ISSN:0975-9646

The Study of Various Server Architectures for Mobile Presence Services in Social Network Applications

Jareena Shaikh[*], Bhandari G. M.[**]

[*] *Computer Engg. Department, Bhivrabai Sawant Polytechnic*
[**] *Head of Department, Bhivrabai Sawant Institute of Technology & Reseach*

***Abstract*- Now a day's Social network applications are becoming increasingly popular on mobile devices. There are various mobile presence services for social network applications. Presence is a service that allows a user to be informed about the reachability, availability, and willingness to communicate of another user. This paper represents the comparative study of IMS presence server and PresenceCloud server. We analyze the performance of IMS presence server and PresenceCloud in terms of the search cost and search satisfaction level. The search cost is defined as the total number of messages generated by the presence server when a user arrives; and search satisfaction level is defined as the time it takes to search for the arriving user's friend list. The results of simulations demonstrate that PresenceCloud achieves performance gains in the search cost without compromising search satisfaction.**

***Index Terms*- Social networks, mobile presence services, distributed presence servers, cloud computing, PresenceCloud, IMS**

I. INTRODUCTION

Mobile devices and cloud computing environments can provide presence-enabled applications, i.e., social network applications/services, worldwide. Facebook [1], Twitter [2], Foursquare [3], Google Latitude [4], buddycloud [5] and Mobile Instant Messaging (MIM) [6], because of ubiquity of Internet. These presence-enabled applications are grown rapidly in the last decade.

Mobile network servives exploit the information about the status of participants including their appearances and activities to interact with their friends. Now a day's because of the wide availability of mobile, social network services enable participants to share live experiences instantly across great distances. Presence is a service that allows a user to be informed about the specified state of another user. The specified state, such as online/ offline status, disposition (out to lunch, away from the computer), activity status (busy, idle), mood (happy, sad) and location of user, reflects the user's accessibility, availability and will to communicate. Presence has become a key enabler for next-generation services such as push-to-talk (PTT), instant messaging (IM) and web2.0, which have facilitated communications among communities of interest, such as groups of friends, colleagues working on the same projects and families [1,2]. There are four fundamental entities in a presence service [3–5]: a principal, a presentity, a watcher and a presence server, which may exist independently or as part of application servers (e.g., PTT, IM and Web2.0). A principal refers to a user who uses presence service and is the owner of presentities or watchers; a presentity isan entity that is capable of providing state information to presence server; a watcher is an entity that subscribes to or requests the state information about a presentity; and a presence server is a network entity which has three main responsibilities: managing the subscription relationships between watcher and presentity; keeping the latest presentity state; and notifying corresponding watchers when the presentity state is updated. A mobile presence service is an essential component of social network services in cloud computing environments. The key function of a mobile presence service is to maintain an up-to-date list of presence information of all mobile users.

The service must also bind the user's ID to his/her current presence information, as well as retrieve and subscribe to changes in the presence information of the user's friends. In social network services, each mobile user has a friend list, typically called a buddy list, which contains the contact information of other users that he/she wants to communicate with. The mobile user's status is broadcast automatically to each person on the buddy list whenever he/she transits from one status to the other. For example, when a mobile user logs into a social network application, such as an IM system, through his/her mobile device, the mobile presence service searches for and notifies everyone on the user's buddy list. To maximize a mobile presence service's search speed and minimize the notification time, most presence services use server cluster technology [7]. Currently, more than 500 million people use social network services on the Internet [1]. Given the growth of social network applications and mobile network capacity, it is expected that the number of mobileIn the last decade, many Internet services have been deployed in distributed paradigms as well as cloud computing applications. For example, the services developed by Google and Facebook are spread among as many distributed servers as possible to support the huge number of users worldwide. Thus, we explore the relationship between distributed presence servers and server network topologies on the Internet, and propose an efficient and scalable server-to-server overlay architecture called PresenceCloud to improve the efficiency of mobile presence services for large-scale social network services.

Billion shared items every month and Twitter receives more than55million tweets each day. In the future, mobile devices will become more powerful, sensing, and media capture devices. Hence, we believe it is inevitable that

social network services will be the next generation of mobile Internet applications. A mobile presence service is an essential component of social network services in cloud computing environments. The key function of a mobile presence service is to maintain up-to-date list of presence information of all mobile users. The presence information includes details about a mobile user's location, availability, activity, device capability, and preferences. The service must also bind the user's ID to his/her current presence information, as well as retrieve and subscribe to changes in the presence information of the user's friends. In social network services, each mobile user has a friend list, typically called a buddy list, which contains the contact information of other users that he/she wants to communicate with.

The remainder of this paper is organized as follows. The next section contains a design of IMS Presence Server. The design of PresenceCloud server is presented in section 3. Section 4 contains some concluding remarks.

II. Design of IMS Presence Server

Presence architecture is shown in Figure2.It has three levels: agents, entities, and a server. The agents collect information from various sources. The agents are various programmes.

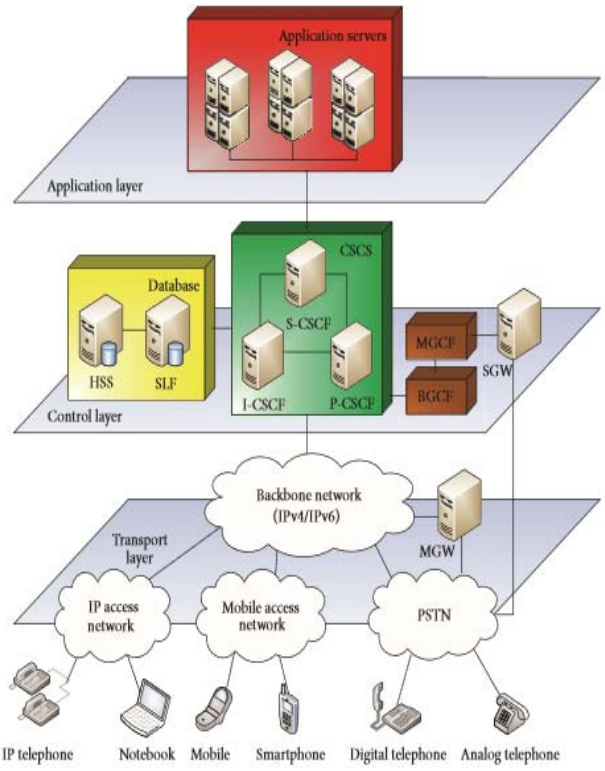

Fig. 1.Architecture of IMS

The presence user agent collects information from user devices. Presence network agent collects information from network elements. Presence external agent collects information from other networks. Watcher presence agent provides information to the watcher.

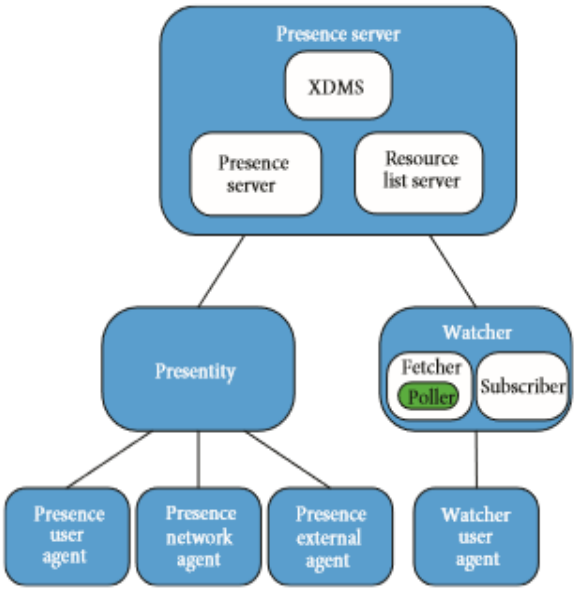

Fig. 2. Presence Architecture

Entities are characterized by the fact that they can process the SIP messages (UE,S-CSCF, and AS). Entities are divided into two types. Presentity (presence entity) provides information about itself and the watcher observes the status of the others. Watchers are divided into three groups. The fetcher is only interested in the current status. Poller is a special kind of Fetcher, which observes status in certain time intervals. The subscriber also observes the changes in the presence of entities [13]. The server collects and sends information about users, which is stored in XML documents. The presence server receives messages and assigns it to the correct user. The resource list server creates lists of users for watchers and sends their status together. The XML document manager server (XDMS) supports other parts of the presence server. For example, XDMS knows that the watcher is authorized to observe the presence entity. Application server is designed so that it could control the number of messages. One of the possibilities is periodic sending of messages. If there are more messages than the server can send, it puts them into a waiting queue. If the waiting queue is full, then the server deletes the messages[14].

2.1. Communication.

There are two processes of exchanging messages in the presence service. Process of publishing is shown in Fig. 3. This exchange of messages has two parts. The first is registration (messages 1–20) and the second is publication of status (messages 21–32). S-CSCF is assigned to the user during registration. User equipment (UE) is a telephony device, which enters IMS through P-CSCF. P- CSCF through I-CSCF determines where to send the Register request. The information about where to send the message and about the user profile for S-CSCF is stored in the HSS.

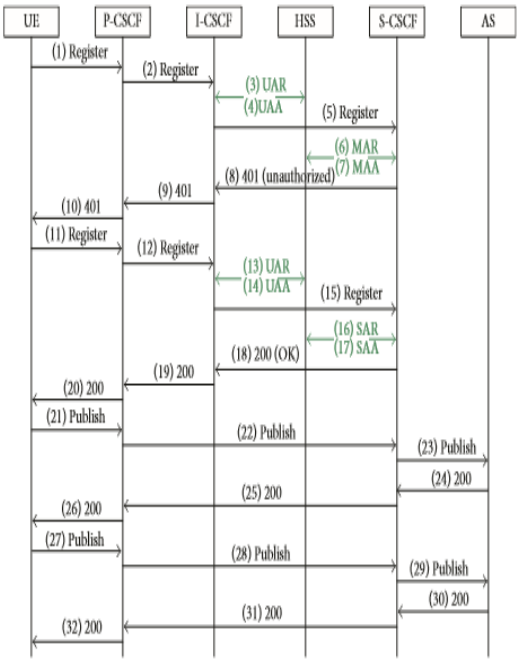

Fig. 3. Publication Status

First, S-CSCF sends a response 401(unauthorized). After receiving answers, UE creates another Register request, after which the user will have successfully registered. A detailed description of the registration is in [15]. In messages 21–23 UE (presentity) sends its whole status in request Publish to the application server (presence server). Messages pass only through P-CSCF and S-CSCF after the registration. S-CSCF knows where the server is according to the initial filter a criteria (iFC). The filter is obtained from the HSS during the registration.

Presence server sends confirmation message 200(OK) as soon as possible, to prevent resending messages. When changing status, UE sends another request Publish, which will go the same way as the first one. The form of the messages is described in[16].The message itself contains only the change of the status.

The process of subscribing is shown in Fig 4. The figure describes a situation, when the watcher is in another IMS network like presence server. Process of registration is the same as in the previous figure and therefore is not listed. Entry Point is I-CSCF to another IMS network. UE(watcher) creates request Subscribe. The filter is in the request [17]. In the filter, there is information about what the watcher wants to know. UE enters into its IMS network through P-CSCF. It continues to S-CSCF. S-CSCF sends subscribe from the watcher presence network to the presentity presence network. I-CSCF finds S-CSCF and S-CSCF sends request to AS, where there is a list of contacts with status.

Upon receiving the request, the application server verifies the user's authority. If it is correct, the application server sends response 200 (OK). AS sends request Notify with the body where it contains the information about the presentity status. Type of watcher is Subscribe in Figure4.

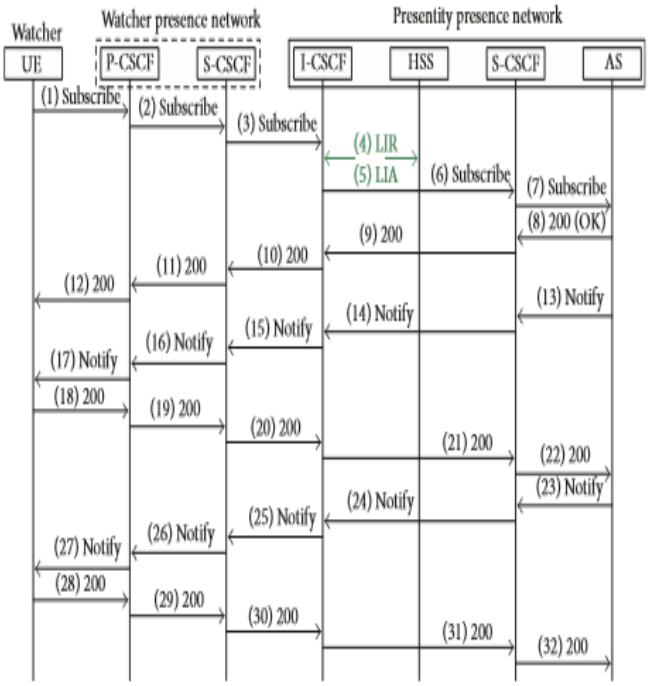

Fig. 4.Subscribe status.

If one of the presentity, which the watcher observes, changes its state, server sends another Notify message without request.

III. DESIGN OF PRESENCE CLOUD SERVER

The past few years has seen a veritable frenzy of research activity in Internet-scale object searching field, with many designed protocols and proposed algorithms. Most of the previous algorithms are used to address the fixed object searching problem in distributed systems for different intentions. However, people are nomadic, the mobile presence information is more mutable and dynamic; anew design of mobile presence services is needed to address the buddy-list search problem, especially for the demand of mobile social network applications. Presence Cloud is used to construct and maintain distributed server architecture and can be used to efficiently query the system for buddy list searches. Presence Cloud consists of three main components that are run across a set of presence servers. In the design of Presence Cloud, the ideas of P2P systems and present a particular design for mobile presence services has been refined. The three key components of Presence Cloud are summarized below:

- Presence Cloud server overlay: It organizes presence servers based on the concept of grid quorum system. So, the server overlay of Presence Cloud has a balanced load property and a two-hop diameter node degrees, where n is the number of presence servers.
- One-hop caching strategy: It is used to reduce the number of transmitted messages and accelerate query speed. All presence servers maintain caches for the buddies offered by their immediate neighbours.
- Directed buddy search**:** It is based on the directed search strategy. Presence Cloud ensures an one-hop

search, it yields a small constant search latency on average.

3.1 Presence Cloud Overview

The primary abstraction exported by our Presence Cloud issued a scalable server architecture for mobile presence services, and can be used to efficiently search the desired buddy lists. We illustrated a simple overview of Presence Cloud in Fig. 5. In the mobile Internet, a mobile user can access the Internet and make a data connection to Presence Cloud via 3G or Wifi services. After the mobile user joins and authenticates himself/herself to the mobile presence service, the mobile user is determinately directed to one of Presence Servers in the Presence Cloud by using the Secure Hash Algorithm, such as SHA-1. The mobile user opens a TCP connection to the Presence Server (PSnode) for control message transmission, particularly for the presence information. After the control channel is established, the mobile user sends a request to the connected PSnode for his/her buddy list searching. Our Presence Cloud shall do an efficient searching operation and return the presence information of the desired buddies to the mobile user. Now, we discuss the three components of Presence-Cloud in detail below.

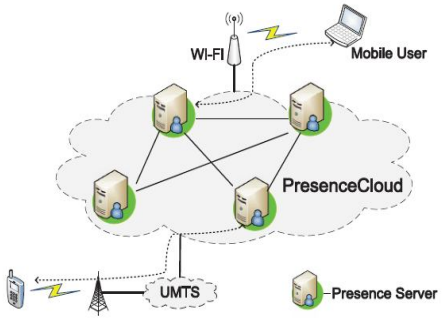

Fig 5. Architecture for presence cloud

3.2 Presence Cloud Server Over relay

The Presence Cloud server overlay construction algorithm organizes the PS nodes into a server-to-server overlay, which provides a good low-diameter overlay property. The low-diameter property ensures that a PS node only needs two hops to reach any other PS nodes.

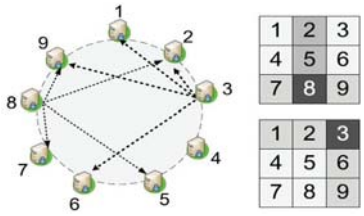

Fig 6. Presence cloud server overlay

Algorithm1. Presence Cloud Stabilization algorithm

1: /* periodically verify PS node n's pslist */
2: Definition:
3: pslist: set of the current PS list of this PS node, n
4: pslist[].connection: the current PS node in pslist
5: pslist[].id: identifier of the correct connection in pslist
6: node.id: identifier of PS node node
7: Algorithm:
8: r ←Sizeof(pslist)
9: for i = 1 to r do
10: node ←pslist[i].connection
11: if node.id ≠pslist[i].id then
12: /* ask node to refresh n's PS list entries */
13: findnode←Find_CorrectPSNode(node)
14: if findnode=nil then
15: pslist[i].connection←RandomNode(node)
16: else
17: pslist[i].connection←findnode
18: end if
19: else
20: /* send a heartbeat message */
21: bfailed←SendHeartbeatmsg(node)
22: if bfailed= true then
23: pslist[i].connection←RandomNode(node)
24: end if
25: end if
26: end for

Our algorithm is fault tolerance design. At each PS node, a simple Stabilization () process periodically contacts existing PS nodes to maintain the PS list. The Stabilization () process is elaborately presented in the Algorithm. When a PS node joins, it obtains its PS list by contacting a root. However, if a PS node n detects failed PS nodes in its PS list, it needs to establish new connections with existing PS nodes. In our algorithm, n should pick a random PS node that is in the same column or row as the failed PS node.

Directed buddy search algorithm:

1. A mobile user logins PresenceCloud and decides the associated PS node, q.
2. The user sends a Buddy List Search Message, B to the PS node q.
3. When the PS node q receives a B, then retrieves eachbi from B and searches its user list and one-hopcache to respond to the coming query. And removes the responded buddies from B.
4. If B = nil, the buddy list search operation is done.
5. Otherwise, if B =nil, the PS node q should hash each remaining identifier in B to obtain a grid ID,respectively.
6. Then, the PS node q aggregates these b(g) to become a new B(j), for each g Sj. Here, PS node j is the intersection node of Sq intersection Sg. And sends the new B(j) to PS node j.

One-Hop Caching

To improve the efficiency of the search operation, PresenceCloud requires a caching strategy to replicate presence information of users. In order to adapt to changes in the presence of users, the caching strategy should be asynchronous and not require expensive mechanisms for distributed agreement. In PresenceCloud, each PS node maintains a user list of presence information of the attached users, and it is responsible for caching the user list of each node in its PS list, in other words, PS nodes only replicate

the user list at most one hop away from itself. The cache is updated when neighbours establish connections to it, and periodically updated with its neighbours. Therefore, when a PS node receives a query, it can respond not only with matches from its own user list, but also provide matches from its caches that are the user lists offered by all of its neighbours. Our caching strategy does not require expensive overhead for presence consistency among PS nodes. When a mobile user changes its presence information, either because it leaves PresenceCloud, or due to failure, the responded PS node can disseminate its new presence to other neighbouring PS nodes for getting updated quickly.

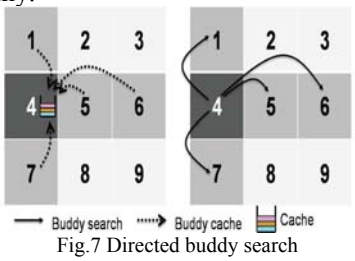

Fig.7 Directed buddy search

When a PS node 4 receives a Buddy List Search Message, B (1; 2; 3; 4; 5; 6; 7; 8; 9), from a mobile user, PS node 4 first searches its local user list and the buddy cache, and then it responds these searched buddies to the mobile user and removes these searched buddies from B. These removed buddies include the user lists of PS node {1,4,5,6,7}. Then, PS node 4 can aggregates b and b to become a new Bð6Þ and sends the new B to PS node 6. Note that the ps list Id of PS node 6 is {3,4,5,9}. Here, PS node 4 also aggregates b and b to become a new and sends the new Bð5Þ to PS node 5. However, due to the one-hop caching strategy, PS node 6 has a buddy cache that contains these user lists of PS node {3,9}, PS node 6 can expeditiously respond the buddy search message . Consequently, the directed searching combined with both previous two mechanisms, including Presence Cloud server overlay and one-hop caching strategy, can reduce the number of searching messages sent. Our caching strategy does not require expensive overhead for presence consistency among PS nodes. When a mobile user changes its presence information, either because it leaves Presence Cloud, or due to failure, the responded PS node can disseminate its new presence to other neighbouring PS nodes for getting updated quickly. Consequently, this one-hop caching strategy ensures that the user's presence information could remain mostly up-to date and consistent throughout the session time of the user.

CONCLUSION

By surveying some papers it is conclude that PresenceCloud achieves performance gains in the search cost without compromising search satisfaction. The IMS model displays the number of incoming and outgoing messages from the network. Users are in various states at the beginning, and gradually they log out, log in, and change their presence status. The change of the number of transmitted messages is related to this. If we observe this long enough without changing probability, the model will be in a stable state. That means that the same number of users changes their state in every step. We can determine the expected number of messages from the model, and according to this, we can design the size of the waiting queue on the presence server. We can see the ratio of messages if we want to assign different priorities too.

REFERENCES

[1] R.B. Jennings, E.M. Nahum, D.P. Olshefski, D. Saha, Z.-Y. Shae, and C. Waters, "A Study of Internet Instant Messaging and Chat Protocols," IEEE Network, vol. 20, no. 6, pp. 16-21, July/Aug. 2006.

[2] Z. Xiao, L. Guo, and J. Tracey, "Understanding Instant Messaging Traffic Characteristics," Proc. IEEE 27th Int'l Conf. Distributed Computing Systems (ICDCS), 2007.

[3] Chi, R. Hao, D. Wang, and Z.-Z. Cao, "IMS Presence Server:aTraffic Analysis and Performance Modelling," Proc. IEEE Int'Conf. Network Protocols (ICNP), 2008. Hadi Sadat, "Power System Analysis, " Tata McGraw Hill, 2002.

[4] P. Saint-Andre, "Interdomain Presence Scaling Analysis for the Extensible Messaging and Presence Protocol (XMPP)," aIETF Internet draft, 2008.

[5] X. Chen, S. Ren, H. Wang, and X. Zhang, "SCOPE: Scalable Consistency Maintenance in Structured P2P Systems," Proc. IEEE INFOCOM, 2005.

[6] N. Banerjee, A. Acharya, and S. K. Das, "Seamless sip-based mobility for multimedia applications." IEEE Network, vol. 20, no. 2, pp. 6–13, 2006.

[7] P. Bellavista, A. Corradi, and L. Foschini, "Ims-based presence service with enhanced scalability and guaranteed qos for interdomain enterprise mobility," IEEE Wireless Communications, 2009.

[8] A. Houri, E. Aoki, S. Parameswar, T. Rang, , V. Singh, and H. Schulzrinne, "Presence interdomain scaling analysis for sip/simple," RFC Internet-Draft, 2009.

[9] M. Maekawa, "A √n algorithm for mutual exclusion in decentral-ized systems," ACM Transactions on Computer Systems, 1985.

[10] D. Eastlake and P. Jones, "Us secure hash algorithm 1 (SHA1)," RFC 3174, 2001.

[11] M. Steiner, T. En-Najjary, and E. W. Biersack, "Long term study of peer behavior in the kad DHT," IEEE/ACM Trans. Netw., 2009.

[12] K. Singh and H. Schulzrinne, "Failover and load sharing in sip telephony," International Symposium on Performance Evaluation of Computer and Telecommunication Systems, July 2005.

[13] H. Sugano, S. Fujimoto, G. Klyne, A. Bateman, W. Carr, and J. Peterson, "RFC 3863: presence information data format (PIDF),"2004.

[14] H. Schulzrinne, U. Columbia, V. Gurbani, P. Kyzivat, and J. Rosenberg, "RFC 4480: RPID: rich presence extensions to the presence information data format(PIDF),"2006.

[15] M.Poikselka, G.Mayer, H.Khartabil, and A.Niemi, The IMS:IP Multimedia Concepts and Services in the Mobile Domain, John Wiley&Sons,NewYork,NY,USA,2004.

[16] M. Day, J. Rosenberg, and H. Sugano, "RFC 2778: a model for presenceandinstantmessaging,"2000.

[17] M. Wuthnow, M. Stafford, and J. Shih, IMS: A New Model for Blending Applications, Taylor & Francis, BocaRaton, Fla, USA, 2010.

[18] G. Camarillo and M. Garcia-Martin, The 3G IP Multimedia Subsystem(IMS):Merging the Internet and the Cellular Worlds, JohnWiley&Sons,NewYork,NY,USA,2ndedition,2006

1. Ms. Jareena N. Shaikh, Lecturer in JSPM's Bhivrabai Sawant Polytechnic Wagholi, Pune, India

2. Mrs. Bhandari G.M, Head of Department (Computer), JSPM's Bhivrabai Sawant Institute of Technology and Research, Wagholi, Pune, India