Enhancing Fault-Tolerance in Ring Topology Based on Waiting Queue and Timestamp

Feras Matarneh [#1], Rami Matarneh[*2]

[#1] Department of computer science, University of Tabuk, Tabuk, Saudi Arabia
[*2] Department of Computer Science, Prince Sattam Bin Abdulaziz University, Saudi Arabia, Al-Kharj

Abstract *— in this paper we present an efficient and fair fault-tolerance in ring topology to recover from token loss situation. We have two requirement in fault tolerance in ring topology (i) safety: of course one node at most can use the shared link (ring) to send data at the same time (ii) livens : a node requests to use the ring will eventually succeed. The proposed solution requires feedback from every other node to recover from token loss. The general idea in this paper is to enhance fault tolerance in ring topology by using queue for each node and timestamp to handle multiple requests when one of nodes uses the ring while another nodes are waiting.*

Keywords — *Fault-Tolerance, Token Ring, Timestamp, Round Robin, Networks, Token-Loss.*

I. INTRODUCTION

There are several kinds of topologies that are used in networks system where topologies show how the nodes connected to each other in a physical manner. Ring topology is one of these topologies, this kind of topologies is relatively easy to install and reconfigure, but has several problems. The most critical problem is the token loss, when one of the connected nodes try to send data frames by the shared link (ring). In fact, the node will not be able to send its own data only if it has a permission to use the shared link, so the problem comes from the possibility of losing the token from last node that used it, where such failure inevitably will causes token loss.

Stations on a token ring LAN are logically organized in a ring topology with data being transmitted sequentially from one ring station to the next one with a control token, which is circulating around the ring [1], [2].

Physically, a token ring network is wired as star with 'hub' and arms out to each station and the loop gang out-and-back through each [3], [4].

Each station passes or repeats the special token frame around the ring to its nearest downstream neighbour, this token passing process is used to arbitrate access to the shared link. Stations that have data frames to transmit must first acquire the token before transmission process [5], [2].

In case when there is no station would like to transmit data frames, a special token frame circulates the loop using repeated token from station to station until arriving a station that needs to transmit data, which is at this moment converts the token frame into a data frame to begin transmission process. After receiving the data frames by the targeted station, the sender station releases the token by converting it back into a token frame to be used by another station. If any error occurs during the transmission due to no token fame or the presence of more than one, a special station referred to as the active monitor, detects the error and reinsert token as necessary [6], [7].

II. RELATED WORK

Mishra and Srimani [8] algorithm uses the concept of circulating a privilege message (token) among the sites. Nodes use timeout and probe messages to detect failure of the current token holder. Their method can deal with only one node failure at a time and have a very high message complexity. In Chang algorithm [9] an election algorithm is used to generate a new token when token is lost.

Singhal's method [10] for handling site and link failures maintains and uses state information about other sites for acquiring token. This has the drawback of possible loss of messages or token [11].

In Nishio algorithm [12] when a site detects token loss, it requires a positive acknowledgment from every other site to generate a new token. This results in very high waiting time in the presence of site or link failures. Moreover, due to false token loss detection, there is a possibility of deletion of the token from the system.

Sopena algorithm [13] uses an election algorithm to generate a new token when token is lost, which introduces extra overhead.

Manivannan token-based fault-tolerant ME algorithm can recover from loss of token by allowing the failed site after getting token to be repaired within a finite period of time [14], [15].

The Enhancing Fault-tolerance in a Distributed Mutual Exclusion Algorithm by Reddy el. [16] base on generating the lost token in a finite time and applies Round Robin.

III.DISCUSSION

In ring topology there are several problems such as: lack of scalability, low speed of data …etc., [1], [17], [18], in this work we will focus on the most important problem which is the loss of token [3], [19] when it released from the latest node in order to

reach another nodes that have a high priority. This we tend to achieve by the following assumptions:

- Assigning a queue to each node to handle multiple requests.
- Using timeout (timestamp) as indicator of priority.
- Using special token format consists of indexes, queue information and a counter.

Let's assume the default format of token to be as follows:

$$\{[a, b, c, d], C[Q\ inf]\}$$

Where

- $[a, b, c, d]$ indexes for all nodes where each index only can be assigned either 1 or 0 (0 means the nodes did not use the ring while 1 means the nodes was used the ring).
- C used as a counter for total usage of the ring by the node.
- $[Q\ inf]$ used for requests from other nodes.

The proposed model bases on the fact that each node in the token-ring topology that want to send data can be transformed into an active monitor and each one also has a queue in order to handle multiple requests from other nodes that want to transmit data frame, timeout (Timestamp) for detecting token loss and a copy of a token. The model also takes into consideration that: (1) any node or communication link may fail (2) token could be lost; because of link or node failure. Fig. 1 represents the basic idea.

IV. Example of Usage

If PC_3 wants to send its own data, at the beginning it sends a request message containing its current time T_3 to all other nodes, when PC_0 receives the request, it should check two conditions:

- If it has an idle token
- The priority of the request (known from the implied time stamp)

If both conditions are satisfied (as in this case the request of PC_0 has a highest priority) then token is sent toPC_3, otherwise, PC_0 queues up PC_3 request and sends it back ACK (acknowledge). After finishing its current uninterruptible job, PC_0 will return back to PC_3 by sending the queue information with the token data structure and clear its queue.

Fig. 2 shows the sequence of steps involved in the ring when there is no failure:

- In the beginning the token in PC_0 seems as follows: $(token:\{[1,0,0,0], 1[3,2]\})$.
- PC_2, PC_3 also want to send their own data by sending request message to all other nodes.
- After receiving PC_2 and PC_3 request, PC_0 queues up the request ordered by timestamp
- When PC_0 finishes, it will send the token: $(token:\{[1,0,0,0], 1[3,2]\})$ to PC_3 because it's timestamp is lower than PC_2 timestamp (high priority)
- After that, PC_3 will receive the token and push PC_2 into its own queue in order to send the token.

In case of node failure, if PC_2 does not receive the token within a timeout period, it sends $"is-token-lost!"$ message to all other nodes and as a response, all non-faulty nodes will send a reply message $"is-token-lost!"$ before expiration of PC_2 timer.

From those received replies within the time, PC_2 comes to know that PC_1 has the most recently token. This information can be found by maintaining a token (copy to all sites and keep the updated copy through token passing). Finally PC_3 will generate and send a token to PC_2, if PC_2 will not receive the token with in another timeout period (this could be as a result of failure of node or message loss), then it will send $"is-token-lost!"$ message to every other site to repeat the token generation process.

Fig. 3 shows the sequence of steps in presence of failure:

- PC_1 failed after it has finished with token: $(token:\{[1,1,0,1], 3[]\})$, while PC_2 is trying to send its own data after timestamp period.
  - PC_2 sends $"is-token-lost!"$ to all other nodes and in reply all non-faulty nodes send their own token-copy information.
- The last value of token before the failure has occurred was $(token:\{[1,0,0,1], 2[]\})$.
- So PC_2 will inform PC_3 to generate a new token and send it back.

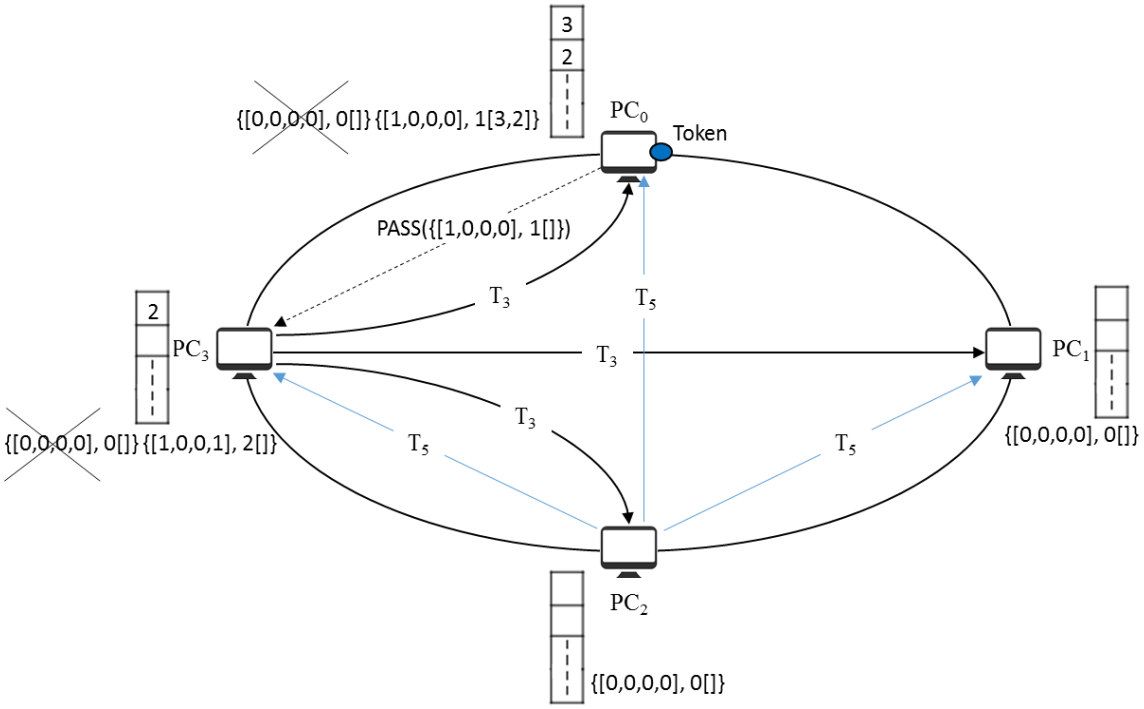

Fig 1: The proposed model of token-ring topology

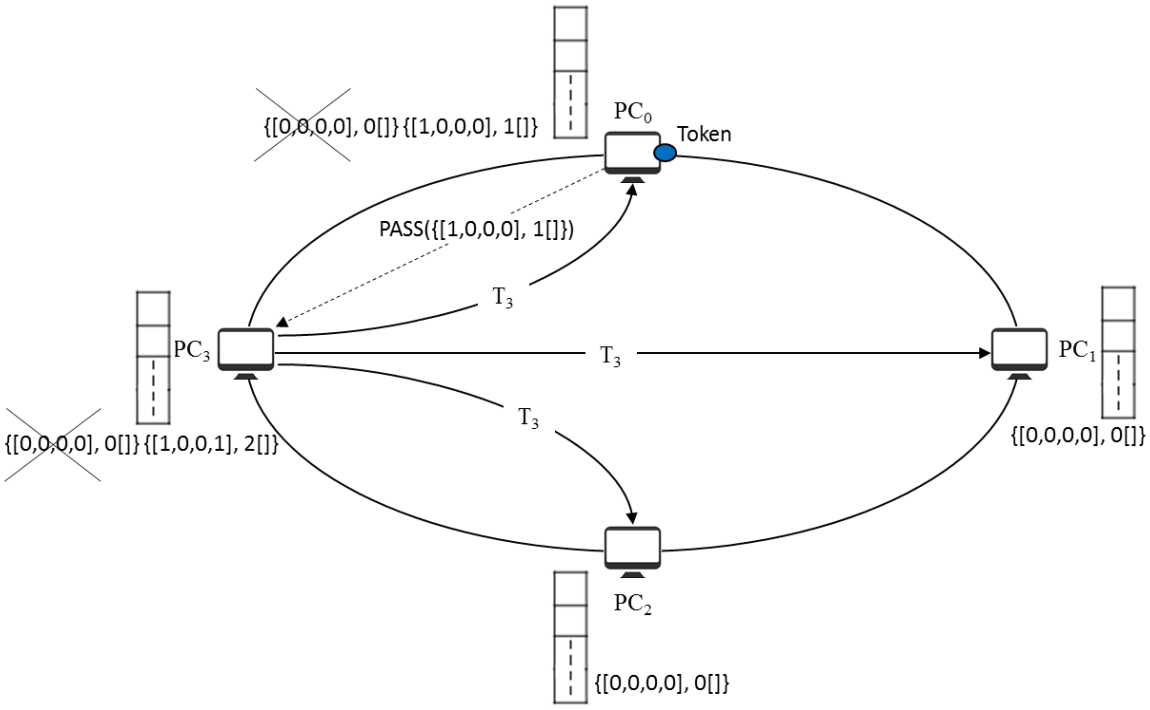

Fig 2: Sequence of steps with no failure

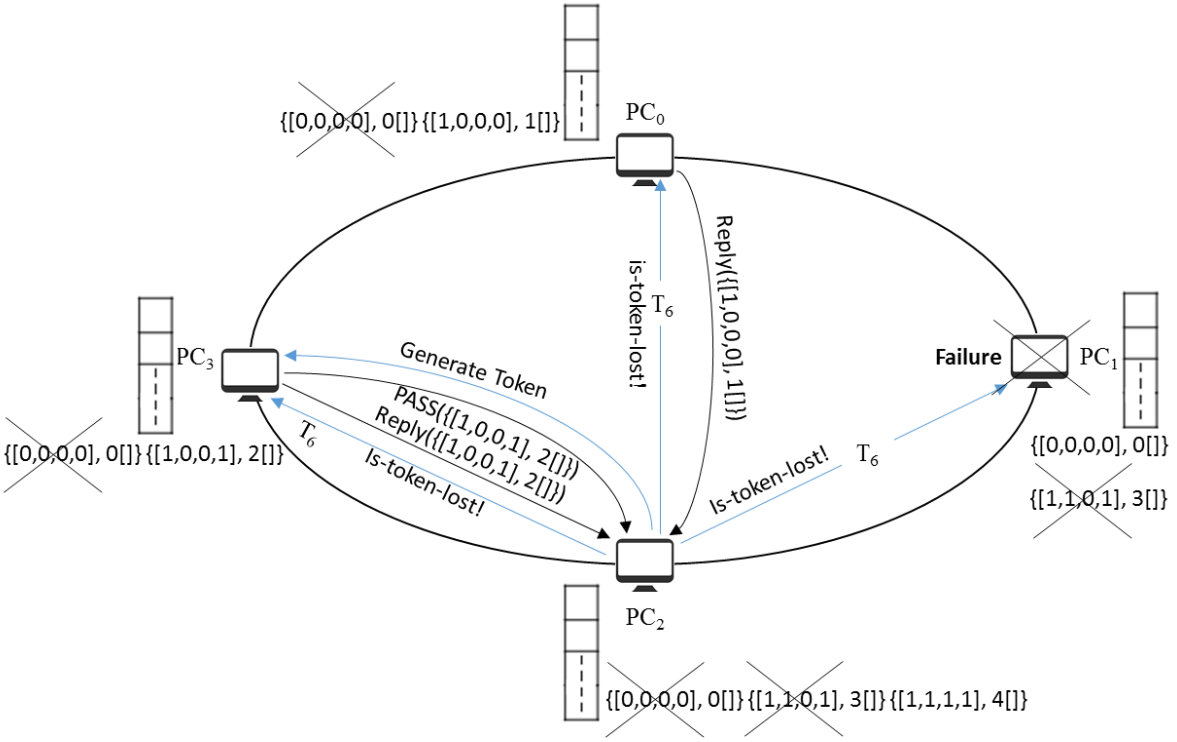

Fig 3: Sequence of steps with failure

V. CONCLUSIONS

The proposed solution gives improved performance to recover from token-loss which is considered the main and critical problem in token ring topology. In token ring topology any random unexpected failure in any node or link will inevitably lead to token-loss problem, thus because of the impossibility of expecting all the errors that may occur, this makes the issue of preventing token-loss problem is costly and ineffective, so we went to the alternative solution to such problem.

In this work we have used a special token format consisting of indexes, queue information and a counter, also we have assigned a queue to each node to handle multiple requests, besides a timestamp to indicate request priority. In general, our solution shows superiority over other similar solutions related with mutual exclusion condition to solve critical section problem.

REFERENCES

[1] Pankaj Rakheja, Dilpreet Kaur, Optimizing Performance of Token Ring for Bal- anced and Unbalanced Load Using OPNET, International Journal of Scientific & Engineering Research Volume 2, Issue 6, 2011.

[2] R.Mueller,"Architecture and Design of a Reliable Token-Ring Network," lEEE Journal on Selected Areas in communications, Vol. SAC-I, No. 5, pp. 756-765, November 1983.

[3] Nasro Min-Allah, Manzoor Elahi, Xing jiansheng, Wang Yong- ji," Enhancing Feasibility Analysis of IEEE 802.5 Token Ring", IEEE 2008

[4] W. Bux, "Token-ring local-area networks and their performance," in Proceedings of the IEEE, vol. 77, no. 2, pp. 238-256, Feb 1989. doi: 10.1109/5.18625.

[5] Computer Networks, Andrew S. Tanenbaum, David J. Wetherall, Prentice Hall, 5th edition, 2010.

[6] M. Ergen, Duke Lee, Raja Sengupta and P. Varaiya, "WTRP - wireless token ring protocol," in IEEE Transactions on Vehicular Technology, vol. 53, no. 6, pp. 1863-1881, Nov. 2004. doi: 10.1109/TVT.2004.836928.

[7] W. Bux, F. Closs, K. Kuemmerle, H. Keller and H. Mueller, "Architecture and Design of a Reliable Token-Ring Network," in IEEE Journal on Selected Areas in Communications, vol. 1, no. 5, pp. 756-765, November 1983. doi: 10.1109/JSAC.1983.1146004

[8] Shivakant Mishra, Pradip K. Srimani, Fault-tolerant mutual exclusion algorithms, Journal of Systems and Software, Volume 11, Issue 2, February 1990, Pages 111-129, ISSN 0164-1212, doi: 10.1016/0164-1212(90)90056-R.

[9] Ernest Chang and Rosemary Roberts. 1979. An improved algorithm for decentralized extrema-finding in circular configurations of processes. Commun. ACM 22, 5 (May 1979), 281-283. doi: 10.1145/359104.359108

[10] J. Singhal (1998). "Optimal Design of a Two-Level Hierarchical Transportation Network with a Different Unit Cost for Each Secondary Link," Decision Sciences, 29, 1, 87-103.

[11] Kishore Singhal, Jiri Vlach, Method for Computing Time Response of Systems Described by Transfer Functions, Journal of the Franklin Institute, Volume 311, Issue 2, 1981, Pages 123-130, ISSN 0016-0032, doi: 10.1016/0016-0032(81)90044-2.

[12] S. Nishio, K. F. Li, E. G. Manning, "A time-out based resilient token transfer algorithm for mutual exclusion in computer networks", IEEE 9th International Conference on Distributed Computing Systems, pp. 386-393, 1989.

[13] Sopena, Julien, et al. "A fault-tolerant token-based mutual exclusion algorithm using a dynamic tree." European

Conference on Parallel Processing. Springer Berlin Heidelberg, 2005.

[14] Manivannan, Dakshnamoorthy, and Mukesh Singhal. "Decentralized token generation scheme for token-based mutual exclusion algorithms." COMPUTER SYSTEMS SCIENCE AND ENGINEERING 11.1 (1996): 45-54.

[15] Mueller, Frank. "Fault-Tolerance for Token-based Synchronization Protocols." IPDPS. 2001.

[16] Reddy, P. Sukendar, Nityananda Sarma, and Rajib Kumar Das. "Enhancing Fault-Tolerance in a Distributed Mutual Exclusion Algorithm." Information Technology, 2006. ICIT'06. 9th International Conference on. IEEE, 2006.

[17] R. Ekwall and A. Schiper, "A Fault-Tolerant Token-Based Atomic Broadcast Algorithm," in IEEE Transactions on Dependable and Secure Computing, vol. 8, no. 5, pp. 625-639, Sept.-Oct. 2011, doi: 10.1109/TDSC.2010.24

[18] D. Agrawal, A. Elabbadi, A Token-Based Fault-Tolerant Distributed Mutual Exclusion Algorithm, Journal of Parallel and Distributed Computing, Volume 24, Issue 2, 1995, Pages 164-176, ISSN 0743-7315, doi: 10.1006/jpdc.1995.1016.

[19] Weigang Wu, Jiannong Cao, Michel Raynal, "A Dual-Token-Based Fault Tolerant Mutual Exclusion Algorithm for MANETs", Mobile Ad-Hoc and Sensor Networks: Third International Conference, MSN 2007 Beijing, China, December 12-14, 2007 Proceedings, doi: 10.1007/978-3-540-77024-4_52.