Autonomous Community Construction Technology to Achieve Service Assurance in ADCS

Khalid MAHMOODa), Xiaodong LU, Nonmembers, and Kinji MORI, Member

SUMMARY Autonomous Decentralized Community System (ADCS) makes its basis on offering customized and dynamic services to group of end-users having common preferences at specified time and location. Owning to extreme dynamism in the system caused by rapidly varying user's demands, and severe mobility of users, it is quite difficult to assure timely service provision to all community members. This paper presents autonomous decentralized community system construction by autonomous division and integration technologies to procure service assurance under dynamic situations, without involving significant communication overhead. By adopting the concept of size threshold, the proposed technique continuously maintains the appropriate size of community in constantly and rapidly changing operating environment, to deliver optimal quality of service in terms of response time. The effectiveness of proposed technology has been shown through simulation, which reveals remarkable improvement (up to 29%) in response time.

key words: autonomous decentralized community system, community division and integration, community size threshold

1. Introduction

Recent headway in the domains of mobile communication and ubiquitous computing has given surge of interest to mobile commerce applications. Current information systems promote concept of information service provision to anyone, anywhere and anytime. Notwithstanding, considering real time requirements of Service Provider (SP) and users of marketing applications, it is imperative to have information delivery to specific users, at specific location and specific time. For example, in the retail business, users are interested to search out attractive services based on their current situations and preferences; while to offer customized services, service providers are eager to grab the current requirements from local majority of users in the vicinity [3]. Due to these underlying overlapping requirements, information services considering users' preferences and their current situation become significant one.

The pervasive deployment of wireless data communications has enabled Location-Based Services (LBS). Owning to its ability to track the geographical position of mobile devices [19], LBS enables provision of services based on the geographical location of the ubiquitous mobile devices [15]–[17]. While, these systems consider service provision on the basis of location, they don’t take into consideration time aspect and user's interests. Moreover, death of time attribute precludes the provision of dynamic services. Furthermore, requirements for mobile commerce to offer services on the basis of correlated spatio-temporal — location specific, time-bounded situations — and common user’s interest can’t be satisfied through global information services of the internet. By incorporating time-awareness and demand-oriented notion, one-dimensional LBS notion could be revamped into three-dimensional situation-aware services. To transform this notion into reality, service-oriented community system has been proposed, which makes its basis not only on location, but also take into account user interests and time attribute [4].

Mobility of users and change in user's demands renders enormous dynamism in ADCS, which complicates attaining the appropriate size of community to achieve timeliness. There might be several solutions to improve the response time of community network by retaining appropriate size of community. One naive approach is to place a centralized observer to monitor the appropriate size of community under severe volatility of community network. However, existence of such centralized approach will render loss in flexibility and deterioration in response time. Yet another possibility is to enable every node of community to monitor response time from the SP all the time, and adjust the community size by employing some election procedures used in clustering techniques to elect appropriate number of cluster-heads [11], [12]. However, such a solution involves significant overhead, which may cause further deterioration in response time. Therefore, we propose novel idea of autonomous decentralized community construction by integration and division technology, which adopts the concept of threshold size to approximate optimal size of community under dynamic situations. This technique achieves timeliness by reducing communication overhead without loss of flexibility.

This paper contributes towards the proposition of community concept, Autonomous Decentralized Community System (ADCS) architecture and Autonomous Community Construction Technique (ACCT). The core objective of ACCT technique is to effectuate high response services, by dividing and integrating community to achieve optimal community size, in dynamic operational environment.

This paper is structured as follows: Next section presents the concept of ADCS, system requirements and exhibits the system architecture in detail. Section 3 describes system model, while Sect.4 clarifies problem formulation. Section 5 illuminates underlying basics and elaborates.
proposed autonomous decentralized community construction technique. Section 6 present evaluation and simulation results. We review related work in Sect. 7. The last section draws conclusion.

2. Autonomous Community System: Concept Requirements & Architecture

2.1 Concept

In LBS [15], [16], SP imparts the service without discerning differences in situation/time. Therefore, mobile-users receive the same contents in all situations. Moreover, SP cannot offer a well-customized service because nodes are passive actors — end-nodes only utilize the service provided by SP. We envisage the situation-aware local-majority based services to be not only location-aware, but also time-aware and demand-oriented. The basis of community system is to provide services to specific users at specific place and specific time by cooperative engagement of community members. Community members having individual objectives, common interests, and demands at specified time somewhere/anywhere are autonomous, cooperative and active actors — end-nodes mutually cooperate in requests aggregation and service provision to enhance the objectives of all.

2.2 System Requirements

Community-service applications are realized with a concept of time-distance oriented system [4], [7], [9], [10]. As the time-distance between nodes changes dynamically, the size of community also fluctuates [4], [7]. Therefore, extreme dynamism is not only caused by change in user preferences but time distance too. To accommodate such changes, the system must satisfy high flexibility. Likewise, to assure instant service provision, service composition and service distribution must be done in real-time. If the system simultaneously complies with the properties of online expansion, flexibility and timeliness under dynamic situations, it is said to provide assurance of the services.

2.3 System Architecture

Inspired from both the spirits of cooperation in the social communities and the Autonomous Decentralized System (ADS) concept [8], Autonomous Decentralized Community System (ADCS) is proposed, which fosters the theme of service-oriented community: An architecture that offers customized service to the group of end-users having common interests in specific service, somewhere/anywhere, at specified time.

Community nodes are attached with each other on bilateral basis. The bilateral logical contact between two community nodes occurs considering that the users of these nodes have the same interests and demands, at specific time and location. The autonomous decentralized community network is a self-organized logical topology with set of nodes V and existence of loops. Each node keeps track of its immediate neighbors in a table. The immediate neighbors' set of the community node x is defined as a set of nodes,

$$NS(x) = \{y; x, y \in V, h(x, y) = 1\}$$ (1)

Where, $h(x,y)$ is the number of the logical hops between nodes x and y. Each node judges autonomously to join/leave the community network by creating/destroying its logical links with its neighbor's members based on its user's preferences. Such autonomous decision of each node is realized through Autonomous Control Processor (ACP), which employs only local information stored in Neighboring Nodes Table (NNT). When a node receives a message, each node autonomously judges, whether the message should be accepted or not, by checking the service property of the message. As a result, those nodes which accept the message form a community [10]. Needless to say, that each node also keeps track of time distance with respect to neighboring nodes of community network and stores it in NNT. The overview of the system architecture is shown in Fig. 1.

To disseminate service information within the community, 1 -> N community hybrid pull/push based communication is used [1], [6]. To meet the flexibility requirement in constantly and rapidly changing operating environment of ADCS, we employ content-code communication [8]. There is no existence of global information such as IP multicast group address in the system. By integrating underlying ADS architecture with proposed ACCT, the ADCS meets service assurance requirement.

3. System Model

Figure 2 shows the overview of proposed system model. Each node along with attached base station in the figure represents the coverage area of cell. Each base station communicates with user's mobile units (PDA/cellular phones) and service provider through backbone network [5]. During handoff mechanism [5] each base station calculates the time-distance: Average estimated time a user requires reaching physically from area covered by one base station to other. For details please refer [4], [7]. By logically connecting the nodes through backbone IP infrastructure, that are in the same situations — for example, nodes that are with in three-hours distant to SP — a group of nodes/base-stations having
same demands of end-users could be formed. Such a group of nodes is called community network; and the area that is formed by the community members (end-nodes) bounded by time-distance is called community service area.

After segregation of community into sub-communities, three different types of nodes could be identified in ADCS: Leader, Peripheral and Normal nodes. Leader is the special node, which on behalf of all members, accesses the SP and propagates the service information to all members of its sub-community; peripheral nodes not only dissociate sub-communities but also act as gateway for intercommunication among them. Apart from peripheral and leaders, we term all other nodes as normal nodes.

3.1 Response Time Model

This section elucidates response time model of ADCS. The service provision in community model comprises of three steps: first, request aggregation of users at all nodes in path from \(x_0, x_1, \ldots, x_n\), where \(x_n\) is the leader node; second, accessing Service Provider (SP) — \(x_n\) to SP; and third, service dissemination to community members — \(x_n\) to all members \(x_i\).

\[
T = T_a + T_s + T_d
\]

Aggregation Time \(T_a\): The nodes perform in-network processing and hence aggregate the users request at each node before forwarding to next node during \((\tau_{agg})\). The transmission delay between each node for a single request is assumed to be constant and is expressed as \(\tau_r\) (time for transmission). Also, the number of hops between the leader and the farthest node within the community is expressed as \(|D_L|\) (size of community).

\[
T_a = |D_L|((\tau_r + \tau_{agg})
\]

Time for SP-access \(T_s\): Assume that message processing at SP takes \(\tau_p\) (time for processing) for a single request. Let transmission delay between the leader and the SP is \(\tau_l\) and total number of requests from leaders be \(m\). The processing time for \(T_s\) could be written as follow,

\[
T_s = m\tau_p + 2\tau_l
\]

Time for Community Service Dissemination \(T_d\): Assume that the definition for \(\tau_l\) and \(|D_L|\) be the same as for aggregation of user’s request messages. Worst time for service distribution could be written as,

\[
T_d = |D_L|\tau_l
\]

4. Problem Formulation

For the sake of simplicity, if we initially assume that every end-node in the community network acts as leader to access service provider individually, \(T_s\) shoots up and surge of simultaneous requests overwhelm SP. Conversely, if some community nodes join together to form one group, where only one leader node accesses the SP on behalf of all community nodes, and all community members coordinate and cooperate in request aggregation and service distribution; intuitively response time for service provision could be improved. However, if the size of this community grows beyond certain limit, the response time starts to deteriorate again due to increase in number of hops between the furthest nodes to the leader of community.

When community size grows and furthest nodes observe delay in response time, this larger size community might be fragmented into two or more sub-communities, with each having its own leader. Subsequently, all sub-communities in the system may aggregate requests and distribute services concurrently. In presence of multiple communities, response time model transforms as follow.

\[
T_a + T_s + T_d \rightarrow |D_L|(\tau_l + \tau_{agg}) + Nm\tau_p + 2\tau_l + |D_L|\tau_l
\]

Where, \(N\) is the number of sub-communities. It is self-evident that as \(|D_L|\) decreases, the number of sub-communities increases; therefore, \(T_a\) and \(T_d\) improves but \(T_s\) deteriorates. On the other hand, when size of sub-community enlarges beyond certain limit, \(T_s\), tends to improve due to less number of leaders accessing SP, but \(T_a\) and \(T_d\) start getting worse. This overall process is shown in Fig. 3.

Moreover, due to extreme dynamism in the community network (nodes join or leave incessantly), response time fluctuates continuously. Thus, the objective function is to find optimum size of community, and it should be adaptive...
according to dynamic situations, to ensure service provision to its all members with in certain maximum latency constraint. We term this latency bounded factor, as Latency Cognizance Constraint (LLC). Each sub-community with set of nodes $x_0$ should satisfy LLC as follows,

$$ S_i = x_0; \sum_{k=0}^{m} \delta(x_k, x_{K+1}) \leq \alpha_i $$

(7)

Where, $x_0$ is edge node and $x_m$ is the leader node; whereas $\delta(x_k, x_{K+1})$ refers to the current end-to-end delay from node $x_k$ to $x_{K+1}$. $\alpha_i$ denotes LLC in $i$-th sub-community.

5. Autonomous Decentralized Community Construction

Before elucidating the procedure of autonomous decentralized community construction, we state the notations, node's parameters and format of messages. Table 1 shows the node's state parameters. By employing control information piggybacked on service and request messages, each node updates its parameters for making autonomous decision about its state (normal/leader). Request messages, carrying aggregated requests of users, are propagated by nodes towards SP via leader node; while, service messages are forwarded by leaders towards all members of their respective sub-communities. The format of both types of messages is shown in Tables 2 and 3 respectively.

5.1 Sub-Community Size Threshold Assessment

Size of each sub-community is determined by maximum threshold ($D_{thH}$) to get optimal performance in terms of response time. This threshold is calculated by leader of the respective sub-community, by monitoring the response time from SP and sub-community size using threshold judgment function, as depicted in Fig. 4. Total response time $T$ can be regarded as function of $f_T(|D_L|)$ and by $f'_T(|D_L|)$, it is possible to determine the tendency of function at the size of sub-community $|D_L|$. Each leader monitors this trend in the function change. If the function is in decreasing trend, the leader increases the sub-community size by extending the threshold value and vice versa. Thus, each community leader is able to adjust sub-community size into optimal one with threshold judgment function, which requires two parameters: $D_L$ and $T_s$. The value of lower threshold $D_{thL}$ is inferred from $D_{thH}$, as described in Sect. 5.2.3.

5.2 Autonomous Community Division and Integration

Suppose that at certain time if existing number of sub-communities is $\beta$, and there exist sub-community with size $|D_i|$ larger than a threshold value $D_{thH}$, then this sub-community is divided to enhance the overall performance. In contrast, if the number of sub-communities is large and there exist sub-communities with size $|D_i|$ and $|D_j|$ smaller than lower threshold $(D_{thL})$, then these sub-communities are integrated. Therefore size of each sub-community lies with in limit of maximum and minimum threshold.

5.2.1 Autonomous Community Division Technology

While distributing service messages from SP, each leader of its respective sub-community piggybacks estimated size threshold value $D_{thH}$ on service message to its all members after fixed time interval. As the message propagates from

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_{thH}$</td>
<td>Upper threshold value of sub-community size</td>
</tr>
<tr>
<td>$D_{thL}$</td>
<td>Lower threshold value of sub-community size</td>
</tr>
<tr>
<td>$D_L$</td>
<td>Distance from leader of sub-community</td>
</tr>
<tr>
<td>$D_B$</td>
<td>Distance from peripheral node (boundary) of current sub-community</td>
</tr>
<tr>
<td>$D_{thi}$</td>
<td>Priority for sub-community</td>
</tr>
<tr>
<td>SubComID</td>
<td>Sub-community ID</td>
</tr>
</tbody>
</table>

Table 1 Node's state parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>Service content code</td>
</tr>
<tr>
<td>SubComID</td>
<td>Sub-community ID</td>
</tr>
<tr>
<td>SPAddr</td>
<td>SP address</td>
</tr>
<tr>
<td>Sender</td>
<td>Address of message sender</td>
</tr>
<tr>
<td>nHops</td>
<td>No. of hops message has traveled</td>
</tr>
<tr>
<td>DthH</td>
<td>Attribute to update node's $D_{thH}$</td>
</tr>
<tr>
<td>DthL</td>
<td>Attribute to update node's $D_{thL}$</td>
</tr>
<tr>
<td>Arbitrator</td>
<td>Priority for sub-community</td>
</tr>
<tr>
<td>Data</td>
<td>Service data</td>
</tr>
</tbody>
</table>

Table 2 Service message parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>Content code of request Message</td>
</tr>
<tr>
<td>SubComID</td>
<td>Sub-community ID</td>
</tr>
<tr>
<td>Generator</td>
<td>Node ID of message initiator</td>
</tr>
<tr>
<td>FlagBond</td>
<td>Flag set by peripheral nodes</td>
</tr>
<tr>
<td>$D_B$</td>
<td>Radius of sub-community</td>
</tr>
<tr>
<td>Data</td>
<td>Service request data</td>
</tr>
</tbody>
</table>

Table 3 Request message parameters.

![Fig. 4 Sub-community size threshold assessment.](image-url)
leader node towards peripheral nodes of community, each normal node increments the nHops field of the received service message. When sub-community size increases by affiliation of new nodes, community leaders calculate new threshold accordingly by employing threshold judgement function; and inform their members by next service message. By employing this information, each community member autonomously keeps track of community size by detecting whether the distance from the leader ($D_{th}$) is exceeding the threshold $D_{thH}$ or not by comparing with nHops field of service message. If any normal node discerns that the community/sub-community it belongs to has become oversized, it initiates division process by declaring itself as leader. Therefore, a new sub-community will be created, and the nodes that are in close proximity to this new leader may assimilate to the new sub-community. Figure 5 shows the overview of autonomous division of community.

Each leader keeps on checking status of size threshold value, and adjusts the optimal size of its sub-community as follow.

Check Threshold()
\[
\begin{align*}
\text{If} (g(|D_{th}|, T_s) < 0) & \quad |D_{thH}| = |D_{thH}| + N; \\
\text{else if} (g(|D_{th}|, T_s) > 0) & \quad |D_{thH}| = |D_{thH}| - N;
\end{align*}
\]

Likewise, each normal node keeps on executing the following procedure to check its eligibility to become leader.

check promotionProcess()
\[
\begin{align*}
\text{If} ((D_{thH} < |D_{th}|) \&\& (|D_{th}| < D_{thL})) & \text{ return (0); } \\
\text{else if} (D_{thH} > |D_{th}|) & \text{ / / perform following steps} \\
\quad \text{Myself} \rightarrow \text{leave}(); & \text{ leave current sub-community} \\
\quad \text{Myself} \rightarrow D_B = 0; & \text{ declares itself to leader by resetting } D_B \\
\quad \text{Myself} \rightarrow D_L = 0; & \text{ reset } D_L \text{ as well} \\
\quad \text{Myself} \rightarrow \text{SubComID} = \text{myself} \rightarrow \text{Sub_id}; & \\
\quad \text{Myself} \rightarrow \text{check Threshold}(); & \text{//start calculating threshold } D_{thH}. \\
\quad \text{Myself} \rightarrow \text{setLower (neighborlist[]);} & \text{return (1); }
\end{align*}
\]

Intuitively, equal-sized sub-communities assist to attain best response time for overall system, therefore when peripheral nodes of existing neighboring sub-communities receive join request from leaders of newly created sub-communities, each node decides to move to new sub-community if it meets following criteria.

- If, its distance from current leader ($D_{th}$) is greater than the new neighboring sub-community leader (nHops+1), then it assimilates to new sub-community with lower $D_{th}$.
- If SubComID of node is Null, as is the case in normalization of leader node (described in autonomous integration technology in Sect. 5.2.2), it gets attached with the newly created sub-community.

When any node relocates itself to new sub-community, it accomplishes following steps:

1. Replace value of its SubComID field with SubComID of newly joined.
2. Assign parameter $D_{th}$ with value equals nHops+1 from new community leader.
3. Set the sender of the message in newly joined sub-community as “upper” in the node’s NNT, while set all other nodes as “lower” nodes (including peripheral nodes of neighboring sub-communities).

5.2.2 Autonomous Community Integration Technology

When nodes isolate from their original community to form new sub-communities during autonomous division process, some of these sub-communities may be undersized. All peripheral nodes — those who receive service message from more than one sub-community — set the $D_{thH}$ field of request message with their distances from the respective leaders. As request message propagates towards leader, request aggregation process lets intermediate (normal nodes) nodes forward only highest value of $D_{th}$. This highest value of $D_{th}$ updates the $D_B$ parameter of leader node. If the leader discerns that the sub-community is undersized ($D_{thH} > D_B$), it becomes normalized and the sub-community vanishes, as shown in Fig. 6. When a leader normalizes, it sets its SubComID field to NULL while sets its $D_B$ field to some undefined value. Moreover, all nodes in NNT are set to “lower”. Later on this node will update its all value upon joining another sub-community, by following three steps.
described in last section. The criterion and conditions for setting value of $D_{thL}$ is described in Sect. 5.2.3.

Each leader node keeps on executing the following procedure to check autonomously, if it needs to normalize.

checkInteg()
If $(D_{thl} < rcvmsg \rightarrow DB \& \& rcvmsg \rightarrow DB < D_{thH})$
return (0); // No Integration required
else if ($D_{thl} > rcvmsg \rightarrow DB$)
   Myself→$DB = \infty$; Myself→subComID = Null;
   Myself→setLower (neighborList[]);
}

5.2.3 Stabilizing Community Construction Process

During process of community division, it is imperative to assure that each segregated sub-community stays stable, which means, after the division each sub-community should not immediately satisfy the conditions for autonomous integration technology. If this occurs, each sub-community keeps on integrating and dividing whole system unstable. In order to preclude such unsteadiness, it is crucial to set the adequate value of $D_{thL}$. When $D_{thH} = n$, and if a new leader promotes at $n + 1$ hop away from the old leader, nodes distant at $[(n + 1)/2]$ hops away from the new leader will be assimilated to the new sub-community. If any leader at this stage decides to normalize, the system may become unstable. To thwart this undesired behaviour, we set criteria of integration to be $[(n + 1)/2] \geq D_{thl}$ instead of $[(n + 1)/2] < D_{thL}$. Therefore, for each sub-community, the following conditions must be satisfied ($k$ is a natural number).

1) $n = 2k + 1$: $n = D_{thH} \geq 2D_{thl} - 1$
2) $n = 2k$: $n = D_{thH} > 2D_{thl}$

6. Evaluation

To evaluate performance of our approach, we consider community network as 6d-regular mesh graph. For the sake of simplicity, we ignore the aggregation time from Eq. (6) and rewrite it as

$$T = |D_L|\tau_l + (M/m) \ast \tau_p + 2\tau_l + |D_L|\tau_l$$  (8)

Where, ‘$M$’ is the total number of nodes in system and ‘$m$’ is number of nodes in one sub-community.

$$\frac{dT}{dD_L} = 2\tau_l - \frac{f'(|D_L|)}{f(|D_L|)}^2 M\tau_p$$
$$\Rightarrow 2\tau_l - \frac{f'(|D_L|)}{f(|D_L|)}(T_S - 2\tau_l) = g(|D_L|, T_S)$$  (9)

Through this size threshold judgment, each leader estimates the optimal size of its sub-community.

Table 4 shows the main parameters used in the simulation to validate the proposed technique.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link transmission delay</td>
<td>$\tau_l$</td>
<td>150 [ms]</td>
</tr>
<tr>
<td>Request processing time</td>
<td>$\tau_p$</td>
<td>100[ms]</td>
</tr>
<tr>
<td>Request aggregation time</td>
<td>$\tau_{agg}$</td>
<td>100[ms]</td>
</tr>
<tr>
<td>Node entry interval</td>
<td>$T_H$</td>
<td>5000[ms]</td>
</tr>
<tr>
<td>Service message interval</td>
<td>$T_{ST}$</td>
<td>3000[ms]</td>
</tr>
<tr>
<td>Peripheral message interval</td>
<td>$T_{Bl}$</td>
<td>9000[ms]</td>
</tr>
<tr>
<td>Leader timer interval</td>
<td>$T_{P}$</td>
<td>5000[ms]</td>
</tr>
</tbody>
</table>

6.1 Simulation Model

Since our assumed topology is hexagonal mesh, so total number of nodes in system could be expressed as, $f(\eta) = 3\eta^2 - 3\eta + 1$ where, $\eta$ is number of levels in regular hexagonal mesh graph. Therefore, the threshold judgment function from Eq. (9) could be rewritten as

$$2\tau_l - \frac{6D_L - 3}{3D_L^2 - 3D_L + 1}(T_S - 2\tau_l) = g(|D_L|, T_S)$$  (10)

6.2 Simulation Results

We compare the proposed autonomous community construction technology (ACCT) with naive approach in terms of response time for service delivery, under the situation where number of members in community increase gradually. To emulate dynamic behavior, we add nodes at different random positions in simulation environment after fix interval $T_{JI} = 5000[ms]$ to monitor behavior of proposed method with respect to response time. Addition of nodes was done periodically until total number of nodes reach up to 720. Moreover, to validate our proposed method, we find the appropriate size of community and response time without considering dynamic situation (static scenario) with sub-community threshold size 2, 3, 4 and 7.

Corroborative evidence of our approach could be realized by the fact that average response time for service delivery in case of proposed autonomous community construction is 3700[ms], when number of nodes reaches to 720. In contrast, the naive approach (no division) gives response time of 5200[ms]. This clearly demonstrates that proposed methodology outdo conventional one, giving significant improvement — up to 29 percent (maximum) — in terms of response time. As shown in Fig. 7 (a), the optimal size of sub-community in non-dynamic situation is seven for large number of nodes, and results of response time and the community size of proposed ACCT conforms to this. The response time fluctuates in proposed methodology due to community transition overhead. The zoom part
MAHMOOD et al.: AUTONOMOUS COMMUNITY CONSTRUCTION TECHNOLOGY TO ACHIEVE SERVICE ASSURANCE IN ADCS

Fig. 7 Simulation result: Response time comparison.

of Fig. 7 (a) is shown in Fig. 7 (b) to demonstrate the effectiveness of ACCT for small number of nodes. The optimal value of sub-community size to achieve best response time for nodes up to 300 is 2 or 3 in static situation; and the results of proposed ACCT are almost close to this. Hence, we may conclude that ACCT performs equally well for both small and large range of nodes to achieve timeliness.

7. Related Work

Service provision on the basis of user profiles has been reported in Service Accelerator (Sea) system [20] and Autonomous Decentralized Service System (ADSS) [21] but these systems don’t cater for local majority based services at certain location.

Context-aware computing [2] proposes flexible information services in ubiquitous environment employing users’ location. However, this approach deals with limited applications, where dynamic changes in environment are not expected. Community service is proposed to deal with dynamic changes in environment where number of members varies greatly.

“Jack-in-the-Net (Ja-net)” [13] based on “Bio-Net Architecture” [14], is proposed to adapt to the dynamic changes in environment, and to generate evolving network services. However, the objective of Ja-Net is to create services that are fully customized for each user, and differs from the objective of community services, where the main objective is to achieve high assurance for massive information services.

Adaptive multi-hop clustering (AMC) is clustering technique of MANET using the concept of adaptive size [18]. In AMC, for merging process cluster $C_i$ tries to find a neighboring cluster $C_j$ to satisfy $|C_i| + |C_j| \leq U$ and maximize the sum value and if $|C_i| + |C_j| > U$ for all neighboring clusters, it tries to find a $C_j$ to minimize the sum value. A cluster $C_i$ with $|C_i| > U$ performs the division mechanism. However, AMC does not address how to select a proper node $v_k$ to serve as the clusterhead for the newly detached cluster. Similarly AMC does not specify how $U$ and $L$ should be determined. In contrast, ACCT doesn’t have such limitations

8. Conclusion

Stirred by the dictum “one for all and all for one”, a group of members (nodes) possessing common information interests of end-users, at certain place and at certain time, forge concept of community. Owing to the synergy among members of the group for information accumulation and service distribution, a continual and timely service information provision could be procured. The community members, users and service providers, mutually collaborate to utilize/provide three dimensional services considering user situation (location, time and demands/interests). This paper has presented autonomous community construction by autonomous community division and integration technologies, by employing maximum and minimum sub-community size threshold, to acquire timeliness under dynamic situations. Finally, we have evaluated the effectiveness of proposed ACCT through simulation. By integrating underlying ADS architecture with proposed ACCT, the ADS meets service assurance requirement.

References


