

Research on data consistency method based on distributed network architecture

Hehui Zhang^{*a}, Xi Song^b, Yong Yang^a, Wenhui Li^b

^aState Grid Gansu Electric Power Research Institute, Lanzhou 730050, Gansu, China; ^bState Grid Gansu Electric Power Company, Lanzhou 730046, Gansu, China

ABSTRACT

In response to the demand for fault tolerance in distributed network systems, this paper analyzes the challenges of distributed architecture to the design of Byzantine Recovery System. As distributed architecture replaces the centralized architecture, Byzantine Fault Tolerance designs using internal bus or direct connection for communication between FCRs have become more and more challenging. This paper proposes a data consistency method based on the distributed network architecture, which can achieve data fault tolerance through independent redundant network communications, which is applicable to the design of fault-tolerant systems for network systems or sensor systems using distributed architectures.

Keywords: Distributed architecture, data consistency, error tolerance

1. INTRODUCTION

The consistency and reliability of data is related to the normal operation of the core system, due to the complexity and diversity of the cosmic space environment, the failure modes of the devices in the space environment are not completely known. This requires the study of fault tolerance methods in the case of unclear failure modes, so that systems in the case of partial hardware failure, are still able to fulfill the expected tasks. Tolerance of arbitrary failure modes is called Byzantine Resilience, and its basic approach is to isolate arbitrary modes of failure occurring during computation by using redundant computers, providing them with identical inputs, executing the same code, running them synchronously, and reach data consistency by comparing the outputs. Since the Byzantine Recovery System is extremely reliable due to the shielding of module or individual machine failures caused by hidden unknown faults through hardware redundancy, it enables the system to be effective in terms of key safety and reliability performance indicators.

However, traditional Byzantine recovery for data agreement assumes that the communication link between redundant computers is perfectly reliable. In practice, to ensure such reliability, the systems are forced to place redundant computers centrally and connect those redundant computers via a reliable internal bus¹. However, with the development of distributed network technologies, centralized architectures are becoming more and more difficult to meet technological needs. Under the distributed architecture, redundant computers can be interconnected through the network, and the reliability of the communication cannot be guaranteed due to the possible failure of the network itself (e.g., failure of the network switch or failure of the network link, etc.)². Based on the above needs, this paper proposes a redundancy architecture and mechanism that can be used to improve the reliability of critical individual computers in distributed architectures by achieving data consistency under the premise of unreliable communication networks.

2. BYZANTINE FAULT TOLERANCE AND CHALLENGES OF DISTRIBUTED ARCHITECTURE

2.1 Problem of the Byzantine Generals

The concept of Byzantine fault tolerance is derived from the solution of the Byzantine Generals Problem, a distribution consistency protocol challenge. Faulty components may send contradictory messages to different parts of the system, and dealing with such faults can be abstractly described by the Byzantine Generals Problem: Imagine that the Byzantine army is divided into several parts stationed outside the enemy's city, and that each part is under the command of its own general, who can only send messages to each other through messengers. After observing the enemy situation, they must make a

*zh950620@163.com

common plan of action, however, some generals may be traitors and will inevitably prevent the loyal generals from agreeing³. This decision making process requires an algorithm to ensure:

- (A) All loyal generals adopt the same course of action.
- (B) A handful of traitorous generals cannot make a loyal general adopt a wrong plan.

Let v_i be the message sent by the i th general, to satisfy condition A, the following condition must hold:

- 1) Every loyal general must be given the same information vector ($v_1, v_2, v_3 \dots$)
- 2) If the i th general is loyal, then all other generals will use the same message sent by the resulting general v_i .

The problem describes a system of n generals with f traitors among the n generals (including the main general), where the main general sends an order to each vice general. The goal is to design such a protocol to ensure that if the main general is loyal, all loyal vice-generals receive messages from that main general, or if the main general is a traitor, all loyal vice-generals at least agree on a value.

2.2 Byzantine restoration systems

The general in the Byzantine general problem corresponds to the processor of the computer system, the traitor corresponds to the processor that suffers a Byzantine failure, the messenger corresponds to the inter-processor communication link, and agreeing on a common battle plan corresponds to the failure-free processor agreeing on the input data. Applying the solution to the Byzantine generals' problem to the design of highly reliable computers, it is possible to develop fault-tolerant computer systems that work correctly when a processor experiences a Byzantine failure. Such a system can be illustrated by the following example: suppose a computer system consisting of four processors, A , B , C , and D , in which processor B fails and B may have the following failure modes⁴:

- Fail-Stop Fault: B fails and B stops working;
- General Fault: B fails, and B continues to work, but B gives the same message to A , C and D ;
- Byzantine fault: B fails, B continues to work, and the information B gives to A , C , and D is inconsistent. For example, the information exchange between B and A is all normal; the information interaction between B and C is all normal; but the interaction between B and D is erroneous; since the states presented by B to A , C , and D are inconsistent, it is very easy to cause misjudgment, such as misjudging that D is bad. The system that prevents this kind of failure phenomenon is called Byzantine resilient system.

A computer system that can tolerate arbitrary, single random failures is called a 1-Byzantine recovery system.

Arbitrary can be interpreted to mean that the faulty processor mentioned above gives contrary, contradictory information to other normal processors, and so requires a more elaborate design to protect against it.

Byzantine Recovery's design requirements for a fault-tolerant computer are as follows⁵:

- (1) All FCRs must be synchronized to a certain time domain (the upper limit of time difference is determined), which is called synchronization requirement.
- (2) Each FCR maintains independent communication links with at least $2f+1$ other FCRs, called connectivity requirement;
- (3) The existence of at least $3f+1$ FCRs (stand-alone or modular) is called the cardinality requirement;
- (4) The need to maintain at least $f+1$ rounds of communication between FCRs is called communication requirement.

An architecture that satisfies the above conditions is called an f -Byzantine recovery fault-tolerant architecture. In a fault-tolerant architecture that satisfies 1-Byzantine recovery, it is necessary to have four FCRs, each connected to the others through three disjoint communication links, and two rounds of exchanges need to be performed between the FCRs to obtain a consistency judgment.

2.3 Challenges of distributed architecture

Traditional computers with Byzantine fault tolerance usually use centralized layouts with interconnections between FCRs via internal buses or direct connections, and the interconnections are considered to be reliable. Distributed architectures, on the other hand, are receiving more and more attention due to their advantages in terms of damage resistance, integrated

resource utilization, and cable network optimization, especially the distributed layout sensor systems and network transmission systems. Figure 1 shows a typical DIMA (Distributed Integrated Modular Avionics) architecture proposed in the field of aeronautics, which is characterized by the sharing of system resources such as processors, memories, and sensor interfaces, and the use of a distributed system architecture to disperse all the integrated modules throughout the entire vehicle and connect all the modules through a real-time and fault-tolerant communication network, thus effectively reducing the cable length and achieves shorter response times. The distributed architecture, represented by DIMA, breaks the traditional centralized layout of the area, which poses a new challenge to the design of the Byzantine Recovery System.

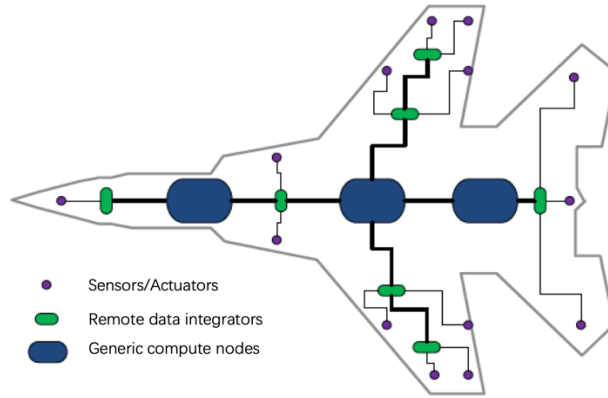


Figure 1. Schematic diagram of DIMA architecture.

In response to this problem, credit-based Byzantine Fault Tolerance is proposed⁶. First, a set of candidate nodes is added to realize the dynamic joining and withdrawal of consensus nodes, so as to ensure that when node replacement is required, nodes can be selected from the candidate node set to replace the consensus nodes. Secondly, introduce a credit evaluation scheme, calculate its credit value according to the node's completion of consensus, and use the credit value to evaluate the credit of the node⁷. Finally, design a node replacement scheme, when the credit value of a consensus node is lower than the set threshold, use a candidate node replaces this node to reduce the consensus participation rate of dishonest nodes⁸. The comparative experiments between VBFT algorithm and PBFT algorithm are carried out, and the results show that the consensus delay of the improved algorithm is reduced, the throughput is increased⁹, and the algorithm efficiency is improved. Voting-based Byzantine Fault Tolerance is proposed¹⁰. When a Byzantine error occurs on the master node, the protocol of PBFT randomly selects a new master node according to the node number¹¹. Although such a selection scheme is simple, it is prone to the situation that the selected node is still a Byzantine node, resulting in a waste of resources.

3. DESIGN OF DATA CONSISTENCY ALGORITHM BASED ON DISTRIBUTED NETWORK ARCHITECTURE

In order to realize data consistency under the distributed network architecture, it is necessary to connect each independent redundant terminal T_i to an independently operating redundant network N_i , and the redundancy fault-tolerant architecture is shown in Figure 2. The redundant terminals T_i are of identical design, the redundant networks N_i are physically isolated from each other, and the interfaces between T_i and each network are independent of each other.

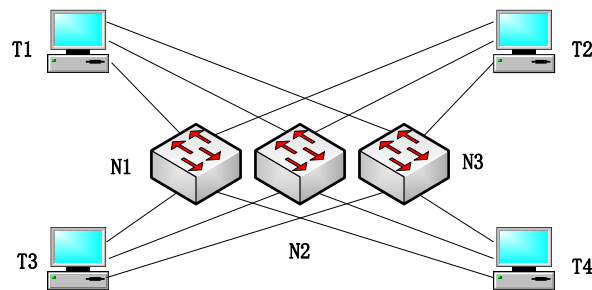


Figure 2. Schematic diagram of fault-tolerant architecture.

The assumptions that need to be met in order to achieve data consistency under a distributed network architecture include:

- (1) All redundant terminals T_i must be synchronized to a certain time domain (upper limit of time difference determined);
- (2) The system tolerates the failure of m arbitrary terminals and requires the presence of at least $3m+1$ redundant terminals;
- (3) The system tolerates n arbitrary network failures and requires the existence of at least $2n+1$ independent networks;
- (4) Each redundant terminal T_i is connected via at least $2n+1$ independent networks and can maintain independent communication links;
- (5) At least $m+1$ rounds of communication must be maintained between redundant terminals.

Where m denotes the number of faulty terminals that the system can tolerate, and n denotes the number of faulty networks that the system can tolerate. For example, the system needs to maintain data consistency in the case of 1 arbitrary terminal failure and 1 arbitrary network failure, which requires at least 4 redundant terminals, 3 independent networks, and requires each redundant terminal to maintain communication links with the other 3 redundant terminals through 3 networks at the same time, and after at least 2 rounds of communication to reach data consistency.

Taking the typical redundant fault-tolerant architecture in Figure 2 as an example, assume that the data generated by T_1-T_4 are A, B, C, D respectively, i.e., the information vector is $[A, B, C, D]$, and the data agreement is reached when T_1-T_4 all receive $[A, B, C, D]$ correctly through the information interaction and the redundant information processing. For this purpose, the data interaction process and redundant information processing are defined as follows:

(1) First round of information interactions

T_1-T_4 send the information generated by this terminal to the other three redundant terminals through $N1-N3$ networks respectively, and the sending method can be unicast, multicast or broadcast (due to the unreliability of the network itself, there is a probability of error for any sending method), take T_1 as an example, the information sent by T_1 to the y terminal through the x network can be written as A_{xy} , and the sending matrix of T_1 can be expressed as

$$\begin{bmatrix} A_{12}, & A_{13}, & A_{14} \\ A_{22}, & A_{23}, & A_{24} \\ A_{32}, & A_{33}, & A_{34} \end{bmatrix}$$

(2) First round of redundant information processing

After the first round of information interaction, each redundant terminal will receive information from other redundant terminals through different networks, taking T_1 as an example, its received information matrix can be expressed as follows

$$\begin{bmatrix} B_{11}, & B_{21}, & B_{31} \\ C_{11}, & C_{21}, & C_{31} \\ D_{11}, & D_{21}, & D_{31} \end{bmatrix}$$

Voting is performed for each row of information in the receiving matrix, and according to assumption condition 3, the receiving matrix after voting can be obtained as $[B_i, C_i, D_i]$, and other redundant terminals adopt the same redundant information processing method.

(3) Second round of information interactions

The redundant terminals will send the information received in the first round and after the completion of voting to all other redundant terminals separately through different networks, taking T_1 as an example, its sending information matrix can be expressed as follows:

$$\begin{bmatrix} B_{112}, & B_{122}, & B_{132} \\ C_{112}, & C_{122}, & C_{132} \\ D_{112}, & D_{122}, & D_{132} \end{bmatrix} (T_1 \text{ sends a message matrix to } T_2)$$

$$\begin{bmatrix} B_{113}, & B_{123}, & B_{133} \\ C_{113}, & C_{123}, & C_{133} \\ D_{113}, & D_{123}, & D_{133} \end{bmatrix} (T_1 \text{ sends a message matrix to } T_3)$$

$$\begin{bmatrix} B_{114}, & B_{124}, & B_{134} \\ C_{114}, & C_{124}, & C_{134} \\ D_{114}, & D_{124}, & D_{134} \end{bmatrix} (T_l \text{ sends a message matrix to } T_l)$$

(4) Second round of redundant information processing

After the second round of information interaction, each redundant terminal will receive information from other redundant terminals through different networks, taking T_l as an example, its received information matrix can be expressed as follows

$$\begin{bmatrix} A_{211}, & A_{221}, & A_{231} \\ C_{211}, & C_{221}, & C_{231} \\ D_{211}, & D_{221}, & D_{231} \end{bmatrix} (\text{Receive } T_2 \text{ information matrices})$$

$$\begin{bmatrix} A_{311}, & A_{321}, & A_{331} \\ C_{311}, & C_{321}, & C_{331} \\ D_{311}, & D_{321}, & D_{331} \end{bmatrix} (\text{Receive } T_3 \text{ information matrices})$$

$$\begin{bmatrix} A_{411}, & A_{421}, & A_{431} \\ C_{411}, & C_{421}, & C_{431} \\ D_{411}, & D_{421}, & D_{431} \end{bmatrix} (\text{Receive } T_4 \text{ information matrices})$$

First vote on the same source information arriving from different networks, according to the assumption condition 3, the voting can exclude the data inconsistency caused by the network failure, after voting the received information matrix can be expressed as

$$\begin{bmatrix} A_{21}, & A_{31}, & A_{41} \\ B_{11}, & B_{31}, & B_{41} \\ C_{11}, & C_{21}, & C_{41} \\ D_{11}, & D_{21}, & D_{31} \end{bmatrix}$$

Then comparing the redundant information sent by different terminals of voting, according to the assumption condition 2, we can get $[A,B,C,D]$ and reach the data agreement.

4. CONCLUSION

In this paper, the limitations of Byzantine fault tolerance mechanism under distributed architecture, a data consistency method based on distributed network architecture is proposed. This method is applicable to distributed architectures such as DIMA, in that key devices can realize data consistency under distributed network through independent redundant communication, which in turn can be used for redundancy backup and data fault tolerance of a distributed network systems

REFERENCES

- [1] Ji, D. and Feng, D., "Asynchronous byzantine agreement protocol based on verifiable signature sharing," *Journal of Electronics*, 23(1), 64-68 (2006).
- [2] Palumbo, D. L. and Butler, R. W., "A performance evaluation of the software-implemented fault-tolerance computer," *Guidance, Control, and Dynamics*, 9(2), 175-180 (1986).
- [3] Gao, S., "T-PBFT: an eigentrust-based practical byzantine fault tolerance consensus algorithm," *China Communications*, 16(12), 111-123 (2019).
- [4] Huang, D. Y., Li, L., Chen, B., et al., "RBFT: a new Byzantine fault-tolerant consensus mechanism based on Raft cluster," *Journal on Communications*, 42(3), 209-219 (2021).
- [5] Ang, Y., Song, Z. and Cheng, T., "Improvement research of PBFT consensus algorithm based on credit," *International Conference on Blockchain and Trustworthy Systems*, Berlin, 47-59 (2019).
- [6] Xu, J., Zhao, Y., Chen, H., et al., "ABC-GSPBFT: PBFT with grouping score mechanism and optimized consensus process for flight operation data-sharing," *Information Sciences*, 624, 110-127 (2023).
- [7] Cui, M., Zhang, J., Xue, Q., et al., "Improvement of practical Byzantine fault tolerance algorithm based on node reputation value matching," *Second International Symposium on Computer Technology and Information Science (ISCTIS 2022)*, 12474, 72-78 (2022).

- [8] Qin, H., Cheng, Y., Ma, X., et al., "Weighted byzantine fault tolerance consensus algorithm for enhancing consortium blockchain efficiency and security," *Journal of King Saud University-Computer and Information Sciences*, 34(10), 8370-8379 (2022).
- [9] Zhong, W., Feng, W., Huang, M., et al., "ST-PBFT: an optimized PBFT consensus algorithm for intellectual property transaction scenarios," *Electronics*, 12(2), 325 (2023).
- [10] Liu, S., Zhang, R., Liu, C., et al., "Improvement of the PBFT Algorithm Based on Grouping and Reputation Value Voting," *International Journal of Digital Crime and Forensics (IJDCF)*, 14(3), 1-15 (2022).
- [11] Jin, B., Hu, Y., Tao, H., et al., "An improved practical Byzantine fault-tolerant consensus algorithm combined with aggregating signature," *7th International Symposium on Advances in Electrical, Electronics, and Computer Engineering*, 12294, 1175-1182 (2022).