A STUDY OF FAILURE NODE DETECTION TECHNIQUES FOR WIRELESS AD-HOC NETWORK

Kailash P. Dewangan[1] ,Padma Bonde[2]

[1]Department of Computer Science and Engineering , KITE, Raipur-Chhattisgarh, India

[2]Department of Computer Science and Engineering , SSGI, SSTC, FET, Bhilai-Chhattisgarh, India

Email- kaishapt@yahoo.com , drpadmabonde@sstc.ac.in

ABSTRACT

Ad-hoc Networks are multi-hop network in which there is no need of central administration. Each node in Ad-hoc networks act as router to send and receive data. A failure detector is an important building block when using ad-hoc network. In ad-hoc network the failed nodes are often in- distinguishable from slow processes. Different classes of failure node detectors have been proposed to solve different kinds of problems. Unfortunately, ad-hoc devices are vulnerable to failure because of various factors, including physical damage due to deployment in harsh environmental conditions, limited energy, and malicious attacks. This paper presents the techniques for failure node detection. Further, this paper also provides the pros and cons of techniques.

Keywords*: Ad-hoc Network, Failure Node Detection, Heartbeat Techniques.*

1.Introduction

There are two types of wireless networks: first is the infrastructure wireless networks and second one is infrastructure-less wireless networks. This paper is concern about infrastructure-less wireless networks, also called Ad-hoc Networks. In Ad-hoc network environment, for establishing a communication between a sender and corresponding receiver, all the intermediate nodes provide equal contribution for finding the route paths and helps to start a communication. Nodes are self-healing and self-configurable because of the topology of Ad-hoc networks changes periodically and nodes in the Ad-Hoc Networks is free to go away anywhere in the network. Ad-Hoc Networks does not contain any infrastructure so there is a bigger challenge to keep up Quality of Service (QOS) in the wireless background. Ad-Hoc networks are vulnerable to many kinds of attacks because of its dynamic nature.Detecting node failure is an important problem that has been widely studied. Recent attention has focused on determining failure when nodes are mobile. Detection of node failure requires additional messages to be sent across the network, which is costly in terms of energy consumption. Node failuredetection in ad-hoc networks is very challenging because the network topology can be highly dynamic due to node movements. Once a node fails, it can no longer communicate with other nodes.The first work to address these properties of failure detectors was by Chandra and Toueg [1]. The authors showed why it is impossible for a failure detector algorithm to deterministically achieve both completeness and accuracy over an asynchronous unreliable network. R.Jin et al. [2], the author take a probabilistic approach and propose two node failure detection schemes that systematically combine localized monitoring, location estimation and node collaboration. R. Badonnel et al. [3] proposes a fault monitoring approach for ad-hoc networks which considers this constraint. This approach is based on an information theory measure suitable to the intermittence of ad-hoc nodes and capable to detect network failures by inference. M. Elhadef at al. [4] introduce the concept of unreliable failure detectors and study how they can be used to solve Consensus in asynchronous systems with crash failures.Gupta at al. [5], the author assume a crash-recovery failure model, and a network model that is

probabilistically unreliable. N. Sridhar [6] present a local failure detector that can tolerate mobility and topology changes. This means that this failure detector can distinguish between a failed process and a process that has moved away from its original location. It also establishes an upper bound on the duration for which a process wrongly suspects a node that has moved away from its neighborhood. Liu and J. Payton [7] present two approaches to dynamically adapting a fault detection algorithm. The author compares their adaptive approaches to existing approaches and evaluate the tradeoffs between cost and accuracy.The following approaches are used for detecting failed nodes in ad-hoc networks like: heartbeating architectures, pinging architectures, gossiping architectures, adaptive timeouts and leases.

1. Heartbeating architectures

Heartbeat protocols are widely used for failure detection in ad-hoc network [8,9]. In these protocols, a node periodically sends a heartbeat message ("I am alive") to a detector node. If the time between consecutive heartbeat messages exceeds a timeout value, then the node is considered failed. Heartbeat architectures are used in many areas: system diagnosis, network protocols, reaching agreement, and fault detection in computer networks. Many variants of heartbeating architectures are found in the literature, depending on the network topology: centralized, all-to-all, ring-based, and cluster-based heartbeating.

A. Centralized heartbeating:

A centralized entity senses the arriving heartbeat messages. Each node periodically sends a heartbeat message to the centralized entity. A node is declared failed if the centralized entity does not hear from it for a defined timeout. This variant is simple to implement. It is often used in devices that control servers to ensure that they are running. When the devices miss a user-defined number of heartbeat intervals, they will reboot the servers. Unfortunately, the centralized entity presents a single point of failure and potential bottleneck when the network scales upwards. This architecture is not appropriate for MANETs since they are inherently fully decentralized.

B. All-to-all heartbeating:

Every node in the network periodically sends heartbeat messages to every other node [9]. If a node does not receive a heartbeat message from a node after a certain period, it declares that node failed. This variant demands a high bandwidth and consequently it does not scale well.

C. Ring-based heartbeating:

In this variant, the nodes are connected logically or physically in a ring. Each node sends a heartbeat message to its successor neighbor when it receives a heartbeat message from its previous neighbor. A node is determined failed if it does not receive a heartbeat message from its neighbor after a timeout. This variant was used in IBM SP-2 [10]. It presents a high detection delay and it is unpredictable for simultaneous multiple failures.

D. Cluster-based heartbeating:

In this variant, the network is partitioned in clusters. Each cluster is maintained by a cluster-head. Different heartbeat styles can be applicable inside clusters and between cluster-heads. For example, it can be all-to-all heartbeating inside clusters and

ring-based between cluster-heads. Cluster-based heartbeating is a compromise between centralized heartbeating and all-to-all heartbeating. By nature, it is decentralized and can be easily made scalable. Moreover, it minimizes the system throughput. Studies of cluster-based failure detection issues for MANET applications are still largely lacking.

The work done in [11] presents a cluster-based failure detection service (FDS) for applications that are made up of large and dense populations of lightweight system resources. Applications are built over ad- hoc wireless networks. The FDS exploits the message broadcasting in wireless networks to build a heartbeat failure-detection service. A cluster-head and the nodes in its range constitute a cluster. Nodes in the range of two clusters may act as gateways for inter-cluster failure forwarding. The heartbeat style is composed of three phases: heartbeat exchange, digest exchange and health-status-update diffusion. In the first phase, every node in the cluster broadcasts a heartbeat message to its cluster-head while the cluster-head broadcasts a heartbeat message to its members.

In the second phase, every node in a cluster sends the cluster-head a digest message, which enumerates the nodes that it heard in the first phase. The cluster-head broadcasts its own digest messages to its members. Finally, in the third phase, the cluster-head analyzes the information collected in the two previous phases, identifies the failed nodes according to a set of failure detection rules and then diffuses an update message to all the nodes. A cluster-formation algorithm ensures the election of cluster-heads and connectivity between the clusters. It does not support routing stability when nodes move. In fact, the current cluster-based heartbeating solution is designed for stationary hosts and does not address node mobility at all, which is critical for MANETs. Consequently, the clusters will disappear once the nodes move. The FDS will then give unpredictable results since it relies on clusters that may no longer be there.

2. Pinging architectures

In this approach, a node sends a ping message ("Are you alive?") to another node. The receiving node replies with an acknowledgement message ("I am alive"). Two strategies are used for detecting failed nodes. A node is considered failed if it does not send an acknowledgement within a timeout or fails to respond to a defined number of ping messages. Pinging architectures are more vulnerable to message loss than heartbeating architectures because of their acknowledgement feature, which increases message loss.

A variant of pinging architectures, randomized pinging, is described in [5]. Each node randomly picks another node to ping. If there is no response, k other nodes are randomly chosen to ping the suspected node. If an acknowledgement is received by one of the k nodes, the acknowledgement is forwarded to the original node. If no acknowledgement is received by the original node, the suspected node is considered failed. This variant considerably decreases the bandwidth, but it presents a high detection delay.

3. Gossiping architectures

Gossiping architecture was presented for the first time in [12]. Subsequently, some new versions of gossiping architecture have been proposed [13]. In this architecture, each node in the network maintains a list of $<M_i, H_i, T_{last}>$ such that M_i is the address of node i, H_i is the heartbeat count and T_{last} is the last time of the heartbeat increase. Every T_{gossip} time, each node increments its heartbeat, selects a random target node (from its list) and sends to it a constant number of $<M_i, H_i>$ entries. A node, upon receiving a gossip message, merges its list with the list received (taking the maximum of heartbeats). If the sum of T_{last} of M_i and a predefined T_{fail} is less than that of the current system time, then node M_i is considered as failed. These architectures are resilient against message loss but they use a large bandwidth because of message length.

4. Adaptive Timeouts

Each node p maintains an adaptive timeout for each neighboring process q [14]. If p suspects q wrongly after, say, time T and later hears from q after delay q, then p updates the timeout period for q to be long enough so that this mistake is not repeated. The new T is now at least T + delay q. Each process thereby keeps extending timeouts until such a time when the timeouts are long enough to account for all forms of incidental delays.

5. Leases

In applications in which nodes sleep for most of the time (as with sensor nets) [15], none of the strategies listed above make sense. In this context, the roles could be reversed. Each process x sends to each neighbor y an "I am alive" message. In addition, x also sends to y a request for a lease for some duration T. Now, x can go to sleep, and all it must do is wake up some time before T expires and send a request for lease renewal to all its neighbors. This is the strategy we use in this paper to implement local failure detection.

6. Discussion

Pinging architectures are not appropriate for ad-hoc network because of their large bandwidth and high detection delay. Gossiping is interesting for ad-hoc network, but large messages are not suitable, especially in large-scale networks. Heartbeating can provide rapid failure detection, which is very important in ad-hoc network since messages pass through multi-hop nodes. Unfortunately, the existing cluster-based heartbeating solution for ad-hoc network applications is designed for stationary hosts. Heartbeating in general is more appropriate for ad-hoc network than the other approaches

7. Conclusion

Soon, ad-hoc network applications will be in demand. The need for failure detection architecture is crucial, especially for real-time applications. Failure detection technology is used to detect survival state of nodes in ad-hoc network to support services running uninterrupted. However, failure detection will consume computing and communications resources of system, reducing the system's ability to supply service. When number of nodes is small, the load brought by failure detection is not obvious. With the increasing scale, detection time of the system extends and amount of detection message increases. Therefore, it is necessary to reduce the load brought by failure detection. In future work we will investigate a real implementation of this failure detection architecture for real time applications.

8. References

[1]. T. Deepak Chandra, Tf J. Thomas, S. Toueg, T. D. Chandra, And S. Toueg, "Unreliable Failure Detectors For Reliable Distributed Systems," J. Acm, Vol. 43, No. 2, Pp. 225–267, 1996.

[2]. R. Jin Et Al., "Detecting Node Failures In Mobile Wireless Networks: A Probabilistic Approach," Ieee Trans. Mob. Comput., Vol. 15, No. 7, Pp. 1647–1660, 2016.

[3]. R. Badonnel, R. State, And O. Festor, "Self-Configurable Fault Monitoring In Ad-Hoc Networks," Ad Hoc Networks, Vol. 6, No. 3, Pp. 458–473, 2008.

[4]. M. Elhadef And A. Boukerche, "A Failure Detection Service For Large-Scale Dependable Wireless Ad- Hoc And Sensor Networks," Proc. - Second Int. Conf. Availability, Reliab. Secur. Ares 2007, Pp. 182–189, 2007.

[5]. Gupta, T. D. Chandra, And G. S. Goldszmidt, "On Scalable And Efficient Distributed Failure Detectors," Proc. Twent. Annu. Acm Symp. Princ. Distrib. Comput., Pp. 170--179, 2001.

[6]. N. Sridhar, "Decentralized Local Failure Detection In Dynamic Distributed Systems," Reliab. Distrib. Syst. 2006. Srds '06. 25th Ieee Symp., Pp. 143–154, 2006.

[7]. Liu And J. Payton, "Adaptive Fault Detection Approaches For Dynamic Mobile Networks," Pp. 735–739, 2011.

[8]. D. Ben Khedher, R. Glitho, And R. Dssouli, "A Novel Overlay-Based Failure Detection Architecture For Manet Applications," Icon 2007 - Proc. 2007 15th Ieee Int. Conf. Networks, Pp. 130–135, 2007.

[9]. Shen, T. Yang, L. Chu, J. L. Holliday, D. A. Kuschner, And H. Zhu, "Neptune: Scalable Replica Management And Programming Support For Cluster-Based Network Services," Usits'01, Pp. 197- 208, Mar. 2001.

[10]. T. Agerwala , J. L. Martin , J. H. Mirza , D. C. Sadler , D. M. Dias, "Sp2 System Architecture," Ibm Systems Journal, V. 34 N. 2, Pp. 152- 184, 1995.

[11]. A. T. Tai, K. S. Tso, W. H. Sanders, "Cluster-Based Failure Detection Service For Large-Scale Ad Hoc Wireless Network Applications," International Conference On Dependable Systems And Networks, Pp. 805-814, 2004.

[12]. R. Van Renesse, Y. Minsky, And M. Hayden, "A Gossip-Style Failure Detection Service," Proc. Of Middleware'98, Pp. 55-70, Sep. 1998.

[13]. S. M. Hedetniemi, S.T. Hedetniemi, Amd A. Liestman, "A Survey Of Gossiping And Broadcasting In Communication Networks", Ieee Networks, Vol. 18, Pp. 319-349, 1988.

[14]. C. Fetzer, U. Schmid, And M. Susskraut. On The Possibility Of Consensus In Asynchronous Systems With Finite Average Response Times. In Icdcs '05, Pages 271–280, 2005.

[15]. R. Boichat, P. Dutta, And R. Guerraoui. Asynchronous Leas- Ing. In 7th Ieee Intl. Wksp. On Object-Oriented Real Time Dependable Systems (Words '02), Pages 180–187, 2002

