Regulating Bus Management System Using Cloud Platform

Ranjith Ramesh, Yokesh Ezhilarasu, Prasanna Ravichandran, and Soma Prathibha, *Member, IACSIT*

***Abstract*—Public transport especially buses are getting crowded day-by-day due to heavy demand of transport facility. More over the frequency of the buses are unregulated. Either the buses lines up at one time or buses get delayed for a long time. This kind of chaos is mainly due to irregular Planning of bus intervals and not knowing the details of the amount of passengers expected at a time. Hence we are proposing a system in which the number of passengers in a bus stop can be calculated and the bus service can be regulated depending on the passenger's arrival. Cloud is the best platform to implement this system as the storage is dynamic in cloud and interface can be easily provided to people using Iaas. In this paper we mainly concentrate on the scheme of the proposed system and resource allocation in cloud using Gossip protocol.**

***Index Terms*—Gossip protocol, GPS system, infrastructure as a service, resource allocation, windows azure.**

I. Introduction

The existing system of our public transit system has not been properly scheduled which leads to overcrowding of passengers in buses. This problem of crowding is mainly due to the unplanned bus management system. Our present Bus transport system does not accounts the total number of passengers in the bus stop. This paper deals briefly about calculating the total number of passengers in a stop and regulate the bus service accordingly using dynamic resource allocation in cloud. This can be implemented using Windows Azure. Azure has capacity to handle data outburst. The development fabric and development storage helps in creating applications in cloud. Gossip protocol can also be inculcated while coding the application.

II. Existing System

There are many tracking system systems to monitor the traffic flow like, taxi tracking in Melbourne, Brisbane and Adelaide used to get the closest free taxi to a waiting customer. Bus Tracking systems are already in use in Perth and Adelaide, but they are used to help customers know when bus is due to arrive.

The public transport management system [1] which is currently used indicates the delay in bus timings during any natural calamities, indicates when the bus load is full. This system uses management and fusion software to monitor this system. This method does not take in to account the number of passengers boarding and departing in the initial stage and hence regulating the transport becomes a difficult task.

The Transit Management System [2] is used in advanced public transport system. They mainly focus on Fleet Management, Traveler Information, Electronic Fare Payment, and Transportation

Demand Management. The automatic vehicle identification helps only in monitoring the vehicle and not the passengers count.

The Transport Asset Management System [3] has the great disadvantage of manual entry of the data which is prone to high error in data entered.

Bus management system with Comprehensive CAD/AVL Passenger information system[4] satisfies the major need of intimating and monitoring the bus routes in an efficient way but they mainly suffer from high cost and they fails in performance when the population is more.

All the above disadvantages can be sorted out by the system which uses cloud computing to monitor passenger population and regulate the frequency of the buses.

III. Proposed System

The proposed system has the following objectives

- To find the passenger population in bus stops using message service and online bus pass.
- Regulate the bus frequency depending on the passenger population
- Create software in cloud to calculate the passenger population and regulate the buses by sending more buses to highly populated stops and reducing the bus flow to less populated stops by intimating the transport authority.
- Allocating the resources properly in cloud by using gossip protocol.
- Using GPS system with cloud to find the delay in buses and indicate the passengers through their cell phone

The following algorithm is followed to calculate the passenger count in a stop.

- Bus pass holders register their source and destination in online through the interface provided to them *as shown in Fig 1.*
- Ad-hoc users are requested to send a message with the source, destination and time frame prior to the time of journey to the cloud software
- The software in the cloud evaluates the request and intimates the passenger with the bus number
- If the passenger goes late to the stop or early to the stop the passenger is requested to send a message to

Manuscript received August 7, 2012; revised September 18, 2012.

Ranjith Ramesh, Yokesh Ezhilarasu, Prasanna Ravichandran, Soma prathibha are pursuing his B.Tech Information Technology in Sri Sairam Engineering College affiliated to Anna University, Chennai (email: ranjith.8593@gmail.com, yokesh93@gmail.com, prasanna.it92@gmail.com, somaprathi25@gmail.com)

the cloud service. The software will intimate the next possible bus timings.

- By this method the amount of passengers in the bus stop can be calculated and the buses can be regulated
- By using GPS system the delay in buses can be intimated to the passengers.

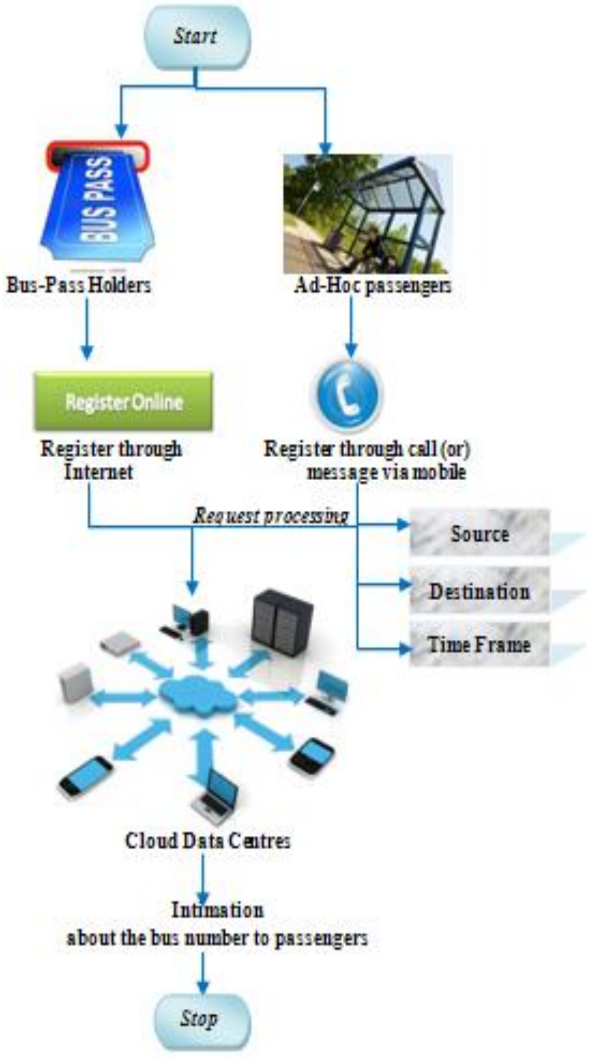

Fig. 1. Flow chart of working of proposed system

The following algorithm is incorporated to intimate the passenger with the bus number and regulate the buses based on population at a stop as shown in Fig. 2.

- The Software in the Cloud Middleware gets the data of source, destination and time frame request from the passengers and calculate the number of boarding and departing passengers at each stop based on the data present in the tables as shown in Table. I which is updated automatically based on the request of Ad-Hoc passengers and bus-pass holders during registration.
- Then, the system checks whether there is any empty seats available in the bus based on the conductor regulation at each stop.
- If the seats were empty, The regular schedule of buses is followed.
- Else, the system instructs the addition of new bus to the route to the Transport Department.
- The passengers are the intimated with the bus number after validation of the seat availability in the bus through a particular route.

TABLE I: SQL DATABASE OF IN AND OUT PASSENGERS AT EACH STOP

Bus Stop	In	Out
X	20	5
Y	10	25

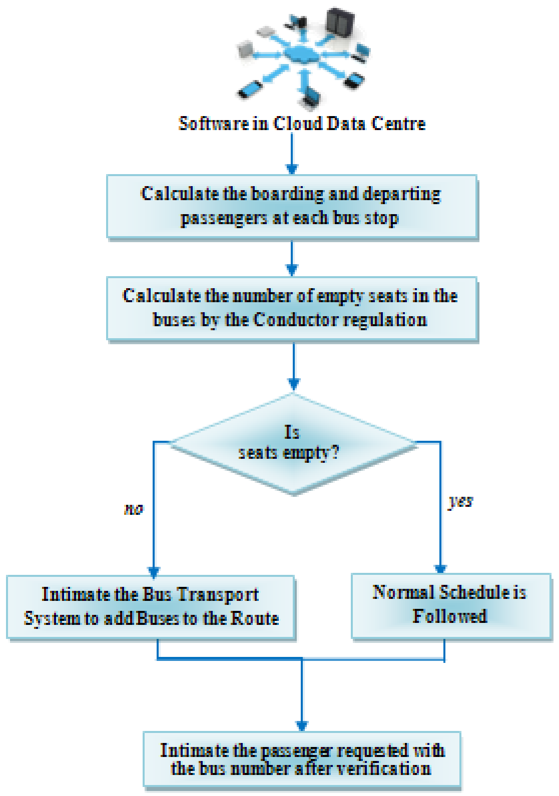
Software in Cloud Data Centre
Calculate the boarding and departing passengers at each bus stop
Calculate the number of empty seats in the buses by the Conductor regulation
Is seats empty?
no
yes
Intimate the Bus Transport System to add Buses to the Route
Normal Schedule is Followed
Intimate the passenger requested with the bus number after verification

Fig. 2. Function of software in cloud data centre

IV. GOSSIP PROTOCOL–DYNAMIC RESOURCE ALLOCATION

A *gossip protocol* has the structure of a round based distributed algorithm [5]. When executing a round-based gossip protocol, each node selects a subset of other nodes to interact with based on distance between the nodes dynamically. Nodes interact via 'small' messages, which are processed and trigger local state changes. Node interaction with the Gossip protocol follows the push-pull paradigm, whereby two nodes exchange state information (gossiping its states information to other node) as shown in Fig .3.

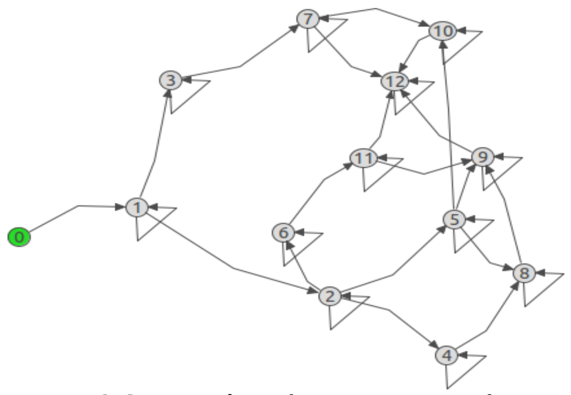

Fig. 3. Gossiping of states between requesting nodes

In the proposed system, we split a big region (for eg. Chennai) into tiny areas(like North Chennai) and each area has a local host(node) which updates its state for each request it handles from users. *In the above diagram*, each circle depicts a node (local host) maintained for each area. The hosts exchange information between themselves during each request thus it reduces the over head on cloud data centre. Compared to alternative distributed solutions, gossip-based protocols tend to be simpler, more scalable and more robust. This type of protocol runs on all machines of the cloud. More precisely, it executes in the resource manager components of the cloud middleware architecture. Initially, the resource manager maintains a *configuration matrix* with the CPU and Memory Demand. Every node connected to the cloud runs the Gossip protocol to process simultaneous request and adapt dynamically to different inputs. The cost of reconfiguration is maintained low to increase the cloud utility. At first, the node *p* executes its *active thread* .This thread chooses a node q dynamically in a periodic time interval, such that the cost of reconfiguration is low .Then, the node *p* send its state *sp* to selected node and in turn the selected node sends its state to the node *p*. There is also a *passive thread* which is run during an incoming request. In which the node receives the state of some other node *sq* and then it sends its status *sp* to the node that is in communication if the request is satisfied. Then after the threads completion, the relative demand between two nodes is equalized by averaging their local states.

After equalization of relative demands, the configuration matrix is updated and thus both the requests of nodes are processed simultaneously and successfully.

The below algorithm shows the execution of the active and passive thread of a generic Gossip protocol given in [6] - executed by node p:

```
ACTIVE THREAD:
do once every δ time units
q= getPeer(state)
sp =prepareMsg(state,q)
send (REQ, sp, p) to q

PASSIVE THREAD:
do forever
receive (t, sq, q) from *
if (t=REQ) then
sp =prepareMsg (state,q)
send ( REP, sp, p) to q
state =update (state, sq)
```

V. Importance of Gossip Protocol in the Proposed System

As the number of people who use the public transit system [1] is very high, it is not preferable to go for ordinary platform like database, networking, e-reservation etc. Database becomes insignificant for a large number of users. Networking and e-reservation fails on the fact that there is a possibility of the server crash. Instead, we could go for *Cloud environment* which allows dynamic resource management. In this paper, we made use of Gossip protocol which ensures,

1) Resource allocation to all people simultaneously.
2) Adapting to the various requests from different users simultaneously.

There may be a numerous number of users requesting for the bus number with source and destination details. In such a case, cloud middleware process the request of many users simultaneously based on the demand of the available resources and the information is updated to the data present in the cloud data centers through the instance created for each request. By this mechanism, we can intimate the user immediately with the bus details without any delay even in the situation of high traffic so that the passengers get boarded and reach their destination on time. This protocol enables us to allocate the resources to the user based on the demand. As the user request changes dynamically, demand for the resources in cloud also varies accordingly. So each time the resources are allocated, the resource manager component of the cloud middle ware maintains information of the CPU and memory demand through a *configuration matrix*. Each time the user request for information, it computes the new configuration matrix for that node p. The instance is created for processing the request and Gossip protocol executes the *active thread*, which select a random node q from a set of nodes that are in the local view of p dynamically continuously after certain time stamp and sends its state *sp*. Gossip protocol uses the *passive thread* to process the incoming request from another node near-by.

After completion of threads, the relative demand is equalized through averaging their local states or shifting the relative demand between communicating nodes and the configuration matrix is updated. After updating the data, the instance is removed and memory space is freed up for other requests. This process of *node-to-node communication* reduces the cost of reconfiguration and processing simultaneous requests efficiently, thus maximizing the cloud utility. Gossip protocol is mainly used because it equalizes relative demand between two nodes and leaving way for other queued requests. In this system, instances are created only after verifying whether the cost of reconfiguration is less or not and the instances executes *gossip protocol* satisfying the dynamic resource allocation and updates the configuration matrix maintained by resource manager after sending an intimation to the user with the bus number based on the query i.e. source, destination and timing. As soon as the intimation is sent, the instance is deleted so that memory is freed up for other user request that is queued.

VI. Implementation of Proposed System

We made use of Cloud Sim tool for the implementation of the proposed system. The below java coding is an illustration of how the request is taken to the datacenter and then the resource is allocated dynamically in cloud.

The following code is used to take the request to the data center and evaluate it.

```
Protected void Characteristics resources (SimEvent ev) {DatacenterCharacteristicscharacteristics=(DatacenterCharacteristics)ev.getData();getDatacenterCharacteristicsList().put(characteristics.getId(),characteristics);if(getDatacenterCharacteristicsList().size()==getDatacenterIdsList().size()){setDatacenterRequestedIdsList(newArrayList<Integer>());createVmsInDatacenter(getDatacenterIdsList().get(0));} }
```

Finally the request is been submitted with the data center id to which it has been allocated

```
protected void processCloudletReturn(SimEvent ev) {Cloudletcloudlet=(Cloudlet)ev.getData();getCloudletRecei
```

vedList().add(cloudlet);Log.printLine(CloudSim.clock()+":" +getName()+":Cloudlet"+cloudlet.getCloudletId()+"received");cloudletsSubmitted--;if(getCloudletList().size()==0&&cloudletsSubmitted==0){***//allcloudletsexecuted***Log.printLine(CloudSim.clock()+":"+getName()+":All Cloudlets executed. Finishing...");clearDatacenters();finishExecution();}else{***//some cloudlets haven't finished yet*** if (getCloudletList().size()>0 && cloudletsSubmitted==0) {**//all the cloudlets sent finished. It means that some bount //cloudlet is waiting its VM be created** clearDatacenters(); createVmsInDatacenter(0);}} }

VII. Conclusion

Thus, the proposed system has high efficiency in allocating the resources and monitoring the passenger population without affecting the efficiency and in a cost effective manner by using cloud computing and gossip algorithm for dynamic resource allocation. It is advantageous from current Bus Management System in the sense that the frequency of buses are regulated based on the demand of passengers through a particular route at each stop i.e.., frequency of buses is more through the route which has more passengers at each bus stop as depicted by the Fig.4 below.

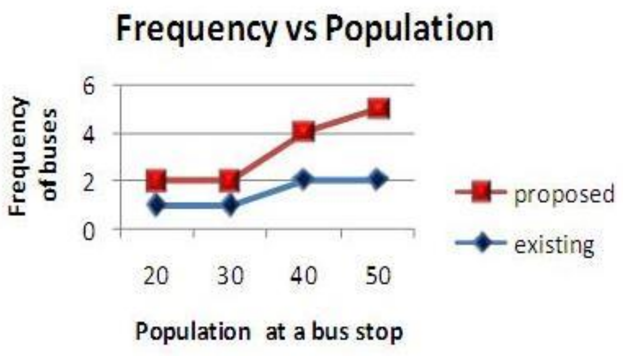

Fig. 4. Graph plotting the frequency of buses VS the population at a stop

References

[1] C. Rizos, *Public Transport Management System*.
[2] “Advanced Transportation Management Technologies-Transit Management system.”
[3] J. Smith and A. Ruffle, *The Transport Asset Management System*.
[4] Mentoring. [Online]. Available: mentoreng.com/solutions/fix-route/index.htm
[5] F. Wuhib, R. Stadler, and M. Spreitzer, “A Gossip Protocol for Dynamic Resource Management in Large Cloud Environments.”
[6] A. Montresor and G. Paolo, “Gossip protocols for large-scale distributed systems ."

Ranjith Ramesh is born in Chennai on May 8 1993. He did his schooling in D.A.V Higher Secondary School and pursuing his B. Tech Information Technology in Sri Sairam Engineering College, Chennai, India. The author is interested in Cloud computing and currently working on this field to carry out further research. He had published a paper in Research Publications. He is currently a member of IACSIT and IAENG. The author is planning to do his further studies in cloud computing**.** The author is interested in debating and writing articles. He is currently authoring a book named “The Malice”.

Yokesh Ezhilarasu is born in Madurai on May 22 1993. He did his schooling in Velammal Matriculation Higher Secondary and pursuing his pre-final year of B. Tech Information Technology in Sri Sairam Engineering College, Chennai, India. The author is interested in Cloud computing, Micro Processors .He is quite crazy and interested towards mathematical concepts in each subject dealt in his undergraduate course. He is looking forward to do future research on Cloud Computing. The author is planning to do his further studies on Cloud computing / Programming.

Prasanna Ravichandran was born on November 24, 1992, in Chennai. He did is schooling in Sita Devi Garodia, Chennai and now he pursuing his pre-final year B. Tech Information Technology in Sri Sairam Engineering college, Chennai. He interested in the field of Cloud computing and also in the field of Networks. He published his Clouding paper in Research Publication.

Soma Prathibha is currently working as an associate professor in the Department of Information Technology at Sri Sairam Engineering College. She did her Post Graduation in Computer Science at Sathyabama University, Tamil Nadu and currently pursuing her PhD in Resource allocation in cloud computing in Anna University, Chennai. She has completed her undergraduate degree in Computer Science at Vijaya nagar Engineering college, Bellary, Karnataka .She is an expert in programming concepts and highly interested in cloud computing**.** She has an total of ten years of teaching experience. She has attended 4 workshops and has published papers in various journals and conferences. She was awarded many times for producing hundred percent results.