Distributed Systems

Federica Consoli

January 15, 2018

1. Explain how active and passive replication systems handle node failures.

Solution: In the case of passive replication there are three possible scenarios.

- Primary fails after the client receives the answer
- Primary fails before sending update messages
- Primary fails after sending update messages and before receiving all ack messages

In the first case, if the response is lost the client will retransmit the request after a timeout. The new primary will recognize the request re-issued by the client as already processed and it will send back the result without updating the replicas.
In the second case, the client will not get an answer and the request will be re-issued after a timeout: the new primary will handle the request as new.
In the third scenario, there is the issue of atomicity: the update needs to be received either by all or by no one. Another primary will be elected among the correct replicas.
With active replication, no recovery action is needed upon the failure of a replica.

2. Define the properties for point-to-point links (fairloss, stubborn, perfect) and explain their relationship with regards to UDP and TCP

Solution: Fair-loss links have three main properties:

- fair loss
- finite duplication
- no creation

Stubborn links have two properties:

- stubborn delivery
- no creation

Perfect links have three properties:

- reliable delivery
- no duplication
- no creation

Both TCP and UDP are fair loss. In the case of UDP, no acks are collected and therefore there is no indication whatsoever on whether the message was delivered or not. Moreover, messages are not retransmitted. On the other hand, TCP does have an ack system, but the reliable delivery is not guaranteed (e.g. packets are lost when a network cable is disconnected).

3. Explain the difference between uniform and non uniform consensus. In addition, describe how to modify the flooding consensus algorithm in order to satisfy the uniform agreement property.

Solution: The difference between uniform and non-uniform consensus lies in the "agreement" property. The uniform agreement property ensures that no two processes decide different values, regardless of whether they are correct or not. This is different from the regular agreement property, which states that no two correct processes decide different values.
The regular flooding consensus algorithm works in rounds: at each round, every process disseminates its set in a proposal message by using the best effort broadcast abstraction. A process decides when it has reached a round during which all proposals by correct processes have been gathered.
To satisfy the uniform agreement property, the flooding algorithm needs to be modified. In the uniform version, the algorithms runs for N rounds and every process decides only in round N. The uniform agreement property is satisfied because all processes that reach round N have the same set of values in their proposal set.

4. Consider the algorithms implementing regular reliable broadcast in a synchronous and in an asynchronous setting. Discuss how the algorithms must change moving from a synchronous setting to an asynchronous one and discuss their complexity in terms of number of exchanged messages.

Solution: The algorithm for implementing regular reliable broadcast in a synchronous system is the lazy algorithm. Messages are broadcast using the best effort broadcast primitive. If the sender does not crash, then the message is delivered by all correct processes. If, however, a process delivering a message detects that its sender crashes, the message is retransmitted to all other messages. This algorithm is called lazy because retransmission happens only upon detecting a crash.
In an asynchronous system, an eager version of the algorithm is used. In this version, the failure detector primitive is not used: each process receiving a message relays it immediately, hence the name "eager".
The eager algorithm has the same complexity in both the best and the worst case scenario: n BEB messages for one RB message (this is the case with $n-1$ failures). For the lazy algorithm, one BEB message is required for one RB message in the best case, while $n-1$ BEB messages are required for one RB message (this is the case with n-1 failure).

5. Discuss why passing from a synchronous system to an eventually synchronous one, it is no longer possible in failure detectors to ensure the property of strong accuracy, while it is possible to ensure eventual strong accuracy

Solution: In a synchronous system, the bounds on communication/processing delays are known. This means that a perfect failure detector can reliably detect crashes by using timeouts: in this setup, the strong accuracy property is satisfied, because a process that has been detected by any other correct process has surely crashed. Moving to an eventually synchronous setup, we know that the system at some point in time will start acting as a synchronous one, but we have no info as to when. During the asynchrony period, bounds on communication and processing delays are not known, so it's not possible for a failure detector to make final decision on processes. In such a case we talk about failure suspicion rather than detection, and every process is able to revert its judgment over one suspected process. This can be done by means of a timeout, where the interval is increased sequentially to ensure that heartbeat messages are captured. In this setup, the eventual strong accuracy is satisfied: during the asynchrony period correct processes can be suspected (if the timeout interval is too small), but the system will eventually become synchronous, meaning that heartbeat messages from alive processes will be detected (and the judgment on them will be reversed) and no correct process will ever be suspected by a correct process.

6. Provide the specification of (i) regular reliable broadcast and (ii) uniform reliable broadcast. Discuss the system model and the assumptions employed by the implementations

Solution: The regular reliable broadcast primitive has four properties: (i) validity (ii) no duplication (iii) no creation (iv) agreement. The agreement property ensures that if a message m was delivered by a correct process, then eventually m is delivered by every correct process. This primitive can be implemented in two ways. The first is the lazy algorithm, which assumes the system is synchronous and that the processes can fail by crashes: in this algorithm, retransmission happens when the source of a message is detected to have crashed. The second algorithm is the eager one, which assumes the system is asynchronous and that faulty procesess cannot be reliably detected. This algorithm is eager in the sense that each process retransmits every message it delivers.
The uniform reliable broadcast primitive differs from the regular one by means of the agreement property: uniform broadcast ensures that if a message m is delivered by some process (whether correct or faulty) then eventually m is delivered by every correct process. Uniform broadcast can be achieved in a synchronous system by using a system of acknowledgments, which allows correct processes to deliver a message only when it has been seen by all correct processes. In the case of an asynchronous system, the algorithm is slightly changed so that deliveries are made when a majority of correct processes has seen the message.

7. Describe how the structure of a consensus algorithm changes when the system is synchronous or eventually synchronous. Discuss the assumptions on the number of correct processes and the number of messages needed to reach consensus (with/without failures).

Solution: In an synchronous system, the algorithm works in round and uses a perfect failure detector and a best-effort broadcast abstraction. Each process maintains a set of proposed values it has seen, which is disseminated at each round through a PROPOSAL message. A process decides when it has reached a round during which it has gathered all proposal that it will ever see by every correct process. When the system is partially synchronous, the approach involves a rotating coordinator system. In this implementation, there is a majority of correct processes $n > 2f$ and each process has access to an eventually perfect failure detector. Firstly, every process sends its current estimation of the decided value to the coordinator c (together with a timestamp). The coordinator gathers a majority of such values, selects the one with the highest timestamp and sends it to every process. Two things can happen at every correct process p: either a) p accepts the new estimates and sends an acknowledgement to the coordinator or b) p suspects c, sends a nack to c and goes to the next round. The coordinator gathers all acks and nacks. If a majority of ack is received, the current estimate is decided as value and it is broadcast to all processes. If one nack is received, c goes to the next round.

8. Describe the best effort, regular and uniform broadcast primitives. Provide a run which is best effort but not regular and one that is regular but not uniform

Solution: In the best effort broadcast primitive, the delivery of a message is ensured as long as the sender does not crash. If the sender crashes, processes may not agree on whether or not to deliver the message. This primitive can be implemented with a straightforward algorithm, where the message that needs to be broadcast is simply sent to all the other processes: the algorithm is fail-silent, since no assumptions are made on failure detection. With regular reliable broadcast, correct processes agree on the set of messages to deliver even when the sender fails (agreement property). This can be achieved in two ways, either with a lazy or an eager algorithm. The lazy algorithm is based on a

synchronous system model where processes can fail by crashing: whenever the crash of a sender is detected (by means of a failure detector), all of its messages are relayed by the correct processes. The eager algorithm is based on an asynchronous system model with no assumption on crash failures: in this version, each process receiving a message immediately relays it to all other processes.
The uniform reliable broadcast ensures that if a message is delivered by some process (faulty or not) then that message is eventually delivered by every other correct process. To implement this primitive, an ack system is used: in a synchronous model (where processes can fail by crashing), delivery of a message can happen only when all the acks are collected, which means that every correct process in the system actually saw the message. If the system is asynchronous, delivery happens when a majority of acks is collected.

9. Discuss how the algorithm for reliable broadcast changes when moving from crash failures assumption to Byzantine failures. In particular, discuss the relationship between number of processes in the system (n) and the number of failures (f).

Solution: In a system with Byzantine processes, the reliable broadcast primitive uses a system of echoes to ensure totality (which means that either every correct process delivers a message or no one does). The implementation works in three rounds. Firstly, a message m is broadcast to every process. Upon receiving a message, a process replies with an "echo" message that is disseminated to every other process. When a message receives a Byzantine quorum of echo messages ($(N+f)/2$), it disseminates a READY message to all process to indicate the willingness to delivery the message. Once $2f+1$ READY messages are received, the message is delivered. Moreover, whenever a process receives $f+1$ ready messages it sends a ready message itself (provided it hasn't already done it): this is an amplification step that ensures the totality property.
Implementing this primitive in a fail-arbitrary model with N processes requires that $N > 3f$.

10. Discuss the Primary Backup replication scheme, point out which inconsistency problems could arise in replicas when the primary crashes and write the steps you need to recover to a consisted state of replicas when a new primary is elected.

Solution: The primary backup replication scheme is made up of two components: primary and backups. The primary receives invocations from clients and returns the appropriate answers, while backups interact with the primary and are used to guarantee fault tolerance by replacing a primary when a crash occurs. When backups receive update messages by the primary, they update their state and send back an acknowledgement. The primary waits for an ack message from each correct backup and then sends back the answer to the client. When a primary fails, there are three possible scenarios; if the primary fails after sending the answer, the client may not get it: in such case, the request will be retransmitted by the client. The new primary will recognize the request as already processed and it will send back the result without updating the replicas.
If the primary fails before sending update message to backups, the client will never get an answer. The request will be re-issued after a timeout and the new primary will handle it as new.
If the primary fails after sending update messages and before receiving all the ack messages, then there is a need to guarantee atomicity, meaning that the update must be received either by all or none of the replicas.

11. Discuss in which system model is best to use probabilistic broadcast rather than reliable broadcast. Discuss the parameters that can be tuned in an implementation of probabilistic broadcast based on gossiping in order to improve the probability of delivery at each process.

Solution: Probabilistic broadcast is the best approach when the system needs to be scalable. With reliable broadcast, processes need to collect acknowledgments in order to ensure reliability in the presence of process crashes. However, when the group of process becomes very large, this task becomes overwhelming and may impact performance/cost. With probabilistic broadcast, a process sends a message to a set of randomly chosen k processes. When a process receives said message for the first time, it relays it to a set of k randomly chosen processes (this concludes a round). With this primitive, the validity property is weaker than the original one, as it accounts for a small failure probability ε. The fanout parameter (k) and the number of rounds (R) can be tuned in order to increase the probability of delivery: by increasing k, the probability of a message to reach every process increases, while the number of rounds decreases. This operation is costly, because it increases the load on each process and the redundancy of message.

12. Discuss the hierarchy of total order primitives and discuss the order and agreement properties. Show an original run that satisfies

    - TO(NUA, SUTO) but not TO(UA, SUTO)
    - TO(UA, WUTO) but not TO(UA, SUTO)

Solution: We can build a total order hierarchy based on the assumption made on the agreement and total order properties. The agreement property can be either uniform (if a process p delivers a message m then all correct processes deliver m) or non uniform (if a correct process p delivers a message m then all correct processes deliver m). The total order property can be either strong or weak. The SUTO property ensures that processes deliver the exact same ordered set until some delivery is omitted: after the omission, the delivery set has to be disjointed from that of other processes. The WUTO property imposes ordering on pairs of messages delivered by pairs of distinct processes: it does not prevent a process from omitting to deliver a message while still delivering the same messages delivered by other processes. As a consequence, the sequence of messages delivered by a process may contain holes with respect to the sequences of messages delivered by other processes. Both SUTO and WUTO have non-uniform counterparts, namely SNUTO and WNUTO. It holds that SUTO $\Rightarrow$ WUTO and SNUTO $\Rightarrow$ WNUTO, but also SUTO $\Rightarrow$ SNUTO and WUTO $\Rightarrow$ WNUTO.
Each specification weaker than TO(UA,SUTO) allows a faulty process to experience one of the following event: delivery of spurious messages, omission of a message in the delivery sequence, delivery of messages in an order not consistent with the one of correct processes.

13. Describe the basic approaches for information dissemination inside publish/subscribe systems discussing characteristics and limitations

Solution: There are four main approaches for information dissemination inside the publish/subscribe systems: event flooding, subscription flooding, filter-based routing and randez-vous routing. With event flooding, events are broadcast from the publisher to the whole broker network. This implementation is straightforward and it does not cause memory overhead, but it has the highest message overhead.
With subscription flooding, each broker maintains a "subscription table" with copies of each subscription. When a publisher generates an event, the tables are used to match the subscriptions and directly notify the interested subscribers. This approach has optimal diffusion (with a large memory overhead) but it is not practical when subscriptions change frequently.
With filter-based routing, subscriptions are partially diffused in the system and used to build routing

tables. When events need to be diffused, these routing tables are used to build a multicast tree that connect the publisher to all the interested subscribers.
Finally, randez-vous routing is based on two functions, SN and EN, which are used to associate, respectively, subscriptions and events to brokers in the system. SN(s) returns the set of nodes that are responsible for storing s and forwarding received events matching s to all the appropriate subscribers. EN(e) returns the set of nodes that must receive event e to match it against their subscriptions. The event routing is a two-phases process: first an event e is sent to all brokers returned by EN(e), which then match it against the subscription they store and notify the corresponding subscribers.

14. Consider the FIFO, Causal and Total Order broadcast primitives. Describe the relations (equivalence, orthogonality and inclusion) that exist among them, providing examples as a motivation to your answer.

Solution: In the FIFO broadcast primitive, messages from the same sender are delivered in the order they were sent. In the Causal Broadcast primitive, messages are delivered such that the happened-before relationships are respected: this means that delivery of a message is delayed until every message that causally precedes it is delivered. Causal order $\Rightarrow$ Fifo order, which means that every run that respects Causal is also Fifo (causal order = fifo order + local order). In the Total order broadcast primitive, all messages are delivered in the same order, even those that are not causally related. There are two important properties: agreement and total order. The agreement property states that correct processes deliver the same set of messages, while the t.o. property states that all correct process deliver the messages in the same order. T.O. is orthogonal to both FIFO and Causal order, which means there may exists some runs that are T.O. but not C.O. and FIFO and vice versa.

15. Provide the specification of the Byzantine Consistent Broadcast communication primitive, describe an algorithm implementing it and discuss the relationship between the number of processes n and the number of Byzantine failures f

Solution: The Byzantine Consistent Broadcast primitive has four main properties: validity (if a correct process p broadcasts a message m the every correct process eventually delivers m), no duplication, integrity (if some correct process delivers a message m with sender p and process p is correct, then m was previously broadcast by p) and consistency (if some correct process delivers a message m and another correct process delviers a message m', then m=m'). In order to achieve these, the algorithm works in round. Firstly, a message m is broadcast by the sender to all other processes. In the second round, every process acts as a witness for the message m and resends it in an ECHO message to all others. When a process receives more than $(N + f)/2$ ECHO messages containing the same message m, m is delivered. In order to make this work in a system with N processes, we neeed $N > 3f$.