Adaptive Service Discovery on Service-Oriented and Spontaneous Sensor Systems

Lu Liu[1], Jie Xu[2], Nick Antonopoulos[1], Jianxin Li[3], Kaigui Wu[4],
[1]School of Computing and Mathematics, University of Derby, Derby, UK
[2]School of Computing, University of Leeds, Leeds, UK
[3]School of Computer Science and Engineering, Beihang University, Beijing, China
[4]College of Computer Science, Chongqing University, Chongqing, China
Email: l.liu@derby.ac.uk

Abstract

Natural and man-made disasters can significantly impact both people and environments. Enhanced effect can be achieved through dynamic networking of people, systems and procedures and seamless integration of them to fulfil mission objectives with service-oriented sensor systems. However, the benefits of integration of services will not be realised unless we have a dependable method to discover all required services in dynamic environments. In this paper, we propose an Adaptive and Efficient Peer-to-peer Search (AEPS) approach for dependable service integration on service-oriented architecture based on a number of social behaviour patterns. In the AEPS network, the networked nodes can autonomously support and co-operate with each other in a peer-to-peer (P2P) manner to quickly discover and self-configure any services available on the disaster area and deliver a real-time capability by self-organising themselves in spontaneous groups to provide higher flexibility and adaptability for disaster monitoring and relief.

Key words: service-oriented architecture, spontaneous networks, self-organisation, self-configuration, sensor systems, social patterns

1. Introduction

Natural and man-made disasters can significantly impact both people and environments. For example, in the 2010 Haiti Earthquake, the Haitian Government reports that between 217,000 and 230,000 people had been identified as dead, an estimated 300,000 injured, and an estimated 1,000,000 homeless. Rescue efforts began in the immediate aftermath of the earthquake with great support from international communities. Real-time monitoring data and information is crucial for the success of disaster relief efforts from international communities.

Sensor networks have the potential to revolutionize the capture, processing and communication of critical data for use of disaster rescue and relief [12]. The functions of sensors need to be integrated to provide a joint service to meet different search and rescue requirements. For the provision of a dependable rescue capability in a dynamic and unpredictable disaster area, the networked sensors should have ability to autonomously support and co-operate with each other to quickly configure any services available on the disaster area to deliver a real-time capability, self-adaptability to modify their behaviours to deliver a sustainable capability according to environmental changes, and ability to share information, generate access and protect information throughout the network.

For provision of dependable capability, the networked nodes should have the ability to:

- autonomously support and co-operate with each other in a peer-to-peer (P2P) manner to quickly discover and self-configure any services available on the disaster area to deliver a real-time capability;
- modify their behaviours to deliver a sustainable capability according to environmental changes;
- self-organise themselves in real time to generate higher flexibility and adaptability for disaster management systems and form groups spontaneously;
- share information and generate access throughout the network.

In this paper, we propose an Adaptive and Efficient Peer-to-peer Search (AEPS) approach for distributed service discovery for dependable service integration based on a number of social behaviour patterns, which demonstrates the desirable functionalities listed above. AEPS has no single point of failure, ensuring greater dependability and availability. AEPS is able to efficiently discover desirable services for decision making of disaster monitoring and relief by interacting with connected nodes with incomplete information and to support dependable dynamic service integration through coping with rapid and significant changes in the disaster area. AEPS builds a "social" network for each sensor node which contributes to effective service discovery.

This rest of paper is organised as follows: Related Work on which the AEPS is based is presented in Section 2. The architecture for dependable service integration for the reliable provision of rescue capability is introduced in Section 3. The AEPS algorithm and associated social behaviour patterns are described and discussed in Section 4. Evaluation of AEPS and other techniques are given in section 5. Demonstration System Development is outlined in section 6. Finally, the summary is given in section 7.

2. Related Work

2.1 Resource Discovery in P2P Networks

Decentralised resource discovery is a fundamental challenge for large-scale P2P networks. Blind flooding is the simplest but the most commonly used method for recourse discovery in P2P networks, which spreads queries to many peer nodes to find a requested file. However, the huge volume of traffic generated by the blind flooding is a serious problem observed in existing P2P networks [18] (e.g. Gnutella), which is an obstacle for further growth of P2P networks. In the P2P architecture, peer nodes can directly access and exchange shared resources. P2P represents a fundamental decentralisation of control mechanisms, which is usually achieved by using a Distributed Hash Table (DHT) method. DHT has become the dominant methodology for resource discovery in structured P2P networks [1], which builds a connected ideal overlay network across the Internet. By using a DHT method, a protocol is specified that allows a peer node to join or leave the network by rearranging the overlay into account for their presence and absence. Unfortunately, DHT ignores the fact that a P2P network is a continuously evolving system. DHT can work well when joins happen sequentially, but can not handle if joins happen concurrently or the faults accumulate over time [9]. The ideal overlay is no longer ideal, once those faults occur.

DHTs provide efficient resource discovery in P2P networks which only need a small number of messages to resolve a query. However, efficiency not only relies on the number of messages to resolve a query, but also the number of messages to maintain the networks. DHTs are not efficient in a dynamic environment, which require a large number of messages to maintain the network topology to keep consistent hash tables at each peer node, although queries can be resolved by a few messages [17, 26]. Any attempts of additional control could be difficult to achieve in the distributed P2P architecture due to the lack of a centralised server. In contrast, self-organisation could be a proper way to solve the control issues in the decentralised P2P architecture which will be discussed in detail in this paper.

2.2. Resource Discovery in Social Networks

In social networks, individual people or aggregate units (such as department, organisation or family) are usually analysed as actors [4]. A social network is a set of actors and the relations that hold them together. These actors exchange resources (e.g. information, goods or social services), which then connect them in a social network.

Watts [23] presented that individuals in social networks were endowed not only with network ties, but also with identities: sets of characteristics which they attribute to themselves and others by virtue of their associations with social groups. Query messages can be directed efficiently with the combined knowledge of network ties and social identities. Watts also showed that individuals in social networks cluster the world hierarchically into a series of layers, where the highest layer accounts for the entire world and each successively deeper layer represents a cognitive division into a greater number of increasingly specific groups.

Social networks were not only investigated as a whole by using sociocentric approaches [4, 23], but also explored with egocentric approaches by analysing each person and his/her connections to other people [15]. A personal network is a special kind of social network that is centred around a person [15]. Generally, a personal network is a set of people that are preferably contacted by an individual person to get information or advice.

Newcomb [16] indicated that one of the primary differences between the behaviour of the newborn infant and that of an adult was that random and diffuse behaviours gradually became more highly organised. Adults' behaviours were mobilised in a selective manner toward specific goals. People tended to manipulate circumstances, so that they can benefit in socialising with the people they choose [16]. The personal network was usually used for personal own benefit. For example, when a person was assigned to a project with a topic, the first thing he/she usually would do was to contact people who know the topic or who can provide background information or gave advice on how to solve his/her problems [21].

The previous study in [7] showed that one of the most effective channels for disseminating of information and expertise as well as finding information within an organisation was his/her social network of collaborators, colleagues and friends. Searching for a piece of information in social networks was most likely a matter of searching for an expert on the topic together with a chain of personal referrals from the searcher to the expert [7].

3. Architecture for Dependable Service Integration

The rapid growth of information services and resources in disaster management systems makes it difficult and challenging to organise them efficiently. A rescue capability is the operation of integrated services to fulfil a rescue mission objective. In order to provide a dependable rescue capability through dynamically integrating businesses, systems and computing, new methodologies are required for the dependable integration of services in heterogeneous environments.

Different types of services across different platforms [10] need to be integrated to provide a joint rescue capability. For example, sensors for detecting survivors could employ any of laser, visible spectre, heat (infrared), radar or sonar sensors and these heterogeneous sensor services can be deployed on recue helicopters, UAVs (unmanned aerial vehicles) and search and rescue personnel. The sensors should provide interoperable service interfaces to the integrating applications taking into account quality of services (QoS) parameters to deliver dependable capabilities in highly critical situations.

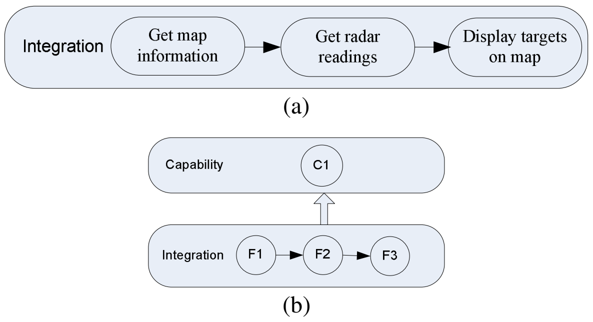


Figure 1: Workflow of service integration for delivery of rescue capabilities

In a network of sensors, a number of radar sensors supply data through services. The network of sensors is modelled conveniently as a dynamic network of services, facilitating ongoing changes. In the modelled system, a surveillance user can submit real-time requests to the system for information of Points of Interest (POIs) in a specified region. A sequence of services (such as "Get map information" and "Get radar reading", "Display targets on map") can be operated in a workflow in order to provide a regional surveillance service, like the one illustrated in Figure 1(a). The service integration can be abstracted by using workflow patterns as shown in Figure 1(b), where function *F1* represents the service of getting map information, *F2* represents the service of getting radar reading and *F3* represents the service of displaying targets on map.

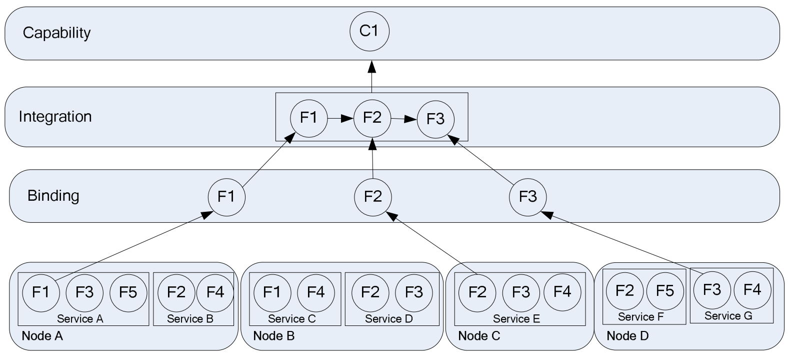


Figure 2: Service-oriented architecture for the delivery of rescue capability

Sensor networks consist of potential a large number of nodes with sensing and wireless communication capability. Each platform node could have one or more sensors supplying data through services. Figure 2 illustrates the integration of services for the delivery of rescue capability on service-oriented architecture (SOA) [10]. In this architecture, each platform node has a number of services, each service performs a set of functions, and some functions can be integrated to form a higher level of functionality to deliver a capability. For example, a rescue helicopter is a platform node which provides services, such as movement, surveillance, and patient delivery services. The surveillance service includes functions for metrological surveillance, situation surveillance and target surveillance. The metrological surveillance function may be combined with other functions to form a higher level weather service that contributes to search-and-rescue missions.

Redundant service binding is a technique to improve the reliability of the provision of rescue capabilities [10]. In order to deliver a reliable rescue capability, the required functions need to be provided by multiple services allocated to different peer nodes. The reconfiguration algorithm can switch to one of backup services in case of failure of initial service. The distributed recovery block (DRB) scheme [8] is applied to minimise the recovery time of integration.

Apart from failures of services, rescue capability evolution could cause potential problems affecting the reliability of provision of rescue capabilities. Requirements often change during a rescue and search operation, in accordance with changes of situation. The rescue capability could be evolved according to changes of environment and users' needs. A reconfiguration algorithm which is able to proactively self-diagnose the evolution, evaluate the impact of evolution and self-configure services to adapt to capability was discussed in [11] to address the issue of changes of requirements.

The above previous research work provides a number of architectural solutions to handle the changes and unknown for provision of dependable and sustainable rescue capability. However, the benefits of integration of services will not be realised unless we have a dependable method to discover all required services in dynamic environments. In the traditional implementations of Service-Oriented Architecture (SOA), a centralised UDDI (Universal Description Discovery and Integration) registry provides the functionalities to advertise and discover services, which centralises all responsibilities for publishing and handling enquiries about services. In disaster management systems, the centralised registry can be considered a single point of failure. Once the service registry failed, all the information about registered services is unavailable. The problem of single-point-of-failure could be more significant in this case, since the service registry is also threatened by disaster attack. Moreover, data links for access to information on remote repository cannot always be guaranteed in the unknown and dynamic disaster area. Additionally, the frequent remote communication could generate too much traffic to the network and the connection to remote registry could be overloaded if too many requests are received at the same time.

In order to dependably integrate services for disaster monitoring and relief, the worst conditions must be considered as the failure of the centralised registry or the failure of the connection to the remote centralised registry. Therefore, each network node should have the capability to efficiently discover desirable services in real time by interacting with connected nodes. In this paper, we propose the AEPS method which is an adaptive and efficient P2P search approach for distributed service discovery for dependable service integration based on a number of social behaviour patterns, which will be discussed from the next section.

4. Dynamic Service Discovery

AEPS is an efficient unstructured P2P search mechanism by self-organising autonomous nodes with human strategies in social networks. These human strategies are categorised into six patterns which are introduced in this section. In the AEPS system, network nodes should have the ability to self-organise themselves in a cooperative manner similarly to social networks. This can improve the adaptability and agility of system.

For resource discovery in social networks, people can directly contact some acquaintances that potentially have knowledge about the services they are looking for. Similarly, a "social" network is also important for disaster management systems, which enables each network node to route queries efficiently and timely. The similarity between communication networks and social networks, where network nodes can be considered as people and connections can be considered as relationships, leads us to believe that theories of social networks are useful for improving the performance of service discovery in communication networks. In this section, a social model is introduced for adaptive and efficient service discovery for dependable service integration by mimicking different human behaviours in social networks. Six social patterns are created based on the social theories that can be used in the service discovery for provision of capability as described in section 2.

4.1 Social Pattern 1: Index Creation

People in social networks can remember and update potentially useful knowledge from social interactions, and then random and diffuse behaviours gradually become highly organised [16]. For example, if a man named Bart successfully borrows an Oxford English dictionary from his friend Jay, this successful interaction will be remembered by Bart. When Bart needs this dictionary again, he will preferentially seek help directly from Jay rather than other people. However, if Jay does not have the Oxford English dictionary and cannot provide any useful information to help Bart find this dictionary, Bart would avoid contacting Jay if he needs the dictionary again in the near future.

Similarly to people in social networks, each AEPS node builds a knowledge index that stores associations between functions and the related nodes by the results of searches. Each knowledge index includes two sub-lists: a friend list and a black list. In the friend list, network nodes that have successfully responded to the query are associated with the requested functions. The query originator will put a neighbouring node into the black list associated with the query function, if no results on this function are found by the node nor its successor(s).

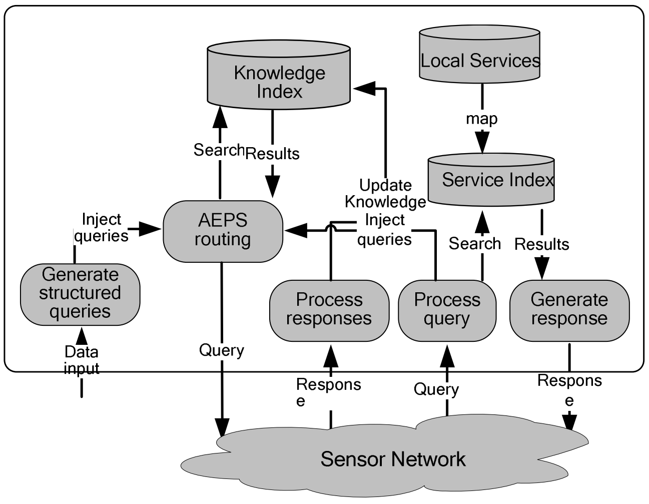


Figure 3: Query initialisation and processing

AEPS uses a TTL (Time-to-Live) to prevent an infinite propagation. TTL represents the number of times a message can be forwarded before it is discarded. In the AEPS network, a newly joined node will send its first query to a set of randomly selected nodes. As shown in Figure 3, if a search is successful, the query originator updates its friend list to associate the nodes that have responded successfully to the query with the requested function. The newly obtained knowledge is stored in the local friend list. In the meantime, the query originator removes invalid cached knowledge in the local friend list according to the results of searches. The black list is also updated with search results. If the query originator sends a request to a node, but no results about the requested function are found by queries via this node, this node will be put into the black list of the query originator associated with the requested function.

Therefore, network nodes can learn from the results of previous searches, which make future searches more focused. When more searches have been done, more knowledge can be collected from search results. If this process continues, each node can cache a great deal of knowledge that is useful to quickly find the network nodes with the required services in the future.

4.2 Social Pattern 2: Knowledge Update & Query Processing

In social networks, some events with associated people fade from a person's memory with time [2] and a personal network is adjustable with changing environments [3]. In the AEPS network, network nodes can update their knowledge about other nodes from daily search results. Some old and invalid knowledge is replaced by newly obtained knowledge. The invalid knowledge that fails to be updated with daily searches will drop to the bottom of the lists and will be removed by using a LRU policy, when the list reaches a maximum.

Figure 4 illustrates an example of knowledge collection process of a node. Node *A* with an empty knowledge index searches for the services with the function "radar sensing". Since no information is cached, node A will send the query to the randomly selected nodes: node *C*, node *D* and node *E*. Then node *D* further forwards the query to its neighbouring node *F*. In this case, the search is successful at node *F*. Thus, node *F* responds to node *A* with the requested function "radar sensing". Node *A* then associates node *F* with the function "radar sensing" into the local friend list. For a following query on "radar sensing", node *A* can find node *F* directly associated with the query from the local friend list and preferentially send the query to node *F* rather than node *D* and *E*.

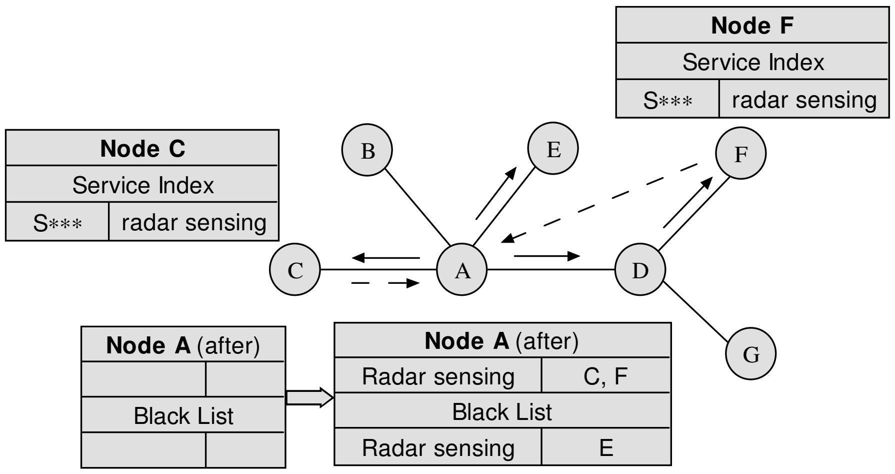


Figure 4: Example of a knowledge collection

As shown in Figure 4, node *A* updates its black list with search results. Node *E,* a neighbour of node *A*, will be added into the black list associated with the requested function "radar sensing", because no results are found by node *E* nor its successors. But node *D* will not be put into the black list, because node *D*'s successor, node *F*, helps node *D* find the requested services successfully. If node *A* does the same search on "radar sensing" again, it will avoid forwarding the query to node *E*, regarding the information in the black list.

When a node generates a query (Figure 4), it will search the local friend list to find some associated nodes. The query originator will prefer to send the query to these nodes found from the friend list and avoid sending the query to associated nodes in the black list. The query originator also attaches a list of “black nodes” associated with the query function regarding the local black list to help message receivers select nodes.

When a node receives a query, it will first check whether the message has been already received. Redundant messages will be discarded without further processing. Then the node will search the local service index to find matched services. If the query needs to be further forwarded ($TTL > 0$), the message receiver will use the local knowledge index to find associated nodes using the AEPS routing algorithm and multicast the query to these nodes as shown in Figure 5.

Analogously to social networks, AEPS utilises a semantic approach to route queries to a selected subset of neighbouring nodes on the overlay. In addition, AEPS also involves a dedicated strategy to address the network overload problem of existing P2P search methods (e.g., Gnutella).

In some existing query routing methods (e.g., [5, 13, 14]), the number of nodes to be forwarded in each hop is set to a static value. A fixed number of nodes are selected in each hop neglecting the probability of finding the requested service in these nodes. In order to conduct a more efficient search in AEPS, the number of nodes to be forwarded is adjustable according to the correlation degree of the selected node to the query between a minimal number $CutOff_{min}$ and a maximum number $CutOff_{max}$. When the correlation degree of a selected node is high, there is a high possibility that the selected node may share a desired service or has the knowledge about who shares the service. The probability of forwarding the query to this node should also be high by defining a high cut-off of node selection. In contrast, if the correlation degree is low, a low cut-off should be set to limit the scope of querying.

4.3 Social Pattern 3: Node Selection

For resource discovery in social networks, people usually recall information in memory to find the right people to contact. The persons recalled from memory may directly relate to their requests. For example, Bart wants to borrow an Oxford English dictionary and remembers that he once borrowed it from his friend Jay. Therefore, he can directly contact Jay again for the dictionary.

However, in most circumstances, people cannot find the persons who are directly related to their requests, but people can find some acquaintances that potentially have knowledge about the resources they are looking for, or can provide background information or give advice on how to find the resources [21]. For example, Bart may never have borrowed or cannot clearly remember whether he has ever borrowed an Oxford English dictionary. But Bart believes his friend Jay, who is a linguist, probably has the dictionary or at least she has more knowledge about who has the dictionary. In this case, the Oxford English dictionary is in the area of linguistics and Bart has found that Jay has abundant knowledge on the interest area of linguistics from previous intercommunications. Jay probably does not have the dictionary, but she will use her own knowledge to help Bart find the dictionary with a high likelihood.

```
Node Selection Algorithm
1.  Get a list of IDs of previously visited nodes by the query
2.  Exclude the previously visited nodes from the node selection
3.  Add the local node ID to the list of previously visited nodes
4.  Decrease the TTL of the query by one
5.  Exclude "black nodes" from the node selection
6.  Select from directly related nodes (first round of selection)
7.  IF $n<CutOff_{max}$ THEN
8.      Select from correlated nodes (second round selection)
9.      IF $n<CutOff_{min}$ THEN
10.         Select random nodes (third round selection)
11.     ELSE
12.     RETURN selected nodes
13.     END IF
14. ELSE
15. RETURN selected nodes
16. END IF
```

Figure 5: Node selection algorithm

The node selection procedure of AEPS is shown in Figure 5, where *n* is the number of nodes that have already been selected in the first and second phase of node selection. Message receivers only trust the information about the "black nodes" provided by the query originator or by their own local black list concerning the security of information. When a node receives a query which needs to be further forwarded, the local routing algorithm will exclude a list of black nodes provided by the query originator from the node selection procedure together with the nodes associated with the requested function in the local black list.

The routing algorithm then starts the three-phase procedure of node selection: searching for the directly related nodes, searching for the correlated nodes and searching for random nodes, from the local friend list. In the first phase, the routing algorithm searches for the nodes directly associated with the requested function from the local friend list and sorts them with the time of receipt. The node that is inputted or updated most recently will be selected first. These directly associated nodes have the greatest likelihood of finding the requested services. Hence, at most $CutOff_{max}$ nodes will be forwarded.

However, the success probability of finding $CutOff_{max}$ directly associated nodes in the first phase is very low, especially for new nodes with little knowledge. If there are not enough directly associated nodes found in the first phase, the algorithm will move to the second phase that searches for the nodes sharing services associated with the interest area of the requested function in the local friend list. An interest area in AEPS is a semantic area with a set of functions. The corresponding interest area of a specific function and the other functions in this interest area can be found from a structured taxonomy, such as Open Directory Categories which is the most widely distributed database with a hierarchical function structure. AEPS users can use the common function hierarchy of the Open Directory to formalise a query.

These nodes sharing services associated with the other functions in the same interest area of the requested function will be sorted according to the degree of correlation to the interest area of the requested function. The routing algorithm prefers to select the nodes with higher degrees of correlation rather than the nodes with lower degrees of correlation. If two or more nodes have the same correlation degree, the node that responded most recently will be put first.

4.4 Social Pattern 4: Search for Experts

Searching for a piece of information in social networks is most likely a matter of searching the social network for an expert on the function together with a chain of personal referrals from the searcher to the expert [7]. In social networks, a person does not need to tell everybody he/she is an expert in the areas which has been indicated with his/her social behaviours.

In the AEPS network, it is not necessary for a node to declare its interest since that has already been implied by its shared services. If a node has a large number of services in a particular area like an "expert", it is very likely that it will also have other services or knowledge on this area. In our simulations, the correlation degree of a selected node in a particular area is generated by how many functions in the area the node is associated with:

$$C = \frac{n_{matches}}{n_{total}}, \qquad (1)$$

where $n_{matches}$ is the number of functions in this area that the node is associated with and n_{total} is the total number of functions in this area.

Different from any other P2P search algorithm, the cut-off criteria R for the second phase are adaptable to different nodes between $CutOff_{min}$ and $CutOff_{max}$. The cut-off criterion R of node with respect to the query is determined by the correlation degree of the node to the interest area of the requested function with the equation:

$$R = Round\left(C \cdot \left(CutOff_{\max} - CutOff_{\min}\right)\right) + CutOff_{\min}, \quad (2)$$

where the function *Round*(x) returns the closest integer to the given value *x*. When the correlation degree of a node is very low ($C \approx 0$), there is a low likelihood to find the requested services from the node. Therefore, the probability of querying the node should be low with a low cut-off ($R \approx CutOff_{\min}$). In contrast, when the selected node is highly correlated with the area of the requested function by matching most functions in this area ($C \approx 1$), the cut-off of the node $R \approx CutOff_{\max}$.

The flowchart in Figure 6 shows the second phase of the node selection algorithm. As shown in the flowchart, the nodes associated with the interest area of the requested function are sorted with their correlation degrees. Because nodes are taken from the list in order, the cut-offs of these nodes R will decrease with the reduced correlation degrees C (according to Equation (2)). But n is increased by one for each new node selected. The query will be sent to the selected nodes only when the number of selected nodes n is smaller than its cut-off to the query R ($n < R$).

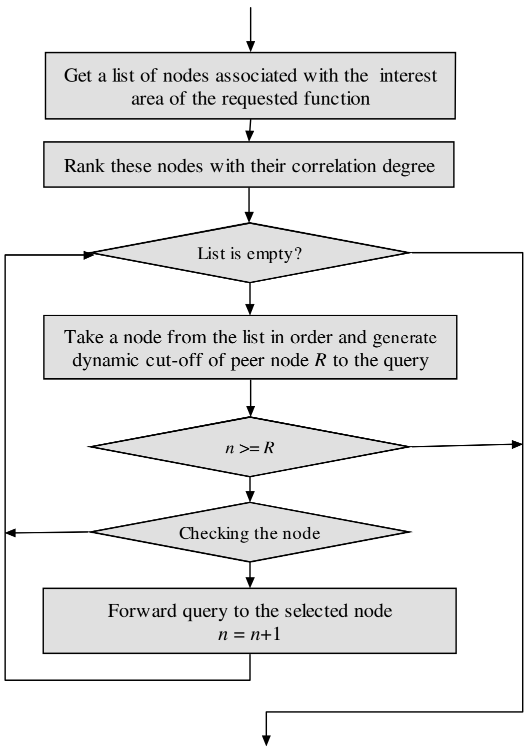
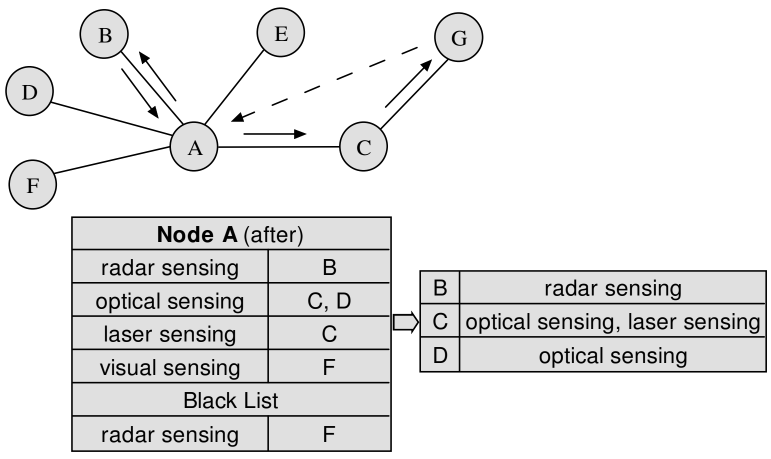

Get a list of nodes associated with the interest area of the requested function
Rank these nodes with their correlation degree
List is empty?
Take a node from the list in order and generate dynamic cut-off of peer node *R* to the query
n >= *R*
Checking the node
Forward query to the selected node
n = *n*+1

Figure 6: Flowchart of the second phase of node selection

If $n \geq R$, node selection procedure is completed in the second phase. If all nodes associated with the area of the requested function ($C > 0$) have been taken from the list in the second phase but there are still not enough nodes selected $n < CutOff_{\min}$, the selection procedure will move to the third phase to randomly pick up nodes from the rest of cached nodes irrelevant to the requested function as well as its interest area ($C = 0$) and forward the query to them, until $CutOff_{\min}$ nodes are forwarded ($n = CutOff_{\min}$).



Node A (after)	
radar sensing	B
optical sensing	C, D
laser sensing	C
visual sensing	F
Black List	
radar sensing	F

B	radar sensing
C	optical sensing, laser sensing
D	optical sensing

Figure 7: Query routing example (two nodes selected)

Figure 7 illustrates a simple example of query routing with AEPS where $CutOff_{\max} = 5$ and $CutOff_{\min} = 1$. Node *A* receives a query with the function "radar sensing". Node *A* will first check the black list to exclude node *F* which is associated with the requested function "radar sensing" in the black list. In the first round of selection, node *B* is selected from the local friend list which is directly associated with the requested function "radar sensing". The number of selected nodes *n* is increased by one: $n=0 \rightarrow n=1$.

Due to $n < FN_{\max}$ ($n=1$, $CutOff_{\max} = 5$), node *A* further searches for the nodes associated with the functions "optical sensing", "laser sensing", and "visual sensing" which are from the same interest area of the requested function "radar sensing". In this case, node *A* gets node *C* and node *D* associated with these functions from the friend list. Since node *C* is associated with two functions "optical sensing" and "laser sensing", but node *D* is associated with only one function "optical sensing" in the area composed by the four functions, node *C* ($C_C = \frac{2}{4}$) is more correlated with the area of general sensing function than node *D* ($C_D = \frac{1}{4}$) according to the cached knowledge. Hence, node *C* will be sorted on the top of node *D* in the list.

The cut-off of node *C* is $R_C = \frac{2}{4} \cdot (5-1) + 1 = 3$ and the cut-off of node *D* is $R_D = \frac{1}{4} \cdot (5-1) + 1 = 2$ by using Equation (2). The query will be sent to node *C*, because the number of selected nodes is smaller than the cut-off of node *C*: $n < R_C$ ($n=1,\ R_C = 3$). Then $n=1 \rightarrow n=2$. The selection procedure will complete and node *D* is not selected, because $n \geq R_D$ ($n=2,\ R_D = 2$). The actual number of queried nodes is *two* in this case.

Node *C* may not have the requested services, but it will use its own cached knowledge to propagate the query further and try to find the nodes for the query that will have a higher likelihood of success. In this example, node *C* knows that node *G* is associated with the requested function according to its local friend list and the requested service is obtained in node *G*.

4.5 Social Pattern 5: Self-Organising

Human society is a self-organising system. Social networks are formed naturally by daily social interactions. A social community is a group of people with common interests, goals or responsibilities which is formed spontaneously. No additional overhead is required for maintenance of social communities. By using social networks, people can find some acquaintances that potentially have knowledge about the resources they are looking for.

Network nodes should be able to self-organise themselves on a non-hierarchical structure and form dynamic organisations with common interests and objectives. If the network nodes in the disaster area can be self-organised like social networks, a large communication cost for group construction, group maintenance and group discovery can be saved, which can significantly improve overall performance of networks.

Similarly to social networks, the links between AEPS nodes are built according to the results of previous interactions, a node has more probability to link to other nodes with the same interests, which have services of interest to him/her, with a high degree of likelihood. Therefore, network nodes that have the same interests will be highly connected to each other and form a virtual community spontaneously, which is a similar environment to Watts's model [22] in social networks.

4.6 Social Pattern 6: Active Query

When a person joins to a new society, he/she will not only passively learn knowledge by remembering useful information from daily occasional events happening in the society, but also actively collect potentially useful knowledge consciously. When he/she seeks out help in a new society, communication with the other people who can be of assistance is not normally limited to a strict exchange of assistance; often, he/she would like to know more about the people who can be of assistance in order to expand his/her knowledge of the new society, which may be useful at a later time.

The query generation strategy of AEPS is similar to the human strategy in social networks where newly joined persons are normally more active to adapt to new environments. In order to efficiently search the network, two kinds of queries are generated at different stages of searches, known as ordinary queries and active queries. For the ordinary queries, the target nodes sharing the desired services will respond to the information related to the requested function only. For the active queries, the target nodes will not only respond to the requested function but also inform the query originator of other associated functions it shares in the same interest area.

With these active queries, the query originator can gather more pieces of knowledge from each successful query, but extra traffic will be generated for shipping such additional knowledge. The extra traffic could be significant if every node generates all queries in this manner, which is difficult to be handled by bandwidth-limited networks. In contrast, if nodes search the network only with ordinary queries, each new node accumulates knowledge slowly by gathering one piece of knowledge from each successful query, especially for those nodes which are seldom online or request the network.

To address these issues, AEPS adopts a trade-off solution for bandwidth-limited networks. The recently joined nodes will utilise active queries to quickly accumulate a large amount of useful information regarding their interests. After the cached knowledge reaches a certain threshold ratio r of the maximum size of the friend list $n_{\max\_knowledge}$, the nodes will use ordinary queries to discover the required services with low traffic cost. If the ratio of cached knowledge $r_{cached} \leq r$, active queries will be used to search the network. Otherwise, ordinary queries will be adopted. Moreover, this process is not only applicable for recently joined nodes, but also enables nodes to quickly recover from unpredictable knowledge loss.

5. Simulation

In this section, the performance of provision of capability on SOA and P2P are evaluated using simulations to see whether the proposed Adaptive and Efficient Peer-to-peer Search (AEPS) approach described above can improve the dependability of provision of capability.

5.1 Simulation Setup

The simulation model was developed using the Java programming language. The main components of the simulation model were illustrated in Figure 8. The simulations were based upon a disaster relief scenario. At the beginning of the simulations, a network was setup containing sixty platform nodes. Sixty different high level functions were generated and distributed to 100 services, and each service performed three functions.

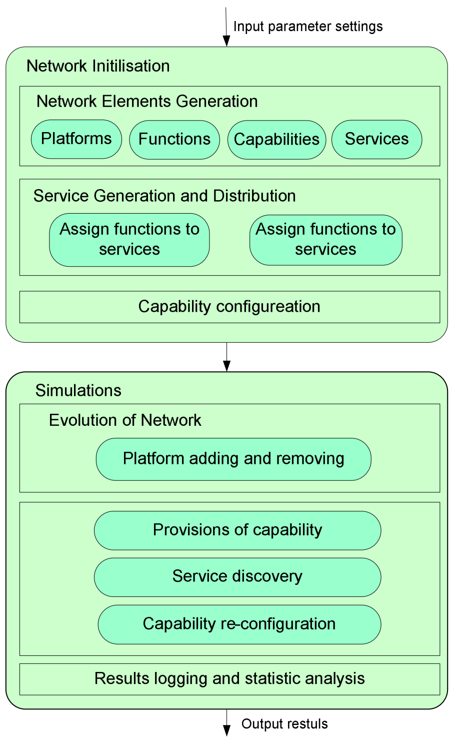

Input parameter settings
Network Initilisation
Network Elements Generation
Platforms
Functions
Capabilities
Services
Service Generation and Distribution
Assign functions to services
Assign functions to services
Capability configureation
Simulations
Evolution of Network
Platform adding and removing
Provisions of capability
Service discovery
Capability re-configuration
Results logging and statistic analysis
Output restuls

Figure 8: Simulation Model

In our simulations, each peer node was randomly assigned a primary functional area and provided three services to the network with a probabilistic method: these services are mostly relevant to the functional area of the node with a probability of 90%, but are occasionally irrelevant to this area. For files relevant to the primary interest area, at least one of the functions of each file should be in the functional area of the hosting node. A total of three functional areas are generated and each covers twenty topics. Each network node randomly connected to four other network nodes bi-directionally and formed a random topology.

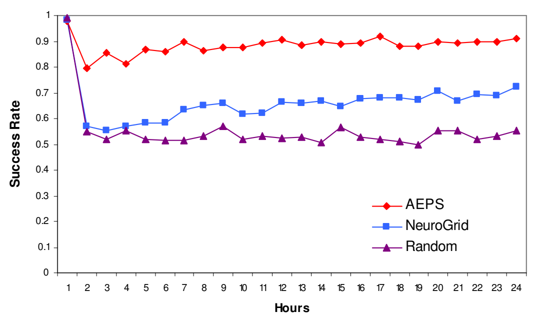
Ongoing changes could be caused by new network nodes joining and exiting the disaster area. To simulate the evolution of networks, one network node joined the network and one network node was removed from the network each hour in the simulations. The availability of each network node was set at 90% in all simulations. Each query is tagged by a TTL to limit the life time of a message to two hops in the simulations, if no other setting is mentioned.

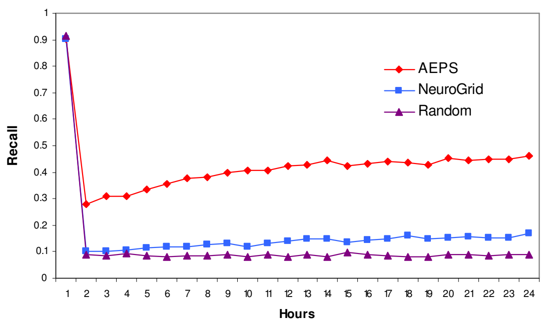
In each time step of simulations, an online network node was chosen randomly as the capability originator to form a capability. The selected node needs to search the network for a specific service on a different network node for integration to successfully deliver a capability. This node will search the centralized service registry for the requested services or use a P2P method for service discovery on the disaster area if the registry fails. In the simulation, the registry failed at one hour after the disaster occurs. The purpose of the simulation was to evaluate whether the distributed network nodes can self-organise themselves in a P2P manner to efficiently deliver capability without the centralized registry.

At each loop of simulation, the performance was measured by the success rate of the provision of capabilities and recall which is defined as the ratio of the number of found services to the number of all possible matched services in the network. Simulations were performed to trace the results of about 1,440 minutes (24 hours, 1,440 simulation loops). Each average result is generated from the experimental results of each hour.

5.2 Simulation Results

In this section, the initial simulation results are presented to provide a comparison among the AEPS and two relevant methods: Random routing [6] and NeuroGrid algorithm [6].

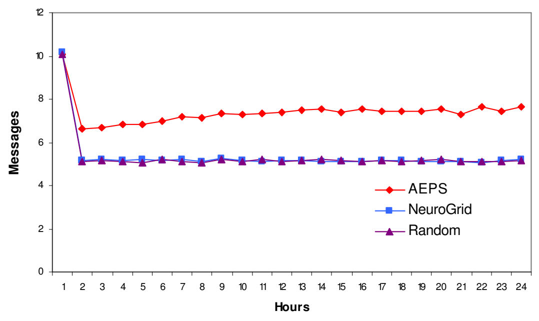


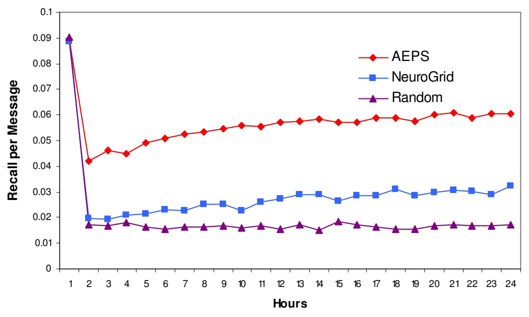


(a) (b)

Figure 9: Performance evaluation. (a) Success Rate; (b) Recall.

From the results in Figure 9(a) and (b), all three mechanisms achieved the highest success rates and recalls by querying the central index server at the beginning of the simulations. The success rate and recall dropped significantly at the beginning of simulations due to the failure of the centralised server. However, AEPS recovered quickly and achieved better performance than NeuroGrid and Random with a higher success rate and recall. In contrast to NeuroGrid, AEPS was capable of retrieving the network nodes which shared the associated services with the relevant functions more often, if network nodes can not find the directly associated nodes with the requested function from the local knowledge index. These selected network nodes which were highly correlated with the interest area of the requested function had more knowledge about the query than randomly selected network nodes. Thus, AEPS can find the requested services more efficiently with the same knowledge. More successful searches, in turn, helped to build the knowledge index more efficiently. Therefore, AEPS had better search ability and better knowledge-collecting ability. With these advantages, AEPS achieved better performance than NeuroGrid and Random.





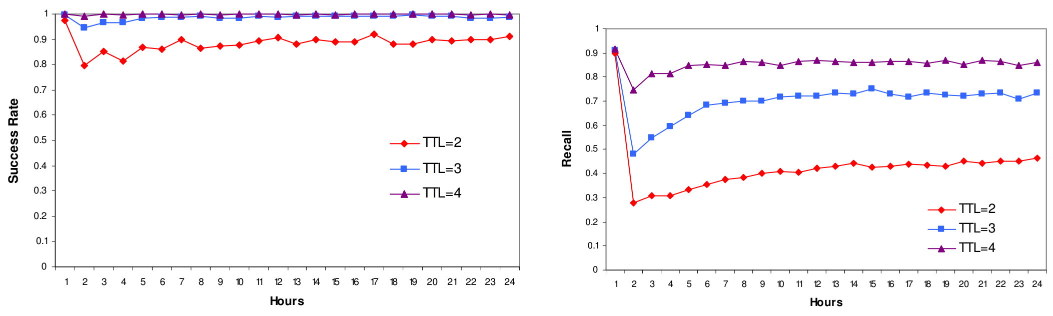
(a) (b)

Figure 10: Overhead and efficiency. (a) Messages; (b) Recall per Message.

As shown in Figure 10(a), AEPS generated a few more query messages introduced by the second phase of node selection procedure, but the search efficiency of AEPS was still better than NeuroGrid by achieving higher recalls per query message as shown in Figure 10(b). AEPS can discover the requested services more efficiently, because they can more rapidly establish the knowledge index than NeuroGrid by gathering more pieces of knowledge from each successful query.

Table 1: Changes to parameters for simulation

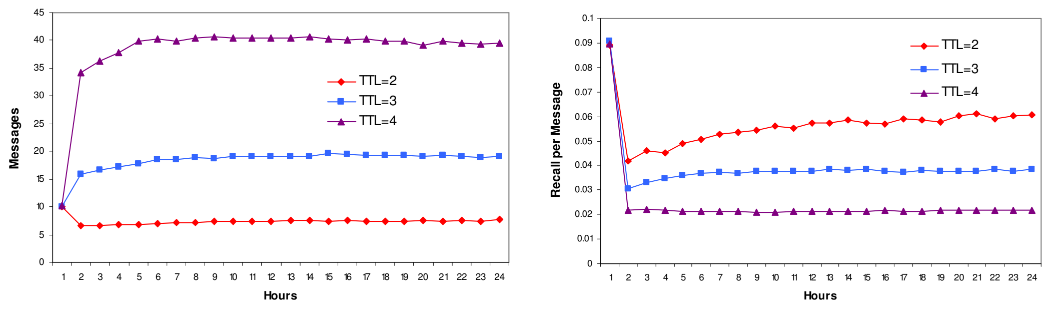
Parameter	Value
TTL	2; 3; 4



(a) (b)

Figure 11: Performance evaluation with different TTLs. (a) Success Rate; (b) Recall.

TTL is an important factor to achieve efficient query forwarding. In this experiment, AEPS was simulated with different TTLs as shown in Table 1 to see the effect that each setting makes. From the results shown in Figure 11(a), AEPS achieved sustainable and dependable capability provision with nearly straight 100% success rate for all searches using *TTL=4*.



(a) (b)

Figure 12: Overhead and efficiency with different TTLs. (a) Messages; (b) Recall per Message.

As shown in Figure 11(b) and Figure 12 (a), recall achieved at the end of simulation increased by 85% by changing TTL from 2 to 4, while messages increased even more significantly by four times as shown in Figure 12(a). This result shows that network traffic is very sensitive to the change of TTLs. Flooding would occur in the network by setting a large TTL. From the results in Figure 12(b), recall per message decreases clearly with increasing TTL. Therefore, TTL should be defined according to the users' requirement and network bandwidth, in order to achieve a good trade-off between search performance and efficiency.

6. Demonstration System Development

A demonstration system has been developed to demonstrate the use of the architectural model for the dynamic service integration of a network of sensors on a disaster area and to provide a search and rescue capability. The core of the approach is the process of mapping high-level requirements for rescue capability onto the discovery of actual services, allowing the establishment of a dynamic workflow of service composition and integration, and dynamic search for services and on the fly planning through dynamic integration of services. The competitive advantage, such as timeliness, reliability and fault tolerance, can be achieved through the dynamic service discovery, composition and integration.

The software demonstrator was developed in NetBeans 6.1 IDE on a GlassFish application server, which enabled our research group members to write, deploy, test and debug SOA applications using the Extensible Markup Language (XML), Business Process Execution Language (BPEL) and Java Web Services. The Software Demonstrator consists of the following main modules:

- Web Services to simulate capabilities of different systems;
- A dynamic workflow module for dynamic Web Services selection and composition for dependable provision of rescue capability;
- A client interface with sensor information display and user input.

The workflow dynamically discovers and integrates the sensor services and is implemented in a Java written Web Service.

The scenario aim of the software demonstrator was to model Region Surveillance using dynamic service integration of sensor networks on a disaster area. The intent is to demonstrate the architectural approach to engineering. The main concepts are:

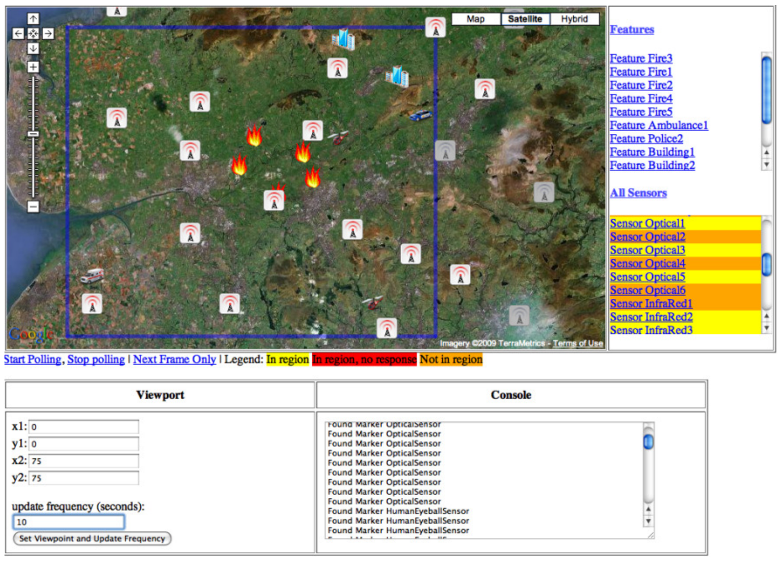
- Use of service-oriented sensor networks for disaster relief enhanced with other architectural styles and patterns;
- Integration of distributed systems in a dynamic environment;
- Coping with changes in availability of distributed components;

Program 1 Simple Sensor Wrapper program in Java

```
/**
* Simple Sensor Wrapper
*/
@WebMethod(operationName = "getSensorImage")
public String getSensorImage()
{
    /**
    * Launch 'readsensor' program and read the output
    * represented by plain text with base64 encoding.
    */
    Runtime runtime = Runtime.getRuntime();
    Process proc;
    StringBuffer programOutput
    = new StringBuffer();
    try
    {
            proc = runtime.exec("readsensor");
            InputStream inputstream = proc.getInputStream();
            InputStreamReader inputstreamreader
               = new InputStreamReader(inputstream);
            BufferedReader bufferedreader
               = new BufferedReader(inputstreamreader);
            // read the program output
            String programOutputLine;
               while (
               (programOutputLine = bufferedreader.readLine())!= null )
      {
      programOutput.append(programOutputLine);
      }
    } catch (Exception ex) { return null; }
    /**
    * Return the wrapped program output to the
    * Web Service client.
    */
    return programOutput.toString();
}
```

A legacy sensor application is exposed as a Web Service in order to allow it be integrated into an SOA enabled workflow for the provision of a regional surveillance capability. The wrapping implementation of the `readsensor' application to a GlassFish hosted Web Service was shown in Program 1.

The surveillance was based on Points of Interest, PoIs and physical features within a geographical area that are detected by a group of simulated sensors. The integration of sensors was achieved by using a workflow to dynamically discover sensors for a given region. The sensor data is then processed to eliminate duplicates and points outside the region of interest and the detected feature positions are displayed on a map. The implementation of region surveillance used Google Maps to display the results from the feature detection workflow.


Map
Satellite
Hybrid
Google
Imagery ©2009 TerraMetrics - Terms of Use
Set Viewpoint and Update Frequency

Features

Feature Fire3
Feature Fire1
Feature Fire2
Feature Fire4
Feature Fire5
Feature Ambulance1
Feature Police2
Feature Building1
Feature Building2

All Sensors

Sensor Optical1
Sensor Optical2
Sensor Optical3
Sensor Optical4
Sensor Optical5
Sensor Optical6
Sensor InfraRed1
Sensor InfraRed2
Sensor InfraRed3

Start Polling, Stop polling | Next Frame Only | Legend: In region In region, no response Not in region

Viewport	Console
x1: 0 y1: 0 x2: 75 y2: 75 update frequency (seconds): 10	Found Marker OpticalSensor Found Marker OpticalSensor Found Marker OpticalSensor Found Marker OpticalSensor Found Marker OpticalSensor Found Marker OpticalSensor Found Marker OpticalSensor Found Marker OpticalSensor Found Marker OpticalSensor Found Marker HumanEyeballSensor Found Marker HumanEyeballSensor Found Marker HumanEyeballSensor

Figure 13. Disaster monitoring showing PoIs

A screenshot from this interface can be shown in Figure 13. In the system, a disaster relief volunteer can submit real-time requests for information of Points of interest, POIs, in a specified region. The system will return the related information about the POIs within that region, e.g., current locations of those POIs. The blue rectangle is the region of interest. The lists on the right of the image show the detected features and the sensors that have been accessed in the workflow.

A screenshot from this interface can be shown in Figure 11. In the system, a disaster relief volunteer can submit real-time requests for information of Points of interest, POIs, in a specified region. The system will return the related information about the POIs within that region, e.g., current locations of those POIs. The blue rectangle is the region of interest. The lists on the right of the image show the detected features and the sensors that have been accessed in the workflow.

The demonstration system incorporates the following innovations to achieve competitive advantage:

- *Information-Rich Information Services*: provide description of services, composition templates with candidate composed services, application workflows, architectural patterns, application patterns, evaluation information [20].
- *Evolving Ontology*: ontology available for dependability, capability, system assessment [25].
- *Service Interoperability*: advanced techniques for dynamic authentication and run-time negotiation [19].

- *Optimisation for On-the-Fly Planning*: based on a tool [19] that supports the use of a variety of optimization techniques and their combination.

7. Conclusions

Enhanced effect can be achieved through dynamic networking of people, systems and procedures and seamless integration of them to fulfil mission objectives with service oriented sensor systems. From the above discussion and evaluation, P2P networking provides dependability in provision of capability based on service oriented sensor systems even in the worst case of failure of service registry. In the AEPS networks, the networked nodes can autonomously support and co-operate with each other in a P2P manner to quickly discover and self-configure any services available on the disaster area and deliver a real-time capability by self-organising themselves in spontaneous groups to provide higher flexibility and adaptability for disaster monitoring and relief. Peer groups are formed and maintained spontaneously, where peer nodes are self-organised based on their daily intercommunications. Each peer node can automatically detect potential interests of other peer nodes in the network according to their previous behaviours and preferentially links to the peer nodes that have similar interests.

AEPS search algorithm has been evaluated for rescue capability provision. The simulation results show that distributed nodes can self-organise themselves in a P2P manner and efficiently discover required service for provision of capability without centralised registry. AEPS improved the scalability and performance of information and service discovery in large-scale distributed systems for the provision of dependable search and rescue capability. Such discovered services and information will provide essential decision support to assist emergency rescue team to make correct real-time decisions at all times even in highly dynamic environments. An interesting future question is how to solve the interoperability issues of involving heterogeneous nodes in networks. A framework [24] is being developing to migrate legacy assets through SOA to realise interoperability between the heterogeneous nodes. Further development of the demonstrator will be used for further evaluation of Adaptive Service Discovery in different dynamic environments. The implementation in the real-world systems will be carried out in the next step for generalised quantitative validation.

Acknowledgment

The work reported in this paper has been supported by Major Program of the National Natural Science Foundation of China, under contact No.90818028.

References

[1] Antonopoulos, N., Salter, J. (2004) Efficient Resource Discovery in Grids and P2P Networks. *Internet Research*, **14**, 339-346.
[2] Cowan, N., Nugent, L.D., Elliott, E.M., Ponomarev, I., Saults, J.S. (1999) The Role of Attention in the Development of Short-Term Memory: Age Differences in the Verbal Span of Apprehension. *Child Development*, **70**, 1082–1097.
[3] Fisher1, K.M., Lipson, J.I. (2004) Information Processing Interpretation of Errors in College Science Learning. *Instructional Science*, **14**, 49-74.
[4] Haythornthwaite, C. (1996) Social Network Analysis: An Approach and Technique for the Study of Information Exchange. *Library & Information Science Research*, **18**, 323-342.
[5] Joseph, S. (2003) P2P MetaData Search Layers. International Workshop on Agents and Peer-to-Peer Computing. Melbourne, Australia.

[6] Joseph, S. (2002) NeuroGrid: Semantically Routing Queries in Peer-to-Peer Networks. International Workshop on Peer-to-Peer Computing. Pisa, Italy.
[7] Kautz, H., Selman, B., Shah, M. (1997) Combining Social Networks and Collaborative Filtering. *Communications of ACM*, **40**, 63-65.
[8] Kim, K.H., Welch, H. (1989) Distributed Execution of Recovery Blocks: An Approach for Uniform Treatment of Hardware and Software Faults in Real-Time Applications. *IEEE Transactions on Computers*, **38**, 626-636.
[9] LibenNowell, D., Balakrishnan, H., Karger, D. (2002) Analysis of the Evolution of Peer-to-Peer Systems. ACM Symposium on Principles of Distributed Computing. Monterey, California.
[10] Liu, L., Russell, D., Webster, D., Luo, Z., Venters, C., Xu, J., Davies, J. (2009) Delivering Sustainable Capability on Evolutionary Service-oriented Architecture. IEEE International Symposium on Object/component/service-oriented Real-time distributed Computing (ISORC 2009). Tokyo, Japan. pp. 12-19.
[11] Liu, L., Webster, D., Xu, J., Wu, K. (2010) Enabling Dynamic Workflow for Disaster Monitoring and Relief Through Service-Oriented Sensor Networks. CHINACOM. Beijing, China.
[12] Lorincz, K., Malan, D.J., Fulford-Jones, T.R.F., Nawoj, A., Clavel, A., Shnayder, V., Mainland, G., Welsh, M., Moulton, S. (2004) Sensor Networks for Emergency Response: Challenges and Opportunities. *IEEE Pervasive Computing*, **3**, 16-23.
[13] Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S. (June 2002) Search and Replication in Unstructured Peer-to-Peer Networks. ACM SIGMETRICS. Marina Del Rey, CA.
[14] Maymounkov, P., Mazi`eres, D. (March 2002) Kademlia: A Peer to Peer Information System Based on the XOR Metric. Internation Workshop on Peer-to-Peer Systems. Cambridge, MA.
[15] McCarty, C. (2002) Structure in Personal Networks. *Journal of Social Structure*, **3**, 1-19.
[16] Newcomb, T.M. (1975) Social Psychology : the Study of Human Interaction. - 2nd ed. Revised. London, Routledge and Kegan Paul.
[17] Rhea, S., Geels, D., Roscoe, T., Kubiatowicz, J. (June 2004) Handling Churn in a DHT. USENIX Annual Technical Conference. Boston, MA.
[18] Ripeanu, M. (2001) Peer-to-Peer Architecture Case Study: Gnutella Network. International Conference on Peer-to-Peer Computing. Linkoping, Sweden.
[19] Townend, P., Huai, J., Xu, J., Looker, N., Zhang, D., Li, J., Zhong, L. (2008) CROWN-C: A High-Assurance Service-Oriented Grid Middleware System. *IEEE Computer*, **41**, 33-38.
[20] Tsai, W.-T., Zhou, X., Chen, Y., Bai, X. (2008) On Testing and Evaluating Service-Oriented Software. *IEEE Computer*, **41**, 40-46.
[21] Waloszek, G. (2002) Personal Networks. SAP AG, Product Design Center.
[22] Watts, D., Strogatz, S.H. (1998) Collective Dynamics of Small-World Networks. *Nature* **393**, 440-442.
[23] Watts, D.J., Dodds, P.S., Newman, M.E.J. (2002) Identity and Search in Social Networks. *Science*, **296**, 1302-1205.
[24] Webster, D., Liu, L., Russell, D., Venters, C., Luo, Z., Xu, J. (2010) Migrating Legacy Assets through SOA to Realize Network Enabled Capability. In: Springer Ed, Web Data Management Trails.
[25] Webster, D., Looker, N., Russell, D., Liu, L., Xu, J. (2008) An Ontology for Evaluation of Network Enabled Capability. Realising Network Enabled Capability (RNEC'08). Leeds, United Kingdom.
[26] Yang, B., Garcia-Molina, H. (2002) Efficient Search in Peer-to-Peer Networks. International Conference on Distributed Computing Systems. Vienna, Austria.