This chapter mainly focuses on architecture of Agent Management System (AMS) of Scalable Agent Pedestal (SAP) system for WSNs. Its main task is management of agents and nodes (SN/BN/BS), i.e., registration, authentication, security, and mobility as per PMADE. AMS can also be extended to support clustering and teaming. We also present a comparative study of the AMS (SAP) and CSCP based WSNs. Results show that AMS improves the performance of WSNs when the number of SNs in the network is very large.

Rest of the chapter is organized as follows. Issues are explored in Section 5.1. System model is presented in Section 5.2. Section 5.3 gives the system architecture of AMS. Section 5.4 presents an optimal migration strategic algorithm and the evaluation of the system is given in Section 5.5. Implementation and Performance study of WSNs under the leadership of AMS (SAP) is explored in Section 5.6. Section 5.7 discusses the outcome of the implementation. Finally, the chapter is summarized in Section 5.8.

5.1 Issues

In traditional DWSNs, data are collected by individual sensors, and then transmitted to a higher level PE which performs data fusion. During this process, a large amount of data are moved around the network, as is the typical scenario in the CSCP. By transmitting MA to SNs, the large amount of sensory data can be reduced or transformed into small data by eliminating the redundancy. The sensory data of two closely located sensors are likely to have redundant or common parts when the data of two SNs are merged. Therefore, data aggregation is a necessary function in densely populated WSNs in order to reduce the sensory data traffic.
But the existing systems take more computing time for finalizing the task. For the limitation of processing power and time to process a request, we need a system which must support device mobility (BS mobility) and computation mobility as per need of the applications.

5.2 System Model
We are required to develop a data dissemination system for WSNs that fulfills most of the above challenges. The developed system should enable the fast and cost-efficient deployment of self-managed computing/communication sensing devices with high overall management cost, but with low management cost at each SNs. With developed system one should be able to deploy large scale computing/communication systems without the need of cost-intensive distributed sensing infrastructure to monitor sensitive area with long life network. This system should facilitate to improve the performance and incorporate new ideas.

5.3 Architecture of Agent Management System
Architecture of Agent Management System (AMS) of SAP for WSNs is given in Figure 5.1. AMS provides energy efficient services using set of agents. In AMS mainly MAs are used to manage the WSNs. It provides common solution to WSNs fault tolerance and load balancing problems and gives true distributed computing and communication environment with the help of MAs. AMS supports code mobility over the mobile/fixed BS and SNs. It also support mobile device (Mobile base station: MBS). A MBS may move near to a region and become the member of a region for time bound emergency cases otherwise BN helps in all kind of tasks.

A BN in a region will work like region head (RH) and maintains information about other members of the region in the form of database. When a BN/BS wants to search some information it requests to BN for members information (viz. IP address, identification certificate, etc.). If the BN is not aware about availability of the type of services a BN/BS is interested and presence of the same in the region then it guides the same to the BN/BS. AMS permits a BN to create a MA for performing its desired task in the present region. Further, if the requested task is not completed with members of the present region, it may take help of AMS running at the BN for multicasting the MA to the BNs in the network. After
completion of the task final result reaches to the BN/BS which was its launching station.

**Figure 5.1: Architecture of Agent Management System (AMS): Multiagent System View**

At present AMS contains nine SAP agents- SAP Mapping Agent (SMAPA), SAP Route Estimating Agent (SREA), SAP Migration Planning Agent (SMPA), SAP Code Container Agent (SCCA) and SAP Result Container agent (SRCA), SAP Sensor Agent (SSA), SAP Broker Facilitator Agent(SBFA), SAP Application Agent (SAA), and SAP Interface Agent (SIA) in future number of agents may be increased as per need of the applications, means developed system is adaptable in nature. These agents are called SAP agents because at any moment of time as per requirement of the applications algorithm/protocols associated with these agents are changed/updated or new amendment can be made. SRCA and SCCA facilitates distributed environment for adapting the nature of the network bandwidth. These agents are identified as single entity known as agent management system (AMS). AMS is named because of its nature to accommodate any kind of changes occurring in the system. Other component of AMS is Network manager (NM) which is responsible to identify the topology of the
network with assistantship of SMAPA. NM provides global identification to mobile devices and MAs {details are given in Chapter 6}. AMS supports code mobility over the mobile/fixed peer device. For balancing the load over the network these agents work together as a SAP multiagent system. Architecture of the functioning of the AMS is given in Figure 5.2. Brief introductions of above agents are as follows.

5.3.1 SAP Sensor Agent (SSA): The SSA is a cognitive agent that resides on a SN and has access to the data source of a SN in that region. Any agent roaming in can dynamically change its itinerary type whenever required. This feature makes the agent fault tolerant, while roaming on the network. SSA makes the sensor information of the represented SN available to the sensor network application. The SSA registers itself with the AMS of the node (SN/BN) on which it resides. In SAP, there is a sensor agent for every SN.

5.3.2 SAP Broker Facilitator Agent (SBFA): The main task of SBFA is to provide lookup and advertising services for the application. SBFA is an agent, which helps other agents and SN/BN to find appropriate partners for a particular conversation.

5.3.3 SAP Application Agent (SAA): The SAA is a MA that is specific to the application. It can possibly communicate, move, cooperate, reason, adapt, learn and perform other application specific tasks. It consists of four components, namely, "identification" to uniquely identify the MA, "Agent Container" to store intermediate integration results, "method" which BN different integration algorithms, and "itinerary" that the MA follows during migration.

5.3.4 SAP Interface Agent (SIA): The SIA assists users in formulating requirements, maps them into requests to other agents for further processing and provides users with answers from the application agents. Its main task is to translate and help refine application specifications into node (SN/BN) controllers, such that the node (SN/BN) behavior is consistent with the application specific and platform adaptive global network behavior.
5.3.5 SAP Mapping Agent (SMAPA)

It is used by a MA to locate services and to access information on network connection qualities. Connection qualities are especially important for the SREA or and the SMPA to achieve optimizations. In addition to throughput, latency and other network status information, this agent collects and distributes information on application-level services provided by the BN in the network. SMAPA cares for precise and up-to-date knowledge (maps) within its local BN and provides a rough summarized view of the linked remote BNs. Utilizing the service descriptions in those maps, a MA is able to locate points of interest within the network and see changes in the network structure. Once a list of interesting BN has been determined, another system component - the SREA can be used by the agent to plan a route.

5.3.6 SAP Route Estimating Agent (SREA)

After getting the list of interesting BN that has been determined by the SMAPA, another system component – the SAP Route Estimating Agent (SREA) as shown in Figure 5.2 - can be used by the agent to plan a route for assigned task. SREA calculates the shortest trip through the network based on the map data. It uses the classic local optimization algorithms. If necessary, a route can be recalculated and amended, for example, in the case of changes in the network or when the agent moves into new BNs and thus shifts its focus with regards to the fisheye paradigm. It facilitates to the MAs for optimizing the sequence of BN to visit, i.e., the route. If an agent chooses a random path through a network, i.e., if an agent is visiting few nodes in region then visiting next region nodes and in future looking into the previously visited regions once again then the sequence may lead to a non-optimal total migration time. The route estimating process itself is basically the Traveling Salesman Problem, which is a NP-complete type of problem. But dividing the problems into set of small subsets of problems eases the complex problems. For optimal migration time agent creates a clone for a region and number of clones being equal to the number of regions of interest where the required services may be available. As a consequence, getting an optimal solution in practical application is ruled out. But there are heuristic algorithms (such as local search, genetic, simulated annealing, neural network algorithms, etc.) that have been supporting this working style of agents applied extensively for solving such problems.
The computation of a route is based on the map data. SREA calculates a kind of SAP Table (SAPT) simply by using the reciprocal values of measured bandwidth. This matrix has to be updated at regular time intervals to fit the environment’s dynamic behavior. Then, a pathfinder algorithm [65] is applied in order to get a SAPT with shortest paths between two places. In some experiments, we figured out that SAPT is not symmetrically in general. This is caused by variation in the bandwidth values and non-symmetrical connections measured by the SMAPA.

![Diagram](image)

**Figure 5.2:** An Agent logical view of the AMS component of SAP

The variation in network throughput influences the result and success of the route estimating, especially short time variations. The SREA generates a route with a fast path through the network on the basis of route matrix. Thereby, some of the best paths may be blocked by short time traffic. At the point in time, when an agent uses the optimized route, the generated path may not be the best one any
more or, in the worst case, is by now the slowest one. The probability that this happens is lower in networks with clearly differing connection qualities. The route estimating is especially useful in networks with different connection qualities and in networks with connections which have different loads over a longer time period. In networks with nearly identical connection qualities, the use of route estimating algorithms makes no sense – just choose a random path instead of spending time to calculate the random path.

5.3.7 SAP Migration Planning Agent (SMPA)

At any point in time, as long as we have a route, a MA may also use a so called SMPA as shown in Figure 5.2 to optimize each single migration included in the route. SMPA is mainly designed to reduce network load by selecting and transmitting only those code and data portions of the agent that are needed at the upcoming remote BN. This is, if necessary, done by a concept called slicing or designated code. Other options are to place code in advance in the network, to send data home to carry fewer luggage’s, to change the transmission protocol, etc. An agent may contain one or more task code to be executed at different nodes in the network. The point of time when an agent’s tasks are transmitted depends on the migration strategy, i.e., how a MA is transmitted over the network? There are so called push strategies which transmit an agent’s tasks along with the agent’s state and data before the agent is started at a remote BN. Using a pull strategy, an agent’s tasks are downloaded dynamically while the agent is executed at a remote BN from its home site/SAP Code Container Agent (SCCA). The agent’s home platform is the Agent Submitter (AS)[65-67] where the agent was started first time, i.e., a client equipped with AS. Furthermore, strategies can be distinguished by which tasks are transmitted: all tasks code at once or only some tasks. For example, the pull-all strategy means: transmit an agent, start it at the remote site and in case that at least one task is required; download all tasks of the agent from its home/SCCA. Using a push strategy, agent’s tasks can be transmitted to the next BN of the agent’s route or even to all BN visited by the agent. For example, the push-tasks-to-all strategy transmits first some of agent’s tasks (those tasks which are needed potentially at remote BNs) to all BN which are visited by the agent. Missed tasks will be downloaded dynamically. Then the agent is migrated to the first BN of its route. For the next hops, only the agent is transmitted.
The SMPA is used to optimize time and network load caused by a transmission. This is done by calculating the expected transmission times for different migration strategies. The results are compared to select a best fit migration strategy. It allows us to calculate network load and transmission time for migration of a MA from home network, between BN of its route and back home. For the computation, it takes in account an agent’s size (state, data, and tasks), data which is collected on its route (increases with a constant factor) and connection qualities (latency and bandwidth). Thereby, a task is used at a remote BN with a certain probability. The data collected by an agent increases by a non constant size and might be transmitted back home from a BN on the agent’s route. There are some technical problems to determine the actual size of an agent at runtime. For the comparison of different migration strategies, this size is constant and needs not to be involved in the computation. The same holds for the collected data. Hence, the SMPA compares the transmission time for the tasks of an agent. The number of tasks and the point in time of transmission differs for different strategies. Possible requests for task downloads have to be taken in account. For details of computing time refer Section {5.5.3}

5.3.8 SAP Code Container Agent (SCCA)

It contains all tasks of an agent. The BN is used to serve as SCCA. Such a server can be used by an agent to download tasks instead of downloading from home site (BS). An automatic SCCA initialization might be useful in a case where it takes more time to download tasks from the home site than from a near SCCA with a fast connection. Such a SCCA is the code base for further migrations as long as there is a good connectivity. This is useful only for pull strategies (downloading tasks code dynamically). The optimization is simply based on a comparison of migration times with and without a SCCA initialization. With a low optimization degree, the module compares the migration time with the home site as a SCCA and with a local SCCA on the current BN. A medium optimization degree is reached, if all available SCCAs are taken into account. As a variation of the low degree optimization, the migration times for further migrations with a dynamic SCCA initialization are computed (high optimization degree).
5.3.9 SAP Result Container Agent (SRCA)

It is used by an agent to upload collected data instead transmitting data to the home site (BN/BS). The initialization of a SRCA depends on whether an agent wants to transmit collected data to home site (BN/BS). Collected data loads the network again and again when the agent migrates. Where a MA does not need this collected data for further computations, the data should be sent home site (BN/BS). SMPA computes that whether it is cheaper to initialize a SRCA to upload data instead of using home site (BN/BS) to upload data. Automatic data upload variant calculates the migration time to the next BN, if all data is carried along with the agent. The result is compared with the time to upload collected data and to migrate without unnecessary data. An agent can initialize code and SRCAs on its route. With this extended network model, the effort and the advantage of initializing and using code and SRCAs can be computed. The introduced optimization variants are some approaches to reduce network load and migration time of a MA. BN is also used as SRCA.

5.4 Optimal Migration Strategic Algorithm

The basic process for an optimization is simple—calculate migration times of different migration strategies. Compare results and choose best migration strategy. A MA might use the SMPA to compute an optimal migration strategy regarding migration times for parts of its route or even for the whole route. A simple variant is to optimize the next hop only by comparing the migration strategies push-all-to-next, pull-all, pull-tasks and push-all-to-all. A agent wants to hop from BN processing element $PE_i$ to $PE_{i+1}$. The agent’s home site is $PE_0$ (home processing element) client’s node (BS). The algorithm looks like this:

/*Calculate transmission times*/

/*Push-all-to-next: Transmit tasks to next BN*/

$T_{-\text{patn}} = \text{delay}(PE_i, PE_j) + \text{Task\_size/bandwidth}(PE_i, PE_j)$;

/*Pull-all: Download tasks from home site (BS/BN) at next BN*/

$T_{-\text{pa}} = \text{delay}(PE_0, PE_j) + \text{Task\_size/bandwidth}(PE_0, PE_j)$;

{Pull-task: Download each task from home site (BS/BN) at next BN}
\[ T_{-pt} = \text{delay}(PE_0, PE_j) + \frac{\text{SUM}(\text{Probability}(k) \cdot \text{Task}(k) + \text{Request})}{\text{bandwidth}(PE_0, PE_j)}; \]

/* Only for the first hop: push-all-to-all: and Distribute tasks from home site (BS/BN) to all BN*/

for \( s \) in servers
{
\[ T_{-pata} = T_{-pata} + \text{delay}(PE_0, PE_s) + \frac{\text{Task\_size}}{\text{bandwidth}(PE_0, PE_s)}; \]
}

/*Select migration strategy*/

\[ T_{-min} = T_{-patn}; \]
\[ MS = "push-all-to-next"; \]
if ( \( T_{-pa} < T_{-min} \))
{
\[ T_{-min} = T_{-pa}; \]
\[ MS = "pull-all"; \]
}
else if ( \( T_{-pt} < T_{-min} \))
{
\[ T_{-min} = T_{-pu}; \]
\[ MS = "pull-tasks"; \]
}
else ( \( T_{-pata} < T_{-min} \))
{
\[ T_{-min} = T_{-pata}; \]
\[ MS = "push-all-to-all"; \]
}

The migration strategy push-all-to-all can be used only at the home site (BS/BN). From there, all tasks are transmitted to all BN visited by the agent. Then, only the agent needs to be transmitted between the BN of the route. No additional tasks are necessary. A special case is also the last hop of a MA. This is the migration back to the home site (BS/BN). Thereby, the collected data and the agent are transmitted only. Thus, there is no optimization for this hop.
A similar optimization variant is to optimize the migration for more than one hop (not only for the next hop). The computation of transmission times is made for all migrations. Thereby, the migration strategy is fixed for all hops. This method can be improved, if the migration strategy is not fixed at all. The complexity of the computation is increased for this method.

5.5 Evaluation Of The System

We evaluate the performance of the MACP and CSCP through both mathematical analysis and simulation. We need to realize that the MACP might not always perform better than the CSCP since MAs also introduce overhead, which mainly comes from the agent creation and dispatch time. However, for the CSCP, there will be increased queuing delays as the number of clients increase. As a result, it may cause longer processing delays and more potential drops at the server side. Unfortunately, in WSNs, the number of nodes could be hundreds or even thousands.

5.5.1 Performance Metrics

We choose two metrics, the execution time and the energy consumption, to evaluate the performance of the CSCP and MACP models in collaborative processing. The execution time is defined as the time spent to finish a processing task. Parameters and assumptions are shown in Table 5.1. Since the number of BNs \( n_b \) in comparison to SNs \( n_s \) is very less. For the CSCP, the total execution time

\[
t_{ex} = \left( n_{ma} * n_{mam} * s_m / r_n \right) + 2 * \left( n_{ma} * n_{mam} * o_m \right) + \left( n_{ma} * n_{mam} * s_m / r_p \right) \ldots (5.1)
\]

Where the data transfer time is \( t_{tr} = \left( n_{ma} * n_{mam} * s_m / r_n \right) \), the overhead time is \( t_{oh} = 2 * \left( n_{ma} * n_{mam} * o_m \right) \) (assuming the time used to read and write the message is the same), and the data processing time is \( t_{pr} = \left( n_{ma} * n_{mam} * s_m / r_p \right) \).

For the MACP the total execution time

\[
t_{ma} = \left( \left( n_{ma} + n_{mam} \right) * s_m / r_n \right) + 2 * \left( \left( n_{ma} + n_{mam} \right) * o_{ma} \right) + \left( \left( n_{ma} + n_{mam} \right) * s_{ma} / r_p \right) \ldots (5.2)
\]

where the time used to transfer the agents is \( t_{tr} = \left( \left( n_{ma} + n_{mam} \right) * s_{ma} / r_n \right) \) since it takes \( \left( n_{mam} * s_{ma} \right) / r_n \) for \( n_{ma} \) MAs to migrate among the \( n_s \) (\( n_b \) and \( n_m \)) SNs.
simultaneously and it takes \((n_{ma} \ast s_{ma})/r_n\) additional times for the BN to receive the agents in sequence after they finish the migration, the agent overhead time is 
\[ t_{oh} = 2 \ast \left((n_{ma} + n_{man}) \ast o_{ma}\right) \] as it takes \(2 \ast (n_{ma} \ast o_{ma})\) for the AMS to dispatch and receive \(n_{ma}\) MAs and \(2 \ast (n_{man} \ast o_{ma})\) for all the local SNs to send and receive each MA, and the time used to execute the processing code locally is 
\[ t_{pr} = \left(n_{ma} + n_{man}\right) \ast s_{ma} / r_p. \]

In order to obtain a more realistic estimation for the data transfer time, we develop simulation models to obtain data transfer time \(t_p\). Similar to the formulation of the execution time, the energy consumption for the two computing paradigms on three components, namely, energy consumed in data transfer \(e_{tr}\), overhead processing \(e_{oh}\), and data processing \(e_{pr}\). Since no matter where the data processing is taking place, be it at the local SN or the BN the energy consumed for the entire sensor network is the same for both computing paradigms we choose to neglect \(e_{pr}\) the same reason, we do not include the energy used for sensing either. The model used to calculate the energy usage is:

\[ e_{cs} = e_{trcs} + 2 \ast \left(n_{ma} \ast n_{man} \ast p_c \ast o_{ma}\right) \] \hspace{1cm} \cdots \hspace{1cm} (5.3)

\[ e_{ma} = e_{трma} + 2 \ast n_{ma} \ast (n_{man} + 1) \ast p_c \ast o_{ma} \] \hspace{1cm} \cdots \hspace{1cm} (5.4)

### 5.5.2 Analysis of the Algorithm

The SMAPA is used by a MA to locate services and to access information on network connection qualities. Connection qualities are especially important for the SREA and the SMPA to achieve optimizations. SMAPA cares for precise and up-to-date knowledge (maps) within its local domain and provides a rough, summarized view of the linked remote BNs. It consists of several network servers for computation. A MA is able to locate points of interest within the network of BN. Once a list of interesting BN has been determined, another system component - the SREA can be used by the agent to plan a route. As long as we have a route, a MA may also use a so called SMPA.
Table 5.1: Assumption and Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time spent in data transfer and receiving</td>
<td>$t_{dp}$</td>
</tr>
<tr>
<td>Overhead</td>
<td>$t_{oh}$</td>
</tr>
<tr>
<td>Processing time</td>
<td>$t_{pr}$</td>
</tr>
<tr>
<td>Data transfer rate</td>
<td>$r_{p}$</td>
</tr>
<tr>
<td>Data processing rate</td>
<td>$r_{p}$</td>
</tr>
<tr>
<td>Message size (the size of raw data that each node collects and which is processed and transmitted by SA)</td>
<td>$s_{ma}$</td>
</tr>
<tr>
<td>MA size</td>
<td>$s_{ma}$</td>
</tr>
<tr>
<td>Overhead of message (time used to read and write a message into a file)</td>
<td>$o_{ma}$</td>
</tr>
<tr>
<td>Overhead of MA</td>
<td>$o_{ma}$</td>
</tr>
<tr>
<td>Number of SNs</td>
<td>$n_{s}$</td>
</tr>
<tr>
<td>Number of agents</td>
<td>$n_{pa}$</td>
</tr>
<tr>
<td>Number of SNs that each agent migrates to (processing center BS/BN is not included)</td>
<td>$n_{man}$</td>
</tr>
<tr>
<td>The power consumption of the node processor in full load.</td>
<td>$P_{C}$</td>
</tr>
<tr>
<td>Simulation result for MACP</td>
<td>$e_{mac}$</td>
</tr>
<tr>
<td>Simulation result for client CSCP</td>
<td>$e_{acs}$</td>
</tr>
<tr>
<td>Number of BNs</td>
<td>$n_{c}$</td>
</tr>
<tr>
<td>Latency</td>
<td>$\hat{e}$</td>
</tr>
<tr>
<td>Available bandwidth between BN</td>
<td>$\tau$</td>
</tr>
<tr>
<td>Size of all task</td>
<td>$B_{C}$</td>
</tr>
<tr>
<td>Amount of time for push-all-to-next</td>
<td>$T_1$</td>
</tr>
<tr>
<td>Amount of time for push-all tasks</td>
<td>$T_2$</td>
</tr>
<tr>
<td>Computation time for pull tasks</td>
<td>$\hat{T}$</td>
</tr>
<tr>
<td>Size of kth task code</td>
<td>$B_{C}^k$</td>
</tr>
<tr>
<td>Size of code which will be downloaded for certain task</td>
<td>$B_{r}$</td>
</tr>
<tr>
<td>Bandwidth of i-th node</td>
<td>$\tau_{i}$</td>
</tr>
<tr>
<td>Ratio of requested packet/total packet available</td>
<td>$u$</td>
</tr>
<tr>
<td>Reciprocal value of $\tau_{ii}$</td>
<td>$a_{ii}$</td>
</tr>
<tr>
<td>Path matrix for i-th route</td>
<td>$P_{i}$</td>
</tr>
<tr>
<td>Shortest route between two nodes</td>
<td>$b_{ii}$</td>
</tr>
</tbody>
</table>

For computing the overhead on the network we have taken few assumptions which are {details about assumptions are shown in Figure 5.1} - Total Nodes = $n_s$, Nodes of Interest = $n_i$, Packets per Node = $n_p$, Size of Packet = $s_p$, Requested Packet = $r_p$, $u$ = Requested Packet $r_p$ / Total Packets Available ($n_p$),
and Traffic due to one node = \( n_p \cdot s_p \). Thus, Total Traffic due to all the nodes of interest can be given by 
\[
(T_{n_i}) = \sum_{i=1}^{n} n_p s_p
\]
and Traffic due to relevant Packet on network will be 
\[
(T_{r_p}) = u.s_p.
\]
Overhead \((o_{SMAPA})\) can be defined as (Total Traffic due to all nodes of Interest - Traffic due to Relevant Packet on network) is given by
\[
\sum_{i=1}^{n} n_p s_p - u.s_p 
\]

SREA component is able to calculate the shortest trip through the network based on the map data. This module may be used by MAs to optimize the sequence of BN to visit, i.e., the route. Route is based on the map data. This component uses classic local optimization algorithms. Minimum path can be found out with the help of route matrix. SAPT is calculated simply by using the reciprocal values of measured bandwidth and this matrix has to be updated dynamically. A path finder algorithm is applied in order to get a SAPT with shortest paths between two places. We have assumed \( \tau_i \rightarrow \) be measured bandwidth of \( i^{th} \) BN and where \( i = 1 \) to \( n_b \), the SAPT distance matrix \( D_i \) for \( n_b \) BNs network is given by

\[
D_i = \begin{pmatrix}
    a_{11} & a_{12} & \ldots & a_{1n_b} \\
    a_{21} & a_{22} & \ldots & a_{2n_b} \\
    \ldots & \ldots & \ldots & \ldots \\
    a_{n_b1} & a_{n_b2} & \ldots & a_{n_b,n_b}
\end{pmatrix}
\]

Thus, \( a_{ii} = 1/\tau_{ii} \) where \( a_{ii} \) is calculated from the reciprocal value of \( \tau_{ii} \).

Where \( a_{ii} \)'s stands for places or point of interest for which we find out the shortest route. Assume \( P_i \) be a SAPT that is find out from \( D_i \) by applying a path finder algorithm. Elements of \( P_i \) gives us the shortest path between two places and \( b_{ii} \)'s are the elements of path matrix \( P_i \). \( b_{ii} \)'s stands for shortest path between two BNs.

\[
P_i = \begin{pmatrix}
    b_{11} & b_{12} & \ldots & b_{1n_b} \\
    b_{21} & b_{22} & \ldots & b_{2n_b} \\
    \ldots & \ldots & \ldots & \ldots \\
    b_{n_b1} & b_{n_b2} & \ldots & b_{n_b,n_b}
\end{pmatrix}
\]

SAPT is not symmetrical because of varying bandwidth values and non-symmetrical connections measured by the SMAPA. As all of the working of
SREA is done locally so no traffic is present on the network to go from one part of network to another only algorithms are used for local calculations of path so we assume the overhead is negligible in this case.

During execution, an agent consists of three parts: the agent’s state and data and its set of tasks. As long as we have a route, a MA may also use a SMPA to optimize each single migration included in the route. SMPA is mainly designed to reduce network load by selecting and transmitting only those code and data portions of the agent that are needed at the upcoming remote BN. This is also used to optimize time during transmission. Network load is the traffic on the network due the migration of MAs. Total traffic and overhead is calculated as given below so that we optimize the network load. MA carries its code and state across the network. For each hop \( j \) traffic is \( T_{SMPA}^j \).

Traffic generated due SMPA is given by \( T_{SMPA}^j = r_p c_{ma} s_{ma}^j s_p \), where \( r_p \rightarrow \) Requested Packet, \( c_{ma} \rightarrow \) Code for MA, \( s_{ma}^j \rightarrow \) Size of the state of agent at hope \( j \), and \( s_p \rightarrow \) Size of a Packet. Thus, size of state of the agent is given by \( s_{ma}^j = d_{list} + \omega + \sum_{i=0}^{j} u s_p \), where \( d_{list} \rightarrow \) Size of the list, \( \omega \rightarrow \) Size of the other internal data structure representing the state of computation \( \sum_{i=0}^{j} u s_p \) indicate the useful information collected by the agent at each, visited nodes. \( d_{list}, u, s_p \) and \( \omega \) do not depend on the node and for simplicity \( \overline{\omega} = d_{list} + \omega \).

Thus, Overall Traffic

\[
T_{SMPA} = \sum_{j=0}^{n_i} (r_p s_p + c_{ma} + \overline{\omega} + \sum_{i=0}^{j} u s_p)
\]

and Traffic Overhead are given by

\[
O_{SMPA} = T_{SMPA} - T_{n_i},
\]

i.e.,

\[
(r_p s_p + c_{ma} + \overline{\omega})(n_i + 1) + (1/2)(n_i + 1) - n_i = (r_p s_p + c_{ma} + \overline{\omega})(n_i + 1) + (1/2)(n_i - 1)
\]
5.5.3 Computing Migration Time

In more detail, a computation of the migration time for different migration strategies for a hop is done according to the following scheme: A agent wants to hop from BN $PE_i$ to $PE_{i+1}$. The agent’s home site is $PE_{0}$ client’s node (BS). The latency between two BN is defined by the function $\delta$. Function $\tau$ denotes the available bandwidth between two BN. The amount of bytes which will be transmitted is $B_c$ (size of all tasks) for amount of time is required push-all-to-next is

$$T_1 = \delta(PE_i, PE_{i+1}) + B_c/(\tau(PE_i, PE_{i+1}))$$

and for pull-all is

$$T_2 = \delta(PE_0, PE_{i+1}) + B_c/(\tau(PE_0, PE_{i+1}))$$.

Furthermore, it is difficult to determine the probability for the usage of a certain task at a remote BN it is not designated if it is designated it can be very easily traced with the database mapping. Thus, we decided to use the worst case assumption that every task has to be downloaded as long as we do not have any other options. A time computation can be made by pull tasks

$$T = \sum_{k=1}^{n} \delta(PE_0, PE_{i+1}) + (B_c^k + B_c)/\tau(PE_0, PE_{i+1})$$,

where $B_c^k$ is the size of the $k^{th}$ task code of the agent. $B_c$ denotes the size of a request for downloading a certain task code.

5.6 Implementation and Performance Study

We have designed several experiments to evaluate the CSCP and MACP based WSN for two metrics execution time and energy consumption.

It is observed from Figure 5.3 that the execution time using both computing paradigms grows as the number of nodes increases. But the CSCP grows much faster than the MACP. This is because as the number of nodes increases, the BS has to deal with more connections requested by the clients at the same time, which extends the execution time. MACP BS is distributed its itinerary computation work in the BNs of the network.
On the other hand, the MACP is less influenced by the number of nodes because there are fewer connections at one time. The simulation also shows that, for $n_s \leq 24$, the CSCP performs similar to MACP. This happens because the MACP needs more connections than the CSCP in order to send and receive MAs. Another reason is that the overhead of the MACP surpasses that of the CSCP.

From Figure 5.4 it is clear that the total energy consumption in the network for the MACP almost always less in comparison to CSCP. Because the amount of data transmitted is across the network is lesser for MACP.
When we fix the number of nodes in the network to 72 and assume that each agent migrates to the same number of nodes, we can evaluate the effect of the number of MAs. Figure 5.5 shows that the execution time of the MACP increases very little as the number of MAs increases and reaches the lowest point when there are two MAs. Then the execution time begins to increase. This is because more MAs will reduce the number of nodes each agent migrates to, thus reducing the execution time. But more MAs also cause more connections and overheads.

![Figure 5.5: The effect of the number of MAs on execution time for the completion of task](image)

When we fix the number of nodes in the network to 72 and assume that each agent migrates to the same number of nodes, we can evaluate the effect of the number of MAs. The performance curve in Figure 5.6 for CSCP is constant since it is irrelevant to the number of MAs. We have also observed from Figure 5.6 that the MACP always has less energy usage than the CSCP because the number of nodes in the network is large.
Figure 5.6: The effect of the number of MAs on total energy consumption in the network.

In the next experiment we have changed the size of the message (data) $s_m$, but fix the other parameters and let $s_{ma} = 2$KB, $n_s = 72$ in order to study the effect of the sensed data size vs. the MA size. We observe from Figure 5.7 and 5.8 that the execution time and energy consumption using the MACP are constant because data are located at the local SNs; only a fixed amount of results are transferred.

Figure 5.7: The effect of the MA size vs. sensed data size on execution time.
When the sensed data size is less than 24KB, the CSCP has less execution time than that of the MACP. However, the larger the sensed data size, the more advantageous is the MACP. This is because as $s_m$ increases, more data need to be transferred, thus increasing the time used by the CSCP. The energy consumption shows similar pattern.

Figure 5.9 and Figure 5.10 show an increase in both accuracy and energy when additional agents are used. This is to be expected as more agents should generate more messages, as well as the potential to deliver more useful data. The increase in energy expenditure does achieve a considerable increase in the amount of data delivered to the BS. Also in Figure 5.9 and Figure 5.10 trend are linear and suggests that the ratio of accuracy gain to energy expenditure is likely to be favorable, offering the potential that designers could tune the approach to their own accuracy or resolution requirements. AMS avoid the elapsed time to return results to save energy and this is measured and shown in Figure 5.11.

![Figure 5.8: The effect of the MA size vs. sensed data size on total energy consumption in the network](image-url)
Figure 5.9: The effect of varying the number of agents deployed on the average energy usage per node for an event covering approximately 20% of the sensed area.

Figure 5.10: The effect of varying the number of agents deployed on the overall accuracy for an event covering approximately 20% of the sensed area.

Figure 5.11: Average elapsed time before the event data was delivered to the BS compared to the number of agents deployed.
In the following experiments, we have studied the effect of size of sensed data. We change the size of the sensed data at each sensor from 0.5KB to 4KB in 0.5KB interval. Figure 5.12 shows that the energy consumption of CSCP is always larger than that of MACP when sensed *data* is varied. Figure 5.13 shows that when sensed *data* is greater than 1.7KB CSCP has a longer end-to-end delay. This happens because MACP saves transmission overhead of each source nod. Thus, the larger sensed *data* is, the more efficient is the MACP. The above simulation results show that the MACP for WSN does not always perform better than the CSCP for WSN. We observe that the end-to-end delay performance varies under different conditions, but in most cases, energy consumption for MACP for WSN is lower than energy consumption for CSCP for WSN. Thus, for the scenarios where energy consumption is of primary concern, the MACP exhibits substantially lower energy consumption and hence potentially longer network lifetime than the CSCP.

![Figure 5.12: Energy Consumption](image-url)
5.7 Results and Discussion
The simulation results also show that the MACP does not always perform better than the CSCP and in different scenarios, the energy consumption is usually the more contingent resource. However, in the context of sensor networks with hundreds or even thousands of SNs, unreliable communication links, and reduced bandwidth, the MACP provides solutions to energy efficient collaborative processing with less execution time. On the other hand, when the number of SNs in a network is small, the MACP suffers longer network latency because of its bigger overhead.

5.8 Summary
In this chapter we have presented Architecture of Agent Management System (AMS) of SAP system for WSNs. AMS provides energy efficient services using set of agents. In AMS mainly MAs are used to manage the WSNs. It provides common solution to WSNs fault tolerance and load balancing problems and gives true distributed computing and communication environment with the help of MAs. AMS supports code mobility over the mobile/fixe BS and SNs. We have also presented comparative study of the MACP and CSCP for WSNs. Results show that MACP improves the performance of WSNs when number of SNs in network is very large. In the next chapter MA based an adaptive and hierarchical system for information fusion in WSNs is presented.