NAME: VIVEK BANSAL

STUDENT ID: 112044493

MAIL ID: VIVEK.BANSAL@STONYBROOK.EDU

COURSE NUMBER: CSE 535 (ASYNCHRONOUS SYSTEMS)

ASSIGNMENT NUMBER: 5

DUE DATE: 02 NOVEMBER 2018

ANS 1:

4 classes of distributed algorithms:

1) **Distributed Consensus**
2) **Distributed Storage**
3) **Security Protocols**
4) **Distributed Programming Support**

i) Distributed Consensus

I googled "Byzantine Failure Tolerance Implementation Github" and found out the following links on the Github:

1) https://github.com/luckydonald/pbft
2) http://bft-smart.github.io/library/
3) https://github.com/vonwenm/pbft-1
4) https://github.com/bft-smart/depspace
5) https://github.com/jTendermint/jabci
6) https://github.com/vmware/concord-bft
7) https://github.com/yagrawala/bft-cr-shuttle

Also when I googled “Blockchain Implementation GitHub” I found out the following GitHub links:

8) https://github.com/dvf/blockchain

When I googled “Raft Algorithm” I found out the following links:

9) https://github.com/zhebrak/raftos
10) https://github.com/simonacca/zatt

Out of these I liked:

1) https://github.com/luckydonald/pbft:
Because it supports a web interface representing what is going on exactly in the system. It also gave insight about which messages get send by which node to which node. It has extra features of webgui to postgres as well.
6) https://github.com/vmware/concord-bft
The code is written in C and C++ which is easy to understand. Also, it is highly scalable and energy-efficient distributed trust infrastructure for consensus and smart contract execution.
7) https://github.com/yagrawala/bft-cr-shuttle
The code is written in distalgo which is very easy to understand. It is the implementation of byzantine fault tolerant replication of servers using chain replication and Shuttle protocol.
10) https://github.com/simonacca/zatt
It is a python implementation of raft algorithm for distributed consensus. It uses UDP in intra-clustering which makes it fast in acknowledgement and retransmission. There is no need of encryption as well. It is safe, stateful and supports TLS on client-server networking architecture as well.

---

ii) Distributed Storage

I googled “distributed storage implementations github” and found out the following links on the Github:

1) https://github.com/Qihoo360/huststore
2) https://github.com/pravega/pravega
3) https://github.com/happyfish100/fastdfs
4) https://github.com/yanghanxy/CloudDB
5) https://github.com/moosefs/moosefs
6) https://github.com/storj/storj
7) https://github.com/douban/gobeansdb

8) https://github.com/chrislusf/vasto
9) https://github.com/fenghui2013/black_hole

Out of these I liked:

2) https://github.com/pravega/pravega
It provides a new storage abstraction – a stream – for continuous and unbounded data. This stream is durable, elastic, append-only, unbounded sequence of bytes that has good performance and strong consistency. It is consistent, high performance storage and ideal for IOT.

1) https://github.com/Qihoo360/huststore
Huststore is an open source high performance distributed database system. It not only provides key-value storage service with extremely high performance, up to hundreds of thousands QPS, but also supports data structures like hash, set, sorted set, etc. Also, it can store binary data as value from a key-value pair, and thus can be used as an alternative to Redis.

3) https://github.com/happyfish100/fastdfs
FastDFS is an open source high performance distributed file system. Its major functions include: file storing, file syncing, and file accessing (file uploading and file downloading), and it can resolve the high capacity and load balancing problem.

5) https://github.com/moosefs/moosefs
MooseFS is a Petabyte Open Source Network Distributed File System. It has higher reliability, parallel data operations, data tiering to support different storage policies for different files/directories. Retired hardware can be removed on the fly which is another big advantage.

---

iii) Security Protocols

I googled "Transport layer security protocol implementation github" and found out the following links on the Github:

1) https://github.com/awslabs/s2n
2) https://github.com/python-tls/tls
3) https://github.com/sundeep/tls

I googled "Diffie-Hellman protocol implementation GitHub" and found out the following links on the GitHub:

4) https://github.com/chrisvoncsefalvay/diffiehellman

5) https://github.com/crypto-browserify/diffie-hellman
6) https://github.com/firatkucuk/diffie-hellman-helloworld
7) https://github.com/cedi/DiffieHellman
8) https://github.com/lowazo/pyDHE

I googled "Needham Schroeder Implementation in Python" and found out the following links on the GitHub:

9) https://github.com/uyenle57/Needham-Schroeder-Protocol?fbclid=IwAR2HpdRBhaBLh6FV3nuou9jA3srZ84Oop4yspYnWuWV5paJMgYH8ZjcqI1M
10) https://github.com/rahulbahal7/needham-schroeder?fbclid=IwAR1wGPRwgAWxz_xH3R9iyVqMJirQC0M8AZJkn-RmGEw-wCINRDMJO6U7B58
11) https://github.com/abender/needham-schroeder

Out of these I liked:

2) https://github.com/python-tls/tls
It is written in Python and supports basic functionalities of tls and it is very easy to understand. It is an open source python implementation of TLS 1.2, using the Python Cryptographic Authority's Cryptography libraries for all cryptographic primitives (e.g. AES, RSA etc.).

3) https://github.com/sundeep/tls
It is only dealing with in-memory buffers, have no global state. It is supporting both client and server operations. It is not allowing disabling of security features like basic security checks, chain validation and hostname validation.

9) https://github.com/uyenle57/Needham-Schroeder-Protocol?fbclid=IwAR2HpdRBhaBLh6FV3nuou9jA3srZ84Oop4yspYnWuWV5paJMgYH8ZjcqI1M
Implementation of Needham Schroeder protocol in a very simplistic manner with detailed steps of how to run it.

10) https://github.com/rahulbahal7/needham-schroeder?fbclid=IwAR1wGPRwgAWxz_xH3R9iyVqMJirQC0M8AZJkn-RmGEw-wCINRDMJO6U7B58
Implementation of Needham Schroeder to achieve authentication in network services and RC4 is used for confidentiality.

iv) Distributed Programming Support

I googled "Distributed Programming Support, verification, testing, monitoring, tracing implementation on Github" and found out the following links on the Github:

1) https://github.com/SyncFree/CISE
2) https://github.com/netdata/netdata
3) https://github.com/naver/pinpoint
4) https://github.com/elastest/elastest
5) https://github.com/theanalyst/awesome-distributed-systems
6) https://github.com/asatarin/testing-distributed-systems
7) https://github.com/jaegertracing/jaeger

Out of these I liked:

1) https://github.com/SyncFree/CISE
   Java Annotations are used in the code. The CISE logic is able to prove that some distributed programs is safe, in the sense that it maintains some application invariant of interest. The project is about developing a SMT-based tool that automates CISE logic, and verified several example applications using the tool. The code is written in Java and well-documented as well.

2) https://github.com/netdata/netdata
   Netdata is a system of distributed real-time performance and health monitoring. It provides parallel insights, in real-time, of everything happening on the systems it runs (including containers and applications such as web and database servers), using modern interactive web dashboards. Proper analysis of different features is given on the project page which made it easier to get in-depth information about the features. The code is well-documented and stable as well.

3) https://github.com/naver/pinpoint
   Pinpoint is an open source APM (Application Performance Management) tool for large-scale distributed systems written in Java. It monitors the application in real-time. It installs APM Agents without changing a single line of code. It have minimal impact on the performance of the system. Code Coverage is 41% and description page have some snapshots about memory, CPU usage taken by the distributed system.

7) https://github.com/jaegertracing/jaeger

Jaeger is an open source for end-to-end distributed tracing. It is used for distributed transaction monitoring, performance and latency optimization, root cause analysis, service dependency analysis and distributed context propagation. Its backend is designed to have no single points of failure and to scale with the business needs. The code is stable and written in a scalable manner.

ANS 3:

Applications of different classes of algorithms:

1) **Distributed Consensus**:

   There are many applications of Distributed Consensus:

   1) Byzantine General Problems: An agreement problem in distributed systems.
   2) Chubby: The Chubby lock service for loosely-coupled distributed systems.
   3) Raft: It is similar to Paxos, which breaks the system into relatively independent subproblems, and it cleanly addresses all major pieces needed for practical systems.
   4) Byzantine Fault Tolerance: The application of the above implementations is in bitcoin, a peer-to-peer digital cash system. The bitcoin network works in parallel to generate a blockchain with proof-of-work allowing the system to overcome the Byzantine failures and reach a coherent global view of the system's state.

   I am explaining about Raft:

   1) Title: Asynchronous replication framework bases on raft algorithm.
   2) Home Page: https://github.com/zhebrak/raftos
   3) Developer: Alexander Zhebrak
   4) Developer Email: alexander.zhebrak@gmail.com
   5) Developer Home Page: https://github.com/zhebrak
   6) Problem: Distributed Consensus
   7) Solution Name: Raft Algorithm
   8) Release Date: 05/02/2017
   9) Release Version: 0.0
   10) Platforms: All
   11) Language: Python
   12) Language Version: Python 3+
   13) Lines Total: 1569
   14) Lines Pure: 994

15) Additional Attributes: commits: 27, stars: 195, forks: 15.
16) List on DistAlgo Web Site: Yes

2) **Distributed Storage**:
There are many applications of distributed storage:
- Dynamo: Amazon's Highly Available Key Value Store
- Bigtable: A Distributed Storage System for Structured Data
- The Google File System
- Cassandra: A Decentralized Structured Storage System
- CRUSH: Controlled, Scalable, Decentralized Placement of Replicated Data
- The Log: What every software engineer should know about real-time data's unifying abstraction, a somewhat long read, but covers brilliantly on logs, which are at the heart of most distributed systems
- Kafka: A Distributed Messaging System for Log Processing
- Ceph: Red Hat’s Ceph offers unified object and block storage capabilities. It’s a distributed storage solution that boasts excellent performance, scalability and reliability.
- GlusterFS: It is a highly scalable file system built for applications like media streaming and big data analytics. Professional support is available through third-party vendors. It has a large and active user community, and the website includes links to many Gluster-related blogs.

I am explaining about Kafka distributed messaging architecture:
1) Title: Kafka: A Distributed Messaging System for Log Processing
2) Home Page: https://github.com/apache/kafka
3) Developer: Apache Software Foundation
4) Developer Email: apache@apache.org
5) Developer Home Page: https://www.apache.org
6) Problem: It solved the problem of low-latency ingestion of large amounts of event data from many websites (such as LinkedIn) and infrastructure into a lambda architecture that harnessed Hadoop and real-time event processing systems.
7) Solution Name: Apache Kafka
8) Release Date: 07/30/2018
9) Release Version: 2.0.0
10) Platforms: All
11) Language: Scala and Java
12) Language Version: Scala 2.11

13) Lines Total: 5,17,736
14) Lines Pure: 3,47,007
15) Applications: Messaging, Kafka-Storm Pipeline, Metrics, Website Activity Tracking. It is designed as a distributed system which is very easy to scale out. It offers high throughput for both publishing and subscribing. It supports multi-subscribers and automatically balances the consumers during failure.
16) Additional Information:
It persists messages on disk and thus can be used for batched consumption such as ETL, in addition to real time applications.
17) Additional Attributes: commits: 5599, stars: 10012, forks: 5625.
18) List on DistAlgo Web Site: Yes

3) **Security Protocols**:
Some security protocols are:
- BAN modified Andrew Secure RPC
- BAN concrete Andrew Secure RPC
- Lowe modified BAN concrete Andrew Secure RPC
- Bull's Authentication Protocol
- Denning-Sacco shared key
- Diffie Helman
- Needham-Schroeder Public Key
- Needham Schroeder Symmetric Key: forms the basis of Kerberos Authentication protocol.
- Transport Layer Security (TLS): TLS is used to secure a web server. If used with a web server, TLS can encrypt online transactions and confidential data relayed between a user's web browser and a website. A secured web server can be identified by a padlock symbol at the bottom of the browser window or in the address bar, as well as by a URL that begins with "https" rather than "http".

I am explaining about Kerberos implementation here:
1) Title: A native Kerberos client implementation for Python on Windows
2) Home Page: https://github.com/mongodb-labs/winkerberos
3) Developer: Bernie Hackett
4) Developer Email: NA
5) Developer Home Page: https://github.com/behackett
6) Problem: Security Problem
7) Solution Name: Kerberos Authentication protocol

8) Release date: 11/08/2017
9) Release Version: 0.7.0
10) Platforms: Windows
11) Language: Python
12) Language Version: 3.5+
13) Lines Total: 3007
14) Lines Pure: 2251
15) Additional Attributes: commits: 61, stars: 25, forks: 7.
16) List on DistAlgo Web Site: No

4) **Distributed Programming Support**:
Some tools for verification, testing and tracing are:
- Dapper,
- Jepsen
- Verdi
- Zipkin, Apache SkyWalking, Pinpoint and HTrace.
- Jepsen
- Verdi

I am explaining about Jepsen here:
1) Title: A framework for distributed systems verification, with fault Injection
2) Home Page: https://github.com/jepsen-io/jepsen
3) Developer: Kyle Kingsbury
4) Developer Email: aphyr+recruiter-emails-will-be-published@aphyr.com
5) Developer Home Page: https://github.com/aphyr
6) Problem: Distributed Programming Support
7) Solution Name: Jepsen
8) Release date: 05/24/2018
9) Release Version: 0.1.9
10) Platforms: All
11) Language: Clojure
12) Language Version: NA
13) Lines Total: 30272
14) Lines Pure: 24594
15) Additional Attributes: commits: 1495, stars: 3361, forks: 392.
16) List on DistAlgo Web Site: Yes
