Distributed System SE 2

Hou, Jue

014695647

[jue.hou@helsinki.fi](mailto:jue.hou@helsinki.fi)

1. Byzantine generals
   1. Problem: The generals of Byzantine are preparing for a war. The army of each general is employed in a place, which is far from each other’s. The messengers, who will be responsible for communication between generals, are assumed to highly reliable. And generals have to stick together to plan a strategy. The problem is that how to identify one traitor among all the generals and sill unite all the other loyal generals to plan an acceptable strategy.

Algorithm: Each general sends messengers to all the other generals to announce his situation, which in this case is his troop strength. So every general will eventually get a vector. And let each value represent troop strength or ID of each general. Exchange these vectors among all the generals. At this time, each general should know the troop strength or ID of every other general and what they know about it. If we form these vectors to a matrix and use rows to represent vectors, all the values in the columns accordingly should be exactly the same except the column represents for the traitor. Also the row vector from the traitor may turn out to be different from others even we do not study the column which represents for his own value of troop strength or ID. Now, every loyal general can be united and meanwhile identify the traitor.

* 1. Suppose there is a distributed storage system with 4 nodes. The system has to get all the values from each node to decide the correct value to be served as the response for users’ request. And now one of nodes fails and it will send random value, which do not make any sense. The problem is how to get a correct response in spite of the fault of that node.
  2. The protocol is designed to solve the problem whether or not to carry out an operation during a network partitioning. Groups cannot communicate. But they have to decide independently whether or not to perform the transaction.
  3. The common point of the topic for generals and for distributed system is that they all require getting majority nodes involved to make the decision. The decision is based on the value from the large fraction of all possibilities of value. The difference between two topics is that for the Byzantine decision will be made after fully information exchanging, while for the Quorum-Based Protocols decision can be made once nodes realize that their group is large enough and meet all the constraints.

1. Two-phase commit
   1. When the coordinator suffers a (temporary) crash, and if this happens after nodes send the agreement message, nodes will keep waiting for the commit message from coordinator until coordinator is able to do so.

When the participant suffers a (temporary) crash, and if this happens after the coordinator sends the prepare message, the coordinator will keep waiting for the response until it reach the time threshold. Then, the transaction will be aborted. On the other hand, if this happens after getting the commit message from coordinator, coordinator will wait as well, but it will send rollback message if expired.

In order to get a strong consistency, different state is needed. The prepare message will query for agreement from all the nodes so that all the nodes will reach the same decision after transection. The agreement message or the abort message from nodes can inform coordinator the state of each node. In this way, the commit can only happen when there achieves a universal agreement. In this phase, the coordinator will make sure all the transactions will get commit after sending the message. Obviously, the commit message will lead to actual commitment of the data. The acknowledge message will inform coordinator if every node process exactly as expected. If not, rollback message will be sent. In this phase, coordinate will commit the data and look after all the nodes. Even though there is something wrong, the system should still remain consistent.

I think it is necessary. The coordinator depends on this message to decide if all the nodes get the data committed. The purpose of two-phase commit is to get a strong consistency. If acknowledge message is removed, it is possible that some nodes fail to get the commit message or get transection done and coordinator will have absolute no idea what is going on. Consequently, the system will become inconsistent.

It cannot be simplified neither with a reliable network nor a reliable nodes alone. With a reliable network, nodes can still crash at any time, it will be even worse if it is permanent. So acknowledge message and agreement message is a must. Also, nodes will have to send abort message due to some internal reason. And it is totally valid and reasonable. Some nodes may get crash during processing commit message. So rollback message is necessary for keeping consistency. With reliable nodes alone, some messages will get lost during transmitting. The acknowledge message and agreement message ensures that message is delivered as expected. Some nodes might receive the commit message while others might not because of the network. In order to keep consistency, rollback message has to be sent. In conclusion, with these conditions alone, the protocol cannot be simplified.

It will work. Because things like network partitioning and node crash might not happen that often. Also, we can assume node fault and network partition are temporary. Although it may take some time for fixing, eventually transection can be committed. If the fault is permanent, the protocol can still work on the rest of nodes. That’s why this protocol can still work.

1. Bully algorithm
   1. Each of the nodes will send a message to nodes with higher ID and wait for reply. If there are nodes with higher ID, these nodes will reply to the starting two nodes. Then the job of starting nodes is done and those nodes who reply will have their own Bully Algorithm started. If one of starting two is already the node with the highest ID, It will claim itself as coordinator. Eventually, the node with currently the highest ID will be the coordinator.
   2. Yes, it will work properly.
      1. The node with non-highest ID. Whenever it goes absent, the algorithm will continue work. Nodes with higher ID will start their own Bully Algorithm. If the node back online before getting the election message, it will also start its own Bully Algorithm. If not, reply from nodes with higher ID will also stop the algorithm of the sender. The whole algorithm will continue whenever the node back online or go absent.
      2. The node with the highest ID. Whether or not this node will be elected as the coordinator depends on the election message from the node with second alive highest ID. Because all the nodes will start their own algorithm and will get it stopped if there are higher one. If it goes absent before receiving the election message, eventually the node with second highest will get time out and claim itself as coordinator, which is also the correct result of the algorithm. If it goes absent after it replies to the second one and claim as a coordinator, some node will detect its absence and start another election. Whenever it back online, once it find itself hold the highest ID, it will broadcast to claim it as coordinator.

In a word, the whole algorithm will work properly even though the state of nodes might change.

* 1. The algorithm is as followed:
     1. Suppose the absence is both detected by Pi.
     2. Pi send ELECTION(Pi) to the next P and store the ID of itself as initial point.
     3. A point Pn receive the election message from its previous one
        1. If Pn has not store the ID of initial point or ID stored is smaller than the first ID value of the election message, Pn sends ELECTION(Pi,…..Pn) to its next point and updates Pi as initial point,
        2. Otherwise, Pn ignore the incoming message.
     4. After Pi receive the election message ELECTION(Pi,….Pj), it announce max(Pi,…Pj) as the coordinator and sends C(max(Pi,…Pj)) to next point.
     5. Circulate the C message until all nodes alive receive it.
     6. Election complete.

1. DNS
   * 1. Cache and TTL: Cache refers to the database maintained by computer system both local and DNS server itself. It contains the information about recent visits so that resolving name can be boosted. When a record is cached, a time threshold will be needed to decide how long the record should remain in the cache until it got deleted. This time threshold is TTL(Time To Live)
     2. Stub resolver: It acts as a DNS client so that application on the client machine can interact with DNS server. Since it does not support recursive query, it has to work with an area resolver. But it has cache to speed up query.
     3. Iterative/recursive lookup: Both lookup will get name resolved if the DNS server which receives the query gets record cached or stored. Otherwise, recursive lookup will contact DNS server and eventually return the name resolution. If required, DNS server will also contact other DNS server. However, iterative lookup will only return the address of another resolver that has the best chance to resolve. Then, the initial DNS server will turn to the resolver provided by the result of iterative lookup.
     4. Zone delegation: Divide the namespace into multiple and related zones. Accessing each of these zones will be delegated to the resolver of that authoritative zone.
     5. Zone transfer: The process of copying the zone information from a master server to a slave server. Master server is the data source. And when the data in master server is changed, that on slave server will also be changed through zone transfer.
     6. Glue record: A record stored in a DNS server and served as a solution to avoid a circular dependency for the zone. Since the delegation is always identified by a name, a circular dependency will be formed if the name of delegation is the name for a subdomain of the domain which is requested for resolution. For example, if querying for [www.example.com](http://www.example.com), DNS server under domain .com will refer to ns.example.com. But this name is under the domain of example.com. So DNS will have to resolve example.com first. And again, it will refer to ns.example.com. Glue record will allow DNS to provide more information for resolving the name.
   1. The enquiry from the application will first be sent to the local DNS server if not locally cached. Then the enquiry from local DNS server will send enquiry to other DNS server iteratively if not cached by itself. Eventually, one DNS server can resolve the name and send back the IP address to local DNS server.
      1. [www.cs.helsinki.fi](http://www.cs.helsinki.fi): The enquiry will be eventually sent to DNS server for domain cs.helsinki.fi. IP address will correspond to the web server of the CS department. It is located in domain cs.helsinki.fi
      2. cs.helsinki.fi: The enquiry will be eventually sent to DNS server for domain helsinki.fi. IP address will correspond to the name resolver of the CS department. It is located in domain helsinki.fi
      3. fi: The enquiry will be eventually sent to root DNS server. IP address will correspond to the name resolver for the whole Finland. It is located in root domain
      4. cs: The enquiry will be eventually sent to root DNS server. IP address will correspond to the name resolver for the domain of cs, if it really exists. Otherwise, there will be no IP address return. It should be located in root domain
      5. [www.google.com](http://www.google.com):
         1. “nslookup” in the department

Server: 128.214.4.29

Address: 128.214.4.29#53

Non-authoritative answer:

Name: www.google.com

Address: 172.217.22.164

* + - 1. <http://network-tools/>

172.217.17.196 is from United States (US) in region North America

Input: www.google.com

canonical name: www.google.com

Registered Domain: google.com

* + - 1. <http://mxtoolsbox.com/>



* + 1. [www.facebook.com](http://www.facebook.com)
       1. “nslookup” in the department

Server: 128.214.4.29

Address: 128.214.4.29#53

Non-authoritative answer:

www.facebook.com canonical name = star-mini.c10r.facebook.com.

Name: star-mini.c10r.facebook.com

Address: 157.240.2.35

* + - 1. <http://network-tools/>

31.13.71.36 is from United States (US) in region North America

Input: www.facebook.com

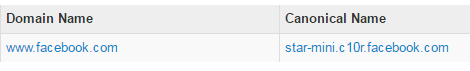
canonical name: star-mini.c10r.facebook.com

aliases:

www.facebook.com

Registered Domain: facebook.com

* + - 1. <http://mxtoolsbox.com/>





* + 1. [www.apple.com](http://www.apple.com)
       1. “nslookup” in the department

Server: 128.214.4.29

Address: 128.214.4.29#53

Non-authoritative answer:

www.apple.com canonical name = www.apple.com.edgekey.net.

www.apple.com.edgekey.net canonical name = www.apple.com.edgekey.net.globalredir.akadns.net.

www.apple.com.edgekey.net.globalredir.akadns.net canonical name = e6858.dsce9.akamaiedge.net.

Name: e6858.dsce9.akamaiedge.net

Address: 2.21.206.148

* + - 1. <http://network-tools/>

23.43.92.127 is from United States (US) in region North America

Input: www.apple.com

canonical name: e6858.dsce9.akamaiedge.net

aliases:

www.apple.com

www.apple.com.edgekey.net

www.apple.com.edgekey.net.globalredir.akadns.net

* + - 1. <http://ping.eu/nslookup>

www.apple.com is an alias for www.apple.com.edgekey.net.

www.apple.com.edgekey.net is an alias for

www.apple.com.edgekey.net.globalredir.akadns.net.

www.apple.com.edgekey.net.globalredir.akadns.net is an alias for

e6858.dsce9.akamaiedge.net.

e6858.dsce9.akamaiedge.net has address 104.81.230.44

e6858.dsce9.akamaiedge.net has IPv6 address 2a02:26f0:18:197::1aca

e6858.dsce9.akamaiedge.net has IPv6 address 2a02:26f0:18:19f::1aca

* + 1. cnn.com
       1. “nslookup” in the department

Server: 128.214.4.29

Address: 128.214.4.29#53

Non-authoritative answer:

Name: cnn.com

Address: 151.101.128.73

Name: cnn.com

Address: 151.101.192.73

Name: cnn.com

Address: 151.101.0.73

Name: cnn.com

Address: 151.101.64.73

* + - 1. <http://network-tools/>

151.101.192.73 is from United States (US) in region North America

151.101.0.73 is from United States (US) in region North America

151.101.64.73 is from United States (US) in region North America

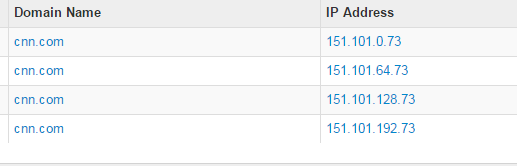
151.101.128.73 is from United States (US) in region North America

Input: cnn.com

canonical name: cnn.com

Registered Domain: cnn.com

* + - 1. <http://mxtoolsbox.com/>



* + 1. IP address mapping (By searching with db-ip.com)

|  |  |  |
| --- | --- | --- |
| Web Site | IP Address | Location(City, Country) |
| www.google.com | 172.217.22.164 | Stockholm, Sweden |
| 172.217.17.196 | Chicago, US |
| 172.217.17.100 | Amsterdam, Netherlands |
| www.facebook.com | 157.240.2.35 | Northway, US |
| 31.13.71.36 | New York, US |
| 31.13.65.36 | Atlanta, US |
| www.apple.com | 2.21.206.148 | Hamburg, Germany |
| 23.43.92.127 | Paris, France |
| 104.81.230.44 | Warsaw, Poland |
| 2a02:26f0:18:197::1aca | Paris, France |
| 2a02:26f0:18:19f::1aca | Paris, France |
| cnn.com | 151.101.0.73 | Palo Alto, US |
| 151.101.64.73 | Palo Alto, US |
| 151.101.128.73 | Palo Alto, US |
| 151.101.192.73 | Palo Alto, US |

1. Sum and substance

**MapReduce: Simplified Data Processing on Large Clusters**

In Google’s MapReduce, the model is deployed and distributed on the commodity machine instead of high-end expensive server. In order to achieve the scalability on hardware and also let the processed by a large amount of CPUs, authors design the whole model on several assumptions for many aspects. The structure of the model can be depicted as master and slaves. Master assigns jobs for slave and keep the record, while slaves do the calculation. For fault tolerance, mechanism like checkpoint and state report will come into picture. Since the map and reduce operation is atomic, the computation result should be consistent. Also, the system guarantees the order of operation to achieve consistency. However, this model did not get too much about security involved.

**The Google File System**

Like MapReduce, the GFS is also designed for inexpensive commodity machine. In order to do so, the system has to detect, tolerate and recover from failure automatically. Mechanisms like shadow master and heartbeat message are good solutions. For data input, it will spread from one chunk server to the closest chunk server. Well-defined consistency model with different type of mutation corresponds to different state of consistency help to identify mutation and keep data consistency. On the other hand, Master takes care of management and consistency. Like the MapReduce, authors did not mention too much about security.

**Bigtable: A Distributed Storage System for Structured Data**

BigTable is a distributed storage system for structure data of large scale and have a flexible control on data. However, it largely depends on the infrastructure of GFS and Chubby service. So, physical scalability, fault tolerant and data consistency can be expected. As an essential part of the system, master server will take care of physical maintenance of BigTable and some other essential matters but not all of them. And the Chubby service will have a more logical control on the system. Things such as locating a tablet and lock service will get Chubby involved. These two infrastructures also join their hands and make the Bigtable even more reliable.

**Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing**

RDD is implemented in Spark with Scala, which also help it to be a scalable data analysis tools. Compared to Google’s MapReduce, RDD is kept persistent in the memory and have more reusability for intermediate result. The data consistency is trivial for RDD because it is not immutable once an RDD is generated in the memory. Besides checkpoint, the mechanism of lineage is also applied not only for easily identifying the parents of an RDD but also help to fault tolerant. Again, RDD did not get to much security issue involved.

**The Sybil Attack**

Sybil Attack mainly focus on security issue. The mechanism of the Sybil Attack is that the faulty remote entities counterfeit and present multiple identities to the local entity so that it can defeat the effect of redundancy. The redundancy is originally used for reducing the dependence on entities which might be hostile. Based on some extremely strict conditions, the Sybil Attack can be defeated. However, it is neither practical nor possible to be practiced on the real world machines as the system requirement.

**Evolve or Dead Die: High-Availability Design Principles Drawn from Google’s Network Infrastructure**

In this passage, authors focus on the issues related to failure which appeared in the Google network. The goal of the whole service is that the downtime can only be a few minutes per month. In order to tackle these failures, authors mentioned a log with unique causes for people to discuss and learn the lesson. Also, for each plane, there are many strategies for failure and fast recovery. For example, redundant control element can provide extra robustness for the system. Even a fallback strategy can be sometime useful when encounter a failure, though it can be pretty embarrassing.

**CAP Twelve Years Later: How the “Rules” Have Changed**

In this passage, authors review the CAP theorem and mentioned a model to handle network partitioning. And author chooses availability over consistency. In this model, nodes have to detect the start of partition and then enter the partition mode, when the data may start to become inconsistent. After the end of partition, data have to be merged or canceled as the system requested or according to a well-defined strategy so that consistency can be restored. And some compensations have to be made.

**Paxos Made Simple**

Author described an algorithm to achieve consensus between processes. The algorithm is based on the fact that two sets of majority of the whole nodes must at least contain one node shared in both set. Based on this fact, a single decision can be made in spite of the existence of multiple decisions. There are three main roles for the algorithm: proposer, accepter and learner. In the phase one, proposer sends “prepare” message to a majority of acceptors. And acceptors will decide to response or not. If responded, which also means promised and value accepted, proposer in phase two will send a numbered proposal with a selected value. And the acceptors will decide to accept it or not according to their current state. For learning an accepted value, acceptors will send their acceptance to a set of distinguished learner. And the value will get passed to all the learners.

**Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System**

In this passage, authors illustrate a mobile computation environment. And authors choose availability over consistency when encounter a network partitioning. By introducing the tentative state with related mechanism, system allows to keep inconsistent data and have it changed or merged later according to the template made by programmer. Together with merge procedure, the system can achieve eventual consistency. Within a proper environment, Bayou can achieve the security of only letting allowable operation performed. Also, by getting application involved, the system can get the advantage of semantics for both conflict detection and resolution.

**Bitcoin: A Peer-to-Peer Electronic Cash System**

Bitcoin framework is aimed to provide an electronic currency without the authorization from the third party. Since all the transactions are published online, the framework of bitcoin is based on cryptography algorithm so that the security of transaction is partially ensured. A transaction between a payer and payee will require previous transaction id, payer’s signature and payee’s public key for verification. By link with others, a block chain formed. Nodes will only continue their work on the longest block chain. So only waiting for some time, a hacker will have to use a large amount of resource to catch up the block chain and override the transaction. In this way, double-spending can be avoided. However, this result is based on the assumption that honest nodes hold the majority of CPUs power and can therefore defeat the hacker.