A study on dynamic state information (DSI) around users for safe urban life
Hoon Ko, Kyungjin An, Taihoon Kim, Goreti Marreiros, Zita Vale, Jongmyoung Choi

ABSTRACT
To select each node by devices and by contexts in urban computing, users have to put their plan information and their requests into a computing environment (ex. PDA, Smart Devices, Laptops, etc.) in advance and they will try to keep the optimized states between users and the computing environment. However, because of bad contexts, users may get the wrong decision, so, one of the users’ demands may be requesting the good server which has higher security. To take this issue, we define the structure of Dynamic State Information (DSI) which takes a process about security including the relevant factors in sending/receiving contexts, which select the best during user movement with server quality and security states from DSI. Finally, whenever some information changes, users and devices get the notices including security factors, then an automatic reaction can be possible; therefore all users can safely use all devices in urban computing.

Keywords
Context-aware, Collaborative, Urban computing, Security, Dynamic state

1. Introduction
The Urban Computing (UrC) focuses on dynamic services for urban life that defined the space for user’s movement, the sensors and the systems including GPS, Smart Devices, RFID readers and wireless access pointers to identify users, their locations, and sometimes their personal information [1,2]. These sensors, devices and systems are generally connected among various heterogeneous networks, and all transferring data are used for providing user-customized or context-aware services to the users [3]. The contexts that are captured by the sensor systems are going to be passed to the organizations that provide context-aware services through the heterogeneous network systems [4,5]. The information which usually comes from the perspective of context security and privacy should be absolutely personal and sometimes valuable. In urban computing, each portion of sensors and device systems are connected over a wireless network and an ad hoc network. Eventually, many attackers may put their attacking devices on a street to get users’ entire information [6]. Finally they can get them to know what users have by analyzing the captured contexts. This problem is surely caused from the weak security [5, 7]. Because of these security issues, users prefer secure services, which is why all network systems should choose the safe paths by selecting the trusted devices or the devices with good security factors. To provide this issue, we define the structure of Dynamic State Information (DSI) which takes a process about security including the relevant factors in sending/receiving contexts, which select the best during user movement with server quality and security states by DSI. This algorithm identifies the device’s status information, manages the information, and updates dynamically. This algorithm also describes the type of encryption algorithms, the length of key, and the update time according to the feature of the context information and devices. This paper is composed of the explanation of the security problems in urban life in Section 2, and we define the
2. Security problems

In [8,9], the authors studied how to send information to users through collaboration among servers nearby. They have already mentioned some problems about the accuracy of similarity evaluation, that is, it was hard to find perfect similarity due to insufficient user information and lack of security level. Consequently they researched the Context-Aware Collaborative Filtering Algorithm with the user’s dynamic context into the old Collaborative Filtering Algorithm [8], they wanted to offer a real time environment based content transferring system using multiple sensors. As a result, the requested contents could be transferred from the nearest server according to user’s location. However, there is still some data loss and receiving delay in this old method. To overcome these problems, [9] presented that they had tried to add the network state information for accuracy during computation. In [6], the authors research about information collection in urban environments; they also studied evolution use by collecting them. However there are problems, like the concentration problem by their gathering. Ad hoc communications can operate them by collecting data from many sensors, and they suggested connecting among the external nodes after setting the multi-hop of ad hoc communication through various and accurate information collecting. Also, because sensors can connect among another object, we can use them anywhere, anytime, etc. In [10], user authentication in a ubiquitous environment was studied, explicitly allowing moving user devices near users by taking an authentication automatically for safety using all contents. Nonetheless this method has a data loss problem due to some steps that it has to be taken towards their certificate and authorization using devices. Also, they had suggested profiling generation skill in order to solve the original problem. In [11,12], they researched location-aware, middle-ware to detect user’s location. Namely, it was for detecting them in case of events arising according to user location. The problem is the data loss because the breaking during their moving in middle-ware’s event based. In [11] the authors tried to solve that problem by suggesting the distributed event recognition method named STEAM. Although [12] had proposed the method, which using Data Mining in order to detect user’s location quickly, usually it takes a long time to analyse data mining. In [13,14], the authors studied how to set the devices in real time. However it still has problems due to the dynamic changing of network states. In [15], the authors had mentioned the server set method to provide a service to moving users. Yet, there are some overhead problems as too many are sending signals to moving users who have frequently/randomly moving and contents re-arrangement.

3. The flow of dynamic state information

3.1. Scenario

Our service scenarios are about the seamless cooperation of small devices in Urban Computing. So, the Fig. 1 shows an example of the services. In the service, user devices, network devices and their organic operations normally cooperate to provide useful services to users. In this service scenario, users allow their information such as a car’s environment or a personal setting in their moving (t2) offices to be sent to the network environment; their offices are set based on the
information from the car (t3), then, they normally connect to the nearby networks according to their device information including security state (t7). While all devices do this processing, the devices protect the information by getting real-time security information from the server, in which the security factors including the security algorithm, the key size, and the validation date are kept updated by propagating the information dynamically (t8).

3.2. DSI definition

In this section, we illustrate how the devices’ security related information is managed, updated, and propagated to neighbor devices. Basically the information is managed and updated on a regular basis. Table 1 shows all notations and their definitions used in DSI. The information is used for determining the secure communication path dynamically. We call this the DSI Algorithm. This algorithm considers devices’ status information and their security capability and determines the encryption algorithm and key length. DSI is updated in two different ways. First it is updated periodically according to the interval time setting. Second it is updated immediately on sending/receiving the notice of the configuration changes. For example, if servers or network devices make their security level higher or lower, they usually broadcast their changed states to other server, network, and user device at once. As soon as all devices, including users’ devices, in the computing area detect this signal, each of them begins to update its own DSI. Furthermore, the user information also is broadcast to nearby devices, and then the devices process the information and update their DSI automatically. DSI has ItemID, Attribute, SecurityRequire, Link structure. ItemID defines the context category, and Attribute defines each item of context information. SecurityRequire is used to define the security states: authentication and authorization. It takes time for devices to process all context information, so this processing may cause the service delay. Therefore, the systems and users should determine the tradeoff between high security and real-time services, so they choose security states according to user requests and some context information. Link is a kind of link to items stored in Repository and it is designed for describing information that is not expressed in DSI. Security context has an authentication state, an authorization state, and the security server ID. All security servers are given SecurityLevel according to the security level evaluated from the governments or commercial policy department for security.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCc</td>
<td>Define a car temperature</td>
</tr>
<tr>
<td>PCr</td>
<td>Define an office temperature</td>
</tr>
<tr>
<td>PCt</td>
<td>Define a traffic condition</td>
</tr>
<tr>
<td>UCI</td>
<td>Define a user location</td>
</tr>
<tr>
<td>UCr</td>
<td>Define a device Information</td>
</tr>
<tr>
<td>CCn</td>
<td>Define a AP Condition</td>
</tr>
<tr>
<td>CCs</td>
<td>Define a network state</td>
</tr>
<tr>
<td>CCo</td>
<td>Define direction condition to destination</td>
</tr>
<tr>
<td>ts</td>
<td>Each time in a step</td>
</tr>
<tr>
<td>A<a href="T">l</a></td>
<td>Define algorithm types</td>
</tr>
<tr>
<td>K[S]</td>
<td>Define key sizes</td>
</tr>
<tr>
<td>SF(f)</td>
<td>Define a security factor</td>
</tr>
<tr>
<td>P(t)</td>
<td>Define a total processing time</td>
</tr>
<tr>
<td>Ps</td>
<td>Define a server processing time</td>
</tr>
<tr>
<td>Pu</td>
<td>Define a user processing time</td>
</tr>
<tr>
<td>PDSI</td>
<td>Define a DSI generating time</td>
</tr>
<tr>
<td>DSM</td>
<td>Define Dynamic State Model</td>
</tr>
<tr>
<td>Pu2D</td>
<td>Define a user device updating time</td>
</tr>
<tr>
<td>PNSI</td>
<td>Define a network setting time</td>
</tr>
</tbody>
</table>

Table 1

Cooperative service algorithm

define f(c) Ca, Cn, Cd; //t8
define Ue Laptop, PDA, Desktop, etc; //t9
detect (Pc); //t2
send (Pc) from User to CAR; //t1
if (Pc) == (Po) then
  return(); //t3
else
  set (Po); //t3
broadcast (F(DSI)); //t6
receive and send Change(I), f(c); //t8
receive and send Change(I), f(u); //t9
receive and send Change(I), f(c), f(u); //t7
set (DSI)in USER; //t10
3.3. DSI operation

In order to reflect the change correctly, efficiently, and in real-time, it is important to keep DSI up to date. Fig. 2 illustrates the update process of DSI. The secure server determines the security algorithm and the key size. Periodically or non-periodically generated updating information is sent to the secure server (t20), which causes the context changes of the server (t23). Then the server notifies the devices about the changes, so that the devices update their security states automatically (t25).

<table>
<thead>
<tr>
<th>DSI.Configuration.Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>update Alg({T}, K{S});     //t20</td>
</tr>
<tr>
<td>inform Alg({T}, K{S});     //t21,t22</td>
</tr>
<tr>
<td>set Alg({T}, K{S});        //t23</td>
</tr>
<tr>
<td>inform SF(f), Alg({T}, K{S}); //t24</td>
</tr>
<tr>
<td>set SF(f), Alg({T}, K{S});   //t25</td>
</tr>
</tbody>
</table>

3.4. Security configuration

In the security area, the weakest points may be attacked by hackers who know those points, so that if there is a need to upgrade the security level, all devices' security levels should be upgraded, and the upgrading process should be automatic because manual processing cannot support that in real-time. The Fig. 3 shows how the change of the security level is propagated to all devices, upgrading from security level #2 to level #3. The DSI change in the Fig. 3 causes the security changes, and the processing follows Security.States.Algorithm. According to the Security.States.Algorithm, the security information is updated (from security level #2 to #3) in the devices: PDA, phone, tablet, and laptop. They usually get update (t11) at once. As a result, they set PDA, phone and tablet as level #2 and the laptop as level #3 (t12). It notifies to near network device or network environment of the final information (t13). After receiving this message, they keep the recent state by updating this information by themselves (t14).

<table>
<thead>
<tr>
<th>Security.Configuration.Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>define Security.Server[LapTop,PDA,Phone,Tablet];      //t11</td>
</tr>
<tr>
<td>upgrade LapTop.#2 to LapTop.#3;                     //t12</td>
</tr>
<tr>
<td>set LapTop.#3;                                      //t13</td>
</tr>
<tr>
<td>inform LapTop.#3 to Network Environments            //t14</td>
</tr>
</tbody>
</table>

3.5. New DSI (for user 1)

Fig. 4 shows the updated DSI and illustrates the change of the key size of the algorithm ‘a’ which is one of the modification values in server changes from [a, 2] to [a, 3], changes f4 of dev2 from [a, 2] to [b, 3], and changes f3 of dev4 from [0, 0] to [a, 3]. When the server is made aware of the credential DSI of user 1, they begin this configuration (t32) (t33). All networks which are using the security factors near users automatically reset according to the new DSI.
4. Experiments and discussion

For secure usage, it needs to make a security process like an authentication and an authorization to all devices used by users. Fig. 5 shows the general flowing steps including devices authentication and authorization. Firstly, it begins as soon as it receives $t_{20}$ which is update $\text{Alg}([T])$, $K([S])$. Then the sensor which receives $t_{20}$ sends this news to server ($t_{21}$, $t_{22}$). After that server sets $t_{23}$ and updates $DSI$ ($t_{24}$, $t_{25}$), next, let each device keep with the new $DSI$ by knowing ($t_{31}$, $t_{33}$) to use four $\text{dev} [1-4]$. The server notices the updated $DSI$ to all user devices and networks to keep the recent states ($t_{13}$) by them. In any situation, servers, users and network members take that processing repeatedly receiving/sending/broadcasting/detecting/resetting in order to set the new configuration ($t_{2}$, $t_{6}$, $t_{7}$, $t_{8}$, and $t_{10}$).

4.1. Simulation time explanation

We conducted this experiment for 3000 s. In this experiment, we included 100 nodes in total, and four of them (node [0, 0], node[1, 8], node[5, 2], and node[7, 7]) were set as destination nodes. User data would be generated randomly in one
or two seconds non-regularly; four destinations were located and equally balanced. It can evaluate the total time of each step by adding the time \( P_1 \) which from server changing and detecting of user devices to the network setting. That is, it can get them by adding as below, the server processing time \( (P_{st}) \), the user processing time \( (P_{Ut}) \), DSI create/update time \( (P_{DSI}) \), the user devices update time \( (P_{UD}) \) and the network configuration time \( (P_{NE}) \) (1).

Server processing time: The reply from the server is very important because of quick response for users. Fig. 6 shows the fast processing time inside the server. \((P1, P2)\) was set as the server processing time. We define the time \( (t21(1, j)) \) taken for finding the number of changed DSIs in sensors as \( P1 \), and the time \( (t21(1, -j)) \) for sending DSIs to the server as \( P2 \).

Then we can define the server processing time (2) and average processing time of server (3) as below;

\[
P_s = P_{st} + P_{st} \cdot (P_{DSI}) + P_{UD} + P_{NE}.
\]

(1)

\[
\text{Server processing time:} \quad \text{The reply from the server is very important because of quick response for users. Fig. 6 shows the fast processing time inside the server.} \quad (P1, P2) \text{ was set as the server processing time. We define the time } (t21(1, j)) \text{ taken for finding the number of changed DSIs in sensors as } P1, \text{ and the time } (t21(1, -j)) \text{ for sending DSIs to the server as } P2.
\]

Then we can define the server processing time (2) and average processing time of server (3) as below;

\[
P_s(P1 \rightarrow P2) = \sum_{x=0}^{n} [3 \cdot t_{st} + f(x)] + [3 \cdot (t_{st} \cdot t_{st})] + 3 \cdot t_{st}
\]

(2)

\[
\text{Avg}(P_s) = \sum_{x=0}^{n} t_{st} + t_{st} + t_{st}/n.
\]

(3)

User processing time: We define the process that users should conduct as \((p2, p3)\). The processed information by the server goes to user devices update. Because it automatically updates the network information via sensors and servers, there are no tasks for the user to take. The user processing time (4) and the average user processing time (5) are as below;

\[
P_u(p2 \rightarrow p3) = \sum_{x=0}^{n} [3 \cdot t_{st} + f(x)] + \sum_{x=0}^{n} [3 \cdot nD_2 \cdot f(x)] \cdot t_{st} + t_{st} + t_{st}.
\]

(4)

\[
\text{Avg}(P_u) = \sum_{x=0}^{n} t_{st} + t_{st} + t_{st} + t_{st} + t_{st} + t_{st}/n.
\]

(5)

User device updating time: It leads to user devices updating after DSI updating. The process for user devices update is \((p3, p4)\). The user devices have to be updated as much as the updated DSI \( (t32[1..n]) \cdot t33[1..n]) \). Total user devices update time (6) and average user devices update time (7) are the next formulas;

\[
P_d(p3 \rightarrow p4) = \sum_{x=0}^{n} [3 \cdot t_{st} + t_{st}] \cdot t_{st}
\]

(6)
Network update time: It has to update the network configuration information soon based on user DSI states, and it process in the same time with user devices update time (6) \((p3, p4)\). Therefore, this update time is dynamically changed according to the number of user devices and network state. The network update time (8) and the average network update time \((9)\) are below;

\[
\text{Avg}(P_{IU}) = \sum_{i=0}^{1} \delta \cdot [t_{11} : t_{12}] \cdot t_i/n. \tag{7}
\]

Fig. 7 contains state information around a user using DSI which were from (10). There is the internal process and external process in each step. The internal process has the user processing time \((P_{UI})\) and the user update time \((P_{UID})\) and the external process has the server processing time \((P_S)\) and the network configuration time \((P_{NE})\). That is:

\[
\text{IF} \ (EF), \ \text{IF}[P_U, P_{UID}] = v_1[x_1, x_2], \ \text{EF} [P_S, P_{NE}] = v_2[y_1, y_2].
\]

In the formula, \(x_1\) is the result of (5), \(x_2\) is the result of (7), \(y_1\) is the result of (3), and \(y_2\) is the result of (9). The values in each cell can be evaluated by (10) from the results of (5), (7), (3) and (9).

\[
v_1 - v_2 = |x_1 - y_1| + |x_2 - y_2|. \tag{10}
\]

The Cell[2,2] generated next value (0.3). This result from the (10) can affect with auto re-configuration for users, user devices and computing environment etc.

\[
v_1 - v_2 = |x_1 - y_1| + |x_2 - y_2| = |0.4 - 0.4| + |0.6 - 0.9| = 0.3.
\]
If it begins the evaluation from start point [1, 1] with the same way (10), the result is obtained as below: Cell[0,0],[0,1] [0,2], [1,0],[1,1],[1,2], [2,0],[2,1],[2,2] = 0.7, 0.6, 0.7, 0.6, -0.5, 0.7, 0.8, 0.3. The minimum value means it takes the minimum time to adapt between the internal process and the external process. Finally, that value is the fastest process.

4.2. Simulation description

Simulation Set: This paper took a simulation with ns2 and C-lang. The network for simulation was designed based on the Fig. 7, the routing set was in from Node [1, 1] to Node [5, 5], and apart from those nodes, the rest of node which was not included from Node [1, 1] to Node [5, 5] was defined to communicate among them. The value in each node was supposed to be set by 10 s according to the result of (10). The user was on Node [1, 1] when this simulation first began, the user's destination was at Node [5, 2], then that user tried to find the optimized route which applied those algorithms that this paper proposed. In this simulation, it set 18 nodes for the experiment, therefore Node [1, 1] could be Node (1), and Node [5, 2] was to be Node (18), (we considered that unused nodes have no paths among them, and we removed a point by multiplying by 100.). Total Rate of Result: The Table 2 shows the rate of the number of packets. After simulation, the result of this paper like acceptance rate or failure rate is to be the Table 2. The total number of packets during simulation is 2,933,007, then 2,877,778 of them obtained succession, 98.1%, the rest of them, 55,229 failed, only 1.9%. Also, the result of experiment and the average time for 100 nodes are in Fig. 8 and Table 3. Each node processing cost range is in between 0 and 8, totally it has 9 costs (c), it is arranged based on each cost. According to this result, there are 49 nodes in range (0–1), and there are 4 nodes which have a maximum cost (7–8) (Fig. 9). Because the sum of all nodes from the start node to destination node means for the total costs, the less the value of cost, the better that performance. There are the evaluated costs for each user in Table 4. Each user moves to next node by an average 2.96 s (1), they get 2.54 s for services (2), their waiting time before services is 3.88 s (3). And, after adding Service Time and Waiting Time, we normally set as Spending Time in Node (4). Finally, according to the result of this simulation, Spending Time in node is an average of 6.42 s (4), Idle Time in each node is an average of 0.444 s (5).

New path with DSI: The value of nodes from this simulation usually uses for user and devices updating which is using near users and environments. Fig. 9 shows the simulation result for two models (without DSI and with DSI), when it is used to find the path by this model, this paper got (b) in case it doesn’t have DSI. Surely, the cost of (b) made 13 (Table 4). It is higher than (c), which got 10 as cost. The result of the cost of (c) was from Model (c) which has DSI. As you see in Table 4, the model which has DSI usually gets the lower cost.

<table>
<thead>
<tr>
<th>Category</th>
<th>Inter arrival time (1)</th>
<th>Spending time in node</th>
<th>Idle time of node (5)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Avg.Time</td>
<td>2.96</td>
<td>2.54</td>
</tr>
<tr>
<td></td>
<td>Service time (2)</td>
<td>2.96</td>
<td>2.54</td>
</tr>
<tr>
<td></td>
<td>Waiting time (3)</td>
<td>3.88</td>
<td>3.88</td>
</tr>
<tr>
<td></td>
<td>Service + waiting time (4)</td>
<td>6.42</td>
<td>6.42</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.444</td>
</tr>
</tbody>
</table>

Fig. 8. The result of experiment.
5. Conclusion and future works

We defined the DSI that can be used in urban computing. On detecting the user’s context changing, it can be applied into DSI, if we use this DSI. At a result, all users can adapt to a changing environment quickly. There are two kinds of processing, one is Internal Factors (IF) and the other is External Factors (EF) in DSI. First, IF involves user processing time ($P_{U}$) that has the same meaning as updating time of user certificate credential (X.509 proxy extend) and user devices updating time ($P_{UDS}$) that user’s device updating time according to user certificate credential. Second, EF is composed by server processing time ($P_{S}$) and Network Configuration Time ($P_{NE}$). We set that the server processing time is the required time for an authentication/authorization for changed context and IF and network re-configuration time considering the server as the network configuration time. As a result, we can keep all information near the user location immediately or periodically using this DSI, therefore, it is possible to update the user credential quickly. Harmonization is important among users, between users and physical objects, among physical objects, between physical objects and software objects in urban computing. We tried to create a harmony between their actions. However, in this research we did not consider an authentication/authorization by user’s biometric information, ex., heart rate, blood pressure etc. Namely, the human interface skill that we can use directly was not involved. This paper has only user’s devices using or network devices using near the user’s location or they operate by user’s asking like user’s button click or user’s ordering by voice etc. That is not real time computing and automatic computing in urban computing include human computing. In future, if we add these human interface skills into this paper, also, we should add all devices’ movement time, then we expect that they can be more active DSI.

Acknowledgment

This work is partially supported by the Portuguese Foundation for Science and Technology (FCT) in Cincia 2007 program for the hiring of Post-Ph.D. researchers.

References


