Chapter 7

Service Migration in Pervasive Environment

7.1 Introduction

Service migration enables continuity of service experiences while changing devices, without terminating running sessions of the service. It has been proposed as a significant application for the emerging pervasive environments. A migration process is needed at once when the delinked routing path is get re-established. Two closely related sub tasks involved in the migration are device mobility and service migration.

A migration process can be initiated by a device itself once the connection is established by finding the proper device to communicate. The other one is by the user, who can select a device from the given list with the desired service int it. A device’s UI and its session state also must be migrated to the device where the migration of service is going to occur. Knowledge on MAC about the new device should be known by the device which is going to initiate the communication; the necessary data will be received from the application layer specification which gives the session description of a particular device.

UDP and TCP are two transport level protocols for the data transfer in this type of communication. UDP is a connection less protocol where TCP is a connection oriented protocol. TCP is distinguished with its reliability, capability in flow control and error control. Data transfer is done with the help of UDP and TCP using a remote procedure which has the capability to continue the service in a detached service path. Device mobility, service migration, UI and MAC identification, and implicit and
explicit migration are the various key aspects which are going to be discussed in the following sections.

7.2 **Background Works**

The ability to move an application between devices, in the context of pervasive, or ubiquitous, computing with its vision of information access anytime and anywhere, introduces a new class of applications that seamlessly follow people as they move through the physical world is discussed in [98]. A new network service framework highlighting the concept of service migration is presented in [99] and the framework take into account essential service quality factors, such as server loading, bandwidth, delay, and so on. More references in service migration are elaborated and discussed in the chapter 3.

7.3 **Mobility of Devices during Service Provisioning**

Device mobility is one of the causes of the route disconnections and the path re-establishment. These can be a cause due to the movement of:

1. Service providing device,
2. Service receiving device.
3. Representative devices in the routing path.

Client applications can highly benefit from continuous service provisioning in many situations. A node running a certain service can become inappropriate to host that service any longer due to the above said mobility, limited resource availability, or network partitioning. In these cases, the client has to discover and restart its interaction with a new service. This can lead to significant performance degradation for a continuous interaction. A mechanism for transferring the service to a new node and resuming its execution is necessary. Figure 7.1 shows a situation how the mobility of device causes disconnections and so service migration.
In the figure, Node A and B are connected up to a distance of 10 kms and when the nodes feel they are on move and the distance is getting increasing they are invoking the packet transfer algorithm which is mentioned in chapter 6. Figure 7.1 (a,b and c) shows the device movement and Figure 7.1 (d) shows the new connection establishment among Node A and Node C.

### 7.4 Service Migration

If a migration policy for a specific service is established, the context manager of the proposed framework decides the migration time based on context [92]. If the context information matches the conditions and the manager finds a target host that satisfies the requirements of a service provider, the service will be migrated to the target host. Specifically, once migration is decided, the proposed method collects the contexts of neighbour hosts and extracts candidate hosts which satisfy minimal
conditions and decides a target host among them. The decision of a target host is based on the priority information provided from a service provider.

Two closely related sub-tasks are involved in the service migration [94][95], namely process migration and connection migration. A seamless service migration scheme is expected after the path re-establishment. Connection migration is a critical issue closely related to the problem of service migration [96]. The connection associated with a migrating process should also be migrated, so that, after the migration, the service can be seamlessly continued on the target device. It can be expected that an ongoing data flow will experience certain degree of degradation and disconnection during the migration. A connection migration scheme is considered to be a good one if all these negative impacts can be minimized. In order to achieve these type of migration first of all, the MAC level interpretation is needed and the next topic depicts the extraction of MAC details from the device which is elected for continuing the dissemination of service.

7.5  MAC Sub Layer Specification

The MAC layer is the sub layer of the data Link layer and since it is where the compatibility of the communication is being validated in the MAC entry level; this layer has to interpret the transmitted data and interpret the type of MAC of the corresponding device which is selected.

To find out the corresponding MAC sub layer [100] of the device the following procedures are used and are:

1. Set_Destination(Destination_MAC) procedure sets the MAC of the devices available in the environment.

2. Cl_Message_Type(Message, Destination_MAC) procedure transfers the cross layer messages with the compatible type of MAC of the destination.
3. *Get_Destination*(return *Destination_MAC*) procedure returns the destination MAC address for further cross layer reference.

![Figure 7.2 MAC Sub Layer specification](image)

Figure 7.2 shows how the broadcasted data fetches the MAC layer representation through the data link layer.

### 7.6 Application Layer Specification

**Context Computing**

The application handles the device, service Details such as IP address, location, peripheral configuration, transmission rate, device type, servicing nodes list, UI etc. A Context Computing Language Repository (CCLR), which contains the most needed programming/scripting languages keywords and their interpretation for processing is needed for interpreting the application level layer contexts. Context Computing Language (CCL) contains its own equivalent interpretation for each keyword inside the CCLR. And it will convert the scripting language keywords to CCL keyword. CCL Message Service will send the CCL output (migration details) to the chosen devices. An example description about a device in the network given as,
<node_id><component_name><size><location><content>. For Example, <mobile1><screen><10,20><10,10> "Hello". In the other device, the text will be decrypted and the component will be adjusted to its screen size percentage and the data will be displayed. The communication can be done by web servicing.

**UI Identification**

The main aim of the application layer interpretation is:

1. UI Adaptation.
2. Service continuation.

Also during data transmission the representative node has to replicate the broadcast data or packets. But during migration the process has to do the additional work like UI adaptation [97], so the packets will be processed at the application layer itself and then it will reach to the physical layer. Figure 7.3 (a) shows the UI identification using context computing and Figure 7.3 (b) shows the simple text transfer regarding migration details between two devices. The application layer has the session details such as:

1. Service details like IP, location, peripheral configuration, transmission rate, device type, servicing nod list, current state of the service and time.
2. CCLR with needed scripting languages keywords.
3. CCL with its own equivalent keyword for each keyword inside the CCLR.

4. CCL message service will send the CCL output as migration details to the chosen devices.

The application layer specification is needed to interpret the device context which is transmitted by the packet in the form of XML document. The UI type and size of a device will be extracted from the device using XML parsers. Figure 7.4 shows the application layer specification for a broadcasted data that passes through all the network layers to reach out the desired device.

![Application Layer specification](image)

An example of a device UI description using XML is referred from [101] is given as:

```xml
<Device
    <UI Type="screen">
        <Size ="screen size" >
            <Colour="RGB Combination" >
                Dest Addr="MAC Addr" 
            Device>
```

The above UI details are described in XML and the interpretations on UI type, size of the screen, colour of image and destination address are parsed by the XML
parsers and can be transferred from one device to another during initiation or
continuation of service provisioning.

7.7 Types of Service Migration

The process of service migration can be initiated by the functions such as
device selection for the continuation of service provisioning, user interface
identification and MAC identification. The selection of eligible device can be done
by the device in the network or by the user from the available service possessing
devices in the list and are categorized into implicit and explicit migration
respectively.

Implicit Migration

The implicit migration has the following characteristics:

1. In the implicit migration the migration will be done automatically
   according to the current context.

2. User doesn’t know where the migration takes place.

<table>
<thead>
<tr>
<th>Algorithm IMPLICIT_MIGRATION()</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong> :- Eligible devices list</td>
</tr>
<tr>
<td><strong>Output</strong> :- Selected Device</td>
</tr>
</tbody>
</table>

If Host.NodeID=BUSY) or (Host.NodeID.MEMORY ≥ THRESHOLD_MEMORY
Or Host.NodeID.Distance≥Threshold_Distance) then
Start New SERVICE DISCOVERY; // start a new service node to migrate
If (New.Host.Node_ID =FOUND) then
migrate.service_to (new.node(id));
send.host.node(id)←"ACK";
Else
Continue Service Until New.Host.Node_ID←FOUND;
End If
End If
END

Algorithm 7.1 Algorithm for Implicit Migration
Algorithm 7.1 shows the selection of devices implicitly by the requesting device itself. If the service providing device is busy or if the memory capacity is less than the threshold size, then the device may search for a device of the desired service and the Figure 7.5 shows the flow of the implicit migration.

![Diagram](image)

**Figure 7.5 Components in Implicit Migration**

**Explicit Migration**

The Explicit Migration has the following characteristics:

1. In the explicit migration the migration will be done manually by the user.
2. The process will wait for the user’s choice. Time delay may occur in compare to the implicit migration.

```algorithm
Algorithm EXPLICIT_MIGRATION(new.node_id, manual_selection_node_id)

Input :- List of eligible devices
Output :- Selected Device

Service_node[i] ← new.node(id); //hold all available service context node id list
If (service_node[i] = manual_selection_node_id) then
    MIGRATE.service to (manually_selection_node_id);
End if
END
```

Algorithm 7.2 Explicit Migration
Algorithm 7.2 shows the procedure involved in explicit migration, where the eligible device list is used for the selection of device manually by the user to where the migration is occurred. Figure 7.6 shows the flow of the activities happen in implicit migration.

![Diagram showing components in implicit migration]

The following procedures referred in [102] are used for sending the service using UDP and TCP sockets:

1. *UDPSender (Send(Packet))* is the procedure used to send the packets in stream from the service provider device.

2. *UDPReceiver(receive(Packet))* is the procedure used to receive the packets by the service requester.

The following procedures in [105] are used to send the UDP packets through XML Web Services:

1. *XML_RPC_Service_Prov()* is the procedure handled by the service provider for initiating to send the packets through UDP by XML using inter process communication.
2. *XML_RPC_Service_Req()* is the procedure used by the service requester to start sending the packets through web services to the service provider.

3. *XML_RPC_Handler()* is the procedures used to control the inter process communication between the RPC Client (Service Requester) and Server (Service Provider).

### 7.8 Migration Metrics

The various migration metrics that are to be evaluated to analyze the performance of the routing scheme are:

1. **Average Path re-establishment time for Service Continuation**, is the time to search a device with compatible MAC and UI for continuing the service during the re-establishment of routing path to continue the service provisioning. This is the summation of routing path establishment time and routing path re-establishment time after fault recovery along with MAC and UI adaptation.

2. **Average Data transfer time**, is otherwise called as the migration time. It is the time taken by the migration scheme to transfer data by using TCP and UDP through the routing path for continuing the service provisioning. This is the summation of initial routing path establishment time with routing path re-establishment time after fault recovery added to the transfer time of the service.

### 7.9 Network Parameters

The various network parameters used during the simulation of service migration are shown in table 7.1.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Area</td>
<td>1500m x 1500 m</td>
</tr>
<tr>
<td>Channel Type</td>
<td>Wired and Wireless</td>
</tr>
<tr>
<td>Propagation Model</td>
<td>Two Way Ground</td>
</tr>
<tr>
<td>Radio Range</td>
<td>200 m</td>
</tr>
<tr>
<td>Radio Delay</td>
<td>10 milli secs</td>
</tr>
<tr>
<td>Traffic Type</td>
<td>CBR with TCP and UDP</td>
</tr>
<tr>
<td>Duration</td>
<td>100 Seconds</td>
</tr>
<tr>
<td>MAC Layer</td>
<td>IEEE 802.3, 802.11, 802.15 and 802.16</td>
</tr>
<tr>
<td>Routing Protocols</td>
<td>SFUSP, AODV, DSR and OLSR</td>
</tr>
<tr>
<td>Mobility</td>
<td>Random Waypoint, Brownian Motion and Manhattan Model</td>
</tr>
<tr>
<td>Node Strength</td>
<td>Energy, Bandwidth and Service Context</td>
</tr>
<tr>
<td>Context</td>
<td>Contexts such as internet, disk, printer, games etc.</td>
</tr>
<tr>
<td>Devices</td>
<td>Mobile devices, stationary devices etc.</td>
</tr>
<tr>
<td>No. of Nodes</td>
<td>50, 100, 200 and 400 with Fixed and Mobility nature. Nodes assigned with different MAC layers.</td>
</tr>
<tr>
<td>Speed</td>
<td>25 m/s with a pause time of 10 milli secs. For Manhattan Model minimum speed of 25 m/s to a maximum speed of 100 m/s.</td>
</tr>
<tr>
<td>Transmission rate</td>
<td>9.6 Kbps</td>
</tr>
<tr>
<td>Data Payload</td>
<td>512 bytes</td>
</tr>
<tr>
<td>Traffic Load</td>
<td>Packet Sent in Every 1 second</td>
</tr>
</tbody>
</table>

### 7.10 Result Analysis

#### Simulation Criteria

The simulation is done with the device density of 50, 100, 200 and 400 nodes in different instant, assigning different mobility patterns such as Fixed, Random, Brownian and Manhattan to the devices in an arbitrary way. The devices are
randomly assigned with various types of MAC layers. It is assigned that, there are N nodes for the simulation with 20 percentage of the nodes are set to fixed network in which 50 percentage of nodes are set to WLAN, 30 percentage of nodes are set to WPAN and 20 percentage of nodes are set to WWAN. Of the 40 percentage of nodes set to random mobility, the above mentioned proportionate of wireless MAC technology is set to the nodes. In the other 20 percentage of nodes that are set as Brownian, the same ratio of MAC is assigned and in the remaining 20 percentage of nodes which are assigned to Manhattan Mobility is assigned with the same ratio of MAC. The simulation is performed for S number of simulations for the same network parameter values and the trace values are used for the calculation of the analysis. 1Mb of data is assigned to the service providing device to transfer to the corresponding device during migration for the simulation purpose.

**Ns-Miracle** is a set of libraries designed to enhance the functionalities provided by the NS2. It provides an efficient and embedded engine for handling cross layer messages and enables the coexistence of multiple modules within each layer of the protocol stack. The Module class provides the following methods for the communication and is referred from [100]:

1. *SendDown(Packet, delay)* is the class to send the packets to the down layers of any network layer.

2. *SendUp(Packet, delay)* is the procedure to send the packets from the down layers to the up layers in the network layers.

This module uses the various procedures and classes from the pervasive discovery protocol, Miracle software, java socket programming and web services. These procedures and classes in combine with the algorithms developed for the proposed routing scheme, make evolve the proposed routing scheme enhanced with an efficient migration scheme. Simulations are done on different platforms since
each and every package needs its own operating system to run. The other routing protocols also made execute along with the procedures used as like SFUSP in order to get a desired result while comparison.

**Simulation Results**

The simulation is done for a number of times under different fault occurrence scenarios that are discussed in chapter 6. The average simulation values of migration are given in table 7.2.

<table>
<thead>
<tr>
<th>Performance Metrics</th>
<th>Number of Nodes</th>
<th>SFUSP</th>
<th>Reactive Distance Vector Protocol</th>
<th>Proactive Link State Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path Re-establishment Time for Service Continuation after MAC and UI Identification (in Seconds)</td>
<td>50</td>
<td>19.45</td>
<td>41.42</td>
<td>44.31</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>23.07</td>
<td>52.22</td>
<td>51.22</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>25.11</td>
<td>58.4</td>
<td>55.21</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>27.02</td>
<td>60.22</td>
<td>62.81</td>
</tr>
<tr>
<td>Migration Time for Data Stream Transfer (in Seconds)</td>
<td>50</td>
<td>23.13</td>
<td>51.32</td>
<td>53.48</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>27.91</td>
<td>60.42</td>
<td>67.29</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>31.54</td>
<td>65.78</td>
<td>71.44</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>34.43</td>
<td>78.1</td>
<td>82.22</td>
</tr>
</tbody>
</table>

The trace files are processed by using Linux commands, C and Java Programs to find out the result for the various metrics. The analysis is done by comparing SFUSP Scheme with reactive distance vector routing protocols and proactive link state protocol. The following graphs show the comparison of the protocols for the various metrics discussed above.
Path Re-establishment Time for Service Continuation

Graph 7.1 shows the performance of SFUSP on average optimal number of nodes with the Reactive distance vector protocols like AODV and DSR. Graph 7.2 shows the performance of SFUSP with proactive link state protocols such as OLSR.
Service Migration Time

Graph 7.3 shows the performance of SFUSP on average optimal number of nodes with the reactive distance vector protocols such as AODV and DSR.

Graph 7.3 Migration Time for Data Transfer - SFUSP Vs Reactive Distance Vector Routing Protocols

Graph 7.4 shows the performance of SFUSP with the proactive link state protocols such as OLSR.

Graph 7.4 Migration Time for Data Transfer - SFUSP Vs Proactive Link State Routing Protocol
7.11 Observations

Path re-establishment time to initiate service migration after identifying MAC and UI is less with respect to SFUSP in comparing with other routing scheme because:

1. SFUSP is added with additional functionalities for migration which is not available in other reactive and routing protocols.

2. Some of the service discovery protocols do the migration process but they also derive the basic functionalities of distance vector or link state protocols which have not been provided with opt functionalities to handle migration of services.

3. SFUSP has both implicit and explicit type of migration scheme which selects the device for service migration by the knowledge of device or by user.

4. The semantics used in context computing helps in identifying the type of device in communication in a less time.

5. The initial route establishment time, fault-tolerant with less path re-establishment time helps to identify the desired device for the continuation of data transfer in a shorter time.

6. The MAC layer and UI Specifications features adds the functionality of SFUSP to work efficiently than the other protocols.

Data Transfer time after the path re-establishment by identifying UI and MAC is less in SFUSP because:

1. There are no procedures related to migration with other routing protocols although they are used for simulation along with the procedures created in this research for the simulation purpose.
2. It is found that proactive protocols are somehow customized with service migration functionality than reactive protocols for a less unstable network and since our proposed scheme is much rely to the proactive and reactive characteristics of routing, SFUSP shows a better result than others.

3. SFUSP is bundled with the necessary procedures for handling the migration for any type of environment and because of the better performance shown by it for the metrics mentioned in chapter 5 and 6, makes it to give better result in data transfer while migration.

7.12 Conclusion

The result shows that SFUSP takes less time in identifying the type of device which it wants to migrate and so the data transfer is also occurred within lesser time when comparing to other reactive and proactive protocols. Thus it is proved that the proposed routing scheme can be used for device communication in pervasive environment.