Lindsey Bieda
Heather Friedberg
10/19/10

CS 2510 – Distributed Operating Systems
Project 1 Report

Token Algorithm

Description
The token algorithm that we implemented is a token ring. The processes are ordered by process ID, and each process is connected to two processes – the one with the process ID that is greater and the one with the process ID that is lower. Lamport clocks are used in order to synchronize the time between all of the processes. Note that in the recording of the time each of the process was expected to take the time written in the file in chunks of 500 ms.

Statistics
The number of messages received, the number of messages sent, the average wait time, and the total time the algorithm was running were the four statistics compiled for each run of the algorithm. This was done three times – for N=4, 8, and 16. The statistics are summarized below:

Number of Processes	Total Messages Sent	Average Messages Sent Per Process	Total Messages Received	Averages Messages Received Per Process	Average Wait Time	Total Runtime of Algorithm
4	14	3.75	29	7.25	2952 ms	16266 ms
8	31	3.9	93	11.625	6992 ms	33851 ms
16	63	3.9	316	19.75	15904 ms	72154 ms

Tokenless Algorithm

Description
The tokenless alogirthm that we implemented is a version of Lamport's Mutual Exclusion Algorithm. Each process is connected to every other process by socket. When a process needs to access the shared memory, it sends a time stamped request to every other process and places its request in its own request queue. It waits to start its work in the critical sections until it has received a response message from all of the processes and its request is the earliest in its queue. When it is done with the critical section it sends a release message to all of the other processes. When a process receives a request message from another process, it sends a time-stamped reply and places the message in its request queue. When a process receives a release message from another process it removes the corresponding

request from its request queue. Once again, in the recording of the time each of the process was expected to take the time written in the file in chunks of 500 ms.

Statistics
The same statistics that were calculated for the token algorithm were also reported for the tokenless; they are summarized below:

Number of Processes	Total Messages Sent	Average Messages Sent Per Process	Total Messages Received	Averages Messages Received Per Process	Average Wait Time	Total Runtime of Algorithm
4	207	51.75	205	51.25	3593 ms	15283 ms
8	1694	211.75	1688	211	6171 ms	31813 ms
16	13500	843.75	13490	843.125	12604 ms	58300 ms

Analysis and Comparison

The token algorithm did fairly well when it was with a small amount of nodes. However, once there were a sizable amount of nodes the wait times increased dramatically. The benefit to the token ring approach seems to be heavily in its simplicity and it appears to have a modest number of messages being passed between the nodes. The average number of messages sent and received per process stayed relatively constant as N increased.

The tokenless approach required significantly greater messages than the token ring, and the number of messages needed per process increased with N. However, the algorithm seemed to less wait time and a shorter total runtime, especially as the nodes increased.

The algorithm that should be selected depends greatly on the number of nodes and the type of network available. If message passing is very expensive, the tokenless approach should be avoided, due the extremely high number of messages required. If tasks are spread somewhat evenly among the processes, the token ring seems to be a good solution, as the token is passed in a designated order, and a process is guaranteed to get it when it is its turn. If tasks are not distributed evenly, however, a tokenless approach is better, as the wait times will be much lower.