Mathematical Statistician and Engineering Applications
ISSN: 2094-0343
2326-9865

# A Structured Minimum-Process Reliable Recovery Line Assortment Protocol for Mobile Ad-hoc Networks

**Dr. SK Wasim Haidar[1], Mohammad Meraj[2], Syed Ahad Murtaza Alvi[3]**

[1]Lecturer, Information Technology Department, College of Computing & Information Sciences, UTAS-Salalah,Oman,

[2,3]Lecturer, College of Applied Computer Sciences, King Saud University, Saudi Arabia,

[1]wasims5@gmail.com, [2]merajjmi@gmail.com,[3]salvi@ksu.edu.sa

***Article Info***

***Page Number: 8905 - 8917***

***Publication Issue:***

***Vol 71 No. 4 (2022)***

***Abstract***

In ad-hoc interconnection every nodule is self-organized and can converse directly with all other nodules. An Ad-hoc mobile interconnection is composed of mobile nodules that converse with each other through broadcast radio transmissions within the transmissions power range. Paralleled to the wired distributed computing framework, ad-hoc wireless interconnections have certain new features. The transient letdown probability of the framework increases greatly with the enlarging of framework scale. If a letdown comes into play in a proceeding and there is not an applicable method to protect it, more cost will be wasted for restarting the program. Coordinated RRL-assortment (Reliable Recovery Line assortment) can be employed to introduce fault tolerance in mobile ad-hoc wireless interconnections environment .In this paper we propose a new minimum proceeding RRL-assortment scheme for ad-hoc interconnections. We assume that Cluster Based Routing Protocol (CBRP) is employed which have its place to the class of Hierarchical Reactive Routing Protocols. The number of synchronized reckoning communications between a cluster head and its ordinary associates is small. The reclamation scheme has no domino effect and the letdown proceeding can roll back from its latest local dependable Restoration-point.

***Article History***

***Article Received****: 25 March 2022*

***Revised****: 30 April 2022*

***Accepted****: 15 June 2022*

***Publication****: 19 August 2022*

## 1. Introduction

Mobile ad-hoc interconnections are accumulation of two or more devices equipped with wireless communication and interconnection capability. These devices can converse with other nodules that lie immediately within their radio range or one that is outside their radio range. For the later, the

nodules should arrange an in-between nodule to be the router to route the packet from the source toward the endpoint. The Wireless Ad-hoc interconnections do not have gateway, every nodule can act as the gateway [12, 13]. The fixed interconnections have fixed and wired gateways or the fixed Base-Stations which are linked to other Base-Stations through wires. Each nodule is within the range of a Base-Station. A 'Hand-off' comes into play as mobile host travels out range of on Base-Stations and into range of another and thus, mobile is able to continue communication impeccably throughout the interconnection. Example applications of this type include wireless local area interconnections and Mobile Phone [8].

The other type of wireless interconnection is known as **Mobile Ad-hoc Interconnections (MANET).** These interconnections have no fixed routers, every nodule could be router. All nodules are capable of relocation and can be linked dynamically in arbitrary manner. The responsibilities for organizing and controlling the interconnection are distributed among the nodules themselves. The entire interconnection is mobile and the individual nodules are endorsed to relocate freely. In this type of interconnections, some pairs of nodules may not be able to converse directly with each other and have to rely on some nodules so that the reckoning communications are delivered to their destinations. Such interconnections are often referred to as multi-hop or store-and–forward interconnections .The nodules of these interconnections functions as routers, which discovered and maintain routes to other nodules in the interconnections. The nodules may be located in or on airplanes, ships, trucks, cars, perhaps even on people or very small devices [10, 11]. Mobile Ad-hoc Interconnections are supposed to be employed for disaster reclamation, battle field Communications, and rescue operations when the wired interconnection is not available. I t can be provided a feasible means for ground communications, and information access.

The topology of the ad-hoc interconnection is represented by an undirected graphic $G= (V, E)$, where $V$ is the set of all the mobile nodules, $E$ is the set of all the mobile links. If edge $(u, v) \in E$, then edge $(u, v), \in E$. Node u and v belong to the communication range of each other, and they are 1-hop neighbors. The set of nodule $i$'s 1-hop neighbor is denoted $N^1_{i.}$ If two nodules share the same 1-hop neighbor, and the shortest path between them is 2 hops, then the two nodules are each other's 2-hop neighbors. The set of nodule $i$'s 2-hop neighbors is denoted $N^2_{i.}$ If the shortest path between two nodules is 3 hops, then they are each other's 3-hop neighbors. The set of nodule $i$'s 3-hop neighbors is denoted $N^3_{i.}$ All the nodules use directional antennae, and have the same transmission ranges [12].

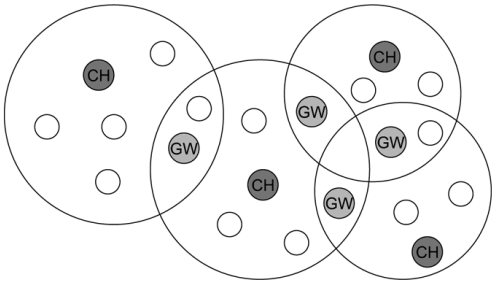


Figure 1:-A topology example after clustering

Each nodule may work as one of the four following roles: cluster-head, gateway, compound gateway and cluster-associate. As shown in figure 1, there are four clusters; each cluster is having one cluster head (denoted by Clust_Hd ). Gateway Nodes (denoted by GW) interconnect the cluster head nodules. First of all, since the nodules in Wireless Ad-hoc Interconnection are free to relocate arbitrarily at any time .So the interconnections topology of MANET may change randomly and rapidly at unpredictable times. This makes routing difficult because the topology is constantly changing and nodules cannot be assumed to have persistent data storage. In the worst case, we do not even know whether the nodule will still remain next minute, because the nodule will leave the interconnection at any minute [8].

Bandwidth constrained is also a big challenge. Wireless links have meaningfully lower capacity than their hardwired counterparts. Also, due to multiple access, fading, noise, and interference conditions etc. the wireless links have low throughput. Some or all of the nodules in MANET may rely on batteries. In this scenario, the most important framework design criteria for optimization may be energy conservation. Mobile interconnections are generally more prone to physical security threats than are fixed cable interconnections. There are increased possibility eavesdropping, spoofing and denial-of-service attacks in these interconnections.

A cluster-based multi-channel ad-hoc wireless interconnection consists of a set of mobile hosts (Mob-Hst), which converse with each other through wireless channels. There exists no shared memory or common clock among these nodules and the communication between the nodules is by reckoning communication-passing only. We assume that wireless channels are all FIFO order. In cluster-based interconnection architecture, the interconnection is partitioned into several clusters. The roles of mobile hosts in a cluster can be categorized into cluster head, gateway, and ordinary associates. The propose of using the cluster-based architecture is to enhance the overall framework

throughput and to achieve resource reuse [10, 11, 12, 13]. In each cluster, it has a unique leader, called a cluster head, to enforce channel allocation. A cluster head is a local manager of all mobile hosts within a cluster. In the same cluster, the mobile host called clusters associates that controlled by the cluster-head. One of the basic functions for a cluster head is broadcasting beacon packets to all mobile hosts in the cluster.

A restoration-point is a local circumstance of a proceeding saved on stable storage. In a distributed framework, since the proceedings in the framework do not share memory, a comprehensive circumstance of the framework is demarcated as a set of local circumstances, one from each proceeding. The circumstance of channels corresponding to a comprehensive circumstance is the set of reckoning communications consigned but not yet acknowledged. A lost or in-transit reckoning communication is one, the dispatching of which has been logged by the dispatcher but whose collecting could not be logged by the collecting proceeding. An orphan reckoning communication is a reckoning communication whose accept event is logged, but its dispatch event is lost. A comprehensive circumstance is said to be "dependable" if it contains no orphan reckoning communication and all the in-transit reckoning communications are logged. To recover from a letdown, the framework restarts its accomplishment from a previous RRL saved on the stable storage during fault-free accomplishment. This saves all the working out done up to the last RRL and only the working out done thereafter prerequisites to be redone [1, 2, 3].

In this paper, we devise a minimum proceeding non-intrusive RRL-assortment mechanism for mobile ad-hoc interconnections. There is no common clock, shared memory or central coordinator. Communication passing is the only mode of communication between any pair of proceedings. Communications are exchanged with finite but arbitrary delays. In our mechanism, we consider that the proceedings which are running in the distributed mobile frameworks are non-deterministic. The mechanism is distributed in nature. There is no centralized controlling nodule. To avoid any waste of bandwidth or CPU consumption, the mechanism is loop free. We assume that Cluster Based Routing Protocol (CBRP) is employed which have its place to the class of Hierarchical Reactive Routing Protocols. Clustering Routing Strategy is highly employed in Ad-hoc Interconnections to surpass scalability problem. By limiting the interconnection view of each nodule, clustering moderates the routing complexity and the size of the routing table. The local relocation of nodules is handled only within the cluster without affecting other parts of the interconnection and so the overhead is highly abridged.

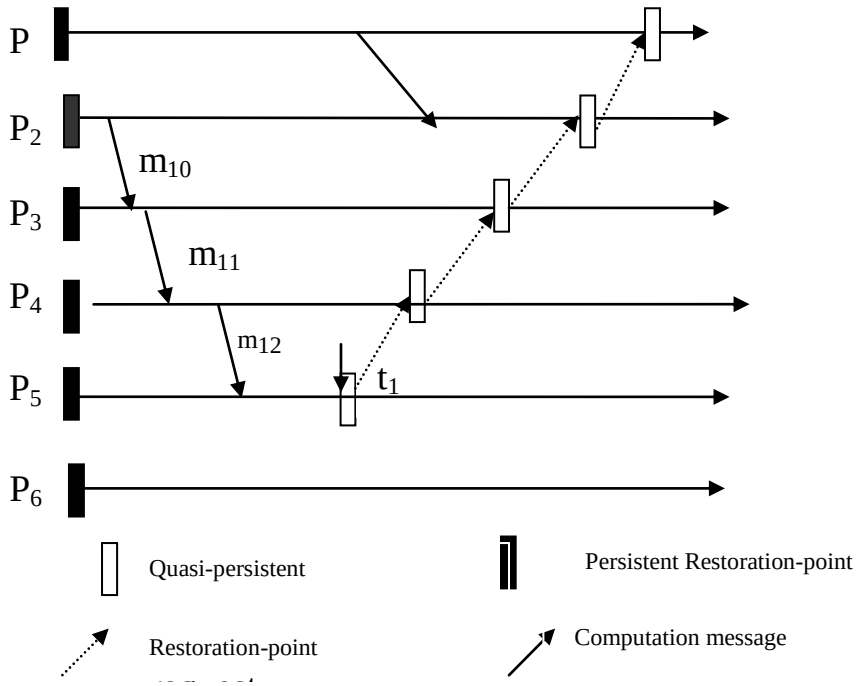


Figure 2

## 2. The Proposed RRL-assortment Algorithm

### 2.1 System Model

Our framework model consists of a number of Mob-Hsts which converse through Cluster Heads (Clust_Hds). Each Clust_Hd provides wireless communication support for a fixed geographical area, called a cluster. Clust_Hds are linked together over the Wireless data interconnections through Gateway Nodes. We assume that wireless channels and logical channels are all FIFO order. If a Mob-Hst relocates to the cell of another Clust_Hd , a wireless channel to the old Clust_Hd is disengaged and a wireless channel in the new cluster is allocated.

There is no common clock, shared memory or central coordinator. Reckoning communication passing is the only mode of communication between any pair of proceedings. Any proceeding can pledge RRL-assortment. It is assumed that proceedings may be flopped during reckoning but there is no communication link letdown. Reckoning communications are exchanged with finite but arbitrary delays. In our mechanism, we consider that the proceedings which are running in the mobile ad-hoc interconnection are non-deterministic.

### 2.2 Basic Idea

In figure 2, at time t1, suppose proceeding P5 pledges RRL-assortment proceeding. It should be noted that our projected mechanism is distributed in nature and any proceeding can pledge RRL-

assortment. If two proceedings contemporaneously pledge RRL-assortment, the restoration-point instigation of the proceeding with lower proceeding_ID will prevail. In this way, contemporaneous instigations will not lead to contemporaneous accomplishments of the projected mechanism. If we use the technique to capture the transitive causal interdependency by direct causal-interdependencies projected by Cao Singhal and other similar mechanisms; the following scenario will take place. P5 dispatches the RRL-assortment appeal to P4 due to m12. On collecting RRL-assortment appeal P4 captures its quasi-persistent restoration-point and dispatches the RRL-assortment appeal to P3 due to m11. Similarly, after arresting its quasi-persistent restoration-point, P3 dispatches the RRL-assortment appeal to P2 due to m10. In this way, RRL-assortment tree of height three is generated and the RRL-assortment time may be exceedingly high in ad-hoc interconnections.

In the projected scheme, every proceeding preserves a causal interdependency array (say CDV[]) of length n where n is the number of proceedings in the ad hoc interconnection. CDVi[j]=1 implies Pi is causally dependent upon Pj. CDVi[j] is set to '1' only if Pi proceedings m acknowledged from Pj such that Pj has not captured any persistent restoration-point after dispatching m. In our mechanism, causal interdependency arrays are maintained as follows. Let the initial causal interdependency arrays of P1, P2, P3, P4, P5, P6 are CDV1 [000001], CDV 2 [000010], CDV 3 [000100], CDV 4 [001000], CDV5[010000], CDV6[100000], respectively. In figure 2, P2 dispatches m10 to P3 along with its causal interdependency array CDV2[000010]. When P3 acquires m10, it appends its causal interdependency array CDV3 by arresting the bitwise logical OR of CDV2[000010] and CDV3 [000100], which comes out to be [000110]. Similarly, P3 dispatches m11 to P4 along with its own causal interdependency array CDV4[000110].

After collecting m11 by P4, CDV4 becomes [001110]. At time t1, P5 pledges RRL-assortment proceeding with the CDV5 [011110], and dispatches the RRL-assortment appeal to P2, P3, P4. In this way no RRL-assortment tree is formed as found in Cao-Singhal mechanism [2] as detailed above. In this way, the time to accumulate the comprehensive circumstance will be significantly low as equaled to Cao-Singhal mechanism [2]. Therefore, the time to accumulate the comprehensive restoration-point will be less and the number of unserviceable restoration-points will also be abridged considerably. The original idea of capturing the transitive causal-interdependencies during normal reckoning was projected by Prakash-singhal [5].

In figure 2, when P2 captures its quasi-persistent restoration-point C21 and finds that P1 is in the causal interdependency set of P2, but is not available in the minimal interacting set {P2, P3, P4, P5} acknowledged from P5. In this case, if P1 does not arrest its restoration-point in the existing instigation, m13 will become orphan. Therefore, P2 dispatches restoration-point appeal to P1 and P1 captures its quasi-persistent restoration-point C11. In this way, we get [ C11, C21, C31, C41, C51, C60] as the RRL.

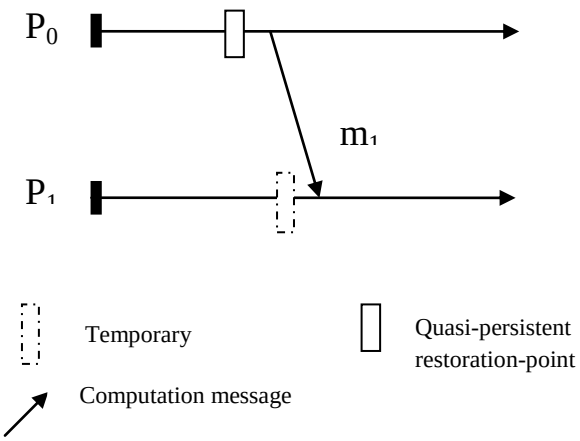


Figure 3

In figure 3, P0 captures its quasi-persistent restoration-point and dispatches m11 to P1. P1 has neither captured its quasi-persistent restoration-point nor acknowledged any RRL-assortment appeal from any other proceeding. By the piggybacked information along with m11 and certain other data structures, P1 concludes that P0 has captured its quasi-persistent restoration-point for some new instigation. In this case if P1 captures its restoration-point after reckoning m11, m11 will become orphan. Therefore, we propose that P1 will arrest a forced restoration-point (say temporary Restoration-point ) before reckoning m. If P1 does not receive any RRL-assortment appeal during the existing instigation, P1 will discard it on commit. In this case, if we find that P1 has not consigned any reckoning communication to any proceeding since its last committed restoration-point, then P1 will process m without arresting its temporary restoration-point . Because, we can say that P1 will not be encompassed in the minimal interacting set in this case.

2.3 Data Structures

The following section describes the notations and data structures employed in our mechanism. In our mechanism, any proceeding can pledge the RRL-assortment operation. Data structures are

initialized on the completion of a RRL-assortment proceeding. We assume that there are n proceedings running in the framework.

- **n_csni :**restoration-point sequence number of proceeding Pi and is incremented when Pi captures a quasi-persistent restoration-point.
- **uncertain_i** A proceeding sets the flag on getting causal interdependency set appeal from instigator. The flag is reset on getting minimal interacting set.
- **CDV i []:**An array of n bits for proceeding Pi . Where CDVi[j] becomes '1' when Pi acquires a reckoning communication from Pj in the existing RRL-assortment interval. In the beginning of every RRL-assortment interval, this array is zero for all the proceedings except for itself which is initialized as '1'. Maintenance of CDVi[] is shown in basic idea.
- **chk_circumstancei:** A boolean which is set to '1' when Pi captures a quasi-persistent restoration-point; otherwise is zero on collecting abandon or commit appeal from the instigator proceeding.
- **Mess_dispatch_flag[i]**:A bit array of size n for n proceedings.
- **Mess_dispatch_flag** $_i$[j]=1 if Pi dispatches m to Pj.
- **set_cdp[]:** An array of size n employed to save minimal interacting set of proceedings on which instigator proceeding is transitively depends on. Initially, when RRL-assortment operation is started, set_cdp is CDVi[] of the instigator proceeding.
- **Chk_set[]:** An array of size n to save information about the proceedings which have captured their quasi-persistent restoration-points. When proceeding Pj captures its quasi-persistent restoration-point then jth bit of this array is set to 1.
- **Timeout_flag:** A flag employed to provide timing in RRL-assortment operation. It is initialized to zero when timer is set and becomes '1' when maximum allowable time for accumulating coordinating restoration-points is expired.
- **P_ Clust_Hd[]:**An array of size n employed to save the information on every Clust_Hd regarding the proceedings which are running in its cell. P_ Clust_Hd [k] = 1 indicate that proceeding Pk is running in the cell of this Clust_Hd . Information about disengaged Mob-Hst, if any, which are supported by this Clust_Hd , is also stored in this array.
- **tent_chk_set[]:**An array of n bits maintained by the Clust_Hd . Tent_chk_set [j]=1 whenever proceeding Pj which is in the cell of Clust_Hd has captured quasi-persistent restoration-point.

- **chk_appeal[]:**An array of n bits maintained also on every Clust_Hd . The jth bit of this array is set to 1 whenever instigator dispatches the restoration-point appeal to Pj and Pj is in the cell of this Clust_Hd.
- **error_flag:**A flag maintained on every Clust_Hd , initialized to '0' and set to '1' when any proceeding in the cell of Clust_Hd flops to arrest quasi-persistent restoration-point.
- **$P_{in}$**:The proceeding which has pledged the RRL-assortment operation.
- **Clust_Hd $_{in}$:**The Clust_Hd which has Pin in its cell.
- **$n\_csn_{in}$:**restoration-point sequence number of instigator proceeding.
- **g_chkpt :**A flag which indicates that some comprehensive restoration-point is being saved.
- **mess_csn_array []:**An array of size n, maintained on every Clust_Hd , for n proceedings .mess_csn_array[i] represents the most recently committed restoration-point sequence number of Pi. After the commit operation, if set_cdp[i]=1 then mess_csn_array[i] is incremented. It should be noted that entries in this array are updated only after converting quasi-persistent restoration-points in to persistent restoration-points and not after arresting quasi-persistent restoration-points.
- **set_cdp1[]:** An array of size n maintained on every Clust_Hd . It contains those new proceedings which are found on getting restoration-point appeal from instigator.
- **set_cdp2[]:**An array of size n. for all j such that set_cdp1[j] $\neq$ o, set_cdp2= set_cdp2$\cup$ set_cdp1.
- **set_cdp3[]:** An array of length n; on collecting set_cdp3, set_cdp, set_cdp1 along with restoration-point appeal [c_req] or on the working out of set_cdp1 locally: set_cdp3=set_cdp3$\cup$ c_req.set_cdp3; set_cdp3=set_cdp3$\cup$set_cdp; set_cdp3=set_cdp3 $\cup$c_req.set_cdp1; set_cdp3=set_cdp3 $\cup$ set_cdp1; set_cdp3 maintains the best local knowledge of the minimal interacting set at an Clust_Hd ;

2.4 The RRL-assortment Protocol

In a mobile ad-hoc interconnection framework, due to less bandwidth of wireless channels and vulnerability of storage of Mob-Hst, all the information regarding the RRL-assortment are stored in the stable storage of the Mob-Hst itself. In the projected mechanism, when a Mob-Hst dispatches a reckoning communication, it is first consigned to its local Clust_Hd over the wireless channel. The Clust_Hd then attaches the causal interdependency array of the proceeding with the reckoning communication and dispatches it to the Clust_Hd for which it was issued. The destination Clust_Hd strips this causal interdependency array from the reckoning communication; and transmits it to the

destination Mob-Hst over the wireless channels. The destination Clust_Hd updates the causal interdependency array of destination Mob-Hst (maintenance of causal interdependency array is explained in basic idea). In this way, no excessive data structures are allowed to travel over the wireless channels. It should be noted that a causal interdependency array of mobile hosts are maintained at Clust_Hds.

We propose that any proceeding in the framework can pledge the RRL-assortment operation. When a proceeding (say Pi) want to pledge RRL-assortment, it captures its quasi-persistent restoration-point and dispatch the appeal to its local Clust_Hd (instigator Clust_Hd). This local Clust_Hd coordinates the RRL-assortment operation on behalf of the instigator Mob-Hst. If two proceedings pledge RRL-assortment at the same time then the RRL-assortment instigation of the lower id will prevail. Clust_Hd in [instigator Clust_Hd ] dispatches the RRL-assortment appeal to all Clust_Hds alongwith set_cdp { set_cdp[ ] = CDVi[] }. It should be noted that the causal interdependency array of Pi i.e. CDVi[] contains all the proceedings on which it is directly or transitively dependent. set_cdp is a quasi-persistent minimal interacting set computed from CDVi[] of the instigator proceeding. When an Clust_Hd acquires the RRL-assortment appeal, it dispatches RRL-assortment appeal to Pj if Pj $\in$ set_cdp[] and Pj is in its cell and stores such proceedings in chk_appealj[ ]. We arrest the following action with every proceeding, say Pj, which is required to arrest its quasi-persistent restoration-point. If there exists any proceeding Pk such that Pk does not belong to set_cdp [] and Pk have its place to CDVj[], then Pj dispatches restoration-point appeal to Pk. During RRL-assortment proceeding, if a proceeding Pj acquires the reckoning communication m from Pi, it takes the following actions. If Pi has captured its quasi-persistent restoration-point before dispatching m and Pj has not captured its quasi-persistent restoration-point at the time of collecting m, in this case, Pj will arrest its temporary restoration-point before collecting m. It should be noted that if Pj captures its quasi-persistent restoration-point after collecting m, m will become orphan and resulting RRL will be undependable.

For a disengaged Mob-Hst that is an associate of minimal interacting set, the Clust_Hd that has its disengaged restoration-point , converts its disengaged restoration-point into quasi-persistent one. When a Clust_Hd learns that its applicable proceedings in its cell have captured their quasi-persistent restoration-points, it dispatches the response to Clust_Hd in. On collecting positive response from all applicable Clust_Hds, the Clust_Hd in issues the commit appeal to all Clust_Hds. On commit when a proceeding learns that it has captured an temporary restoration-point

and has not acknowledged the formal quasi-persistent RRL-assortment appeal from any proceeding, it discards its temporary restoration-point .

### 2.5 Handling Failures during RRL-assortment

A Mob-Hst may miscarry during RRL-assortment. If a Mob-Hst flops after arresting its quasi-persistent restoration-point or if it is not an associate of minimal interacting set, then the RRL-assortment procedure can be completed uninterruptedly. If a proceeding flops during RRL-assortment, then our straight forward approach is to discard the whole RRL-assortment operation. The flopped proceeding will not be able to respond to the instigator's appeal and the instigator will detect the letdown by timeout and will discard the complete RRL-assortment operation. If the instigator flops after dispatching commit, the RRL-assortment proceeding can be considered complete. If the instigator flops during RRL-assortment, then some proceedings, waiting for commit will time out and will issue abandon on his own.

Kim and Park [6] projected that a proceeding commits its quasi-persistent restoration-points if none of the proceedings, on which it transitively depends, flops; and the dependable reclamation line is advanced for those proceedings, that committed their restoration-points. The instigator and other proceedings, which transitively depend on the flopped proceeding, have to abandon their quasi-persistent restoration-points. Thus, in case of a nodule letdown during RRL-assortment, total abandon of the RRL-assortment is avoided.

## 3. Conclusion

We have projected a minimum-proceeding synchronized RRL-assortment mechanism for mobile ad-hoc framework; where no intrusive of proceedings takes place. We try to moderate the number of unserviceable restoration-points by avoiding RRL-assortment tree which may be formed in Cao-Singhal [2] mechanism. We captured the transitive causal-interdependencies during the normal accomplishment. The Z-causal-interdependencies are well taken care of in this mechanism. We also avoided accumulating causal interdependency arrays of all proceedings to find the minimal interacting set as in [1]. In this way, we abridged the reckoning communication complexity to a significant extent, as compared to these mechanisms. Thus the projected mechanism is simultaneously able to attain the zero intrusive time and to moderate the unserviceable restoration-points to bare minimum, by preserving exact causal-interdependencies among proceedings and

piggybacking RRL-assortment sequence number and causal interdependency array on to the normal reckoning communications.

**References**

[1] Cao G. and Singhal M., "On the Impossibility of Min-proceeding Non-intrusive RRL-assortment and an Efficient RRL-assortment Algorithm for Mobile Computing Frameworks," Proceedings of International Conference on Parallel Processing, pp. 37-44, August 1998.

[2] Cao G. and Singhal M., "Mutable Restoration-point s: A New RRL-assortment Approach for Mobile Computing frameworks," IEEE Transaction On Parallel and Distributed Frameworks, vol. 12, no. 2, pp. 157-172, February 2001.

[3] Koo R. and Toueg S., "RRL-assortment and Roll-Back Recovery for Distributed Frameworks," IEEE Trans. on Software Engineering, vol. 13, no. 1, pp. 23-31, January 1987.

[4] Parveen Kumar, Lalit Kumar, R K Chauhan, V K Gupta "A Non-Intrusive Minimum Proceeding Synchronous RRL-assortment Protocol for Mobile Distributed Frameworks" Proceedings of IEEE ICPWC-2005, pp 491-95, January 2005.

[5] Prakash R. and Singhal M., "Low-Cost RRL-assortment and Failure Recovery in Mobile Computing Frameworks," IEEE Transaction On

[6] Parallel and Distributed Frameworks, vol. 7, no. 10, pp. 1035-1048, October1996.

[7] J.L. Kim, T. Park, "An efficient Protocol for RRL-assortment Recovery in Distributed Frameworks," IEEE Trans. Parallel and Distributed Frameworks, pp. 955-960, Aug. 1993.

[8] L. Kumar, M. Misra, R.C. Joshi, "Low overhead optimal RRL-assortment for mobile distributed frameworks" Proceedings. 19th IEEE International Conference on Data Engineering, pp 686 – 88, 2003.

[9] Murthy & Manoj, "Ad hoc Wireless Interconnections Architectures and Protocols", Pearson Education, 2004.

[10] D.J. Baker and A. Ephremides, "The Architectural Organisation of a Mobile Radio Interconnection via a Distributed mechanism", IEEE Trans. Commun., vol. 29, no. 11, pp 1694-1701, Nov., 1981

[11] D.J. Baker, A. Ephremides and J.A. Flynn "The design and Simulation of a Mobile Radio Interconnection with Distributed Control", IEEE J. sel. Areas Commun.., pp 226-237, 1984

[12] B.Das, R. Sivakumar and V. Bharghavan, "Routing in Ad-hoc interconnections using a Spine",Proc. Sixth International Conference, 1997.

[13] B.Das, R. Sivakumar and V. Bharghavan, "Routing in Ad-hoc interconnections using Minimum linked Dominating Sets",Proc. IEEE International Conference, 1997.

[14] M.Gerla, G. Pei, and S.J. Lee, "Wireless Mobile Ad-hoc Interconnection Routing", Proc. IEEE/ACM FOCUS'99, 1999.

[15] M. Singhal and N. Shivaratri, Advanced Concepts in Operating Frameworks, New York, McGraw Hill, 1994.