An Implementation of Failure Detection for Large-scale Distributed Systems

Yu Xiangzhan
(Department of computer science of Harbin Institute of technology, China)

Abstract: In this paper we discuss the problems that the failure detecting of the large-scale distributed system faces, analyze the advantages and disadvantages of the methods proposed before, and propose a new failure detecting implementation combining hierarchical protocol and gossip-style protocol. This implementation adopts hierarchical design and divides the nodes on the large-scale distributed network into several groups. Push model is adopted for the detecting of the nodes of the same group, while the transmission of heartbeat message between different groups uses a protocol similar to gossip-style protocol. It not only shortens detecting time, but also decreases network load. And the leader node can conveniently realize the participation and withdrawal of nodes of the system, which increases the extensibility greatly. Through the experiment and analysis it shows that our implementation possesses the timeliness, accuracy, adaptability and extensibility.

Keywords: failure detection, fault tolerance, distributed system, large-scale, hierarchical.

1. Introduction

Failure detectors are well-known as basic building blocks for fault-tolerant distributed systems. Failure detectors are used in a wide variety of settings, such as network communication protocols [1], computer cluster management [2], group membership protocols [3,4,5], etc.

In a broad sense, failure detectors running in a distributed system provide some information on which processes have crashed. Chandra and Toueg [6] provide the first formal specification of unreliable failure detectors and show that they can be used to solve some fundamental problems in distributed computing, namely, consensus and atomic broadcast.

With rapid development of network technology, applications spread from LAN to WAN, and failure detection is extended into large scale of distributed system. Hayashibara[7,8] discussed some problems of implementing a failure detection service for a large-scale distributed system.

In order to reduce network load, people carry out research for the structure of failure detection system, and hierarchical and gossip-style protocol are proposed. In [9], multi-layer gossip-style protocol is to perform the transmission of detection message by introducing the means of virus infection. Bradley has carried out simulative testing on the performances of gossip-style protocol [10]. In the gossip-style protocol, topology of the systems can be changed dynamically, and this protocol can reduce the network load. In contrast, there are drawbacks. First, this protocol does not work well when a large percentage of components crash or become partitioned away. Second, detecting the occurrence of a specific failure can potentially take a fairly long time. Third, the gossip protocol models assume that the system begins at steady state; all live nodes are included in the gossip list on start-up. If this assumption were not made, each member could broadcast its list in order to be located initially. If a network does not support broadcasts, one or more gossip servers could be added to the system. A gossip server is an additional member.
with a well-known address that is likely to be available even in the case of network partitioning.

In [11,12,13], hierarchical structure is proposed. In this structure, failure detectors obtain information of suspicions by either messages from other failure detectors or by monitoring target components directly. Therefore its network load is less than traditional approaches and it can address the scalability problem. However these approaches are based on a static tree structure and thus fail to address the dynamism problem. As for the approach proposed in [11], if a detector fails, a lot of configuration information will lose, and the corresponding detection will stop. As for the approach proposed in [12], the leader itself is a weakness. If a group leader fails, the detection operation of the group will stop.

Our implementation combines hierarchical protocol and gossip-style protocol, adopts hierarchical design, and divides the nodes on the network into several groups. Every monitor keeps a list including the information of all monitors. In a group, monitors detect each other, whereas the means of notification is adopted to transmit heartbeat messages among different groups. Each monitor updates the content of the list by analyzing the heartbeat message received. A monitor can know the state of every node by inquiring the content of the correlative item of the list. In addition, there is a leader node in each group, which is responsible for the management of the nodes.

The rest of this paper is organized as follows: section 2 describes the distributed system model. In section 3, we describe our system model. Section 4 presents the basic layer. And in section 5, we present hierarchical protocol of our implementation. Finally, in section 6, we show the performance experimentation result, and we get a conclusion in section 7.

2. Distributed System Model

In order to carry out research for failure detection better, distributed system model needs to be established. As for the distributed system, two properties are most important: transmission delay of messages and relative operation speed of process. According to the two properties, distributed system can be divided into Synchronous System and Asynchronous System.

2.1 Synchronous System

In the model of Synchronous System, the upper limits of transmission delay of messages and the relative operation speed of process are all known, in addition, it is supposed that a process has synchronous clock. In such system, applying timeout mechanism can realize reliable failure detection. Because the model of Synchronous System restricts the transmission delay of messages and the relative operation speed of process, few of the distributed systems meet this kind of model.

2.2 Asynchronous System

In the model of Asynchronous System, the transmission delay of messages and the relative operation speed of process have no upper limits, the clock between processes is not synchronous. At present, for the distributed system based on WAN, the transmission delay of messages and the relative operation speed of process are all unknown, therefore, this belongs to the model of Asynchronous System. Work in this paper focuses on this type of system model.

2.3 Failure Model

Typical distributed system is composed of many processes. The failure of process is unavoidable. According to the behavior after process fails, failure can be divided into two kinds:

- Fail-stop failure: after such failure appears, the process will stop and there will not be any output. Before such failure appears, the process runs with completely normal state.
- Byzantine failure: this kind of failure will cause the process into a random state. Operation is not stopped, but it is not performed according to predetermined rules, and the output is not correct. Generally, this kind of failure is caused by logic error or malicious attacks.

In this paper, we focus on the first type of failure. This kind of failure is generally caused by the error of hardware, software or natural disasters.

3. Model and Frame

3.1 Model of System
Failure detection service of our implementation aims at a large-scale distributed system. Logically, the entity of system is divided into four types: process, monitor, node and group. Process means performing some function procedure in distributed system. Node means one computing node in distributed system, i.e. one computer or one host. At each node, many processes may run, however, only one monitor runs. A group consists of many nodes. The failure of some process will not affect the other processes and monitors. But if a node collapses completely, the above processes and monitor will fail completely.

3.2 Design Goals

Given this system model, the main concerns that should be addressed in the design of failure detection service for a large-scale distributed system are:

- Accuracy and completeness. The failure detection must identify failure accurately, with both false positives and false negatives being rare.
- Timeliness. Failure must be identified in a timely fashion.
- Low load. The heartbeat message in failure detection is surely to influence network. With the extension of scale of distributed system, failure detection services should not cause too large load to network.
- Self-adaptability. In the asynchronous distributed system, the network situation changes rapidly, especially, the transmission delay of messages and the loss of messages will affect the accuracy of failure detection, therefore, the self-adaptability of failure detection must be consolidated.
- Scalability. The design of the failure detection service must be capable of scaling to large numbers of processes and nodes.
- Extensibility. In a large-scale distributed system, topological structure may change frequently. This requires the failure detection system to adapt to such change.

3.3 Frame of System

As illustrated in Figure 1, our failure detection service architecture is defined in terms of three types of entities:

1) A monitor is responsible for observing the state of both the node on which it is located and any monitored process on that node. It generates periodic heartbeat messages, summarizes this status information, and sends it to other monitors.

2) Leader is a special monitor. It has all functions of a monitor, and at the same time leader manages the participation and withdrawal of nodes in the group.

3) A group is composed of many monitors. The division principle of group is determined according to requirements. In general, all monitors in a LAN are taken as a group; it will become the basic layer of distributed failure detection system; many groups constitute the second layer.

![Figure 1. Architecture of the failure detection service](image)

As shown in Figure 1, the distributed failure detection system is a two-layer structure; the nodes in broken line, in the same LAN, make up of a group; many such groups constitute a large distributed failure detection system.

Each monitor maintains a list, a comprehensive list, in which the information of all nodes is included. We denote this list as `entire-list`. Each item in the list records the information of each node, such as to which group a node belongs to, attribution information of the group, IP address, and information of the states of processes at the node. The heartbeat message sent by each monitor is a list with the structure same as the `entire-list`, and in the list, only one item or some items may be contained. The intervals that each monitor sends heartbeat message are the same.
In every interval, each monitor sends a heartbeat message, a list, to the other monitors in the group. It does not send the *entire-list* to the other monitors. In the list, only one item is included, i.e. its own item. After receiving such message, other monitors will replace the corresponding item in their *entire-list*.

The detection between monitors in different groups introduces gossip-style protocol. In every interval, one monitor per group will send heartbeat message to other groups on the average. This message is also a list only containing the items of all nodes in the local group. After the monitors of other groups receive such information, they will replace the corresponding contents in their *entire-list*. Subsequently, in the next interval, the list will be sent to the other members in their groups. From here, we know, averagely, through three intervals, the detection between the monitors of different groups can be completed.

4. Basic Layer

In this section, we will introduce the operation process of each monitor of a group in detail.

![Frame of the basic layer](image)

4.1 Failure Detection

As shown in Figure 2, in a LAN, there are many nodes, and on each node, many processes and one monitor run. The monitor inquires the state of each process, and at the same time, each process can query the process state of other nodes from the monitor. In each interval, each monitor sends heartbeat message to the other monitors in the group.

Each monitor keeps a list, *entire-list*. Each item in the list contains all information of a node. The information is composed of two parts: one part reflects the group attribution of the node, i.e. subnet, netmask and flag; the other part reflects the attribution of the node, including IP address, state, length and content. The information format and definition is as follows:

- **Subnet**: group attribution
- **Netmask**: group attribution
- **Flag** represents whether this monitor is a leader or not.
- **Member ID**: identify this node.
- **IP address**: the IP address of this node on which the monitor runs.
- **State** shows in which state this node is. It can be crashed or normal.
- **Length** represents the length of the content.
- **Content** includes the state of processes on this node.

After monitor receives heartbeat message, it will replace the corresponding items in its *entire-list*. The processes operating at nodes will inquire the *entire-list* at any time to learn the state of the other nodes or processes. The state information will be shown in state field and content field.

For the detection between the monitors in the same group, chen’s algorithm [14] is applied. Within the estimated time, if the heartbeat message sent by a detected node is not received, the detected node will be judged as a failure. And the state filed in the item of the detected node in *entire-list* will be changed into crash.

Failure is composed of process failure and node failure. A monitor will consider the processes that run on the same node as failure by inquiring these processes, and the monitor will revise the content field in the item of the node in its *entire-list*, then in the next interval, the monitor will send heartbeat message, a list, which includes only one item of this node to all nodes in the group. The node failure can be detected only by applying chen’s algorithm. Within the estimated time, in the same group, if node A does not receive the heartbeat message sent by node B, node A will consider node B as...
a failure; it will revise flag field in the item of node B in its entire-list; the content of flag field will be revised as crash.

We can know that the process failure is notified to other monitors in the same group, while the node failure is detected by other monitors in the same group.

4.2 Node Management

The function of leader in a group is to manage the nodes in the group. As shown in Figure 3, when a new node will join the system, it only needs to know the IP address of any monitor in the system, and then sends an application to the monitor. The monitor will find to which group the new node should belong according to the attribution information of the group in its entire-list. And, it will send the application of the new node to the leader of the group. After the leader receives the information, if it finds it is a new member, it will assign a Member ID to the new node, and add a new item for this node in the entire-list, then send the renewed entire-list to all members in the group. At the same time, it will send a list containing items of all the nodes in the group to the leaders of other groups, and the leaders of other groups will transmit the information to their members, then all members renew the corresponding contents of their entire-list.

The withdrawal of node is similar to the participation of the node. The node to withdraw sends information to the leader of the group, and the leader will delete the item of this node in the entire-list, and then notify other nodes.

The Member ID can only be increased. After a node withdraws, its original Member ID will not be used again. Even when the node requires joining the system again, the leader will assign a new ID to it. Such operation is to prevent inconsistent entire-list caused by frequent participation and withdrawal requirements of one node. In addition, when a node fails, its entire-list loses; after the node recovers, the node will reapply to participate; so long as its IP address does not change, it is not a new member and its item will not be deleted; the leader will not assign a new ID to it, and it will still use its original ID.

In the process that a new member applies to participate, if message loses, the participation process of the new node will not be completed. If the leader does not receive the application, the new node will reapply. If the leader receives application, in the process of notifying the other nodes, due to loss of message, some nodes can not know that the entire-list is changed. Therefore the entire-list in different nodes can be inconsistent. However, we can assure that the leader of each group can grasp the complete item of all nodes in its group. When some node receives heartbeat message, if it finds that in the heartbeat message there are new items, which does not exist in its entire-list, it will send the application of renewing entire-list to the leader of the group. At the time, if the leader has the items on its entire-list, it will return the items to the node; if the leader does not have the items on its entire-list, it will analyze the group attribution of these items. If these are the nodes of other groups, the leader will send renewing application to the leader of these nodes; if these are the nodes of the same group, the leader will send reapplication order to these nodes. (Such problem will appear after a new leader elected again).

4.3 Election of A Leader

As for the election of leader, we adopt the method described in [12]. After the election succeeds, the new leader will notify other leaders with a list that includes items of all nodes in the group. Other leaders will send the list to members in their groups, and at same time send a list that includes all nodes in their groups to the new leader. The new leader will renew the corresponding item of its entire-list, and the new leader will send its entire-list to all members in its group. All members in the group will renew their own entire-list and keep consistent with the leader.

Two kinds of situations will trigger the election of
leader: failure of leader node and withdrawal of leader. These two kinds of situations are different. If leader node fails, its item will still be kept in the *entire-list*. While for the withdrawal of leader, its item will be deleted from the *entire-list*. Before a leader withdraws, it should notify the leaders of other groups, then all members in its group. After receiving notices, the members in its group will delete its item from the *entire-list*, and then begins to elect a new leader.

4.4 Participation of a Leader

The participation of a leader is a little complex. The participation of leader means increasing a new network partition and a new group. The first member in the new group is the leader of the group, and it sends the application to any monitor of system. The monitor will find it is a new node and belongs to a new group, and then send the application to its leader. The leader will notify the new node that you are a leader of a new group. Then the new leader will continue the process of section 4.3.

4.5 Withdrawal of a Group

If a group will withdraw completely, firstly, all members in the group should withdraw, and then the leader withdraws.

5. Hierarchical configuration

5.1 Failure Detection among Groups

![Figure 4. Detection among groups](image)

As shown in Figure 4, we adopt the means of notification to implement failure detection among groups. In every interval, one monitor per group will send heartbeat message to other groups on the average. The concrete description is shown as follows: In a group including N nodes, every N-1 intervals every monitor will select a member randomly per group, and send a list including items of all nodes of its group to them. If there are K groups, it will send K-1 lists. Thus, in other groups, some members will receive the list, and in the next interval, when they send heartbeat messages to other monitors in their groups, they will attach the list with the heartbeat message and send it to other monitors. The monitors receiving the list will renew their corresponding items in their *entire-list*.

5.2 Analyses

We can see that the failure detection among groups is similar to the multi-layer gossip-style protocol. But there are some differences: 1) the failure detection for the members in different groups is not judged by timeout mechanism, but determined by the means of notification. A member of any group knows the state of processes and nodes of other groups by querying its *entire-list*. The items of all nodes in other groups in *entire-list* are renewed by receiving the heartbeat message from other groups. 2) in gossip-style protocol, only a node is selected for sending heartbeat message at random. Here, in every other group, a node will be selected as randomly as a heartbeat message receiver. If there are K groups, in one time, K-1 pieces of heartbeat messages will be sent.

6. Analysis

6.1 Network Load

Under stable state, all heartbeat messages are transmitted from the detected node to the monitor. Under the situation that the number of nodes is fixed, the network load is fixed. Let us compare the network load when applying gossip-style protocol with the network load when applying our implementation. Supposing that there are K groups in the system, and each group has N nodes, thus, in the *entire-list* there are N*K items; supposing that each item has B bytes, then a complete *entire-list* occupies N*K*B bytes. For gossip-style protocol, averagely in every interval, the network load is N^2*K^2*B bytes. In our implementation, we know within an interval, each member sends a list
including one item to all other members in the group, and the network load is \( N \times (N-1) \times B \) bytes. At the same time, in each group, averagely, a member will still send a list including \( N \) items to other groups, which will produce the network load: \( (K-1) \times N \times B \) bytes. In addition, in each group, one member will send a list including \( N \) items to other members in the group, and this will produce the network load: \( (N-1) \times N \times B \) bytes. Then the total load averagely within an interval is:

\[
(N \times (N-1) \times B + (K-1) \times N \times B + (N-1) \times N \times B) \times K \text{ bytes.}
\]

We suppose

\[
(N \times (N-1) \times B + (K-1) \times N \times B + (N-1) \times N \times B) \times K \leq N^2 \times K^2 \times B
\]

then,

\[
\begin{align*}
2N+K-3 & \leq N \times K \\
2(N-1) & \leq K \times (N-1) \\
2 & \leq K
\end{align*}
\]

So, when \( K \geq 2 \), the network load of our implementation is smaller than that of gossip-style protocol.

6.2 Detection Time

The members in the same group monitor each other. Therefore, according to chen’s algorithm, within the time that is a little longer than an interval, each member will know the process state and node state of other members. In addition, in section 5.1, we can know, averagely within at most 3 intervals, the items of all members in a group can be transmitted to all members of other groups.

7. Conclusion

This paper proposes a new implementation of failure detection for a large-scale distributed system. In this implementation, all nodes are uniform; when any node fails, it will not affect the operation of other nodes; the format of all heartbeat messages is same. And the logical structure and configuration of our implementation are very simple, so its robustness is good. It is adaptive to the system with a large number of nodes.

As for network load, in section 6.1, we can know that my implementation is better than gossip-style protocol. And the network load is fixed.

As for timelessness, in section 6.2, we can know that the detection time is correlative with the interval mainly.

As for accuracy, Chen’s algorithm itself has high accuracy, and this can be known in [15]. In addition, in a group, the possibility of message loss and transmission delay is small. So long as we select a proper safety margin, the accuracy of Chen’s algorithm can be assured. Because we adopt the means of notification to implement failure detection among different groups, the wrong suspicion will occur rarely.

As for the flexibility, chen’s algorithm itself has the ability of adapting to network change. In the detection for members among different groups, we apply the means of notification, so the change of transmission delay will not affect the detection very much.

As for the extensibility, my implementation can adapt to the change of network topology. The existence of leader easily realizes the participation and withdrawal of nodes. Each leader preserves the most authoritative and complete member information in its group. At the same time, the existence of leader will not become the weakness of the entire system. Except completing the function of member management, each leader is equal to other monitors. The failure of the leader will not affect the failure detection of other nodes. In addition, by election, new leader can be produced immediately when old leader fails.

References


