



# Efficient Fault Identification Protocol for Dynamic Topology Networks Using Network Coding

Hazim Jarrah<sup>1</sup>, Peter H. J. Chong<sup>1</sup>(✉), Nurul I. Sarkar<sup>2</sup>,  
and Jairo Gutierrez<sup>2</sup>

<sup>1</sup> The Department of Electrical and Electronic Engineering,  
Auckland University of Technology, Auckland, New Zealand  
{Hjarrah, peter.chong}@aut.ac.nz

<sup>2</sup> The Department of Information Technology and Software Engineering,  
Auckland University of Technology, Auckland, New Zealand  
{nurul.sarkar, jairo.gutierrez}@aut.ac.nz

**Abstract.** This paper considers the problem of fault identification in dynamic topology networks using the time-free comparison model. Here, we introduce an efficient self-diagnosis protocol that can identify faulty nodes in dynamic networks. This protocol can correctly diagnose various fault types including permanent, dynamic, and soft faults. The protocol consists of a testing stage and a disseminating stage. During the testing stage, each node identifies the state of a part of nodes using the time-free comparison model. Afterward, nodes share their views employing a random linear network coding (RLNC) technique in the disseminating stage. The design of the disseminating stage is crucial for diagnosis efficiency. Using RLNC obviates the need for disseminating the views individually, and hence it reduces the number of messages required to diagnose the network. The OMNeT++ simulation has been used to evaluate the performance of the proposed protocol regarding the communication complexity. Results show that the proposed protocol is robust, scalable and energy-efficient.

**Keywords:** Self-diagnosis · Dynamic networks · Dynamic fault  
RLNC

## 1 Introduction

Faults are the origins of impairments of network dependability [1]. That is, they hinder reliance on services provided by the network. However, they are inevitable and may evolve into errors and failures [2, 3]. In dynamic networks that have been deployed and operated in critical situations, faults hit more often as a result of unpredictable circumstances, and that may cause risks for people, environment and finance [4]. The development of dependable networks requires dealing with faults. Traditionally, system-level fault diagnosis problem [5] considers automated fault identification in wire and wireless networks [6–9]. Recently, we have developed a time-free fault diagnostic model that respects the design requirements of dynamic networks [10]. In particular, it takes into consideration the asynchronous communications and the topology changes in this kind of networks.

The time-free comparison model is so named because it removes time constraints imposed by earlier models. Therefore, it suits asynchronous systems such as dynamic networks. Its design obviates the need for timers that are extremely hard to set into dynamic networks. Moreover, it is robust to dynamic topology changes. It is based on comparison approach where nodes examine each other states and collaborate to identify faulty and fault-free nodes in the network. In comparison approach, Nodes execute a task appointed, and their outputs are compared to identify whether any node is faulty. The main idea behind the comparison approach is that fault-free nodes processing the same inputs (task) will produce the same outputs.

This paper introduces a distributed self-diagnosis protocol to solve the diagnosis problem in dynamic networks. That is, this protocol enables fault-free nodes to determine the status of each node in the network correctly. Our proposed protocol consists of two main stages namely comparison stage and disseminating stage. The comparison stage employs the time-free comparison protocol to identify faulty nodes. This stage is executed by each node to test its neighbour nodes and generates a partial view of them. The disseminating stage exploits random linear network coding (RLNC) technique to exchange the partial views among nodes and generate a global view about the network. Using RLNC aims to reduce the number of diagnosis messages and hence enhance the protocol efficiency.

Network coding (NC) is an emerging communication paradigm introduced in 2000 [11]. This revolutionary paradigm remodels how nodes communicate with each other to improve the performance of networks. Nodes traditionally store and forward each packet received whereas they combine packets and forward the combination at once in NC [12]. To date, NC has been employed in numerous applications for wire and wireless networks [13]. The studies demonstrated that NC benefits network operation and design regarding throughput, scalability, robustness, energy, and reliability [14, 15]. Different coding techniques have been proposed. Nonetheless, random linear network coding (RLNC) considers more realistic assumptions for mobile networks [16]. In RLNC, Three operations are crucial; namely encoding, recoding, and decoding. Where source nodes encode native packets, intermediate nodes recodes coded packets, and destination nodes decode the packets. Firstly, source nodes generate an encoded packet that embodies linear combinations of native packets over a finite field using random coding coefficients. Intermediate nodes, then, recode these packets creating a linear combination of encoded packets. In this sense, these nodes can encode packets even though they have not been decoded yet. Finally, destination nodes decode coded packets and regenerate native packets once they get a sufficient number of linearly independent encoded packets.

The paper presentation includes sections as follows. Section 2 describes the system, the fault, and the diagnostic model. Section 3 presents a time-free fault diagnosis protocol using RLNC. Section 4 shows the simulation results obtained along with analysis. Finally, Sect. 5 concludes the paper.

## 2 Preliminaries

### 2.1 System and Fault Model

This subsection describes dynamic system considered in this research. The system consists of  $n$  mobile nodes that communicate wirelessly via packet radio network. It is an asynchronous system that alleviates time restrictions on node speed, transmission delay, and computation time. Nodes have no access to global clock. The dynamic network is represented by a communication graph that has dynamic topology and hence the connections may change over the time. At specific time  $t$ , a graph  $G_t = (V_t, E_t)$  characterizes the network;  $V$  is the set of nodes and  $E$  is the set of edges;  $E \subseteq V \times V$ . A graph  $G' = (V', E')$  indicates the fault-free nodes in the graph  $G$  at time  $t$ . The graph  $G'$  is assumed to comply with Assumption 1 which is an essential rule to preserve the properties of fault self-diagnosis protocols, i.e., correctness and completeness.

**Assumption 1. Connectivity over Time:** Let  $G' \subseteq G$  be a subgraph that consist of fault-free nodes in  $G$  at time  $t$ . Then, there must be at least one path between every two nodes  $u, v \in G'$ . That is,  $\forall u, v \in V', u \rightarrow v$ .

We assume nodes are mobile. Thus, the neighbour nodes may change. Also, we assume that nodes adopt the passive mobility model where nodes are unaware of moving. Hence, they cannot inform neighbours about that. As a consequence, the neighbour nodes cannot distinguish between migrated out or undergoing a fault node.

Faults have been investigated from different perspectives. In particular, a fault could be soft or hard considering its impact on node communications. That is, hard fault (e.g., fail-stop, fail silence, and crash) prevents node communications while soft fault may impact node operations but not communications. Another perspective of fault is based on its duration. Permanent faults endure until be fixed by external interventions such as battery depleted or node crash. On the other hand, temporary faults, i.e. intermittent and transient faults, cease to exist spontaneously. Fault occurrence time has also been used to distinguish between static and dynamic faults. Static faults exist before the start of a diagnosis session whereas dynamic faults emerge during the diagnosis session.

This research assumes that links experience no faults. Mainly, no message creation, alteration or loss may happen through these links. Nodes, however, may undergo any type of faults except temporary faults.

System diagnosability is an upper limit to the number of faults that a system can undoubtedly diagnose. That is, if the number of faulty nodes exceeds that limit, then the system will be disconnected, and it is unsure to diagnose faults correctly and completely. Here, we utilise a local fault model that bounds the upper limit of faults locally. Assume  $\sigma_v$  is the upper limit of faulty nodes in  $v$ 's neighbourhood. The  $\sigma_v$  is limited by the degree of the node  $v$ ,  $\deg(v)$ , i.e.,  $\deg(v) > \sigma_v$ . In case of reliable broadcast the bound, should be,  $\deg(v) > 2\sigma_v$ . This is because, it is uncertain to achieve reliable broadcast if half or more of nodes are faulty [17, 18].

**Definition 1. Local Diagnosability:** A dynamic network is locally  $\sigma$ -diagnosable at node  $v$  if each fault-free node can unambiguously identify the fault status of all nodes given that the number of neighbour faulty nodes does not exceed  $\sigma_v$ .

Fault-free nodes should be able to reply to  $\sigma + 1$  test requests within the first  $\alpha$  replies. This assumption implies that every fault-free node will be correctly diagnosed by at least one fault-free node. That is, fault-free nodes are winning nodes and achieve Assumption 2.

**Assumption 2. Winning Nodes:** Every fault-free node,  $u$  has a set of best neighbours that can communicate with  $u$  faster than with the other nodes.

## 2.2 Time-Free Diagnostic Model

The time-free diagnostic model has been developed for dynamic networks [10]. It adopts the comparison approach to identify faulty nodes. In the comparison approach [19], a task is appointed to nodes, and their outputs are compared to detect the state of nodes examined. This model assumes tasks are complete, i.e. they can detect faults in nodes. Also, it assumes that fault-free nodes executing the same task continually generate exact outputs while faulty nodes generate different outputs. Fault-free nodes can compare the outputs and generate comparison outcomes. This model utilises the asymmetric invalidation model rules [20] shown in Table 1.

**Table 1.** Comparison outcomes for the gMM model [20]

Nodes under comparison	Node performing the comparison	
	Fault-free	Faulty
None is faulty	0	0 or 1
One is faulty	1	0 or 1
Both are faulty	1	0 or 1

The following describes the steps to perform the comparisons in the time-free diagnostic model:

- **Test Request Generation:** A node  $u$  generates a test task  $T_i$ , where  $i$  is an integer number depicting the test number. Next, it broadcasts the test request message  $m = (TEST, T_i)$ , where  $TEST$  indicates the message type. Afterwards,  $u$  waits for responses from  $\alpha u$  nodes. It is noteworthy that  $u$  operates no timers.
- **Test Request Reception:** Once a node  $v$  receives the test request message  $m$  from  $u$ , it produces the result  $R_u^v$  of the task  $T_i$ . Then it broadcasts the test response message  $= (RESPONSE, T_i, R_u^v)$ ,  $RESPONSE$  is the message type. After that, the node  $v$  starts its diagnosis session through generating its test request message. As we consider a dynamic topology,  $v$  could be just moved into  $u$ 's transmission range, i.e.,  $v \notin N_u$  at the test generation time.
- **Test Response Reception:** Consider a node  $w \in V$ . Upon receiving responses from  $\alpha w$  nodes,  $w$  stops waiting. Then,  $w$  takes either of the two following actions:
  - **Case 1:**  $w$  knows the expected result of the task  $T_i$ ; it compares them. If  $R = R_u^w$ , then  $w$  can conclude that  $v$  is fault-free, and hence,  $v$  will be added with an associated timestamp to the list of fault-free nodes diagnosed by  $w$ , i.e.,

$FF_w = FF_w \cup \{v, ct\}$ . Otherwise,  $v$  is added to the list of faulty nodes with a timestamp  $ct$ , i.e.,  $F_w = F_w \cup \{v, ct\}$ .

- Case 2:  $w$  does not know the expected result of the task  $T_i$ . Hence,  $w$  executes the task  $T_i$  first and then compares the result with  $R_u^v$ . If the comparison outcome is 0 then, it will add  $v$ , with a timestamp  $ct$ , to the fault-free list,  $FF_w = FF_w \cup \{v, ct\}$ . Otherwise,  $v$  will be added to the faulty nodes list,  $F_w = F_w \cup \{v, ct\}$ .

It is clear that this model obviates the need for timers and alleviates time constraints. Moreover, it tolerates the topology changes. Seemingly, fault-free nodes may improperly be diagnosed as faulty just because they have not replied fast. However, the correct state of any node is held with the highest timestamp by at least one fault-free node. Besides, nodes collaborate to identify a complete and correct view of the network. Therefore, in the end, the objective of having a complete and correct diagnosis of a network is guaranteed.

### 3 Network Coding-Based Self-diagnosis Protocol

This section introduces a novel distributed self-diagnosis protocol for dynamic topology networks considering static and dynamic permanent faults. Nodes collaborate with each other to diagnose the whole network in a distributed fashion. Every fault-free node executing this protocol obtains a complete and correct view of the faulty status of all the nodes in the network. The diagnosis session includes two main stages namely testing and disseminating. In the testing stage, nodes diagnose their neighbours creating partial views about the network. Next, they exchange these views with each other in the disseminating stage creating a global view.

The diagnosis session could be started either at fixed intervals or when a node detects an unusual event. A node then initiates the testing step. The messages that are transmitted during the diagnosis session are called diagnosis messages. The end of the session is when all nodes stop running the protocol.

The primary design objective of our protocol is to reduce the number of diagnosis messages. In particular, the design of the disseminating stage affects the protocol efficiency regarding communication complexity. Reducing the number of diagnosis messages is crucial for real deployment. In particular, the lesser the diagnosis messages during the diagnosis session, the more scalable and energy-efficient protocol. Also, the protocol design should be robust to topology changes. Considering these design requirements, we propose an RLNC-based distributed self-diagnosis protocol (RLNC-DSDP) for dynamic networks.

In our proposed protocol, the testing stage is based on the time-free diagnostic model described briefly in Sect. 2. Besides, the disseminating stage employs RLNC technique to improve the protocol efficiency. This protocol is executed on every node in the network. In the following, we detail these stages at a node,  $u$ .

### 1. Testing Stage

The testing stage at node  $u$  commences by sending a message of type *TEST* to neighbour nodes. This message includes a test task,  $T_i$ . That is,  $u$  sends a message  $m = (TEST, T_i)$  to every one-hop neighbour nodes at that time  $v \in N_u$ . It is clear that neighbour nodes may change over the time as a result of network dynamics. Once a node  $v$  receives a diagnosis message for the first time, it performs the following procedures. First, it generates a message of type *TEST* including its test task if it have not started yet;  $m_v = (TEST, T_i)$ , and broadcast  $m_v$  to its neighbours;  $N_v$ . Second, it sends back a message of type *RESPONSE* including the task received ( $T_i$ ) and the results calculated ( $R_u^v$ );  $m = (RESPONSE, T_i, R_u^v)$ . The response messages may be received by non-testers. However, including the test task in the response message helps other nodes to diagnose its state. Once  $u$  receives  $\alpha$  messages of type *RESPONSE*, it generates a partial view based on the time-free comparison protocol in Sect. 2. That is, nodes will be diagnosed as fault-free if they produce the same results, soft-fault if they produce different results, and hard-fault if they send no reply. The node partial view contains two lists, namely fault-free list,  $FFu$  and Faulty list,  $Fu$ . These lists consists of members of form  $(ID,)$ ,  $ID$  and  $ct$  represent node identifier and current timestamp respectively.

The design of this protocol uses message exchange pattern and obviates the need for timers. The result is a practical protocol for dynamic networks that is robust to topology changes. By the end of this stage,  $u$  maintain a partial view about adjacent nodes. Mobile or slow nodes may erroneously consider as faulty because they move away from  $u$  or they did not reply within the first  $\alpha$  node. However, the system and diagnostic model assumptions guarantee that the correct state of any node is held by at least one fault-free node with the highest timestamp.

### 2. Disseminating Stage

At the end of the comparison phase, each node has a partial view of the network. The dissemination phase considers conveying these views to other nodes over the network to gain a complete view of all the nodes in the system. Our proposed protocol employs RLNC to perform this task. In the following, we describe how RLNC improves the efficiency during this stage.

Each node during this stage plays mainly two roles. First, it creates and transmits its partial view about the network. Second, it updates its view upon receiving other partial views and relays them to other nodes. In this sense, every node,  $u$  has an information message named *PartialView* that consist of the lists of faulty and fault-free nodes at  $u$ ;  $p_u = (PartialView, Fu, FFu)$ . Hence, there are a set of messages to be exchanged among all nodes in the network;  $\{p1, p2, p3, \dots, pn\}$ . Our proposed RLNC-based dissemination proceeds as the following. First, each node,  $u$  transmits its message  $p_u$ . Upon collecting  $\alpha$  dissemination messages,  $u$  generates a coded packet,  $e$ , combining linearly packets received. That is,  $e$  is the total of multiplying each packet with the corresponding value in the coding vector,  $c$  as in the following relation:

$$e = \sum_{i=1}^n c_i P_i \quad (1)$$

Where  $c = (c_1, c_2, \dots, c_n)$  is a coding vector that composes coefficients selected randomly from a finite field,  $F$ . The node  $u$  generates and sends a message of type *ENCODED* containing the information vector along with the coefficient vector;  $e_u = (ENCODED, DataV, CoeffV)$ . During this session,  $u$  will add to its decoding matrix any message of type *ENCODED* that increase the rank of the decoding matrix. Also, this message will be forwarded to other nodes. However, the message received will be discarded if it has no innovative packet. Later, full ranked decoding matrix will be solved and the native messages will be retrieved, by Gaussian elimination. Hence,  $u$  has the partial views of all nodes except nodes experiencing a hard fault as they cannot communicate with other nodes. Therefore, it can generate a complete view about the system considering the most recent information.

It is clear that RLNC implements NC in a distributed fashion and it requires no earlier awareness about packets received by other nodes. However, RLNC adds computational overhead for nodes and transmission overhead attaching coefficient vectors to messages. These additional overheads may hinder RLNC uses under some scenarios.

## 4 Simulation Results and Analysis

This section presents the simulation results obtained under different scenarios to evaluate the performance of our proposed protocol. We have used OMNeT++ simulator to conduct our simulations.

First, we evaluate the protocol efficiency and scalability with regard to network's size. That is, various networks with different sizes varies from 10 – 100 nodes. The nodes distributed randomly into the simulation area and 10% of nodes were considered faulty. The network topology is fixed in this scenario. The area size is 600 m × 600 m and the transmission range is 150 m for each node. Figure 1 shows the results under this scenario. The number of messages exchanged increases with the increasing number of nodes. However, it is clear that using RLNC significantly reduces the number of required diagnosis messages.

Second, we evaluated the protocol performance and robustness for increasing the number of faults. In this scenario, a network of 80 nodes had experienced 2 to 30 faults. The network had fixed topology and constant connectivity. Both static and dynamic faults have been considered in this scenario. Figure 2 depicts the results obtained under this scenario. It compares the communication complexity of our protocol experiencing static faults (RLNC-DSDP (Static)) as well as dynamic faults (RLNC-DSDP (dynamic)). It is clear that the increase in the number of faults leads to a decrease in the number of messages. The reason is that fewer number of nodes got involved in the diagnosis session. This figure also shows that our protocol is robust for various type of faults. That is, the diagnosis messages required for identifying dynamic and static faults are very close.

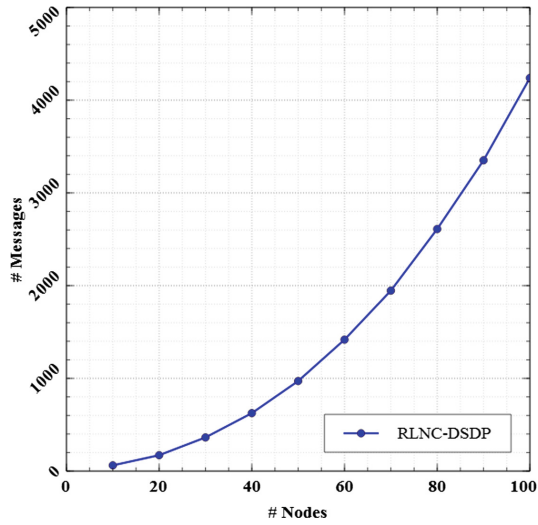


Fig. 1. Communication complexity under scenario 1

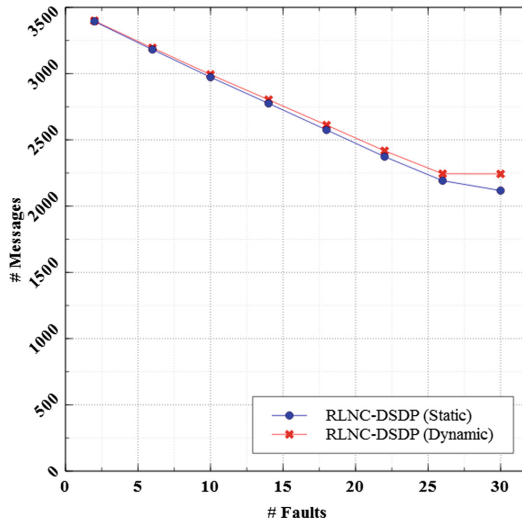
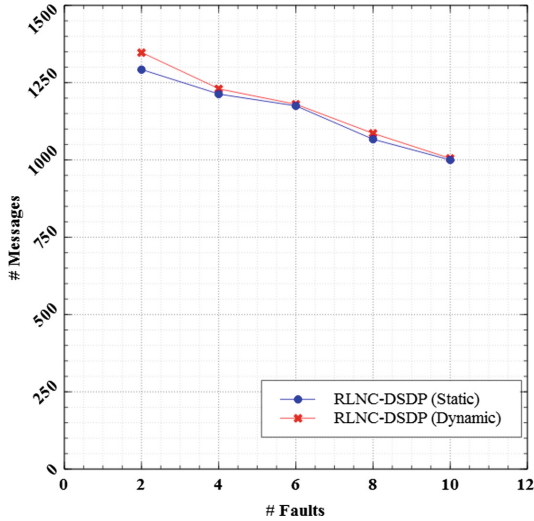


Fig. 2. Communication complexity under scenario 2

Finally, we evaluated the protocol performance under dynamic topology network. A network of 50 nodes has mobile nodes ranging from 2 – 10. The mobility of nodes caused random topology changes. Static and dynamic faults also considered under this scenario. Figure 3 shows the figures obtained under this scenario. This figure compares the performance under static and dynamic faults as well. The number of messages exchanged is reduced when more mobile nodes are there. It means that the mobility of





**Fig. 3.** Communication complexity under scenario 3

nodes aids in distributing the diagnosis messages. Also, it shows the robustness under various fault types.

The results show that the RLNC-DSDP protocol can efficiently identify the various type of faults in static and dynamic topology networks. That is, it provides a scalable, robust, and energy-efficient diagnosis algorithm. Further research is needed to investigate the computation overhead and delay caused by using RLNC. However, we believe that the advantages gained outperform the potential overhead.

## 5 Conclusion

In this paper, we proposed a new self-diagnosis protocol based on the time-free comparison model. Our protocol implements a simple RLNC technique to share the partial views. The partial views are grouped together instead of broadcasting them individually. Therefore, it obviates the need for broadcasting every local view received individually and hence the number of diagnosis messages required is significantly reduced. The future work concentrates on temporary faults identification.

## References

1. Basile, C., Killijian, M.-O., Powell, D.: A survey of dependability issues in mobile wireless networks (2003)
2. Santoro, N.: Design and Analysis of Distributed Algorithms, vol. 56. Wiley, Hoboken (2006)
3. Jarrah, H., Sarkar, N.I., Gutierrez, J.: Comparison-based system-level fault diagnosis protocols for mobile ad-hoc networks: a survey. *J. Netw. Comput. Appl.* **60**, 68–81 (2016)

4. Silva, I., Leandro, R., Macedo, D., Guedes, L.A.: A dependability evaluation tool for the Internet of Things. *Comput. Electr. Eng.* **39**, 2005–2018 (2013)
5. Preparata, F.P., Metze, G., Chien, R.T.: On the connection assignment problem of diagnosable systems. *IEEE Trans. Electron. Comput.* **EC-16**, 848–854 (1967)
6. Blough, D.M., Brown, H.W.: The broadcast comparison model for on-line fault diagnosis in multicomputer systems: theory and implementation. *IEEE Trans. Comput.* **48**, 470–493 (1999)
7. Chessa, S., Santi, P.: Comparison-based system-level fault diagnosis in ad hoc networks. In: 2001 Proceedings of 20th IEEE Symposium on Reliable distributed systems, pp. 257–266 (2001)
8. Elhadeif, M., Boukerche, A., Elkadiki, H.: Diagnosing mobile ad-hoc networks: two distributed comparison-based self-diagnosis protocols. In: Proceedings of the 4th ACM International Workshop on Mobility Management and Wireless Access, pp. 18–27 (2006)
9. Elhadeif, M., Boukerche, A., Elkadiki, H.: Performance analysis of a distributed comparison-based self-diagnosis protocol for wireless ad-hoc networks. In: Proceedings of the 9th ACM International Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems, pp. 165–172 (2006)
10. Jarrah, H., Chong, P., Sarkar, N.I., Gutierrez, J.: A time-free comparison-based system-level fault diagnostic model for highly dynamic networks. In: Proceedings of the 11th International Conference on Queuing Theory and Network Applications, p. 12 (2016)
11. Ahlswede, R., Cai, N., Li, S.-Y., Yeung, R.W.: Network information flow. *IEEE Trans. Inf. Theor.* **46**, 1204–1216 (2000)
12. Ho, T., Lun, D.: *Network Coding: An Introduction*. Cambridge University Press, Cambridge (2008)
13. Deb, S., Effros, M., Ho, T., Karger, D.R., Koetter, R., Lun, D.S., et al.: Network coding for wireless applications: a brief tutorial (2005)
14. Matsuda, T., Noguchi, T., Takine, T.: Survey of network coding and its applications. *IEICE Trans. Commun.* **94**, 698–717 (2011)
15. Bassoli, R., Marques, H., Rodriguez, J., Shum, K.W., Tafazolli, R.: Network coding theory: a survey. *IEEE Commun. Surv. Tutorials* **15**, 1950–1978 (2013)
16. Chou, P.A., Wu, Y., Jain, K.: Practical network coding. In: Proceedings of the Annual Allerton Conference on Communication Control and Computing, pp. 40–49 (2003)
17. Koo, C.-Y.: Broadcast in radio networks tolerating byzantine adversarial behavior. In: Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing, pp. 275–282 (2004)
18. Bhandari, V., Vaidya, N.H.: Reliable broadcast in radio networks with locally bounded failures. *IEEE Trans. Parallel Distrib. Syst.* **21**, 801–811 (2010)
19. Maeng, J., Malek, M.: A comparison connection assignment for self-diagnosis of multiprocessor systems (1981)
20. Sengupta, A., Dahbura, A.T.: On self-diagnosable multiprocessor systems: diagnosis by the comparison approach. *IEEE Trans. Comput.* **41**, 1386–1396 (1992)