due to other transfers and (iii) transport protocol limitations.

A DFSP file transfer is divided into two main phases: a distribution phase and a collection phase. After a *client request* the server sends the data at maximum speed to the client and multiple intermediate nodes as shown in figure 1 (*striping phase*). When it has sent all blocks it can retreat from the transfer. The client then collects the missing blocks from the intermediate nodes.

As indicated in figure 1, the DFSP server performs a *block partitioning* of the data. The block size is configurable and to ensure security and fault tolerance of the data, the blocks can be encrypted and checksums can be computed. The intermediate nodes can be chosen from a list or are provided dynamically by a Meridian based peer-to-peer overlay [2]. The overlay allows to dynamically build the list of possible intermediate nodes and to choose nodes which have a network location between sender and receiver. This improves the reliability of the protocol, since Meridian pro-

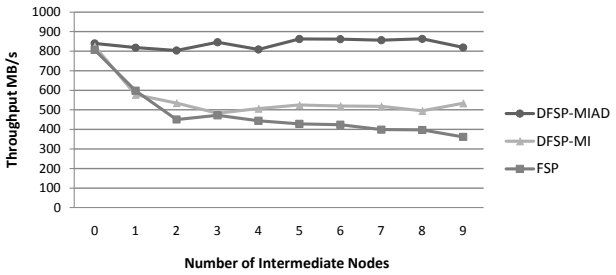

Figure 2: Effective Throughput of DFSP vs. FSP

vides a bootstrap procedure and gossip protocol that make the protocol tolerant to node outages. Depending on the throughput, the DFSP increases or decreases the number of intermediate nodes. The heuristic for the number of nodes increases the number of nodes by a multiplicative factor and decreases it one by one. The number of intermediate nodes is increased until the throughput of the sender is maximized. If the throughput decreases after new nodes were added, the number of nodes is decreased again. Furthermore, nodes with bad throughput will be replaced. The dynamic adaption improves the sending time, but it also improves the receiving time.

FSP uses user name and password to connect to the intermediate nodes. This is insecure, since the passwords have to be stored on all clients. Therefore, DFSP additionally provides challenge response authentication with certificates. To increase the scalability and speed of authentication a trust server is used as authority.

The distribution of the data increases the possibility that parts of the data are lost due to failures. The more nodes are involved in the data transfer, the higher are the chances for that one of the nodes fails. As long as the sender is still online this is no problem since it will transfer the data to another node. However, if the sending phase is finished, the server is not part of the transfer any more. Hence, a failure in the collection phase results in an abort of the transfer in FSP. Since the goal of the protocol is fast data evacuation this is not acceptable. Therefore, DFSP provides a k-safe transfer. In this mode, it is ensured that each block of the data which is not sent to the client directly is replicated at least $k + 1$ times. Therefore, the transfer can tolerate the loss of k intermediate nodes and still succeeds.

Due to the dynamic approach DFSP is usually faster than FSP. FSP has an optimal sending throughput if the number and choice of intermediate nodes is optimal. However, this is hard to determine beforehand. Furthermore, a slow intermediate node will degrade the sending time and especially the overall transfer time. If the number of intermediate nodes is too small the sending time suffers, since the servers capabilities are not exploited. If the number of intermediate nodes is too large, the additional overhead during collection reduces the total throughput noticeable. This can be seen in the following evaluation.

3. EVALUATION

In contrast to the fixed selection of intermediate nodes in FSP, DFSP features a *Multiplicative Increase* (MI) and a *Multiplicative Inscrease, Additive Decrease* (MIAD) strategy. The first method (MI) starts with a fixed set of nodes

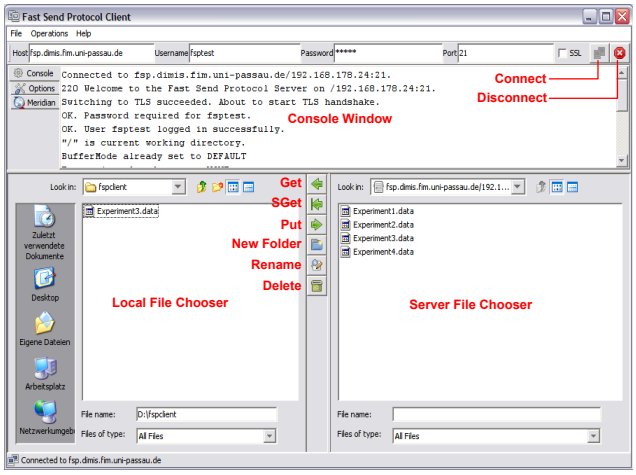

Figure 3: Screenshot of the DFSP GUI

which is multiplicative increased if the effective throughput is higher than the previous one. The second approach (MIAD) is similar to the MI-method except that the amount of intermediate nodes is decreased if the current effective throughput is smaller or equal to the previous one. To demonstrate the advantage of the two different methods in contrast to FSP, figure 2 presents the results of our evaluation.

The evaluation was executed at our cluster with one server and 16 nodes. The server is connected by a 1 Gbit switch and all nodes by 100 MBit switches. The x-axis presents the variable amount of intermediate nodes (0-9). The y-axis shows the effective throughput (MBit/s), where detailed numbers are given in the table. The evaluation verifies that both strategies (MI and MIAD) improve the throughput over FSP by any configuration, whereas MIAD demonstrates a more stable behavior.

4. DEMONSTRATION

In the demonstration we will show the functionality of the protocol which is administrated by the DFSP graphical user interface (see figure 3). Apart from standard file manager functionality, like moving files, creating folders and deleting files or folders, the GUI shows the status of the connection and makes it possible to configure the data transfer and the Meridian overlay.

In order to demonstrate a data evacuation scenario, we will initiate a third party transfer. For the demonstration we will use a laptop connected to the Internet that can issue the third party transfer. The transfer itself will be issued in our lab at the University of Passau. For the test, we will transfer a 1 Gbyte sized file.

5. REFERENCES

[1] T. Rabl, C. Koch, G. Hölbling, and H. Kosch. Design and implementation of the fast send protocol. *Journal of Digital Information Management*, 7(2):120–127, 2009.

[2] B. Wong, A. Slivkins, and E. G. Sirer. Meridian: A Lightweight Network Location Service without Virtual Coordinates. *ACM SIGCOMM Computer Communication Review*, 35(4):85–96, 2005.