DISTRIBUTED SYSTEMS COURSE

Exercise 3 - Final Report

Erfan Sharafzadeh Farnaz Yousefi
e.sharafzadeh@jhu.edu f.yousefi@jhu.edu

10/29/2019

1 INTRODUCTION

Following the second exercise, we want to use Spread to handle all the hard work in ensuring the agreed receive order of the data messages. We use spread's API to handle membership, when every process ensures that everyone has joined the multicast group, processes start with sending a burst of new messages. Then after they receive any message they will decide to deliver it to file and send a new message based on our flow control condition. We mark finalize process by sending a message with index=0. When a process receives finalize from every other process, it changes its state to Exiting and sends a special Exit message to everyone. On the receipt of this Exit message from all, processes will exit.

2 PERFORMANCE RESULTS

As Spread itself controls the flow, not much can be done in our mcast program for the flow control. The only means to do flow control in our program are *STARTING BURST* which specifies the number of packets we send at the start of transmission, when do we send our new data. Since the rate of sending is controlled by Spread, these packets are put in Spread's buffer and sent by Spread's own mechanism. Therefore, there must always be enough packets in the buffer so that maximum capacity of Spread is used. However, we observed that when *STARTING BURST* is high (higher than the capacity of the buffer), the buffer overflows and the program stops. Table 2.1 presents the transfer duration with different starting bursts (SB). As we increase the the burst from 1, we observe that the transfer latency gradually decreases. This is because we are trying to keep Spread's buffer full enough to send its window (Starting bursts between 1-20 will not be enough to fill Spread's sending buffer). However, as we increase the burst further, we don't see any positive effects in the transfer duration. Starting bursts higher than 800 will make buffer overflow. Figure 2.1 depicts the same results and the fact that no improvements are seen after setting starting burst to 20. Based on the global minimum in transfer duration, we select our burst to be 80.

Also, to further prevent buffer from overflowing, we only send new data (one data packet) whenever we receive a data packet from our own machine – meaning that everyone has received this packet successfully so it would be deleted from Spread's buffer and a new message would replace it. This way, we make sure that buffer would not overflow in the middle of the execution.

SB	1	10	20	30	70	80	90	100	120	200	300	500	700
Duration	259.3	76.3	33.5	34.1	34.6	29.6	30.9	32.3	33.6	32.5	33.3	33.00	35.5

Table 2.1: Effect of changing the starting burst in transfer duration

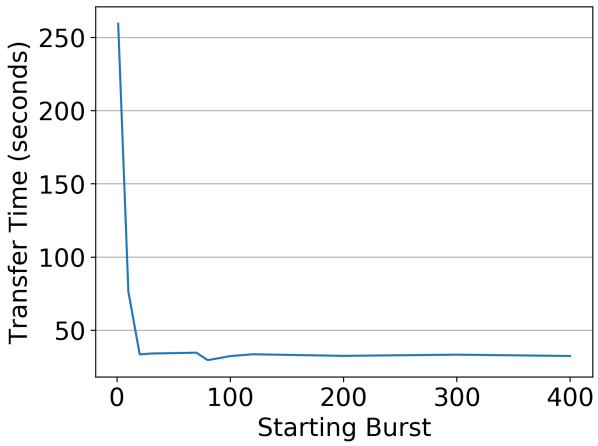

Figure 2.1: Effect of changing the starting burst in transfer duration