**Distributed Algorithms**

**California University of Pennsylvania**

**CSC 360 Analysis of Algorithms**

**Professor Sible**

**Fall 2020**

**Kevin Andor Scutaru**

**Nathaniel DeHart**

**Cole Stewart**

The Internet is made up of a massive collection of distributed systems giving these systems very important roles in most peoples’ lives. The distributed Bellman-Ford, distributed search, consensus, and mutual exclusion problems are all examples where distribution algorithms can be applied. This paper will briefly go over what a distributed algorithm is, and will explore the workings of the aforementioned problems.

Distributed algorithms are algorithms designed to take advantage of a group of interconnected computers with separate memory banks. This cluster of computers is known as a distributed system. One way that distributed algorithms can work is by designated one computer as the “master” and distributing pieces of an algorithm to other computers from there. Another way that a distribution algorithm may work is by attempting to symmetrically distribute the tasks across different computers. In either case of a distributed algorithm, delays happen while one computer awaits the output of another computer. The ideal case of implementing a distributed algorithm in such a system would be dividing the time cost of the problem by the number of computers. While it is ideal, it is not fully obtainable due to necessary inter-computer communication. Thus, designers of such algorithms can only attempt to get as close as possible to fully divide the runtime of a problem by the number of computers.

There exists a distributed form of the Bellman-Ford algorithm that can be used to find the shortest path from a given start node to the destination node. It does this by generating a table for each node in the system. These tables start empty and are filled out by sending out the current tables to neighboring nodes continually until there are no changes in these tables. The tables that are sent contain the ID of each node and the lowest cost to reach that node if it is known. When a table is received from a neighboring node, the costs to move to other nodes are added to the receiving node’s table and then compared with the original receiving values. If any lower send costs are found, then the new costs are placed into the receiver’s table. The receiver is then set to send its updated table to all its neighboring nodes. If no updates were made, then that receiver does not send its table anywhere. This process can be continued concurrently between all nodes through the use of the distributed system until none of the nodes are sending out updated tables. Using these final tables, systems can now identify the shortest path between any two nodes.

Another example of a problem that utilizes distributed algorithms includes the distributed search problem, a search engine model where the processing of queries, indexing, and web crawling (a script that browses the web on its own) are distributed among many computers and networks, rather than relying on one computer to handle the processing. Search engines such as Google use the model to handle the task of web crawling from multiple locations around the world in order to update their database. Their model tasks it’s crawlers with viewing parts of the web, going to different sites and going to the sites that are linked to that site, recording all results along the way. Then, each crawler compresses all the results and sends them to a central server, where it gets stored in a database. For queries, Google’s DNS software sends them to clusters of computers, taking into account factors such as the location of the user, how busy they are, etc. When the query is received at a cluster, the software that receives the query distributes it for all the computers within that cluster to search for results, where an index server, document server, and page builder will gather all the necessary indexing, titles and summaries from the results and build the page. The earliest attempts of distributed search can be dated back to the year 2000 when the first decentralized peer-to-peer network Gnutella was in its infancy. A prototype search engine based on Gnutella, named InfraSearch, was created. It worked by having participating websites crawl their own data and upload it to the InfraSearch network, where users can access the site through the InfraSearch search engine.

The consensus problem, which starts with multiple processes that must agree upon the usage of a singular data value, is another example of how distributed algorithms are used. To come to an agreement on the data value, the consensus algorithm will use four properties: termination, which is where each process decides a value; validity (or, in some cases, integrity), where—if all processes chose some value *v*—then any other correct processes must also choose *v*; and agreement, where every process agrees on that same value. Commonly used distributed algorithms for consensus problems are the Paxos protocols and the Raft protocols. The Paxos family focuses on creating a fault-tolerant distributed algorithm, so as to mostly negate unreliable processors that would prohibit progress while guaranteeing consistency. This protocol is mostly used in database systems that would require durability. In cloud computing, there is a study that shows that Paxos protocols are actually outperformed by most other alternatives and are, in that environment, extremely inefficient and can “seriously impact failover” (Benz *et al.*). The Raft algorithm is much different from the Paxos family in that it will elect a singular processor as the leader and the other processors will count on that specific processor to do things correctly. The leader processor will contact the follower processors to let them know that it is still alive and, if the followers do not receive a message, they will elect a new leader to take over the processes. The followers will also communicate with the leader processor. While Raft works similarly to Paxos in terms of efficiency, it is much easier to understand than Paxos is (Ongaro and Ousterhout).

Mutual exclusion is another important application of distributed algorithms, which is important in determining how to continue when messages are unpredictably delayed or there is incomplete knowledge within the system. A mutual exclusion algorithm will ensure that only one process has access to a critical section at any given time. There are multiple ways to solve mutual exclusion, such as a token-based approach, in which a token is shared among the processes and a process is only allowed to enter a critical section if it has said token; a non-token based approach, in which messages are sent between two processes to determine which will enter the critical section next; and a quorum-based approach, in which processes are put into subsets called quorums and two quorums will have a common site that determines which one enters the critical section. A mutual exclusion agreement has a lot to focus on in its algorithm, such as fairness to each process and how it must give each process a chance to enter the critical section. One well-known mutual exclusion distributed algorithm is the Lamport algorithm. In Lamport’s algorithm, a node will push its request onto a stack with a timestamp, send a request to every node, wait for a reply from every node, and—if it is at the head of the queue—enter the critical section, exit, and remove itself from the queue. Lamport’s algorithm has a high message complexity of 3(N-1) messages per entry/exit and also is extremely unreliable, as a single failure will stop progress (Kshemkalyani and Singhal). There is an improvement upon Lamport’s algorithm, which is the Ricart-Agrawala algorithm. It has a message complexity of 2(N-1) and is a lot less unreliable. Even if a process were to fail and starve, it can be solved if the failure can be detected after it has timed out for a set amount of time (Agrawala and Ricart).

While they may not be as intuitive to develop as single machine algorithms, distributed algorithms are made to take advantage of multi-machine setups. Some of the algorithms produced could be either inconsistent, faulty, inefficient, or all of the above. The choice of usable algorithms for a given problem could come down to picking the least of these negative attributes. With that being said there are reasons these algorithms are used. The vast speed and memory benefits of interconnected systems should not be ignored. The Internet is only growing, and along with it the demands placed on distributed systems grow as well.

**References**

Burch, C. (2009, August). *Introduction to Parallel and Distributed Algorithms*. Retrieved from Toves: http://www.toves.org/books/distalg/

Kozlovski, S. (2018, April 27). *A Thorough Introduction to Distributed Systems*. Retrieved from Free Code Camp: https://www.freecodecamp.org/news/a-thorough-introduction-to-distributed-systems-3b91562c9b3c/

Web crawler. (n.d.). Retrieved November 18, 2020, from https://www.sciencedaily.com/terms/web\_crawler.htm

Rouse, M. (2008, April 24). What is distributed search? - Definition from WhatIs.com. Retrieved November 18, 2020, from https://whatis.techtarget.com/definition/distributed-search

Distributed search engine. (2020, October 16). Retrieved November 18, 2020, from https://en.wikipedia.org/wiki/Distributed\_search\_engine

Agrawala, Ashok, Ricart, Glenn. “An optimal algorithm for mutual exclusion in computer

networks.” *Communications of the ACM*, Jan. 1981. https://dl.acm.org/doi/10.1145/358527.358537.

Benz, Samuel, Birman, Ken, Marandi, Parisa, Pedone, Fernando. “The Performance of Paxos in

the Cloud.” *Cornell University*.https://www.cs.cornell.edu/projects/Quicksilver/public\_pdfs/SRDS14.pdf. Accessed 17 Nov. 2020.

Kshemkalyani, Ajay, Singhal, Mukesh. “Chapter 9: Distributed Mutual Exclusion Algorithms.”

*Cambridge University Press,* 2008. https://www.cs.uic.edu/~ajayk/Chapter9.pdf.

Ongaro, Diego, Ousterhout, John. “In Search of an Understandable Consensus Algorithm

(Extended Version).” *Stanford University*. https://raft.github.io/raft.pdf. Accessed 17 Nov. 2020.