

Lecture Notes in Electrical Engineering 1112

Ji Su Park
Hiroyuki Takizawa
Hong Shen
James J. Park *Editors*

Parallel and Distributed Computing, Applications and Technologies

Proceedings of PDCAT 2023

 Springer

Series Editors

Leopoldo Angrisani, *Department of Electrical and Information Technologies Engineering, University of Napoli Federico II, Napoli, Italy*

Marco Arteaga, *Departament de Control y Robótica, Universidad Nacional Autónoma de México, Coyoacán, Mexico*

Samarjit Chakraborty, *Fakultät für Elektrotechnik und Informationstechnik, TU München, München, Germany*

Jiming Chen, *Zhejiang University, Hangzhou, Zhejiang, China*

Shanben Chen, *School of Materials Science and Engineering, Shanghai Jiao Tong University, Shanghai, China*

Tan Kay Chen, *Department of Electrical and Computer Engineering, National University of Singapore, Singapore, Singapore*

Rüdiger Dillmann, *University of Karlsruhe (TH) IAIM, Karlsruhe, Baden-Württemberg, Germany*

Haibin Duan, *Beijing University of Aeronautics and Astronautics, Beijing, China*

Gianluigi Ferrari, *Dipartimento di Ingegneria dell'Informazione, Sede Scientifica Università degli Studi di Parma, Parma, Italy*

Manuel Ferre, *Centre for Automation and Robotics CAR (UPM-CSIC), Universidad Politécnica de Madrid, Madrid, Spain*

Faryar Jabbari, *Department of Mechanical and Aerospace Engineering, University of California, Irvine, CA, USA*

Limin Jia, *State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University, Beijing, China*

Janusz Kacprzyk, *Intelligent Systems Laboratory, Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland*

Alaa Khamis, *Department of Mechatronics Engineering, German University in Egypt El Tagamoa El Khames, New Cairo City, Egypt*

Torsten Kroeger, *Intrinsic Innovation, Mountain View, CA, USA*

Yong Li, *College of Electrical and Information Engineering, Hunan University, Changsha, Hunan, China*

Qilian Liang, *Department of Electrical Engineering, University of Texas at Arlington, Arlington, TX, USA*

Ferran Martín, *Departament d'Enginyeria Electrònica, Universitat Autònoma de Barcelona, Bellaterra, Barcelona, Spain*

Tan Cher Ming, *College of Engineering, Nanyang Technological University, Singapore, Singapore*

Wolfgang Minker, *Institute of Information Technology, University of Ulm, Ulm, Germany*

Pradeep Misra, *Department of Electrical Engineering, Wright State University, Dayton, OH, USA*

Subhas Mukhopadhyay, *School of Engineering, Macquarie University, Sydney, NSW, Australia*

Cun-Zheng Ning, *Department of Electrical Engineering, Arizona State University, Tempe, AZ, USA*

Toyoaki Nishida, *Department of Intelligence Science and Technology, Kyoto University, Kyoto, Japan*

Luca Oneto, *Department of Informatics, Bioengineering, Robotics and Systems Engineering, University of Genova, Genova, Italy*

Bijaya Ketan Panigrahi, *Department of Electrical Engineering, Indian Institute of Technology Delhi, New Delhi, Delhi, India*

Federica Pascucci, *Department di Ingegneria, Università degli Studi Roma Tre, Roma, Italy*

Yong Qin, *State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University, Beijing, China*

Gan Woon Seng, *School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, Singapore*

Joachim Speidel, *Institute of Telecommunications, University of Stuttgart, Stuttgart, Germany*

Germano Veiga, *FEUP Campus, INESC Porto, Porto, Portugal*

Haitao Wu, *Academy of Opto-electronics, Chinese Academy of Sciences, Haidian District Beijing, China*

Walter Zamboni, *Department of Computer Engineering, Electrical Engineering and Applied Mathematics, DIEM—Università degli studi di Salerno, Fisciano, Salerno, Italy*

Junjie James Zhang, *Charlotte, NC, USA*

Kay Chen Tan, *Department of Computing, Hong Kong Polytechnic University, Kowloon Tong, Hong Kong*

The book series *Lecture Notes in Electrical Engineering* (LNEE) publishes the latest developments in Electrical Engineering—quickly, informally and in high quality. While original research reported in proceedings and monographs has traditionally formed the core of LNEE, we also encourage authors to submit books devoted to supporting student education and professional training in the various fields and applications areas of electrical engineering. The series cover classical and emerging topics concerning:

- Communication Engineering, Information Theory and Networks
- Electronics Engineering and Microelectronics
- Signal, Image and Speech Processing
- Wireless and Mobile Communication
- Circuits and Systems
- Energy Systems, Power Electronics and Electrical Machines
- Electro-optical Engineering
- Instrumentation Engineering
- Avionics Engineering
- Control Systems
- Internet-of-Things and Cybersecurity
- Biomedical Devices, MEMS and NEMS

For general information about this book series, comments or suggestions, please contact leontina.dicecco@springer.com.

To submit a proposal or request further information, please contact the Publishing Editor in your country:

China

Jasmine Dou, Editor (jasmine.dou@springer.com)

India, Japan, Rest of Asia

Swati Meherishi, Editorial Director (Swati.Meherishi@springer.com)

Southeast Asia, Australia, New Zealand

Ramesh Nath Premnath, Editor (ramesh.premnath@springernature.com)

USA, Canada

Michael Luby, Senior Editor (michael.luby@springer.com)

All other Countries

Leontina Di Cecco, Senior Editor (leontina.dicecco@springer.com)

**** This series is indexed by EI Compendex and Scopus databases. ****

Ji Su Park · Hiroyuki Takizawa · Hong Shen ·
James J. Park
Editors

Parallel and Distributed Computing, Applications and Technologies

Proceedings of PDCAT 2023

Editors

Ji Su Park
Department of Computer Science
and Engineering
Jeonju University
Cheonjam-ro, Korea (Republic of)

Hong Shen
School of Applied Sciences
Macao Polytechnic University
Macao SAR, China

Hiroyuki Takizawa
Cyberscience Center
Tohoku University
Sendai, Japan

James J. Park
Department of Computer Science
and Engineering
Seoul National University of Science
and Technology
Seoul, Korea (Republic of)

ISSN 1876-1100

ISSN 1876-1119 (electronic)

Lecture Notes in Electrical Engineering

ISBN 978-981-99-8210-3

ISBN 978-981-99-8211-0 (eBook)

<https://doi.org/10.1007/978-981-99-8211-0>

© The Editor(s) (if applicable) and The Author(s), under exclusive license
to Springer Nature Singapore Pte Ltd. 2024

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd.
The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

Paper in this product is recyclable.

Message from the PDCAT 2023 General Chair

The 24th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT) is a major forum for scientists, engineers, and practitioners throughout the world to present their latest research, results, ideas, developments, and applications in all areas of parallel and distributed computing. Beginning in Hong Kong in 2000, PDCAT 2023 will be held in Jeju, Korea, after 24 years of successful journey through various countries/regions including Taiwan, Japan, China, Singapore, Australia, New Zealand, and Korea across Asia-Oceania. We are inviting new and unpublished papers.

The conference papers included in the proceedings cover the following topics: PDCAT of Networking and Architectures, Software Systems and Technologies, Algorithms and Applications, and Security and Privacy. Accepted and presented papers highlight new trends and challenges of Parallel and Distributed Computing, Applications and Technologies. We hope readers will find these results useful and inspiring for their future research. Our special thanks go to the Program Chairs: Ji Su Park (Jeonju University, Korea), Hiroyuki Takizawa (Tohoku University, Japan), Hui Tian (Griffith University, Australia), and all Program Committee members and all reviewers for their valuable efforts in the review process that helped us to guarantee the highest quality of the selected papers for the conference.

James J. Park
Hong Shen
PDCAT 2023 General Chairs

Organization

Honorary Chair

Doo-soon Park

SoonChunHyang University, Korea

General Chairs

James Park

SeoulTech, Korea

Hong Shen

Sun Yat-sen University, China

Program Chairs

Ji Su Park

Jeonju University, Korea

Hiroyuki Takizawa

Tohoku University, Japan

Hui Tian

Griffith University, Australia

Workshop Chairs

Michael Hwa Young Jeong

Kyung Hee University, Korea

Neil Y. Yen

The University of Aizu, Japan

Publicity Chairs

Sushil Kumar Singh

SeoulTech, Korea

Yong Zhang

Shenzhen Inst. of Adv. Tech., China

Byeong-Seok Shin

Inha University, Korea

Kwang-il Hwang

Incheon National University, Korea

Joon-Min Gil

Daegu Catholic University, Korea

Deok-Gyu Lee

Seowon University, Korea

Byoungwook Kim

Gangneung-Wonju National University, Korea

Local Arrangement Chairs

| | |
|----------------|---------------------------|
| Yan Li | Inha University, Korea |
| Yeongwook Yang | Hanshin University, Korea |

Registration and Finance Chair

| | |
|-------------|----------------------------------|
| Jungho Kang | Baewha Women's University, Korea |
|-------------|----------------------------------|

Program Committee

| | |
|------------------|--|
| Yuebin Bai | Beihang University, China |
| Raj Bayyar | University of Melbourne, Australia |
| Guoliang Chen | U. of Sci. & Tech. of China, China |
| Lin Chen | Sun Yat-sen University, China |
| Yawen Chen | Otago University, New Zealand |
| Luo Cao | Wuyi University, China |
| Huanwei He | Zhongshan College, China |
| Zhenxiong Hou | Northwestern Polytechnical University, China |
| Shi-Jin Horng | Nat. Taiwan U. of Sci. and Tech, China |
| Longkun Guo | Fuzhou University, China |
| Mingyu Guo | University of Adelaide, Australia |
| Mirjana Ivanovic | Univ. of Novi Sad, Serbia |
| Teofilo Gonzalez | Univ. of Calif. Santa Barbara, USA |
| Ajay Gupta | Univ. of Western Michigan, USA |
| Huaxi Gu | Xidian University, China |
| Francis Lau | Univ. of Hong Kong, China |
| Shuangjuan Li | South China U. of Agriculture, China |
| Yidong Li | Beijing Jiaotong Univ, China |
| Yamin Li | Hosei University, Japan |
| Weifa Liang | Australian National Univ, Australia |
| Manu Malek | Stevens Institute of Technology, USA |
| Rui Mao | Shenzhen University, China |
| Depei Qian | Beihang University, China |
| Yingpeng Sang | Sun Yat-Sen University, China |
| Michael Sheng | Macquarie University, Australia |
| Jigang Wu | Guangdong Univ. of Tech., China |
| Chengzhong Xu | University of Macau, China |
| Jingling Xue | Univ. of New South Wales, Australia |
| Haibo Zhang | Otago University, New Zealand |

| | |
|----------------------|--|
| Yong Zhang | Shenzhen Inst. of Adv. Tech, China |
| Zonghua Zhang | Huawei France, France |
| Xiaofan Zhao | Police University of China, China |
| Cheng Zhong | Guangxi University, China |
| Albert Zomaya | University of Sydney, Australia |
| Dongwann Kang | SeoulTech, Korea |
| Jun-Ho Huh | Korea Maritime and Ocean University, Korea |
| Alireza Souri | Department of Computer Engineering, Iran |
| Ka Man | Xi'an Jiaotong-Liverpool University, China |
| Vimal Shanmuganathan | Ramco Institute of Technology, India |
| JongBeom Lim | Pyeongtaek University, Korea |
| Jaehwa Chung | Korea National Open University, Korea |
| Yeong-Seok Seo | Yeungnam University, Korea |

Contents

| | |
|---|-----|
| A Blockchain System for Fake News Detection | 1 |
| <i>Janusz Bobulski</i> | |
| Formalization and Verification of the Zab Protocol Using CSP | 12 |
| <i>Wenting Dong, Jiaqi Yin, Sini Chen, and Huibiao Zhu</i> | |
| Using MPIs Non-Blocking Allreduce for Health Checks in Dynamic Simulations | 25 |
| <i>Jana Gericke, Harald Klimach, Neda Ebrahimi Pour, and Sabine Roller</i> | |
| Parallelizable Loop Detection using Pre-trained Transformer Models for Code Understanding | 32 |
| <i>Soratouch Pornmaneerattanatri, Keichi Takahashi, Yutaro Kashiwa, Kohei Ichikawa, and Hajimu Iida</i> | |
| SAHF-LightPoseResNet: Spatially-Aware Attention-Based Hierarchical Features Enabled Lightweight PoseResNet for 2D Human Pose Estimation | 43 |
| <i>Ali Zakir, Sartaj Ahmed Salman, and Hiroki Takahashi</i> | |
| List-Based Workflow Scheduling Utilizing Deep Reinforcement Learning | 55 |
| <i>Wei-Cheng Tseng and Kuo-Chan Huang</i> | |
| Federated Learning for Skin Cancer Classification | 62 |
| <i>Zhe-Kai Xu and Yen-Wen Lin</i> | |
| A Task Offloading and Content Caching Strategy for the Internet of Vehicles in Cloud-Edge Environment | 68 |
| <i>Yaping Wang, Junye Qiao, Zekun Hu, and Pengwei Wang</i> | |
| Privacy-Preserving Retrieval Scheme Over Encrypted Medical Records with Relevance Ranking | 81 |
| <i>Wanting Lei, Xiehua Li, Yingzhu Wang, and Xiaoyu Mei</i> | |
| A Data-Centric Approach for Efficient and Scalable CFD Implementation on Multi-GPU Clusters | 93 |
| <i>Ruitian Li, Liang Deng, Zhe Dai, Jian Zhang, Jie Liu, and Gang Liu</i> | |
| Research on Psychological Testing Methods of Criminal Suspects Based on Multi-features of EEG | 105 |
| <i>Yijie Peng and Xiaofan Zhao</i> | |

| | |
|---|-----|
| Insider Trading Detection Algorithm in Industrial Chain Based on Logistics Time Interval Characteristics | 118 |
| <i>Fulin Chen, Kai Di, Hansi Tao, Yuanshuang Jiang, and Pan Li</i> | |
| Link Attributes Based Multi-service Routing for Software-Defined Satellite Networks | 130 |
| <i>Xueyu Lu, Wenting Wei, Liying Fu, and Dong Zhang</i> | |
| A Fuzzy Logical RAT Selection Scheme in SDN-Enabled 5G HetNets | 143 |
| <i>Khitem Ben Ali and Faouzi Zarai</i> | |
| SSR-MGTI: Self-attention Sequential Recommendation Algorithm Based on Movie Genre Time Interval | 154 |
| <i>Wen Yang, Ruibo Yue, Yawen Chen, and Jun Zhao</i> | |
| Fine Time Granularity Allocation Optimization of Multiple Networks Industrial Chains in Task Processing Systems | 167 |
| <i>Pan Li, Kai Di, Xinlei Bai, Yuanshuang Jiang, and Fulin Chen</i> | |
| ϵ -Maximum Critic Deep Deterministic Policy Gradient for Multi-agent Reinforcement Learning | 180 |
| <i>Yuanshuang Jiang, Kai Di, Zhongjian Hu, Fulin Chen, Pan Li, and Yichuan Jiang</i> | |
| Effective Density-Based Concept Drift Detection for Evolving Data Streams | 190 |
| <i>Zelin Cui, Hui Tian, and Hong Shen</i> | |
| An End-to-End Multiple Hyper-parameters Prediction Method for Distributed Constraint Optimization Problem | 202 |
| <i>Chun Chen, Yong Zhang, Li Ning, and Shengzhong Feng</i> | |
| Dynamic Priority Coflow Scheduling in Optical Circuit Switched Networks | 215 |
| <i>Hongkun Ren, Hong Shen, and Xin Wang</i> | |
| Deep Reinforcement Learning Based Multi-WiFi Offloading of UAV Traffic ... | 227 |
| <i>Zhiyong Liu and Hong Shen</i> | |
| Triple-Path RNN Network: A Time-and-Frequency Joint Domain Speech Separation Model | 239 |
| <i>Yu-Huan Zhai, Qiang Hua, Xiao-Wen Wang, Chun-Ru Dong, Feng Zhang, and Da-Chuan Xu</i> | |

Design of Query Based Gallery Selector and Mask-Aware Loss for Person Search 249
Qiang Hua, Ao Sun, Yu-Chen Liu, Feng Zhang, Chun-Ru Dong, and Da-Chuan Xu

A Privacy-Preserving Blockchain Scheme for the Reliable Exchange of IoT Data 260
Mnar Alnaghes, Nickolas Falkner, and Hong Shen

R-RPT-A Reliable Routing Protocol for Industrial Wireless Sensor Networks 272
Kripanita Roy and Myung-Kyun Kim

Action Segmentation Based on Encoder-Decoder and Global Timing Information 283
Yichao Liu, Yiyang Sun, Zhide Chen, Chen Feng, and Kexin Zhu


Security Challenges and Lightweight Cryptography in IoT: Comparative Study and Testing Method for PRESENT-32bit Cipher 295
Van Nam Ngo, Anh Ngoc Le, and Do-Hyeun Kim

The Prediction Model of Water Level in Front of the Check Gate of the LSTM Neural Network Based on AIW-CLPSO 306
Linqing Gao, Dengzhe Ha, Litao Ma, and Jiqiang Chen

Author Index 313



A Blockchain System for Fake News Detection

Janusz Bobulski^(✉) 

Department of Computer Science, Czestochowa University of Technology, Czestochowa, Poland
januszb@icis.pcz.pl

Abstract. The media greatly impacts the world’s perception and what is happening in it. Nowadays, the Internet is a medium where information flowing from all over the world meets. Every year, more and more information websites are created from which we learn about the reality surrounding us. The problem is that many of these websites do not use primary and reliable sources but only use sources that have already been processed and duplicated. Very often, these portals omit not only important facts for a given event but also reproduce false information. There are also situations where fake news is deliberately created and disseminated. The phenomenon called “fake news” has long been known as disinformation or propaganda. Contemporary scientific analyses focus primarily on its new dimension, i.e. the dissemination of untrue or incorrect information on the Internet, the reasons for the popularity of fake news on the Internet, and the speed and manner of information dissemination. However, the most important problem is preventing the spread of fake news. In the article, the authors propose a method of authorising content using blockchain technology. The presented method is resistant to manipulation attempts, confirmed by the tests.

Keywords: blockchain · fake news · cybersecurity

1 Introduction

Blockchain is a technology that enables the decentralised, secure, and immutable storage, transmission, and verification of information. As a result, it has found applications in various fields, including [1]:

- **Cryptocurrencies and finance:** Blockchain is widely used in cryptocurrencies such as Bitcoin, Ethereum, and Litecoin. It allows for direct storage and transfer of value between users without the involvement of financial intermediaries. Blockchain also enables the secure verification of financial transactions, identities, and audits.
- **Supply chains:** With blockchain, products can be traced from producer to consumer, allowing for better control over production processes, fraud prevention, and increased food safety.
- **Medical data management:** Blockchain enables the secure and unalterable storage and transfer of medical data, which can help protect patient privacy and enhance the effectiveness and speed of healthcare.

- Electronic voting: Blockchain provides secure and reliable electronic voting, which can prevent electoral fraud and ensure transparent and fair elections.
- Identification systems: Blockchain can securely and unalterably verify user identities, helping combat fraud and preventing cyber-attacks.
- Smart contracts: Blockchain enables the creation and execution of smart contracts that automatically fulfil the terms of the contract, without the need for intermediaries.
- Artificial intelligence: Blockchain can store and process large amounts of data required to train machine learning and artificial intelligence algorithms.

These are just a few examples of blockchain applications. This innovative technology has the potential to revolutionise many fields and industries. Blockchain allows for the decentralised, direct, and immutable storage and transmission of information, which makes it a promising tool in the fight against false information, particularly in verifying the credibility of content and information sources. Here is a literature review of the use of blockchain in combating fake news.

In their article [2], the authors systematically reviewed several scientific papers on using blockchain to verify fake news. They found that blockchain could detect and ensure correct information. The findings revealed that the suggested technique is satisfactory and efficient in recognising rumours and preventing their spread.

Due to availability of news and also for the free scope of sharing, most of the time rumours are being extensive in a short period of time. Detecting and preventing rumors and false information remains a significant challenge for social network. The introduction of blockchain technology has paved the way for the development of decentralised apps in order to address this issue. In this technology any information is recorded permanently. The authors in [2] explore a strategy to eliminate bogus news on social media by utilising the benefits of peer-to-peer network ideas. By issuing non-fungible token content rating we can detect and ensure appropriate news. The findings revealed that the suggested technique has a satisfactory performance and efficiency in recognising rumours and preventing their spread.

Prior research has proposed adding a quorum, a group of appraisers trusted by users to verify the authenticity of digital content, to the fake news prevention systems. In the paper [3], the authors propose an entropy-based incentive mechanism to diminish the negative effect of malicious behaviours on a quorum-based fake news prevention system. To maintain the Safety and Liveness of our system, they employed entropy to measure the degree of voting disagreement to determine appropriate rewards and penalties. They use Hyperledger Fabric, Schnorr signatures, and human appraisers to implement a practical prototype of a quorum-based fake news prevention system. The outcomes of the case analyses and experiments show that the presented mechanisms are feasible and provide an analytical basis for developing fake news prevention systems.

In conclusion, scientific literature suggests blockchain technology can help combat fake news by increasing data transparency and immutability. With further research and innovation in the field of blockchain, there is a promising prospect for its development in the fight against fake news.

2 The Structure of the Blockchain

Blockchain, i.e. a chain of blocks, is a technology used to store data securely, immutably and decentralised way. The blockchain structure consists of the following elements:

- **Blocks** - each block contains a set of transactions grouped and added to the chain at a particular time. Blocks are chained using hashes of cryptographic functions, preventing tampering with the blocks' data.
- **Block header** - The block header contains information about the block itself, such as the block number, its creation time, the hash of the previous block, and the proof of work used to validate the block's authenticity.
- **Proof of Work** - This is a mechanism used in the blockchain to protect the network against attacks and confirm the block's authenticity. It consists of solving mathematical problems using high computing power, making it difficult for attackers to falsify data.
- **Peer-to-peer network** - blockchain works based on a peer-to-peer network in which nodes are connected directly, creating a decentralised network.
- **Network Nodes** - Network nodes are the computers that create and maintain the blockchain by processing transactions, validating blocks, and sending them to other nodes in the network.
- **Addresses** - each blockchain account has its unique address used to carry out transactions.
- **Transactions** are saved in blocks and contain information about transferring values between addresses. Network nodes must verify and accept each transaction before being added to the block.
- **Time Company** - The blockchain uses a Time Company system to ensure that transactions are added to blocks in the correct order and time.

Thanks to such a structure, blockchain provides security, immutability and decentralised control over stored data.

3 The Problem of the Byzantine Generals

The blockchain network structure should ensure resistance to failures, disruptions in transmission and attempts to attack the content of individual blocks. A properly functioning decentralised system must implement mechanisms ensuring reliability in any situation.

In the case of analysing the situation with the appearance of incorrect messages in the blockchain network, the problem of Byzantine generals is often explored. Its assumptions are as follows: many Byzantine generals are planning an attack on the city. All generals must attack simultaneously or decide to retreat together to be victorious. They can only communicate with each other through messengers. The problem is that there are traitors among the generals who send the wrong messages. The solution to this theoretical problem must consider that the messages are open to all generals, including traitors. This thought experiment is described in detail in [4]. The problem of Byzantine generals' problem directly impacts distributed systems, for example, those based on blockchain.

The nodes in a distributed network, equivalent to the generals in the theoretical problem, can send and receive messages from each other. Those of them that work incorrectly is called Byzantine knots. Any node action that goes against the assumptions can be considered Byzantine action.

One of the conditions necessary for the proper operation of the blockchain network is creating a mechanism that ensures the creation of new blocks correctly. Such a mechanism is correct if it allows the product of blocks according to generally accepted consensus rules. At the same time, it shows high resistance to such problems as nodes with limited connectivity or nodes sending incorrect messages (intentionally or as a result of a failure). It is crucial that after the voting process, the servers make the same joint decision and the related action - adding a new block to the local chain or rejecting it. If even one server decides otherwise, it can lead to anomalies in the future. Supposing any server chooses to include a block that the rest of the network will not, then in future votes, that server will consider each subsequent block incorrect. When the node adds a new block, it checks whether the hash provided as the hash of the previous block in the candidate block is equal to the hash of the most recent block stored in the local copy. If the hash is incorrect, the server votes against that block.

Similarly, the system will behave when it has not previously attached a block that most servers have joined. Also, there will be a conflict when checking the consistency of the previous block's hash. Skipping a block is more likely than adding a block without majority approval. It can happen due to the temporary unavailability of the server. Let's assume, during this time, other servers vote to add a new block and accept it. The inactive server will never know about this vote unless it decides to compare its blockchain instance with one of the other servers later. However, the server does not have to be unavailable not to accept a new block. There may be a situation where a server fails to propagate its address when joining the network sufficiently. As a result, other servers fail to send their opinions about candidate blocks to it. So, the decision is made by a majority vote. It is assumed that consensus is reached if more than 50% of the servers consider the new block correct. In general, most nodes in the network will work fine.

Blockchain-based fake news detection.

Another serious problem is when servers send erroneous messages not because of an error but because of deliberate human interference. These types of situations are more dangerous because the attacker can act in a more coordinated manner and attack several servers simultaneously. One of the assumptions of a decentralised system is that each user or server has an equal impact on the network, and actions that interfere with the system's proper operation are not allowed and must be countered. Situations, where nodes send misleading messages, are closely related to the problem of Byzantine generals.

In the case of a problem with disseminating fake news, misleading messages will be untrue and should not be published. To analyse the problem, we chose a simple network consisting of four servers (A, B, C and D) during the voting process on the correctness of the new block, i.e. the truth of the information. All servers received the block and validated it. Servers B, C, and D determined the block was correct, while server A determined it was invalid and forwarded its vote information to the other servers. Even though server A cast the vote differently from the others in this case, this is not incorrect. Each server has the right to vote following the actual state of its blockchain.

In the analysed situation, the block will be accepted despite one vote against it because 75% of the servers say that the block is correct. In this case, server A will also decide to include a new block based on the votes from other servers. Since its vote was different, its blockchain is broken, and it needs to download the correct version from one of the other trustworthy servers. However, when server A (called the Byzantine or misleading server) knows that the block is correct (his correct vote would be a vote for block inclusion). It sends information to servers B and D that it thinks the block is correct, while it sends a misleading message to block C that the block is incorrect. In this way, it may disrupt the voting process in individual nodes; server B and D, after counting, consider that 100% of servers are in favor of including the block, while server C finds that only 75%.

One change increases the risk of making a wrong decision, in this case, on server C. For networks with less unanimity or more Byzantine servers, it can lead to an error in one or more nodes. In a situation where servers B and D, receive information that 75% of nodes found the block correct, they attach it to the local blockchain instance. Server C, on the other hand, as a result of the operation of the Byzantine server, receives information that the compatibility is 50%. In the case of the analysed example of fake news, a match greater than 50% is needed to reach a consensus, so server C does not accept the block. In this case, the Byzantine server achieves its goal - it manages to corrupt one of the servers, leading it to a situation where it has an outdated string.

The above analysis shows that the operation of Byzantine servers consists in sending misleading messages in such a way as to lead some servers to make an incorrect decision. Such servers may perform attacks randomly or direct them to a specific server. It is possible when the attacker controls multiple servers simultaneously and sends the wrong message only to the selected node.

The authors of this theoretical problem have already proposed the basic algorithm for solving the problem of Byzantine generals. The authors of the problem proposed the OM(m) algorithm, which solves the above problem for a group of not less than $3m + 1$ generals, where m is the number of traitors [4].

The algorithm looks like this: for the value of OM(0), the commander sends his vote to every other general, and each general uses the received value to decide. If m is greater than zero, the OM(m) algorithm assumes that each general. After receiving a message, it will additionally send it to the other generals, omitting the original author of the message; it will therefore perform the OM(m-1) algorithm.

The creators also described the operation of this algorithm for the simplest case - four generals, one of whom is a traitor. For one traitor, this is the minimum number of generals, according to the following equation:

$$3 * m + 1 = 3 * 1 + 1 = 4, \quad (1)$$

where m is the number of traitors.

If the server sending the voice is not a traitor, it sends its valid vote to all servers. Byzantine server B sends server D a false vote in the second voting round. However, server D also gets a vote for adding a block from server C. So it has three votes - two for (from the original server and server C) and one against adding a new block. So, based on the majority's decision, it accepts the new block. The algorithm must also work correctly when the original message's sender is a Byzantine server. The byzantine server sends

three messages to other servers; it is not essential what the messages are because, in the second round of sending messages, the servers will exchange received messages. Therefore all nodes that are working correctly will have the same pool of messages (1, 2, 3) and will make the same decision.

The above algorithm has several disadvantages. As the analysis shows, for n nodes, the $OM(m)$ algorithm first calls $n-1$ separate $OM(m-1)$ algorithms, each of which calls the $OM(m-2)$ algorithm $n-2$ times, which causes further recursive calls the algorithm until $OM(0)$ is reached.

It follows that for networks with more than 1 Byzantine server, a large number of messages are required to be sent. Therefore, it has a negative impact on network performance, and a mechanism should be implemented that will allow for the differentiation of messages at successive levels of nesting.

3.1 A Simplified Solution with Signed Messages

The problem of Byzantine generals is much easier to solve when we can say with certainty that a given voice is coming from a specific server. As the analysed article shows, the problem under consideration becomes much easier to solve when the following assumptions are added: the signature of a loyal general cannot be forged, and every other general can verify the signature's correctness. As the authors of this algorithm claim and then prove, a solution can be used that works independently of the number of Byzantine generals [4].

The algorithm assumes that each general receives a signed order from the commander and then sends it to the other generals. As emphasised, generals, in contrast to the oral communication solution, can determine with certainty whether a commander is a traitor by comparing the received messages [4].

For the analysed problem of verifying the correctness of messages, a simplified algorithm with signed messages can be implemented based on the above idea. After receiving the voice from the server, each other server additionally marks the received vote and distributes it to other nodes. After receiving a certain number of votes or after the voting time has elapsed, each server checks the received votes. However, to determine what vote a particular server cast, it does not make decisions based on the majority - if it finds even one vote that contradicts the others, it means that the server is a Byzantine server. It is because each vote is cryptographically signed and unquestionably correct. The server thus has evidence that another server has attempted to interfere illegally with the network by sending conflicting messages to different nodes.

Attempted forgery when sending messages about the received vote is even easier to detect. First, the transmitted voice must be cryptographically correct, so you cannot effectively convince the other servers that the given server made a different decision than it was. Each server must provide proof of the original vote they previously received. In addition, there is no possibility of impersonating another server because also, at the second level, the voice must be cryptographically signed [5].

So the algorithm looks like this:

1. Server A sends its signed voice to the other servers.
2. For each server and different from A:
 - 2.1. The server i saves the received vote.
 - 2.2. Server i signs the received vote and distributes it to all other servers except A.
 - 2.3. Server i starts the process of collecting information about what vote other servers received from server A. He gets a vote from every server except A:
 - 2.3.1. The server i validates the voice. If it is incorrectly signed, it rejects it. If it is correct, it adds it to its list.
 - 2.4. After the set time or receiving votes from all servers except A, the server verifies the received votes. If at least one of them is different from the others, it means that server A is a Byzantine server and is trying to make an unauthorised interference with the network. The server i notes this fact in a separate list and does not consider server A's vote.

The above algorithm applies to accepting a vote only from server A. It is performed as many times as there are servers in the network - each time, one of the servers sends its vote on a given block. Based on the stored list of Byzantine servers, each server can determine which votes to take into account and which to ignore during the final counting of votes.

3.2 Selection of the Consensus Algorithm

The correct operation of the application depends on the mechanisms based on which users decide whether a new block can be attached to the blockchain. Such tools are called consensus algorithms. There are many different types of mechanisms.

In public blockchains, it is necessary to introduce restrictions on who and on what basis can create new blocks. It avoids problems such as double spending or filling up the chain with unnecessary blocks with artificial transactions. An appropriate algorithm for reaching a consensus positively impacts the stability of the entire application, as it allows you to establish an actual rate of new data creation. For example, the Bitcoin chain implements a mechanism that determines the difficulty of creating a new block in a given period (difficulty matching period), considering the time stamp in the blocks of a given interval. Thanks to such calibration, regardless of the users' involvement, it always takes an average of 10 min to extract one block [6].

In the case of verifying the authenticity of the message, we suggest using an algorithm for reaching a consensus based on the reputation of a given user/server in the network. So it implements the Proof of Authority algorithm (PoA). In the case of this type of solution, PoA blockchain networks are secured by nodes that are uncompromisingly selected as reliable units [7, 8]. Anti-fake news portals, such as Snops.com, FactCheck.org, PolitiFact, and ABC, will play the role of these nodes. The second group of trusted servers will be recognised by large, credible publishers such as CNN, BBC, Times, Washington Post, etc. [9, 10].

3.3 Network Effectiveness Testing

A messaging application and a Byzantine server implementation were built as part of the research. Such a server sends a random message to each server during voting. It will be possible to examine the behaviour of both the server trying to influence the network and the incorrect action due to a failure. The probability for both vote values is 50%. The result of running the program for four servers, one of which is a Byzantine server, is as follows:

```
Node A | my choice: true | ratio: 75%
Node B | my choice: true | ratio: 75%
Node C | my choice: true | ratio: 100%
Byzantine Node D | my choice: false/false/true | ratio: 100%
Votes for: 4
Votes against: 0
Real ratio: 100 %
Final consensus: true
```

Even though the block was accepted, nodes A and B received a vote from the Byzantine server, resulting in a lower-than-real vote rate (ratio) for adding the block.

In another attempt, the Byzantine server was successful in attacking network integrity. The actual result of the vote is 75% - this is the result that should be in each of the servers, if there were no wrong messages sent. The Byzantine server sent node A a vote for adding a block and nodes B and C against it. Therefore, nodes B and C made a decision contrary to the actual state and rejected the block that, according to the objective state of the network, they should join. In addition, due to the fact that as many as 50% of the servers made a different decision, there was an unintended branching of the blockchain.

```
Node 0 | my choice: false | ratio: 75%
Node 1 | my choice: true | ratio: 50%
Node 2 | my choice: true | ratio: 50%
Byzantine Node 4 | my choice: true/false/false | ratio: 50%
Votes for: 3
Votes against: 1
Real ratio: 75 %
Final consensus: false
```

The test shows that less compatibility of properly functioning nodes increases the probability of error and the performance of Byzantine servers. The research conducted 1000 tests on a network of 100 servers. We conducted a second step or experiment to test the network's performance against the number of Byzantine servers.

The tests were repeated 100 times for each case with a different number of malfunctioning nodes. Since the total number of servers is 100, the number of Byzantine servers also equals their percentage. The obtained results are presented in Table 1 and graph 1. The results show that the network achieved 100% efficiency even when as much as 42% of the nodes were Byzantine servers. In further testing, there were isolated cases where the servers made a decision different from the general conclusion, but the efficiency remained around 99% until Byzantine servers made up 58% of the network; above this

value, the efficiency of the network began to drop drastically. With 73% of the nodes defective, more than half of the tests were negative, and with 82%, almost all.

Therefore, a large number of nodes positively affect the correct operation of the network. It should be noted, however, that in the simulations, the Byzantine servers made decisions at random, which was more similar to the behaviour of servers sending incorrect messages as a result of a failure than as a result of intentional interference. One would expect that Byzantine servers could perform targeted attacks, i.e. send multiple messages to specific servers to achieve certain benefits. These types of actions are more difficult to simulate. On the other hand, more such groups of servers could try to exert an unfair influence on the network, and they would pursue different goals, so these tests seem reliable. Regardless of whether we consider Byzantine servers to be random or directed, our tests clearly show that such servers have a very large impact on the network and can easily lead to inconsistencies between individual nodes. The implementation of algorithms that counteract this type of attack is therefore necessary for the proper operation of the network.

Table 1. Correctness of network operation depending on the percentage of Byzantine servers

| Percentage of Byzantine servers | Network correctness (%) |
|---------------------------------|-------------------------|
| 20 | 100 |
| 30 | 100 |
| 40 | 100 |
| 42 | 100 |
| 43 | 99,99 |
| 45 | 99,99 |
| 50 | 99,97 |
| 55 | 99,81 |
| 60 | 98,39 |
| 65 | 91,45 |
| 70 | 69,75 |
| 75 | 31,16 |
| 80 | 4,10 |
| 85 | 0,03 |
| 86 | 0,01 |
| 87 | 0 |
| 90 | 0 |

Therefore, a large number of nodes positively affect the correct operation of the network. It should be noted, however, that in the simulations, the Byzantine servers made decisions at random, which was more similar to the behaviour of servers sending

incorrect messages as a result of a failure than as a result of intentional interference. One would expect that Byzantine servers could perform targeted attacks, i.e. send multiple messages to specific servers to achieve certain benefits. These types of actions are more difficult to simulate. On the other hand, more such groups of servers could try to exert an unfair influence on the network, and they would pursue different goals, so these tests seem reliable. Regardless of whether we consider Byzantine servers to be random or directed, our tests clearly show that such servers have a very large impact on the network and can easily lead to inconsistencies between individual nodes. The implementation of algorithms that counteract this type of attack is therefore necessary for the proper operation of the network (Fig. 1).

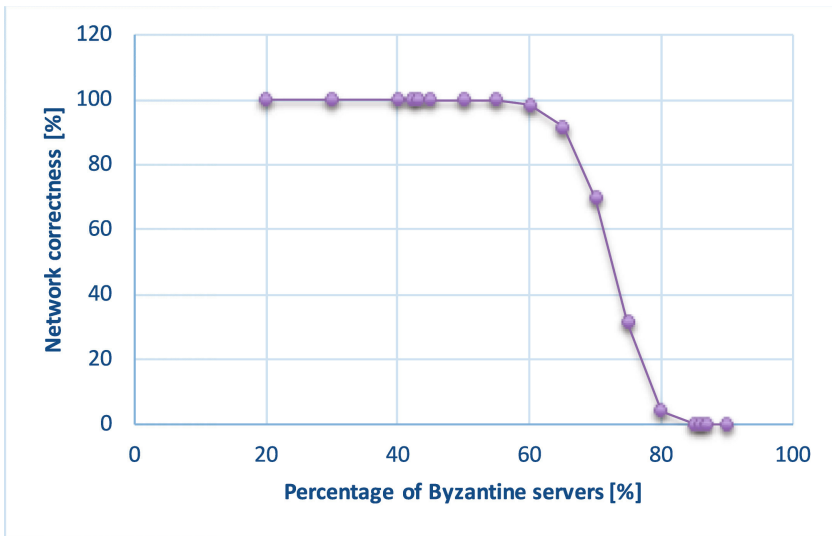


Fig.1. Correctness of network

4 Conclusion

Blockchain technology will find future applications in areas not yet associated with it. This technology can significantly reduce the real scourge of the Internet, fake news. Blockchain technology will allow you to track the way information is created and all attempts to modify it at every stage, thus preventing attempts to create false information. It has not been possible until now. Only blockchain technology allows you to do this by supporting the mechanism of interconnected transaction chains, which are 100% protected against outside interference. This article contains a proposal for the practical application of blockchain to protect against fake news. The study confirms the effectiveness of the proposed method. Its simple structure will undoubtedly be a vote for its use in practice.

References

1. Rutkowski, B.: Blockchain - technological aspects and examples of applications. <https://www.lazarski.pl/pl/nauka-i-badania/instytuty/wydzial-ekonomii-i-zarzadzania/centrum-technologiei-blockchain/blockchain-aspekty-technologiczne-oraz-przyklady-zastosowan/>. Accessed 28 Feb 2023
2. Shahid, I.U., Anjum, M.T., Shohan, M.S., Tasnim, R., Al-Amin, M.: Authentic facts: a blockchain based solution for reducing fake news in social media. In: 4th International Conference on Blockchain Technology and Applications (2021)
3. Chen, C.C., Du, Y., Peter, R., Golab, W.M.: An implementation of fake news prevention by blockchain and entropy-based incentive mechanism. *Soc. Netw. Anal. Min.* **12** (2022)
4. Lamport, L., Shostak, R., Pease, M.: The Byzantine generals problem. *ACM Trans. Program. Lang. Syst.* **4**(3), 387–388 (1982)
5. Bashir, I.: *Advanced Blockchain Applications*, Helion SA (2019)
6. Song, J.: *Understand Bitcoin. Cryptocurrency Programming from Scratch*, Helion SA (2019)
7. Kozik, R., Choraś, M., Kula, S., Pawlicki, M.: Distributed architecture for fake news detection. In: Herrero, Á., Cambra, C., Urda, D., Sedano, J., Quintián, H., Corchado, E. (eds.) *CISIS 2019. AISC*, vol. 1267, pp. 208–217. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-57805-3_20
8. Zhou, X., Zafarani, R.: Fake news: a survey of research, detection methods, and opportunities. *arXiv preprint arXiv:1812.00315*, 2 (2018)
9. Bobulski, J.: Hidden Markov models for two-dimensional data. *Adv. Intell. Syst. Comput. Intell. Syst. Comput.* **226**, 141–149 (2013)
10. Kubanek M., Bobulski J.: The use of hidden Markov models to verify the identity based on facial asymmetry. *Eurasip J. Image Video Process.* **2017**(1), Article No. 45 (2017)



Formalization and Verification of the Zab Protocol Using CSP

Wenting Dong¹ (✉), Jiaqi Yin², Sini Chen¹, and Huibiao Zhu¹

¹ Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, China

51255902036@stu.ecnu.edu.cn

² Northwestern Polytechnical University, Xi'an, China

Abstract. ZooKeeper Atomic Broadcast (Zab) is a high-performance atomic broadcast protocol, which is a key component of Apache ZooKeeper. By ensuring strong consistency and fault tolerance, the Zab protocol plays a crucial role in building robust and resilient distributed systems. However, the correctness and reliability of the Zab protocol have received limited attention in research. Thus, we employ Communicating Sequential Processes (CSP) to analyze and evaluate of the Zab protocol's properties and behavior. We utilize Process Analysis Toolkit (PAT) to verify six important properties, including Deadlock Freedom, Divergence Freedom, Data Reachability, Consistency, Sequentiality and Atomicity. The verification results demonstrate that the Zab protocol provides assurance of correctness and reliability.

Keywords: Zab protocol · CSP · Modeling · Verification · PAT

1 Introduction

With the rapid advancement of distributed systems and blockchain technology, a large number of novel protocols and algorithms have been proposed, such as Paxos [14], Raft [16], Zab [11], etc. However, numerous widely-utilized protocols and algorithms in distributed systems have yet to undergo further analysis and verification to ensure the security and correctness of the system.

To cover the gap, formal methods provide a rigorous and systematic approach to distributed system development, analysis, and verification [2, 8]. Formal methods are powerful technology and widely applied in numerous domains, such as operating systems [13], blockchain [17], cyber-physical systems [3], artificial intelligence [12] and so on.

Most recently, there are also some works using formal methods to verify protocols and algorithms. Wilcox et al. [19] presented a framework Verdi and verify Raft [16]. Chand et al. [4] used TLAPS to describe formal specification and verification of Multi-Paxos algorithm. Yin et al. [20] employed the TLA+ to specify and verify the limited properties of Zab. Inspired by these works, we choose process algebra CSP [7] to further model and analyze the Zab protocol.

In addition, the research on the correctness and reliability of Apache ZooKeeper and the Zab protocol has received limited attention. Apache ZooKeeper [9, 10] is a centralized and highly reliable distributed coordination service that is widely used in distributed systems. The fundamental protocol employed by ZooKeeper is the ZAB (Zookeeper Atomic Broadcast) protocol [11], which serves as a foundational building block for developing robust, scalable, and reliable distributed applications, and plays a crucial role in the frontier research and innovation in the field of distributed systems. Thus, it is crucial to conduct formal verification of the Zab protocol.

In this paper, we aim to provide a solid foundation for the analysis and verification of the Zab protocol by using process algebra CSP (Communicating Sequential Processes) [7]. We utilize model checker PAT (Process Analysis Toolkit) [18] to verify a broader range of properties, including Deadlock Freedom, Divergence Freedom, Data Reachability, Consistency, Sequentiality and Atomicity. The verification results show that the correctness and reliability of Zab protocol are guaranteed.

The remainder of this paper is organized as follows. Section 2 provides a brief introduction of the Zab protocol and the process algebra CSP. In Sect. 3, we illustrate the detailed modeling of the Zab protocol with three phases. In Sect. 4, we adopt PAT to implement the constructed model and verify six properties. Finally, Sect. 5 provides a summary of the contributions and potential future work.

2 Background

In this section, we give a concise explanation of the Zab protocol. At the same time, we give a brief introduction to the process algebra CSP.

2.1 Workflows of the Zab Protocol

Before introducing the Zab protocol, it is essential to clarify the entities involved in the protocol.

- **Client:** Clients interact with the distributed system and send update requests to the leader.
- **Leader:** Servers in the leading state are responsible for coordinating the replication of data updates across the followers. In case of failures or leader re-election, a new leader is elected among the followers.
- **Follower:** Servers in the following state replicate data updates received from the leader. They maintain a copy of the leader’s log and execute the updates in the same order as the leader.
- **Looker:** Servers in the looking state actively participate in the leader election process by requesting votes from other servers, which occurs when there is no leader present in the system.

The Zab protocol operates in three main phases: Discovery phase, Synchronization phase and Broadcast phase. Next, we delineate them respectively.

Discovery Phase. In this phase, the servers in the ZooKeeper system discover each other and determine their roles. Initially, all servers start in the looking state, indicating that there is no leader. In addition, servers communicate and exchange information to elect a leader. They send election requests and respond with election requests from other servers to establish a new primary. Once a server receives votes from the majority, it switches to the leading state, indicating that it is recognized as a leader. The workflows of this phase are illustrated in Fig. 1.

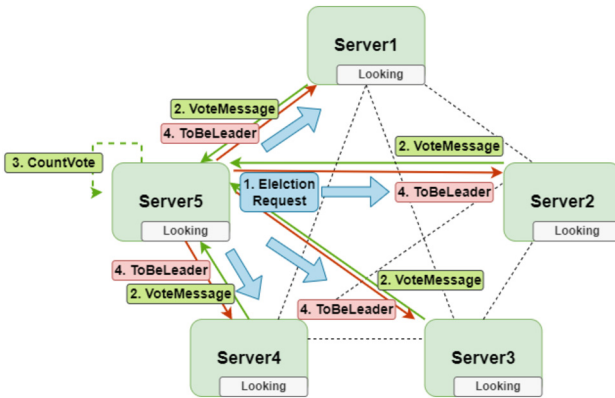


Fig. 1. The Workflows of the Discovery Phase

Synchronization Phase. After the leader is elected, the synchronization phase begins. In this phase, the followers synchronize data with the leader. It sends synchronization requests to the followers, which reply with their last known committed proposal. The leader then compares its own transaction log with the followers’ logs and sends the missing proposals to bring them up to date. This ensures that all followers have an identical copy of the leader’s log and brings them into a consistent state. The workflows of this phase are shown in Fig. 2.

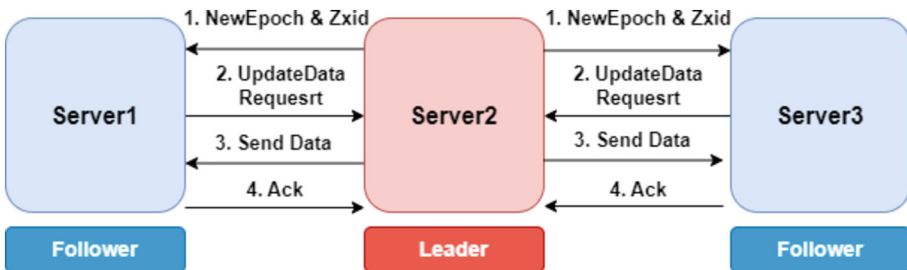


Fig. 2. The Workflows of the Synchronization Phase

Broadcast Phase. Once synchronization is complete, the servers enter into the broadcast phase. During this phase, the leader receives client’s requests and proposes new

proposals. It orders the proposals, assigns unique identifiers, and broadcasts them to all followers. The followers replicate the proposals. Then, when the leader receives the ACK message from more than half of the followers for the transaction proposal, it will send a commit message to all the followers. This ensures durability and guarantees the consistency of transactions across the entire cluster. Figure 3 shows the workflows of this phase.

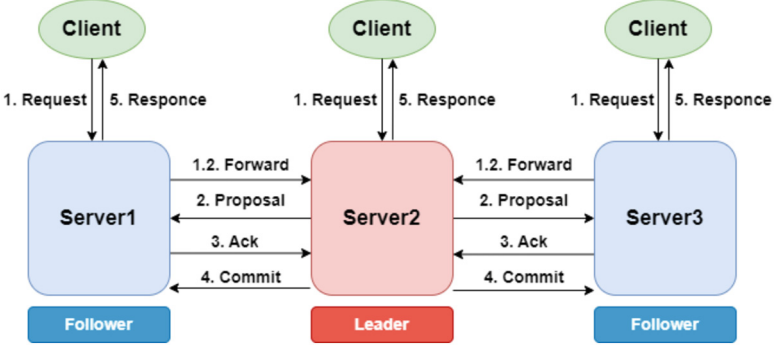


Fig. 3. The Workflows of the Broadcast Phase

2.2 CSP

CSP (Communicating Sequential Processes) [7] provides a rigorous mathematical theory and language. Additionally, CSP has been successfully applied to model and verify various concurrent systems and communication protocols [1, 5, 15]. Part of CSP syntax used in our model are described as follows.

$$P, Q = \text{Skip} | \text{Stop} | a \rightarrow P | c!x \rightarrow P | c?v \rightarrow P | P \square Q \\ P || Q | P ||| Q | P \triangleleft b \triangleright Q | P; Q | P[X] | Q$$

- *Skip*: The process does nothing, but terminates immediately.
- *Stop*: The process reaches deadlock.
- $a \rightarrow P$: After the execution of event a , process P is executed.
- $c!x \rightarrow P$: The process receives a message through channel c and assigns it to variable x , then behaves like P subsequently.
- $c?v \rightarrow P$: The process sends a value v through channel c , and then starts executing process P .
- $P \square Q$: It stands for general choice between the processes P and Q .
- $P || Q$: Processes P and Q run in parallel.
- $P ||| Q$: Processes P and Q run concurrently without barrier synchronization.
- $P \triangleleft b \triangleright Q$: If the Boolean expression b equals true, then process P is executed, otherwise process Q is executed.
- $P; Q$: Processes P and Q execute in sequence.
- $P[X] | Q$: The parallel composition of P and Q performs the concurrent events on the set X of channels.

3 Modeling

In this section, we present the formal modeling of the Zab protocol with CSP. Firstly, we give the whole structure of our model and then we model each phase of the Zab protocol respectively.

3.1 Sets, Messages and Channels

To facilitate the procedure of modeling and clarify the whole system, we give the definitions of sets, messages and channels used in this model.

Based on the mechanisms and processes involved in the Zab protocol, some sets have been defined. The set **Module** denotes all modules within the Zab protocol, which comprises clients, servers, and proposals. The set **ID** represents the unique identifier for each of the above module. The set **Status** is composed of all status that a server may be in. The set **Data** includes the data transmitted between modules. The set **Req** defines request messages and the set **Ack** contains feedback messages. Furthermore, we show a detailed definition of the relationship between relevant sets and constants in Table 1.

Table 1. The Relationship Between Involved Constants and Predefined Sets

| Set | Constants |
|---------------|--|
| Module | C (client), S (server), P (proposal) |
| ID | CID (client id), SID (server id), ZXID (proposal id) |
| Status | Leading, Following, Looking, Crashing |
| Data | Data |
| Req | ReadData (request for data), ProposalMsg (request to store proposal), Election (request for election), LeaderMsg (request to be the leader), ReqProposal (request for proposal), CommitProposal (request to commit proposal) |
| Ack | True/1 (positive feedback), False/0 (negative feedback), VoteMsg (voting result) |

Based on the above sets, we define messages transmitted among components. Depending on the type of messages, we abstract and classify the messages, and the definitions are as follows:

$$MSG_{req} = \{msg_{req}.A.B.content \mid A \in Module, B \in Module, content \in Req\}$$

$$MSG_{rep} = \{msg_{rep}.A.B.content \mid A \in Module, B \in Module, content \in Ack\}$$

$$MSG_{data} = \{msg_{data}.A.B.content \mid A \in Module, B \in Module, content \in \{Data, ID\}\}$$

where, MSG_{req} is composed of the request messages transmitted from module A to module B , MSG_{rep} denotes the response messages and MSG_{data} represents the data messages. A and B are the sender and the receiver respectively, and $content$ represents content contained in each message.

Then, we define that MSG consists of the above three types of messages.

$$MSG = MSG_{req} \cup MSG_{rep} \cup MSG_{data}$$

Furthermore, we define the channels for communication among various modules. These channels are denoted as COM_PATH , shown as below:

- $ComCS$: The channels are between the client and server. In a ZooKeeper system, there are multiple client and server processes interacting with each other. So, the corresponding channels will also be generated, and we use subscript i to distinguish each channel expressed as $ComCS_i$.
- $ComSS$: The channels are between servers. Similarly, there are multiple channels. We use subscript j to distinguish each channel, denoted as $ComSS_j$.

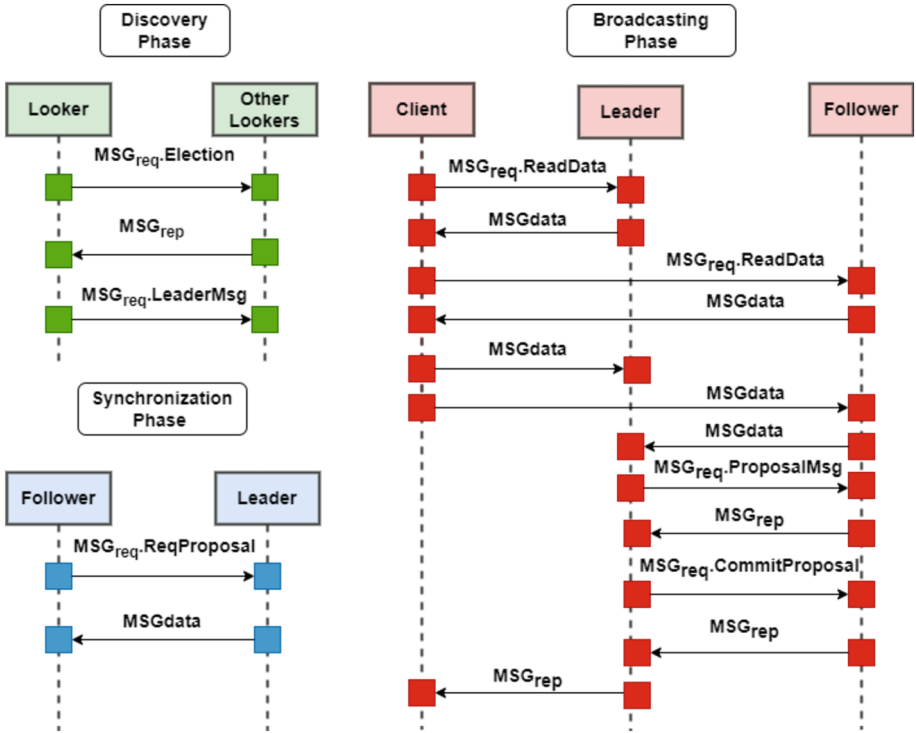


Fig. 4. The Communication Flow of the Zab Protocol

3.2 Overall Modeling

As described in Sect. 2, the behavior and status of servers differ depending on the phase of the Zab protocol. Therefore, the definition of $Server$ is as follows:

$$Servers_{sid}() =_{df} \text{Looker}_{sid}() \triangleleft \text{Status}[sid] == \text{looking} \triangleright \\ \left(\begin{array}{l} \text{Leader}_{sid}() \triangleleft \text{Status}[sid] == \text{leading} \triangleright \\ (\text{Follower}_{sid}() \triangleleft \text{Status}[sid] == \text{following} \triangleright \text{Faild}_{sid}()) \end{array} \right)$$

where, the array $\text{Status}[sid]$ records the current status of server with ID sid . $\text{Looker}_{sid}()$, $\text{Leader}_{sid}()$, $\text{Follower}_{sid}()$ and $\text{Faild}_{sid}()$ are processes.

The ZooKeeper system can be abstracted as a system consisting of clients and servers. The communication flow of the Zab protocol is shown in Fig. 4. We formalize the whole model $\text{System}()$ as below:

$$\text{System} =_{df} \parallel_{cid \in C; sid \in S; zsid \in P} (\text{Client}_{cid} [|COM_PATH|] \text{Servers}_{sid})$$

3.3 Discovery Phase

During this phase, the servers do not process requests from clients. Instead, they communicate with each other using the ComSS channels to conduct a leader election. This phase contains three core processes, $\text{Discovery}_{sid,lsid}()$, $\text{SendElection}_{sid,lsid,T}()$ and $\text{SendLeader}_{sid,lsid}()$.

Firstly, the process $\text{Discovery}_{sid,lsid}()$ represents what the server needs to do during the discovery phase. If the $\text{leaderCount} == 0$, the server needs to initiate election requests, handle election requests from other servers, and process leader election requests. Otherwise, the server needs to synchronize data with the existing leader. The model is shown as follows:

$$\text{Discovery}_{sid,lsid}() =_{df} \\ \left(\begin{array}{l} \text{Vote}(sid,lsid) \rightarrow \text{SendElection}_{sid,lsid,T} \\ \square \left(\begin{array}{l} \text{ComSS}_j? \text{msg}_{req}.lsid.sid.Election\{\text{Vote}(lsid,sid)\} \rightarrow \\ \text{ComSS}_j! \text{msg}_{rep}.sid.lsid.VoteMsg \rightarrow \text{Discovery}_{sid,lsid}() \end{array} \right) \\ \square \text{ComSS}_j? \text{msg}_{req}.lsid.sid.LeaderMsg \rightarrow \text{Synchronization}_{lsid,sid}() \\ \square \text{fail}.sid\{\text{Status}[sid] = \text{crashing}\} \rightarrow \text{Faild}_{sid}() \end{array} \right) \\ \triangleleft \text{leaderCount} == 0 \triangleright \\ \left(\begin{array}{l} \text{Synchronization}_{\text{leaderID},sid}() \\ \text{fail}.sid\{\text{Status}[sid] = \text{crashing}\} \rightarrow \text{Faild}_{sid}() \end{array} \right)$$

In the above formula, $\text{Vote}(sid,lsid)$ is utilized to cast a vote from server sid to server $lsid$. leaderCount Records the number of leaders in the current system.

Secondly, the process $\text{SendElection}_{sid,lsid,T}()$ means that server sid initiate an election request. It sets a time parameter T , so the process will wait for the reply message from the server $lsid$ until the time exceeds T . Within the time limit, if the server receives votes from more than half of the servers, it will be elected as the prospective leader. The model is depicted as follows: