# AN ADEPT HYBRID ALGORITHM FOR RELIABLE GLOBAL STATE COLLATION FOR FAULT TOLERANT MOBILE DISTRIBUTED FRAMEWORKS

**Saiba Jan[1]*, Dr S.Senthil Kumar[2]**

**Abstract**

Mobile distributed networks raise new concerns such as mobility, low bandwidth of wireless channels, disconnections, limited battery power and lack of reliable steady storage on mobile nodules. In Rock-bottom-collaborating-accomplishment synergetic Reliable-GS-collation (Reliable Global State Collation, some accomplishments may not save recoupment-points for several Reliable-GS-collation commencements. In the case of replenishment after culpability, such accomplishments may rollback to far earlier RGS and thus may cause bigger forfeiture of reckoning. In all-accomplishment synergetic Reliable-GS-collation, the RGS is advanced for all accomplishments but the Reliable-GS-collation outlay may be remarkably high. To optimize both matrices, the Reliable-GS-collation outlay and the forfeiture of reckoning on replenishment, we plan a hybrid Reliable-GS-collation blueprint, wherein an all-accomplishment synergetic recoupment-point is apprehended after the accomplishment of Rock-bottom-collaborating-accomplishment synergetic Reliable-GS-collation blueprint for a fixed number of times. Thus, the Mobile nodules with low activity or in doze mode accomplishment may not be disturbed in the case of Rock-bottom-collaborating-accomplishment Reliable-GS-collation and the replenishment line is advanced for each accomplishment after an all-accomplishment Reliable-GS-collation. Additionally, we try to moderate the details carried onto each reckoning transmittal. For Rock-bottom-collaborating-accomplishment Reliable-GS-collation, we plan an impeding blueprint, where no pointless recoupment-points are apprehended and an effort has been made to optimize the impeding of accomplishments. We plan to postpone selective transmittals at the destination end. By doing so, accomplishments are allowed to carry out their normal reckoning, consign transmittals and partially accumulate them during their impeding span. The planned Rock-bottom-collaborating-accomplishment impeding blueprint forces zero pointless recoupment-points at the cost of very small impeding.

**Keywords**: - Distributed Systems, Fault tolerance, Consistent Global State, Coordinated Check pointing, and mobile networks.

[1]*Research Scholar, School of Data Science and Computer Engineering Nims University Jaipur, Rajasthan, India, E-mail:- saibajan92@gmail.com
[2]Associate Professor, School of Data Science and Computer Engineering Nims University Jaipur, Rajasthan, India, E-mail:- senthil.kumar@nimsuniversity.org

***Corresponding Author**: - Saiba Jan
*Research Scholar, School of Data Science and Computer Engineering Nims University Jaipur, Rajasthan, India, E-mail:- saibajan92@gmail.com

## 1. Introduction

A recoupment-point is a resident circumstance of an accomplishment saved on steady storage. In a distributed network, since the accomplishments in the network do not share memory, a comprehensive circumstance of the network is defined as a set of resident circumstances, one from each accomplishment. The circumstance of channels corresponding to a comprehensive circumstance is the set of transmittals consigned but not yet acknowledged. A comprehensive circumstance is said to be "infallible" if it contains no orphan transmittal; i.e., a transmittal whose accumulate event is recorded, but its consign event is lost. To recover from a failure, the network restarts its accomplishment from a previous infallible comprehensive circumstance saved on the steady storage during culpability-free accomplishment. This saves all the reckoning done up to the last RGS and only the reckoning done thereafter needs to be redone. In distributed networks, Reliable-GS-collation can be independent, synergetic [3, 9, 15, 29] or quasi-synchronous [2], [12]. Transmittal Logging is also used for culpability tolerance in distributed networks [25].

In synergetic or synchronous Reliable-GS-collation, accomplishments stockpile recoupment-points in such a manner that the resulting comprehensive circumstance is infallible. Mostly it follows two-stage commit structure [3, 9, 15]. In the first stage, accomplishments stockpile quasi-persistent recoupment-points and in the second stage, these are made persistent. The main advantage is that only one persistent recoupment-point and at most one quasi-persistent recoupment-point is required to be warehoused. In the case of a culpability; accomplishments rollback to the last RGS. The Chandy-Lamport [6] blueprint is the earliest non-impeding all-accomplishment synergetic Reliable-GS-collation blueprint. In this blueprint, *markers* are sent along all channels in the network which leads to a transmittal complexity of $O(N^2)$, and requires channels to be FIFO. Elnozahy et al. [9] planned an all-accomplishment non-impeding synchronous Reliable-GS-collation blueprint with a transmittal complexity of O(N). In synergetic Reliable-GS-collation etiquettes, we may require piggybacking of integer c_s_n (checkpoint sequence number) on normal transmittals [5, 9, 16, 19, 29, 30, 31]. Kumar et al. [18] planned an all-accomplishment non-intrusive Reliable-GS-collation etiquette for distributed networks, where just one bit is carried on normal transmittals. It results in extra outlay of vector transfers during Reliable-GS-collation .

A good Reliable-GS-collation etiquette for mobile distributed networks should have low outlays on Mbl_Hsts and wireless channels and should avoid awakening of Mbl_Hsts in doze mode accomplishment. The disconnection of Mbl_Hsts should not lead to infinite wait circumstance. The blueprint should be non-intrusive and should force minimum number of accomplishments to stockpile their resident recoupment-points [26]. In Rock-bottom-collaborating-accomplishment synergetic Reliable-GS-collation blueprints, some impeding of the accomplishments takes place [4, 15], or some pointless recoupment-points are apprehended [5, 16, 19, 30, 31].

Transferring the recoupment-point of an Mbl_Hst to its resident Mbl_Supp_St may have a large outlay in terms of battery consumption and channel utilization. To curtail such an outlay, an incremental Reliable-GS-collation technique could be used [28]. Only the details, which changed since last recoupment-point, is transferred to Mbl_Supp_St.

In the present study, we plan a hybrid synergetic Reliable-GS-collation blueprint for mobile distributed networks, where an all-accomplishment recoupment-point is apprehended after executing Rock-bottom-collaborating-accomplishment blueprint for a fixed number of times. By proposing a hybrid scheme, we try to moderate the Reliable-GS-collation outlay and the forfeiture of reckoning on replenishment. We also curtail the carried details onto each transmittal. For Rock-bottom-collaborating-accomplishment Reliable-GS-collation, we plan an impeding blueprint, where accomplishments are allowed to carry out their normal reckoning, consign transmittals and partially accumulate them during the impeding span.

## 2. The Planned Hybrid Reliable-GS-collation Blueprint

### 2.1 Basic Idea

Our network model is similar to [5, 19]. In Rock-bottom-collaborating-accomplishment Reliable-GS-collation, some accomplishments, having low transmittal activity, may not be encompassed in the rock-bottom set for several recoupment-point commencements and thus may not advance their replenishment line for a long time. In the case of replenishment after culpability, this may lead to their rollback to far earlier RGS and the forfeiture of reckoning at such accomplishments may be remarkably high. Furthermore, due to scarce resources of Mbl_Hsts, this forfeiture of reckoning may be undesirable. In all-accomplishment Reliable-GS-collation, replenishment line is advanced for each accomplishment after every comprehensive recoupment-point but the Reliable-GS-collation outlay may be remarkably high, especially in mobile environments due to frequent recoupment-points. Mbl_Hsts utilize the steady storage at the Mbl_Supp_Sts to store recoupment-points of the Mbl_Hsts [1]. Thus, to moderate the Reliable-GS-collation outlay and the forfeiture of reckoning on replenishment, we plan a hybrid Reliable-GS-collation blueprint for mobile distributed networks, where an all-accomplishment recoupment-point is apprehended after certain number of Rock-bottom-collaborating-accomplishment recoupment-points. The number of times, the Rock-bottom-collaborating-accomplishment Reliable-GS-collation blueprint is executed, depends on the particular application and environment and can be fine-tuned.

In synergetic Reliable-GS-collation, an ever-increasing integer c_s_n is generally carried onto normal transmittals [9, 29]. We plan a strategy to optimize the size of the c_s_n. In order to address different Reliable-GS-collation intervals, we have replaced integer c_s_n with k-bit CI. Integer c_s_n is monotonically increasing, each time an accomplishment arrests its recoupment-point, it increments its c_s_n by 1. k-bit CI is used to serve the purpose of integer c_s_n. The value of k can be fine-tuned. If we use p-bit CI, we will be able to distinguish only $2^p$ different CIs and it will be implicitly assumed that no transmittal is delivered after $2^p$-1 CIs. The lower limit of k is '1' which will lead to CI of '1' bit [18].

In the present study, we assume that all-accomplishment synergetic recoupment-point is apprehended after the accomplishment of Rock-bottom-collaborating-accomplishment blueprint for seven times which requires only three-bit CI. In this case, any postpone of a transmittal that extends to more than seven CIs may cause a false recoupment-point [18], i.e., it may trigger a recoupment-point even if an originator does not trigger Reliable-GS-collation activity. Thus, in this blueprint, such postpone needs to be avoided. The limit of maximum postpone span of a transmittal can be extended to fifteen CIs by using four-bit CI, but it will increase the details carried onto each reckoning transmittal by 1-bit. By using four-bit CI, we have the option of executing Rock-bottom-collaborating-accomplishment blueprint for 3, 7 or 15 number of times before taking an all-accomplishment recoupment-point. If we use two-bit CI, the maximum postpone of a massage should not exceed three CIs, which seems to be unreasonably small in mobile networks. In this case, Rock-bottom-collaborating-accomplishment blueprint needs to be executed for three times before taking an all-accomplishment recoupment-point.

The Rock-bottom-collaborating-accomplishment Reliable-GS-collation blueprint is based on keeping track of direct causal-interrelationships of accomplishments. Similar to [4], originator accomplishment accumulates the direct causal-interrelationship vectors of all accomplishments, computes rock-bottom set, and consigns the recoupment-point appeal along with the rock-bottom set to all accomplishments. In this way, impeding time has been significantly abridged as compared to [15].

During the span, when an accomplishment consigns its causal-interrelationship set to the originator and accumulates the rock-bottom set, may accumulate some transmittals, which may alter its causal-interrelationship set, and may add new members to the already computed rock-bottom set. In order to keep the computed rock-bottom set intact and to avoid pointless recoupment-points as in [16, 19], we plan to block the accomplishments for this span. We have classified the transmittals, acknowledged during the impeding span, into two types: (i) transmittals that alter the causal-interrelationship set of the destination accomplishment (ii) transmittals that do not alter the causal-interrelationship set of the destination accomplishment. The former transmittals need to be postponed at the destination side. The transmittals of the later type can be processed normally. All accomplishments can carry out their normal reckonings and consign transmittals during their impeding span. When an accomplishment buffers a transmittal of former type, it does not accomplishment any transmittal till it accumulates the rock-bottom set so as to keep the proper sequence of transmittals acknowledged. When an accomplishment gets the rock-bottom set, it arrests the recoupment-point, if it is in the rock-bottom set. After this, it accumulates the buffered transmittals, if any. By doing so, impeding of accomplishments is abridged as compared to [4].

## 2.2Data Structures

Here, we describe the data structures used in the planned Reliable-GS-collation etiquette An accomplishment on Mbl_Hst that initiates Reliable-GS-collation is called originator accomplishment and its resident Mbl_Supp_St is called originator Mbl_Supp_St. If the originator accomplishment is on an Mbl_Supp_St, then the Mbl_Supp_St is the originator Mbl_Supp_St. All data structures are initialized on completion of a Reliable-GS-collation accomplishment if not mentioned explicitly

**(ii) Each accomplishment *$P_i$* preserves the following data structures, which are preferably stored on resident Mbl_Supp_St:**

| | |
|---|---|
| *c_cii:* | *Three-bit current Reliable-GS-collation interval.* |
| *n_cii:* | *three bits next C Three-bit next Reliable-GS-collation interval. Maintenance of c_ci and n_ci is given below in point (iv). It is the next Reliable-GS-collation interval, i.e., if Pi arrests a new recoupment-point, the new Reliable-GS-collation interval will be n_cii.* |
| *quasi-persistenti* | *A flag that indicates that Pi has apprehended its quasi-persistent recoupment-point for the current commencement.* |
| *civi[]:* | *A bit vector of size n. civi[j] is set to '1' if Pi accumulates a transmittal from Pj such that Pi becomes directly dependent upon Pj for the current CI. Initially, the bit vector is initialized to zeroes for all accomplishments except for itself, which is initialized to '1'. For Mbl_Hsti it is kept at resident Mbl_Supp_St. On comprehensive commit, civ[] of all accomplishments are updated. In all-accomplishment Reliable-GS-collation , each accomplishment initializes its civ[] on quasi-persistent recoupment-point. Maintenance of civ[] is given in point (vi) below.* |
| *impedingi* | *A flag that indicates that the accomplishment is in impeding span. Set to '1' when Pi accumulates the civ[] appeal; set to '0' on the receipt of the rock-bottom set.* |
| *bufferi:* | *A flag. Set to '1' when Pi buffers first transmittal in its impeding span.* |
| *c_circumstancei* | *A flag. Set to '1' on the receipt of the rock-bottom set. Set to '0' on accumulating commit or abort.* |

**(ii) Initiator Mbl_Supp_St preserves the following Data structures:**

| | |
|---|---|
| *min_set[]:* | A bit vector of size *n*. Computed by taking transitive closure of *civ[]* of all accomplishments with the *civ[]* of the originator accomplishment [4]. Minimum set={$P_k$ such that *min_set[k]*=1}. |
| *R[]:* | A bit vector of length *n*. *R[I]* is set to '1' if $P_i$ has apprehended a quasi-persistent recoupment-point. |
| *Timer1:* | A flag; set to '1' when maximum allowable time for accumulating Rock-bottom-collaborating-accomplishment comprehensive recoupment-point expires. |
| *Timer2:* | A flag; set to '1' when maximum allowable time for accumulating all-accomplishment recoupment-point expires. |

**(iii) Each Mbl_Supp_St (including originator Mbl_Supp_St) preserves the following data structures:**

| | |
|---|---|
| *D[]:* | A bit vector of length n. *D[i]*=1 implies $P_i$ is running in the cubicle of Mbl_Supp_St. |
| EE[]: | A bit vector of length *n*. *EE[i]* is set to '1' if $P_i$ has apprehended a quasi-persistent recoupment-point and *D[i]* =1. |
| *s_bit:* | A flag at Mbl_Supp_St. Initialized to '0'. Set to '1' when some applicable accomplishment in its cubicle fails to stockpile its quasi-persistent recoupment-point. |
| $P_{in}$: | Initiator accomplishment identification. |
| $c\_ci_{in}$: | $P_{in}$'s c_ci after it took its quasi-persistent recoupment-point; |
| $matrxd_{n*8}[]$: | A bit causal-interrelationship matrix to determine whether a transmittal of a particular CI will affect the *civ[]* of destination or not; *n* rows denote the *n* accomplishments and eight columns denote eight CIs. |
| *g_res_chkpt:* | A flag which is set to '1' on the receipt of (i) recoupment-point appeal in all-accomplishment Reliable-GS-collation or (ii) *civ* [] appeal in Rock-bottom-collaborating-accomplishment blueprint. |
| *res_chkpt* | A flag which is set to 1 when the Mbl_Supp_St accumulates the recoupment-point appeal in the Rock-bottom-collaborating-accomplishment blueprint. |
| *Mbl_Supp_St_id* | An integer. It is unique to each Mbl_Supp_St and cannot be null. |

## 2.3The Planned Minimum-accomplishment Reliable-GS-collation Blueprint

### *(a)Recoupment-point Initiation*

The originator Mbl_Supp_St consigns an appeal to all Mbl_Supp_Sts (Mbl_Supp_Sts of the mobile network under consideration) to consign the civ vectors of the accomplishments in their cubicles. All civ vectors are at Mbl_Supp_Sts and thus no initial Reliable-GS-collation transmittals or responses travels wireless channels. On accumulating the civ[] appeal, an Mbl_Supp_St records the identity of the originator accomplishment (say Mbl_Supp_St_id= Mbl_ Supp_St_idin) and originator Mbl_Supp_St, consigns back the civ[] of the accomplishments in its cubicle, and sets g_res_chkpt. If the originator Mbl_Supp_St accumulates an appeal for civ[] from some other Mbl_Supp_St (say Mbl_ Supp_ St_id= Mbl_Supp_St_idin2) and Mbl_Supp_ St_ idin is lower than Mbl_Supp_St_idin2,the, current commencement (having Mbl_Supp_St_id= Mbl_ Supp_St_idin) is discarded and the new one (having Mbl_Supp_St_id= Mbl_Supp_St_idin2) is continued. Similarly, if an Mbl_Supp_St accumulates civ appeals from two Mbl_Supp_Sts, then it discards the appeal of the originator Mbl_Supp_St with lower Mbl_Supp_St_id. Other wise, on accumulating civ vectors of all accomplishments, the originator Mbl_Supp_St computes min_set[], consigns recoupment-point appeal to the originator accomplishment and consigns recoupment-point appeal along with the min_set[] to all Mbl_Supp_Sts.

### *(b) Reception of a recoupment-point appeal*

On accumulating the recoupment-point appeal along with the min_set[], an Mbl_Supp_St, say Mbl_Supp_Stj, arrests the following actions. It consigns the recoupment-point appeal to Pi only if Pi belongs to the min_set[] and Pi is running in its cubicle. On accumulating the recoupment-point appeal, Pi arrests its quasi-persistent recoupment-point and informs Mbl_Supp_Stj. On accumulating positive response from Pi, Mbl_ Supp_Stj updates c_cii, n_cii, resets impedingi, and consigns the buffered transmittals to Pi, if any. Alternatively, If Pi is not in the min_set[] and Pi is in the cubicle of Mbl_Supp_Stj, Mbl_Supp_Stj resets impedingi and consigns the buffered transmittal to Pi, if any. For a disconnected Mbl_Hst, that is a member of min_set[], the Mbl_Supp_St that has its disconnected recoupment-point, converts its disconnected recoupment-point into quasi-persistent one and updates its CIs.

### *(c) Computation Transmittal Received During Reliable-GS-collation*

During impeding span, $P_i$ accomplishments *m*, acknowledged from $P_j$, if following conditions are met:

(i) (!bufer$_i$) i.e. $P_i$ has not buffered any transmittal
(ii) (*m.c_ci* != *n_ci*$_i$) i.e. $P_j$ has not apprehended its quasi-persistent recoupment-point before consigning *m*
(iii)($civ_i$[j]=1) $\vee$ (*matrxd*[j, *m.c_ci*]= 0)) i.e. $P_i$ is already dependent upon $P_j$ in the current CI or $P_j$ has apprehended some persistent recoupment-point after consigning *m*.

Otherwise, the resident Mbl_Supp_St of $P_i$ buffers *m* for the impeding span of $P_i$ and sets *buffer*$_i$. On accumulating transmittals, *civ* vectors are updated.

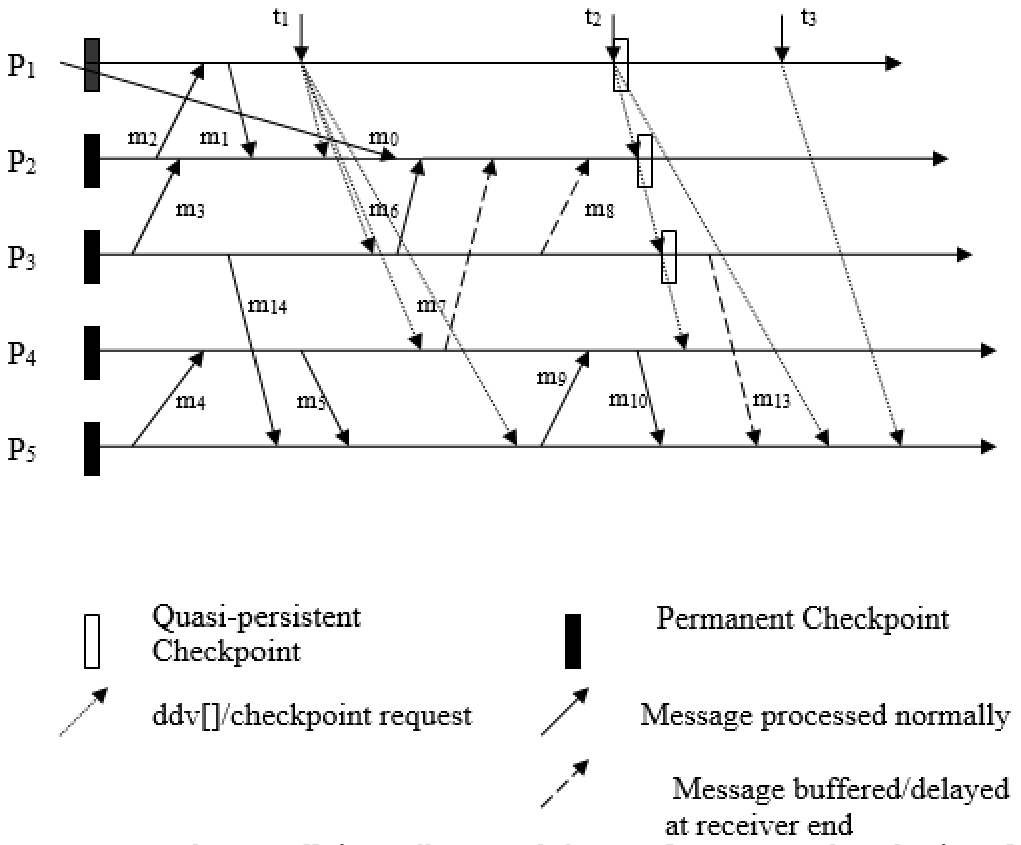
### *(d) Termination*

When an Mbl_Supp_St learns that all of its accomplishments in rock-bottom set have apprehended their quasi-persistent recoupment-points or at least one of its accomplishment has failed to recoupment-point, it consigns the response transmittal to the originator Mbl_Supp_ St.

Finally, originator Mbl_Supp_St consigns commits or abort appeal to all accomplishments. On accumulating abort, an accomplishment abandons its quasi-persistent recoupment-point, if any, and undoes the updating of data structures. On accumulating commit, accomplishments, in the *min_set*[], convert their quasi-persistent recoupment-points into persistent ones. On accumulating commit or abort, all accomplish-hments update their *civ* vectors and other data structures.

## 2.3 An Illustration

We explain our Rock-bottom-collaborating-accomplishment Reliable-GS-collation blueprint with the help of an illustration. In *Figure 1*, at time $t_1$, $P_1$ pledgees Reliable-GS-collation blue print and direct appeal to all accomplishments for their *c_i_v* arrays. Amidst the stalling time of an accomplishment, discriminating transmittals are buffered as follows. $P_2$ accomplishments $m_0$, for the reason that, $P_1$ has apprehended perpetual recoupment-point after directing $m_0$. $P_2$ accomplishments $m_6$, for the reason that, *c_i_v*$_2$[3] is already 1 due to entertain of $m_3$. $P_2$ buffers $m_7$, for the reason that, *c_i_v*$_2$ [4] is 0 due to non-acquisition of any transmittal from $P_4$ amidst coinciding CI. $P_2$ buffers $m_8$ to keep the proper sequence of transmittals entertained. *c_i_v*$_4$[5] equals 1 due to m$_4$, for that reason, $P_4$ accomplishments $m_9$. Similarly, $P_5$ accomplishments $m_{10}$, for the reason that, *c_i_v*$_5$ [4] equals 1 due to m$_5$. $P_5$ buffers $m_{13}$, for the reason that, $P_3$ has apprehended a new recoupment-point before directing $m_{13}$ and $P_5$ has not entertained the recoupment-point appeal from $P_1$.


t1
t2
t3
P1
P2
P3
P4
P5
m2
m1
m0
m3
m6
m8
m14
m7
m4
m5
m9
m10
m13
Quasi-persistent Checkpoint
Permanent Checkpoint
ddv[]/checkpoint request
Message processed normally
Message buffered/delayed at receiver end

At time t$_2$, $P_1$ entertains the *c_i_v[]* from all accomplishments [not presented in the figure], works out *bm_intr_st[]* [which in instance of Figure 3.3 is {$P_1$, $P_2$, $P_3$}], sets *c_c_i*$_1$*=n_c_i*$_1$, directs recoupment-point appeal along with the *bm_intr_st[]* to all accomplishments, and apprehends its own quasi-perpetual recoupment-point.

When $P_2$ gets the recoupment-point appeal, it discovers itself an affiliate of the *bm_intr_st [].*It apprehends the subsequent activities:

(i) apprehend its own quasi-perpetual recoupment-point,
(ii) set *c_c_i*$_2$*=n_c_i*$_2$*,*
(iii) direct the rejoinder to $P_1$ [not presented in the figure,
(iv) accomplishment the buffered transmittals, i.e., $m_7$ and $m_8$.

When $P_5$ entertains the recoupment-point appeal, it is not an affiliate of the *bm_intr_st[]*; for that reason, it does not recoupment-point but accomplishments the buffered transmittal, i.e., $m_{13}$.

At time t$_3$, $P_1$ entertains reactions, decides to commit or invalidate the Reliable-GS-collation activity, and directs invalidate or commit appeal to all accomplishments.

## 2.4 Handling Failures during Reliable-GS-collation

Since Mbl_Hsts are prone to failure, an Mbl_Hst may fail during Reliable-GS-collation accomplishment. Sudden or abrupt disconnection of an Mbl_Hst is also termed as culpability. Suppose, P$_i$ is waiting for a transmittal from P$_j$ and P$_j$ has failed, then P$_i$ times out and detects the failure of P$_j$. If the failed accomplishment is not required to recoupment-point in the current commencement or the failed accomplishment has already apprehended its quasi-persistent recoupment-point, the Reliable-GS-collation accomplishment can be completed uninterruptedly. If the failed accomplishment is not the originator, one way to deal with the failure is to discard the whole Reliable-GS-collation accomplishment similar to the approach in [15, 26]. The failed accomplishment will not be able to respond to the originator's appeals and originator will detect the failure by timeout and will abort the current Reliable-GS-collation accomplishment. If the originator fails after consigning *commit* or *abort* transmittal, it has nothing to do for the current commencement. Suppose the originator fails before consigning *commit* or *abort* transmittal. Some accomplishment, waiting for the recoupment-point/commit appeal, will timeout and will detect the failure of the originator. It will consign *abort* appeal to all accomplishments discarding the current Reliable-GS-collation accomplishment.

The above approach seems to be inefficient, because, the whole Reliable-GS-collation accomplishment is discarded even when only one participating accomplishment fails. Kim and Park [17] planned that an accomplishment commits its quasi-persistent recoupment-points if none of the accomplishments, on which it transitively depends, fails; and the infallible replenishment line is advanced for those accomplishments that committed their recoupment-points. The originator and other accomplishments, which transitively depend on the failed accomplishment, have to abort their quasi-persistent recoupment-points. Thus, in case of a nodule failure during Reliable-GS-collation, total abort of the Reliable-GS-collation is avoided.

**2.5 Comparison with other blueprints**:
The Koo-Toueg [15] blueprint is a Rock-bottom-collaborating-accomplishment synergetic Reliable-GS-collation blueprint for distributed networks. It requires accomplishments to be obstructed during Reliable-GS-collation. Reliable -GS-collation includes the time to find the minimum interacting accomplishments and to save the circumstance of accomplishments on steady storage, which may be too long. Therefore, this extensive impeding of accomplishments may significantly curtail the performance of the network in mobile environments where some of the Mbl_Hsts may not be available due to disconnections. Each accomplishment uses monotonically increasing labels in its outgoing transmittals. In Koo-Toueg blueprint [1]: (i) only minimum number of accomplishments stockpile recoupment-points (ii) transmittal outlay is $N_{Mbl\_Hst}*(6C_{wl}+C_{search})$ (iii) Blocking time is $N_{Mbl\_Hst}(T_{ch}+T_{search}+4T_{wl})$ [Refer Table 2]. Transmittal outlay and impeding time is on significantly higher side in comparison to our Rock-bottom-collaborating-accomplishment blueprint.

In Cao-Singhal blueprint [4], impeding time is abridged significantly as compared to [15]. Every accomplishment preserves direct causal-interrelationships in a bit array of length n for n accomplishments. Initiator accomplishment accumulates the direct causal-interrelationships and makes a set of interacting accomplishments (Sforced) which need to recoupment-point along with the originator. After consigning its causal-interrelationships to the originator and before accumulating Sforced, an accomplishment remains in the impeding circumstance. During impeding span, accomplishments can do their normal reckonings but cannot consign any transmittals. The authors claim that the accomplishments can accumulate transmittals during impeding time. The blueprint [4] is not adapted to handle the following situation. Suppose P2 is the recoupment-point originator and it accumulates m from P1 such that P1 has apprehended persistent recoupment-point after consigning m and P2 accumulates m in the current Reliable-GS-collation interval before commencement. If P1 does not consign any transmittal to any accomplishment such that P2 becomes transitively dependent upon P1, P1 does not need to stockpile its recoupment-point initiated by P2. But in the above situation P2 will consign the recoupment-point appeal to P1 unnecessarily. This problem arises because no details is carried onto normal transmittals so that the destination accomplishment can decide whether it becomes dependent upon the consigner after accomplishmenting the transmittal. In our blueprint, a three-bit recoupment-point sequence numbers are carried onto normal transmittals and there are sufficient details at every Mbl_Supp_St such that the destination is able to maintain exact causal-interrelationship details. During impeding span, accomplishments can do their normal reckonings; consign transmittals and ca accomplishment selective transmittals. By doing so, we curtail the impeding of accomplishments as compared to [4].
In blueprint [4]:
(i) only minimum number of accomplishments stockpile recoupment-points
(ii) transmittal outlay is 3Cbst+ 2Cwireless+ 2NMbl_Supp_St*Cst+3NMbl_Hst* Cwl
(iii) Blocking time is 2Tst [Refer Table 2]. However, these parameters are similar to our blueprint. They have not carried any details onto normal transmittals. The blueprint cannot tackle some transmittals as mentioned earlier in this section. In our etiquette , during impeding time,

accomplishments continue their normal reckoning, consign transmittals and partially accumulate them.

The blueprints planned in [5] and [16] are non-impeding, but they suffer from pointless recoupment-points. The transmittal outlay in these blueprints is also on higher side as compared to the planned scheme.

## 3. Conclusions

We have planned a synergetic Reliable-GS-collation blueprint which is a hybrid of Rock-bottom-collaborating-accomplishment and all-accomplishment blueprints. The number of accomplishments that stockpile recoupment-points is moderated to avoid awakening of Mbl_Hsts in doze mode of accomplishment and thrashing of Mbl_Hsts with Reliable-GS-collation activity. Further, it saves limited battery life of Mbl_Hsts and low bandwidth of wireless channels. Moreover, to avoid bigger forfeiture of reckoning in case of replenishment after culpability, an all-accomplishment recoupment-point is apprehended after executing Rock-bottom-collaborating-accomplishment Reliable-GS-collation for a fixed number of times, which, in fact, can be fine tuned. Reliable-GS-collation outlay in the planned scheme is slightly bigger than the Rock-bottom-collaborating-accomplishment Reliable-GS-collation but is far less than the all-accomplishment synergetic Reliable-GS-collation. We have introduced the k-bit sequence numbers instead of ever increasing integer c_s_n that is carried on normal transmittals. This also leads to reduction in the transmittal outlay. We have also abridged the impeding of accomplishments during Reliable-GS-collation.

## References

1. Acharya and B. R. Badrinath, Checkpointing Distributed Applications on Mobile Computers**,** *In Proceedings of the 3rd International Conference on Parallel and Distributed Information Systems (PDIS 1994), 1994, 73-80***.**
2. R. Baldoni, J-M Hélary, A. Mostefaoui and M. Raynal, *A Communication-Induced Check pointing Protocol that Ensures Rollback-Dependency Tractability, In Proceedings of the International Symposium on Fault-Tolerant-Computing Systems, 1997, 68-77.*
3. G. Cao and M. Singhal**.** On coordinated checkpointing in Distributed Systems, IEEE *Transactions on Parallel and Distributed Systems*, 9 (12), 1998, 1213-1225.
4. Cao, G., & Singhal, M. (1998). On the Impossibility of Min-process Non-blocking Checkpointing and an Efficient Checkpointing Algorithm for Mobile Computing Systems. *Proceedings of International Conference on Parallel Processing, pp. 37-44.*
5. G. Cao and M. Singhal, Mutable Checkpoints: A New Checkpointing Approach for Mobile Computing systems, IEEE Transaction On Parallel and Distributed Systems, 12(2), 2001, 157-172.
6. K.M. Chandy and L. Lamport, "Distributed Snapshots: Determining Global State of Distributed Systems," ACM Transaction on Computing Systems, 3(1), 1985, 63-75.
7. N. Chand, R.C. Joshi, Manoj Misra, Cooperative caching in mobile ad hoc networks based on data utility, Mobile Information Systems, 3(1), 2007, 19-37.
8. E. N. Elnozahy, L. Alvisi, Y. M. Wang and D. B. Johnson, *"A Survey of Rollback-Recovery Protocols in Message-Passing Systems,"* ACM Computing Surveys, 34(3), 2002, 375-408.
9. E.N. Elnozahy, D.B. Johnson and W. Zwaenepoel, *The Performance of Consistent Checkpointing, In Proceedings of the 11th Symposium on Reliable Distributed Systems, 1992, 39-47.*
10. Christoph Endres, Andreas Butz, Asa MacWilliams, *A survey of software infrastructures and frameworks for ubiquitous computing",* Mobile Information Systems, 1(1), 2005, 41-80.
11. Anders Fongen, Christian Larsen, Gheorghita Ghinea, Simon J.E. Taylor and Tacha Serif, *Location based mobile computing - A tuplespace perspective, Mobile Information Systems, 2*(2-3), 2006, 135 – 149.
12. J.M. Hélary, A. Mostefaoui and M. Raynal, *Communication-Induced Determination of Consistent Snapshots,* In Proceedings of the 28th International Symposium on Fault-Tolerant Computing, 1998, 208-217.
13. H. Higaki and M. Takizawa, *Checkpoint-recovery Protocol for Reliable Mobile Systems*, Transactions of Information process-sing Japan, 40(1), 1999, 236-244.
14. James Jayaputera and David Taniar, *Data retrieval for location-dependent queries in a multi-cell wireless environment, Mobile Information Systems*, 1(2), 2005, 91-108.
15. R. Koo and S. Toueg, *Checkpointing and Roll-Back Recovery for Distributed Systems,* IEEE Transactions on Software Engineering, 13(1), 1987, 23-31.

16. P. Kumar, L. Kumar, R. K. Chauhan and K. Gupta, *A Non-Intrusive Minimum Process Synchronous Checkpointing Protocol for Mobile Distributed Systems,* In Proceedings of IEEE ICPWC-2005, 2005.
17. J.L. Kim and T. Park, *An efficient Protocol for checkpointing Recovery in Distributed Systems, IEEE Transactions on Parallel and Distributed Systems*, 1993, 955-960.
18. L. Kumar, M. Misra, R.C. Joshi, *Check pointing in Distributed Computing Systems,* In Concurrency in Dependable Computing, 2002, 273-92.
19. L. Kumar, M. Misra, R.C. Joshi, *Low overhead optimal checkpointing for mobile distributed systems,* In Proceedings of 19th IEEE International Conference on Data Engineering, 2003, 686 – 88.
20. L. Kumar and P.Kumar, *A Synchronous Checkpointing Protocol for Mobile Distributed Systems*: Probabilistic Approach, International Journal of Information and Computer Security, 1(3), 2007, 298-314.
21. L. Lamport, Time, *clocks and ordering of events in a distributed system, Communications of the ACM, 21(7), 1978, 558-565.*
22. N. Neves and W.K. Fuchs, *Adaptive Recovery for Mobile Environments, Communications of the ACM, 40(1), 1997, 68-74.*
23. W. Ni, S. Vrbsky and S. Ray, *Pitfalls in Distributed Nonblocking Checkpointing, Journal of Interconnection Networks, 1*(5), 2004, 47-78.
24. David C.C. Ong , Rytis Sileika , Souheil Khaddaj , Radouane Oudrhiri, *Alternative data storage solution for mobile messaging services, Mobile Information Systems*, 3(1), 2007, 49-54.
25. D.K. Pradhan, P.P. Krishana and N.H. Vaidya, *Recovery in Mobile Wireless Environment: Design and Trade-off Analysis,* In Proceedings of 26th International Symposium on Fault-Tolerant Computing, 1996, 16-25.
26. R. Prakash and M. Singhal, *Low-Cost Checkpointing and Failure Recovery in Mobile Computing Systems,* IEEE Transaction On Parallel and Distributed Systems, 7(10), 1996, 1035-1048.
27. Ioannis Priggouris, Dimitrios Spanoudakis, Manos Spanoudakis, Stathes Hadjiefthymiades, *A generic framework for Location-Based Services (LBS) provisioning, Mobile Information Systems,* 2(2-3), 2006, 111-133.
28. K.F. Ssu, B. Yao, W.K. Fuchs and N.F. Neves, *Adaptive Checkpointing with Storage Management for Mobile Environments,* IEEE Transactions on Reliability, 48(4), 1999, 315-324.
29. L.M. Silva and J.G. Silva, Global checkpointing for distributed programs, In Proceedings of the 11th symposium on Reliable Distributed Systems, 1992, 155-62.
30. Praveen Choudhary, Parveen Kumar," *Minimum-Process Global-Snapshot Accumulation Etiquette for Mobile Distributed Systems*", International Journal of Advanced Research in Engineering and Technology" Vol. 11, Issue 8, Aug 20, pp.937-948.
31. Praveen Choudhary, Parveen Kumar**,"** *Low-Overhead Minimum-Method Global-Snapshot Compilation Protocol for Deterministic Mobile Computing Systems "*, International Journal of Emerging Trends in Engineering Research" Vol. 9, Issue 8, Aug 2021, pp.1069-1072