Towards Relevancy Aware Service Oriented Systems in WSNs

Syed Yousaf Shah, *Student Member, IEEE* Boleslaw K. Szymanski, *Fellow, IEEE* Petros Zerfos, *Member, IEEE* and Christopher Gibson

Abstract—The increasing interest of researchers in service oriented architecture (SOA) for wireless sensor networks (WSNs) is opening new unexplored venues in the field of WSNs. In service oriented systems, services are configured and composed of various other services and thus perform complex tasks. In such composite services, the geospatial locations of services and their coverage is of vital importance as they signify the geospatial relevance of the service to the area of interest to the user. In this paper, we present a service-oriented system for WSNs that is capable of performing service configuration under geospatial and relevancy constraints. We present and evaluate "*Cost Based Model (CBM)*" and "*Gain Based Model (GBM)*" approaches to capture the relevancy of services hosted on WSN nodes in composite service configuration. The system is resilient to failures and can operate in *manual* or *autonomous* recovery modes. The system supports three service configuration methods namely, *distributed*, *centralized* and *hybrid*. Furthermore, we present a novel emulation mechanism for testing the performance of our proposed relevancy models and show that our system efficiently configures services.

Index Terms—Science of service composition, Services Computing, Service-oriented architecture, Service Management

1 Introduction

Service oriented architecture (SOA) for WSNs has received significant attention from researchers. The SOA for WSNs focuses on how applications for WSNs can be designed and developed as services so that they can be interconnected and composed in service oriented fashion to create complex services. With ever-increasing hardware capabilities, sensors can execute services on the edge of the network. These services can be interconnected in order to perform intelligent tasks. But there are still open research questions in this emerging field of service oriented architecture. Tools and techniques for WSNs that would enable secure, fault-tolerant and robust configuration of complex services are still rapidly evolving. The geospatial aspect of the data that is collected and transferred using WSNs is adding a novel aspect to the design of complex services in WSNs. Such geospatial tagging of data was never a design factor in service oriented architectures earlier, neither in WSNs nor in the web services.

In a sensor oriented WSNs, the nodes host services and these services can be configured on-the-fly to perform more sophisticated operations. The service configuration in pervasive wireless sensory systems (WSNs) is quite challenging as the requirements of the applications/services hosted on WSNs change over time and these changes must be reflected in the system configuration. As events in WSNs (e.g., node failures making services residing on the node unavailable etc.) occur over time, the configuration mechanism should dynamically reconfigure the system according to the new requirements. An efficient configuration mechanism should be able to configure services in a way that ensures their inputs and outputs to be interoperable during execution of complex tasks. Moreover, the selection of services in service configuration should be taking into account various constraints (including configuration and operational policies) and various performance metrics. Intuitively, relevancy of a service is the volume of relevant information that the service produces to the user's request. In other words, information in which user is actually interested. Relevancy of a service can be defined along various dimensions such as, coverage of the service in the area of interest, accuracy of the measurements provided by the service, latency, temporal utility of a service or geospatial/spatiotemporal closeness of a service to the area of interest. Typically, in service configuration mechanisms, the relevancy that a service brings to the overall system is ignored and only the flat cost of using a service is considered. In contrast, our service configuration approach considers both aspects, cost and relevancy.

In this paper, we use the word "source service/source node" as in Figure 1 for a service provided by a sensor or a group of sensors without any input from other services and we require that the relevancy is defined for each source service. Compos-

- *S. Yousaf Shah is with the Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, USA. E-mail:shahs9@rpi.edu*
- *Boleslaw Szymanski is with the Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, USA.*
- *Petros Zerfos is with the IBM T.J. Watson Research Center, Yorktown, NY, USA.*
- *Christopher Gibson is with the IBM Hursley, UK.*

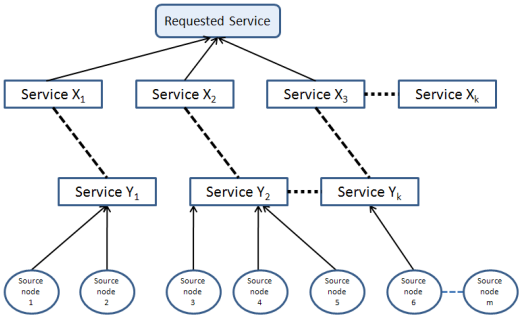

Fig. 1. Sensor service configuration hierarchy

ite services are the services that require input from other services. All services produce output and their relevancy is established by their service providers which directly or indirectly provide input to them. Therefore, a service can be either composite service or a source service.

Mobile and sensor devices are producing huge volume of data these days. From plethora of these data sources, fusing data from most relevant sources to produce information that is relevant to the user's interest is a challenging task. This work explores, how we can dynamically configure complex services that produce information highly relevant to the user's interest. In this paper, we focus on the design of a service-oriented system that can capture the concept of relevancy and can compose complex services from services hosted on sensor or mobile nodes. The sensor service configuration is more elaborate form of service composition. Service composition is a problem of composing services together to form a composite service with complex functionality. The process of service configuration not only performs service composition but also takes into account the implicit and explicit requirements specified by the user of the service and incorporate these requirements into the system as hard and soft constraints. Some of such requirements include, user specified policies and restrictions, spatiotemporal relevancy constraints on the services hosted at the edge of the network and modes of operation of the service.

In this paper, we focus mostly on relevancy of geospatial services to the area of interest defined by the user. Yet, the concept of relevancy and the models for capturing relevancy presented in this paper are more general. These models can be applied to any user defined relevancy metric which can then be modeled as soft constraint in the configuration of services. Figure 1, shows an example of the a service hierarchy and how services at the edge of the network are configured to form a complex service.

Considering relevancy is an attempt to establish some measure of value of information based on its attributes, such as physical location at which a sensor measurement was taken, time and timeliness of information, its type, etc. Our system applies spatial constraints on service selection, and configures the system by choosing low-cost and spatially relevant component services improving the spatial relevancy of the overall system. We show that our proposed mechanism is tolerant to failures, i.e., in the case of failures the system automatically (*in automatic mode*) reconfigures using the most relevant alternative services available. Following are the main contributions of the paper:

- A *Cost Based Model (CBM)* for users sensitive to cost of configured service. *CBM* aims to optimize service configuration for cost and relevancy based on user preferences.
- A *Gain Based Model (GBM)* for users sensitive to gain in terms of relevancy that they get from a configured service. *GBM* optimizes the gain a user gets from the configured service in terms of relevancy based on user defined relative values of irrelevant to relevant information.
- Ability to ensure service configuration compliance with budget constraints using *Return On Investment (ROI)* based mechanism. This utility model aims at users who want to optimize their *ROI* from multiple services.
- A novel self-recovering and fault tolerant system for the configuration of services with spatial and other policy constraints.
- Centralized, distributed and hybrid service configuration mechanisms.

The remainder of the paper is organized as follows: Section 2 describes real-life application scenarios for relevancy-aware services. Section 3 describes the spatial relevancy in service configuration as well as utility models for relevancy. Section 4 provides details on the design and implementation of the system whereas Section 5 presents evaluation results of various relevancy models and configuration modes. Section 6 provides overview of the relevant literature and Section 7 concludes the paper.

2 Application Scenarios for Relevancy Aware Services

Many applications require relevancy aware services and in such applications maximizing relevancy to user's interest or intent is very critical. In this section, we present application scenarios in which relevancy plays an important role.

2.1 Information Relevancy Maximization Scenario

Suppose a user is interested in a composite service, that is capable of monitoring an area for activity and visualizing the location of events. Our system configures the composite service that uses component services such as a camera service, a set of acoustic detection services and a localization service to geolocate

the events. For such a scenario, we have developed a map interface to the system. The user is provided with a map of the area with services hosted at different locations. Using this GUI, the user encircles the area which the user wishes to monitor. The system automatically selects services that maximize the coverage in the area of interest and filters out any unrelated services. After selecting appropriate services, the system configures those services and links them together to create a composite service that performs the monitoring task. Figure 2 shows the map client also accessible on an iPad connected to the system. The interface displays services on the map and the links between services denote the wiring among different services. The system shows how spatially relevant services are composed together to configure a complex service that utilizes three 'Line Of Bearing (LOBR)' readings using "JOIN" service that calculates 'Line Of Camera Reading (LOCR)'. The big yellow circle shows the area of interest specified by the user. As we can see in the Figure 2 the service "LOBR_4" is not used by the composite services as it does not link to any other services on the map. This is because "LOBR_4" is out of the area of interest specified by the user, so an alternate service provider of the same outputs, in this case "LOBR_3" is selected for configuring the required monitoring service.

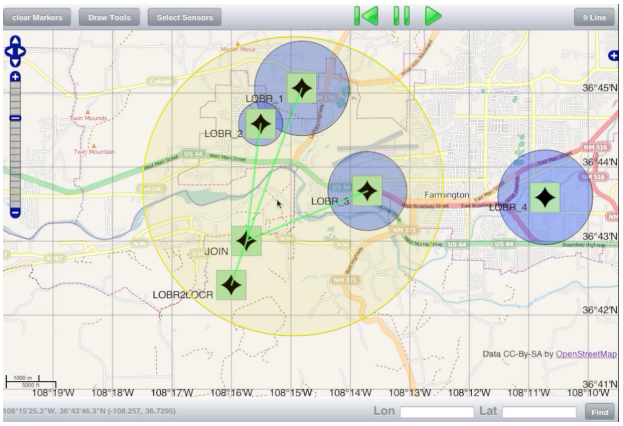

Fig. 2. Map client of service configuration showing services and their interaction.

2.2 Wildfire Modeling

Consider example of *wildfire modeling* in a forest. Suppose there are various kinds of sensors, such as relative humidity (RH) sensors, temperature sensors, anemometer (for wind speed) and wind direction sensors installed in the forest. In order to model the wildfire, we need complex services that take input from various sensors and produce the required fire spreading models. In this case, a RH sensor will require input from the temperature sensors in the fire area; the *originLocator* service will require input from RH sensors as well as various temperature sensors to calculate the origin of the fire. The *wildfire modeler* service will require input from various sensors such as wind direction sensor, anemometer, RH sensors and temperature sensors to model the path and speed of the fire spreading. In such a way, different services are configured together to produce information about wildfire. The accuracy of wildfire modeling highly depends on the efficient selection of services. These sensors should not only produce accurate information, but information that is highly relevant to the area of interest, which in this case is the area on fire. Services in close vicinity of the wildfire are of higher relevance than those far away from the fire, even though they produce the same kind of outputs but are not relevant to the actual fire.In this example, the semantic matching of services as well as the relevancy of services both are of high importance for configuring and composing complex services.

3 Service Configuration with Spatial Relevancy

Suppose a WSN is deployed as a support system for a disaster relief effort. A monitoring system configured in such a scenario might use audio and video feeds produced by other services to surveil the area for operation coordinators. The service configuration in such a scenario should not only consider input/output portability [1], but also other factors, such as energy cost and spatial relevancy of services to the area of interest. In such a scenario, services that are more relevant (e.g., have a larger sensing range in the operation area) are more useful than services that provide the same outputs but with lower relevancy. In this section, we present two different models for relevancy incorporation in service configuration. Based on user's preferences, these models can be used to incorporate service relevancy in the service configuration.

3.1 The Coverage Model for Relevancy

Sensing and mobile devices can collect various kinds of data from the environment and the relevancy of data produced by a sensor, a mobile device or a service can be defined accordingly. Here, we model relevancy in terms of geospatial coverage that a sensor provides in the area of interest. For the scope of this paper, we model the area covered by a sensor in the form of a disk of radius r around the sensor location, which is defined by latitude, longitude and altitude. Each user's service request also specifies the area of interest using two parameters, a point (latitude, longitude, altitude) on the map and radius R. The area of interest is calculated as a circular disk of radius R around the specified point. Our design and relevancy utility models are not restricted to a disk coverage model; the same mechanism can be applied to different coverage models (hexagonal,

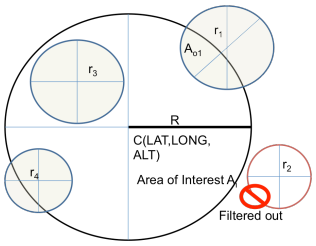

Fig. 3. Disk Coverage Model

polygon etc.). Our system is designed in such a way that new coverage/overlap formulations can easily be plugged into the system.

If a source service/sensor has no coverage in the area of interest then the node is not considered in the configuration process. The relevancy of a service increases with its coverage in the area of interest. Figure 3 shows the disk coverage model, the large disk centered at a particular *LAT, LONG, ALT* with radius *R* represents the area of interest to the user and the small circles depict the sensing area of different services. As shown in the figure, services with sensing radius r_2 is filtered out of candidate set of services for configuration because it does not provide any coverage in the area of interest. In Figure 3, A_{o1} is the overlapping area of service with radius r_1. If *R* is the radius of the area (disk) of interest A_i centered at *C(LAT, LONG, ALT)* and r_x is the radius of the disk covered by a service x centered at $c_x(lat, long, alt)$, then the relevancy of a service x is defined by equation 1.

$$Relevancy = \Gamma(x) = \frac{|C_R \cap c_{r,x}|}{|C_R|} \quad (1)$$

where C_R denotes area of interest with radius R and $c_{r,x}$ denotes the area of coverage of service x with radius r.

3.2 The Price of Relevancy

We define the *Price of Relevancy* as the price that the user is willing to pay for getting certain value from a service; this "value" is a function of relevancy. We believe, such an approach is important in evaluating models for relevancy as it gives the users a way to customize used services according to their needs. Depending upon the constraints that user might have in the application space such as budget or amount of relevancy, user can control budget spending and achieve the required relevancy. The *Price of Relevancy* is represented by Eq. (2).

$$Price\ of\ Relevancy = \frac{AdditiveBaseCost}{Value} \quad (2)$$

The "value" in the simplest case could be just the relevancy that the configured service provides in the area of interest. Then the equation can be written as,

$$Price\ of\ Relevancy = \frac{AdditiveBaseCost}{Relevancy} \quad (3)$$

The values of *AdditiveBaseCost* and *Relevancy* are normalized to same range. The *Relevancy* here is the overall relevancy of the configured service to the area of interest and the *AdditiveBaseCost* is the total cost i.e., accumulated cost of all the services used in the service configuration. In a service configuration, use of a particular service incurs certain cost to the hosting node. This cost can be singular, such as energy consumed, or a combination of different factors such as edge transmission delay, battery consumption and processing time delay; we refer to such a cost as the *BaseCost*. The *BaseCost* of a service can be measured in different ways, it can be monetary cost of using the service, energy consumed, communication and processing cost etc., or composition of various such costs. As we can see from the price formulation (Eq. 3), if the prices are very high that means the cost incurred to achieve certain relevancy is high and a user may not achieve appropriate level of relevancy for the cost to be justified. Therefore, according to our price formulation, low prices are beneficial. The best suitable price also depends on the user specific situation as we will see later in the results sections; we use it as one of the evaluation criteria for our relevancy models.

3.3 Cost Based Optimization of Relevancy

Cost and relevancy of a service play major roles in the configuration of services in mobile and sensor networks. Users often are sensitive to the cost of composite services but still require relevant information, and want to control the weight they put on to cost and relevancy; for such users we propose the *Cost Based Mode (CBM)*, represented by Eq. (4). This utility model aims to configure services that are both low-cost and spatially relevant to the requested area of interest. Every service has a *BaseCost* associated with it, which is incurred when the service is used, since we defined relevancy in Eq. (1) as ratio, we can easily convert it to irrelevancy to the area of interest as shown below. The irrelevancy of a service is also a cost, thus we combine both *BaseCost* and *Irrelevancy* into an *AggregatedCost* using Eq. (4). Using a balancing coefficient α where $0 \leq \alpha \leq 1$, user can adjust the model to its needs with lower cost or higher relevancy. Both the relevancy cost and base cost are normalized to the same range before the aggregated cost is calculated.

$$Irrelevancy(x) = \bar{\Gamma}_x = 1 - \Gamma_x$$
$$AggregatedCost = \alpha \times BaseCost_x + (1-\alpha) \times \bar{\Gamma}_x \quad (4)$$

Under *CBM*, the system uses *AggregatedCost* of services in service selection process and uses greedy

heuristics of *Set Cover* problem (following Geyik et al. [1]) to select service with minimum *Aggregated Cost* at each step of service composition. The minimum cost service composition and achieving maximal relevancy in services are NP-hard problems ([1], [2]) which is why we use the greedy heuristics.

Figure 4 shows that, as we increase the value of α in the *CBM*, the price of the relevancy increases and it goes to infinity at $\alpha = 1$. This shows that as user puts more weight on *BaseCost* in the model the system selects services with lower relevancy and as the value of α goes to 1 the service is composed with minimum *BaseCost* but may produce totally irrelevant information. Please note that at $\alpha = 0.5$, the price of relevancy remains below 1 and there is a good balance between both the *BaseCost* and *Irrelevancy*. In general user can set α according to the application scenario, 0.5 is good value in cases where user wants both the low cost as well as relevant services. If user wants lower cost or more relevancy this can be accomplished by adjusting value of α above or below 0.5.

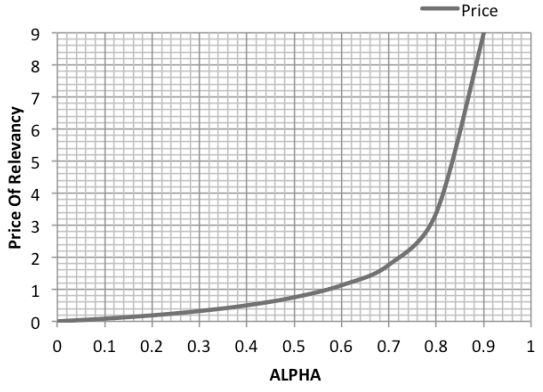

Fig. 4. Price with changing α

CBM is useful in many scenarios in which minimizing the overall cost of services is needed but reasonable relevant of information is also desired. For example, if a user is interested in overall temperature trend in an geographical area (e.g., New York City) or a forest, *CBM* can configure services with low cost yet maintaining reasonable relevancy to the specified area. Other applications of this model are, water quality monitoring of streams in the city, marketing surveys in the cities etc., in such scenarios relevant information from the area of interest is required but the information does not need to be precisely localized.

3.4 Gain Based Optimization of Relevancy

Some users might be sensitive to the overall cost of the service but others might be very sensitive to the actual gain in terms of relevancy they get from the system. Users sensitive to gain need to maximize the relevancy of information to their area of interests and may not be restricted by cost constraints. For such users, we propose a *Gain Based Model (GBM)*. *GBM* enables users to control the amount of relevancy that they get from the service and adjust according to their needs. In addition to relevancy *GBM* incorporates the idea that information outside the area of interest may not be totally irrelevant, thus allows users to specify how much they value such information. *GBM* is represented by Eq. (5).

$$Gain = (1 - \kappa) \times \Gamma_x + \kappa - BaseCost_x$$
$$or$$
$$Gain = \Gamma_x + \kappa \times \bar{\Gamma}_x - BaseCost_x \quad (5)$$

The coefficient κ where $0 \leq \kappa \leq 1$, is the ratio of value of irrelevant information to relevant information. Using κ users can specify how much they value the information that is outside of the direct interest. The total value of all the information produced by service x is one unit, the $BaseCost_x$ of the useful service x is a fraction of this value, i.e., $0 \leq BaseCost_x < 1$ (otherwise service is unusable because its cost exceeds its value even if fully relevant). The solution is to include in the configuration all services for which Eq. (5) is positive. This model tries to configure highly relevant services thus maximizing gain while taking into account the value of κ. Please note that this balancing coefficient κ is different from that of *CBM* which is α. The two models serve users with different concerns. The coefficient α in *CBM* enables user to control "Cost" whereas κ enables customization of gain in terms of relevancy.

For example, if the data outside of the area of direct interest of the user are worthless, then $\kappa = 0$ and the service x is included *if and only if*,

$$Gain_x(0) = \Gamma_x - BaseCost_x > 0$$
$$when\ \Gamma_x > BaseCost_x$$

On the other hand if irrelevant information is as valuable as the relevant one then $\kappa = 1$ and we have,

$$Gain_x(1) = 1 - BaseCost_x > 0$$

so correctly all useful services qualify if their cost is less than 1.

Finally, if irrelevant information is half of the value of the relevant one, we get the following condition.

$$Gain_x(1/2) = (1 + \Gamma_x)/2 - BaseCost_x > 0$$
$$\Gamma_x > 2 \times BaseCost_x - 1$$

Thus, more services will qualify than in case of $\kappa = 0$ because $2 \times BaseCost_x - 1 < BaseCost_x$. In general the qualifying condition is directly derivable from equation 5,

$$\Gamma_x > \frac{(BaseCost_x - \kappa)}{1 - \kappa} \quad (6)$$

So when $BaseCost_x < \kappa$ and $\kappa > 0$ the service qualifies no matter how relevant it is. Since for each service we know κ, Γ_x and $BaseCost_x$ the optimization is very simple, for each service we just compute if inequality (6) is satisfied. The value of κ is application dependent, as shown above it can be set to, above or below 0.5. Setting κ above 0.5 will make more services qualify for configuration giving increasing importance to information from outside the direct area of interest. The system allows user to adjust value of κ, so user can always try different values of κ in order to choose most suitable value for the application.

GBM is useful in many scenario where the relevancy of information is of prime value. In wildfire scenario mentioned in section 2.2, the information from services in the wildfire region is of high value, therefore services should be configured and optimized to produce highly relevant information about the fire area as well as its immediate surroundings. Monitoring water contamination in a localized area (e.g., a county) or information regarding spreading of virus infection in a county are other strong candidate scenarios for *GBM*. In such life-threatening scenarios, highly relevant information is very valuable as opposed to cost factor, precise information is needed to track and stop spread of disease locally.

Return on Investment (ROI)

With *GBM*, we can measure the gain achieved from using certain service. If a user has multiple services running and wants to balance and optimize spending on them, *GBM* can be used to calculate the *Return on Investment (ROI)* of each service. The *ROI* of services is very useful measure for users wanting to create a portfolio of services within budget constraints. Using Eq. (5), we can define *ROI* of a service as,

$$ROI_x(\kappa) = \frac{Gain(\kappa)}{BaseCost_x} = \frac{(1-\kappa)\times\Gamma + \kappa}{BaseCost_x} - 1 \quad (7)$$

where $ROI_x(\kappa)$ is the ROI of service x with coefficient κ.

In order to maximize gain for a given budget based on *ROI*, let *RealCost* denote the monetary measure in the same units as Budget and let *BaseCost* be defined as the ratio of the *RealCost* to the value of a service. Then for the given κ, we can sort all services in the decreasing order of their $ROI_x(\kappa)$ renumbering services so the services with highest *ROI* has lowest index and so on. If two services have the same *ROI*, then larger $RealCost_x$ service is placed before lower one. Then in a loop we do,

$index \leftarrow 0$
while $Budget > 0$ **do**
  $index \leftarrow index + 1$
  $Budget \leftarrow Budget - RealCost_{index}$
  Add $Service_{index}$ to configuration
end while

As an example, suppose we have two services $Service(1)$ and $Service(2)$.

$Service(1)$ is worth \$100 if all of its information is relevant and \$20 if nothing is relevant, and it costs \$80 to run and has relevancy of 90%. So,
$RealCost_1 = \$80, BaseCost_1 = \$80/\$100 = 0.8$, $\kappa = \$20/\$100 = 0.2$
$ROI_1 = (0.8 * 0.9 + 0.2)/0.8 - 1 = 0.15$

$Service(2)$ is worth \$200 if all of its information is relevant and \$50 if nothing is, and it costs \$160 to run and has relevancy of 80%. So,
$RealCost_2 = \$160, BaseCost_2 = \$160/\$200 = 0.8, \kappa = \$50/\$200 = 0.25$
$ROI_2 = (0.75 * 0.8 + 0.25)/0.8 - 1 = 0.0625$

If our budget is \$120 then, if $Service(2)$ is chosen, it will bring \$7.5 gain and exhaust the budget. If instead we select $Service(1)$ for \$80 and $Service(2)$ for the remaining \$40, we get \$12 gain from $Service(1)$ and \$2.5 from $Service(2)$ so in total $\$14.5 > \$7.5$.

It is important to notice that the *ROI* allows the user to assign resources across several applications. With many applications, setting the minimum *ROI* for all applications tells the user how to assign resources (but the budget could be high) or the procedure that we showed could be used, where we allocate first the highest *ROI* assets for all applications and stop when entire budget is optimized. The optimization based on *ROI* has its benefits both in case of atomic resources as well as resources which can be partially shared. In the above example, though $Service(2)$ is worth \$200, but the *ROI* is low if $Service(2)$ is used alone. On the other hand the *ROI* of $Service(1)$ and partial use of $Service(2)$ leads to higher *ROI*. The partial use of services is possible in different situations. Consider example of a 'video camera' that provides high definition video for a certain cost (that can be cost of bandwidth), it can also provide lower resolution video for a lower cost i.e., lower usage bandwidth. In such a way camera is partially used for lower cost.

4 System Design and Implementation

We present a service-oriented system for service configuration with relevancy constraints. Our proposed design provides user with the flexibility to configure system in various modes (*Centralized, Hybrid and Distributed Modes*) as well as run system in *Autonomous Recovery Mode* and *Manual Recovery Mode*.

4.1 The WSN Framework

We prototyped our system in Java using the ITA Information Fabric [3], a SOA-based middleware for sensor networks. A sensor network built using the Information Fabric consists of set of fabric nodes, each of which manages a set of assets and offers a set of services. Fabric is a fully distributed infrastructure with federated nodes (WSN nodes) that form a service

bus across the WSN. The information fabric enables users to develop and deploy applications on sensor nodes and as a framework it provides basic functionalities, such as message routing, node discovery and connectivity services among the sensor nodes. Fabric uses Gaian Database [4] as a backend database for storing assets and services available across different parts of the network.

The service configuration with spatial constraints checking is implemented as fabric service that runs on every Fabric node. Service configuration is designed with full consideration of extensibility. New components (e.g., a new coverage model, a more elaborate cost function, an additional system checker component) can be easily plugged into the system as services. Moreover, the service oriented design of the system enables the user to customize various aspects of the system such as enable/disable the spatial constraints, the depth of service composition on certain node, the cost function, the meta-data functionality, system recovery and configuration modes.

To achieve the flavor of realistic system, we run Fabric nodes inside Common Open Research Emulator (CORE [5]) nodes. Each *Core* node emulates a virtual machine with separate process space and independent network stack. Using *CORE* we emulate the network layer of our WSN testbed. In order to emulate the data link and physical layer in combination with *Core*, we use Extendable Mobile Ad-hoc Network Emulator (EMANE [6]). Figure 5 shows the overview of the emulation testbed and how different components of the system integrate. As shown in the figure, the Fabric node runs inside a CORE node, which is an independent Linux container. Each CORE node is integrated with the EMANE in order to emulate *WIFI* communication among Fabric nodes. Our system is novel in a way that it uses state of the art simulation and emulation platforms to prototype the sensor service configuration. We believe that this is a novel way to test WSN applications and architecture because the emulation platform provide hardware like support to the Information Fabric framework within which we run our service configuration system. Moreover, with such a testbed it is possible to emulate large WSNs as well as emulate path losses and physical layer errors.

4.2 Service Provisioning

The user sends service configuration request to system through a client. One such client is the map interface shown in Figure 2. The *Configuration Service* handles the request accordingly based on the mode of configuration. Figure 6 shows how service configuration request is handled on node level. After the request is received from the user, the system retrieves services from the Fabric registry. The services set retrieved from *registry* is based on input/output parameter matching therefore the set may contain services that

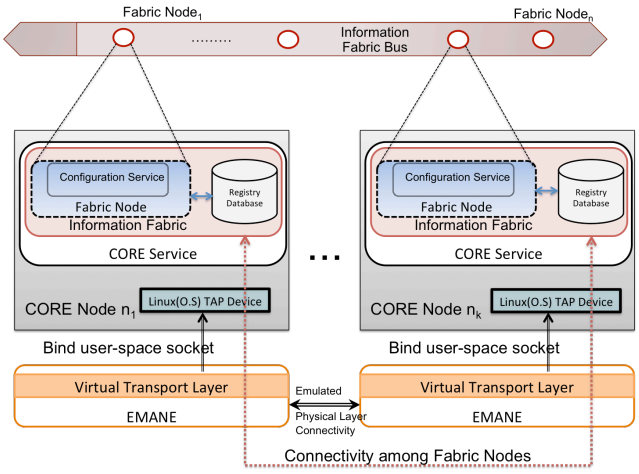

Fig. 5. Distributed mode of configuration

do not provide coverage in the requested area of interest. Services that do not provide coverage in the area of interest to the user are filtered out (except the critical services), thus leaving only those which have non-empty coverage in the area of interest. Then, a Set Cover heuristic is applied to the services, we implemented Set Cover heuristic algorithm similar to the one described in [1] for optimization based on *GBM* or *CBM*. If there are any hard constraints applied to services, such as policies then the corresponding policies are fetched from the *Policy Repository*. Only policy compliant services are selected for configuration and the final service composition graph is written back to the registry [7]. If the policies cannot be satisfied, the service is not configured and user is notified. The policies in our system are enforced as *hard constraints* while spatial constraints are enforced as *soft constraints*. Failure to meet *hard constraints* results in failure of configuration whereas if *soft constrains* are not met, user of the system can still get service configured, as user can choose to ignore spatial constraints. Policies that are *hard constraints* can be relaxed or negotiated using policy relaxation and negotiation techniques [8], however policy relaxation and negotiation are out of scope of this paper.

4.3 Service Maintenance

Each node runs a *Maintenance Fablet* along with the *Configuration Service*. The *Maintenance Fablet* is a fabric bus plug-in that runs as an independent thread and is responsible for checking the composition status of the service hosted on the node. It makes sure that all the services which are known to be composed are indeed composed and in case any of the services becomes unavailable, the *Maintenance Fablet* detects the failure. The failure detection is done through monitoring, at specified intervals the composition graph of each service is retrieved from *Registry* and the availability of the node hosting this service is verified. The mapping

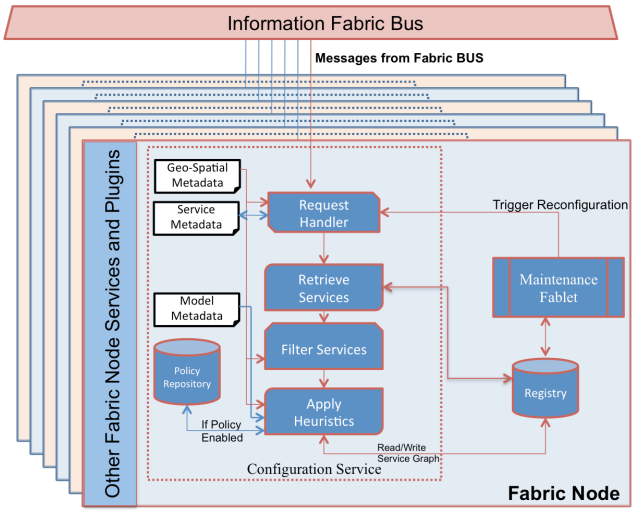

Fig. 6. Node level information flow

of what service is hosted on which node and the current status of the node is stored and maintained by Fabric middleware in the registry, so *Maintenance Fablet* can easily use that information. In case node hosting the service has failed, the *Maintenance Fablet* triggers reconfiguration of the services affected by the failed service(s) or reports the failure in case auto reconfiguration is not possible. In such a way the state of the system is checked and maintained. The monitoring parameters (such as frequency of the system checks, enable/disable system checking) of the *Maintenance Fablet* are configurable and can even be specified by the user in the request, if the user wants to disable the automatic reconfiguration or wants to modify other configuration parameters.

The user can request to run the system in *Autonomous Recovery Mode* or in *Manual Recovery Mode*. In *Autonomous Recovery Mode*, the system is self-resilient to different kinds of failures, such as malfunctioning of a service node, failure of service on a node or service node going out of range etc. When any of such failures is detected, the system automatically reconfigures itself with a valid most cost-effective solution or reports a problem in case configuration cannot be run due to failures or user constraints. In *Manual Recovery Mode*, the system will not reconfigure itself automatically in case of failures. This mode is designed to allow human intervention and the reconfiguration is triggered by the user.

4.4 Modes of Configuration

The system is designed to operate in *Centralized*, *Distributed* and *Hybrid* mode. In order to make service composition faster and more efficient, we incorporate caching mechanism for composite service graph. The composite service parts of composite service graph are cached in the fabric registry for later use. These cached service graphs are periodically validated by the *Maintenance Fablet*; if the composite service graphs are not valid due to unavailability of their service providers, they are removed from the registry database. Users can compose service by utilizing the cached services or opt to generate fresh configuration.

4.4.1 Centralized Mode of Configuration

In the centralized mode of configuration, all the services are configured using a centralized node. All the nodes store information about services they are hosting in a centralized registry. All the services are serially configured in bottom up approach by the centralized node. This mode is prone to single point of failure and can be inefficient as every single change (in any component service) will trigger recomposition of the complete composite service graph. However, this mode of configuration incurs low bandwidth cost because configuration messages are sent to a central node which then communicates with other nodes.

4.4.2 Distributed Mode of Configuration

The distributed mode of service configuration makes the system robust against single point of failure. In this mode each service is hosted by a separate node. In order to configure services in a distributed manner, all the component services are sent configuration request as message via the node hosting these services. In our emulation setup, these services are hosted on different *Core* nodes and are connected to the registry via Fabric middleware. At the system startup, every node publishes its functional capabilities, location information and its sensing range to the registry. Upon receiving request for configuration, a service looks for input services that it needs for getting configured. If all the services required by the service are configured, it configures itself otherwise it retries after a specified sleep time τ. After the timer τ expires, the service checks again for the required services and attempts reconfiguration for a specified number of tries denoted by χ, before terminating the configuration process. In such a way, all the services independently configure themselves and finally the requested service composed of other services, is configured. Whenever a service is configured, an entry is inserted in the registry database which is available to other services to peek at and check status of this service. The registry is used for status synchronization among services. In case any service fails, all those services that are not dependent on the failed service are still available and can be utilized without interruption while *Maintenance Fablet* tries to reconfigure the services that are dependent on the failed service.

The benefit of distributed mode is that it adds high level of robustness and configuration granularity to the system and improves the efficiency. Moreover, the capabilities of Gaian database can also be used to seamlessly access disparate databases which increases

the reliability. The distributed mode is the most scalable, yet its scalability is limited by the network bandwidth. As the number of services grows, so does data replication (if fully distributed registry is used) and the number of messages sent during reconfiguration process increasing burden on networking. In this mode one configuration message per service is sent, therefore, the number of messages is dependent on the number of services in the system. In a large and fully distributed system, not only the number of messages passed among distributed Gaian database instances will be large (if distributed Gaian DB instances are used), but also service configuration request messages will be large leading to significant bandwidth consumption in a resource constrained environment. This mode uses Gaian DB at no extra cost because Fabric already uses Gaian DB as backend database (registry) and maintains as well as optimizes the lifecycle of connections with it. Details on performance of Gaian DB can be found here [9].

4.4.3 Hybrid Mode of Configuration

The hybrid mode is an attempt to achieve the best of both centralized and distributed modes of configuration. The aim of this mode is to make the system robust against single point of failure while reducing the system specific messages. In this mode, there are more than one configuration nodes that are responsible for configuration of different sets of services. Moreover, if a distributed registry is used, these sets of services are served by separate registry instances which reduces total number of registry instances. This way we can increase the node-to-services ratio (number of services configured by a node) as well as the nodes to registry database ratio (number of nodes served by one instance of registry). Having one registry database serve more than one node decreases the level of data and query distribution and consequently the number of messages transferred across distributed databases. With one node hosting more than one services, we reduce the number of messages sent to the nodes as only one service request message per node is sent to the node for all the services configured through the node.

Along with the benefits, this mode introduces new challenges and solutions to these challenges is out of scope of this paper. One such challenge is the optimization of parameters, i.e., how many services should be configured by a single node, how the services should be co-located on nodes, how many registry databases should there be in the system. Another problem is, if one registry serves more than one nodes then the registry can also become a single point of failure for all the nodes dependent on this registry instance. We handle this specific challenge by introducing a backup channel to the registry hosting nodes to make the connection of nodes with registry more reliable and resilient. This backup channel is an alternative connection to the registry which is used only for registry related communication, in case the main connection breaks. The backup channel may not offer the same set of functionalities as the main channel but maintains connectivity with nodes and other distributed databases in case main communication channel fails. Such a backup communication channel enables the distributed database to stay alive and respond to queries.

In our experimental testbed, we configure the CORE nodes hosting registry with two interfaces. One channel which is used as main emulated channel and all the communication (service specific, e.g., sensor readings) among the nodes is done through the emulated channel. For establishing backup channel in our testbed, we use the *CORE* control channel to the registry hosting nodes. The networking of the experimental setup is done in such a way that nodes can connect to registry database via the CORE control channel in case of failure of main channel, but use the main channel for other communication. In such a way, even if the main channel gets disconnected and services on that node fail, the backup channel (*CORE* control channel) provides connectivity to registry database.

5 Evaluation and Experimentation

We have performed experimentation on the distributed emulated testbed described in section 4. For experimentation, we have used the monitoring scenario and evaluated the models presented in this paper. We also conducted experiments to judge the efficiency of our system. In our experiments, we configured and composed six different composite services shown in charts as SINK-0-0 to SINK-0-5. These are the sink services providing spatially constrained information to the user. Each sink service is composed of various other component services in fashion depicted in Figure 1, a hierarchy with six levels. On average a single sink service is composed of 23 other services selected out of various alternatives services based on the relevancy model in use. In evaluating our relevancy models, we have compared the *Price of Relevancy* of services in addition to *Additive Cost* and *Increase in Relevancy*. In our experiments α and κ were set to 0.5.

We compared *CBM* and *GBM* with a *Simple Relevancy Model (SRM)* for service configuration. The *SRM* essentially minimizes the overall base cost of the configured services but, the *SRM* only selects services with non-empty coverage in the area of interest during configuration process. Therefore, *SRM* does have basic notion of relevancy because all those services which have no coverage in the area of interest are already filtered out before actual configuration takes place using any of the models. We measured the relevancy of the configured services in *SRM* case and

compared it with our proposed models to verify that our models actually improve the level of relevancy.

5.1 Performance Comparison of the Configuration Modes

We compared the running time of *Centralized*, *Distributed* and *Hybrid* service configuration modes on our emulation testbed. Plots in Figure 7 show configuration time of three different modes of configurations. As we can see, the configuration time in all configuration modes remains about the same for configuration of individual service, but time for both *Centralized* and *Hybrid* modes increase as the number of services in a configuration grow (or the composite graph enlarges). In our experiment, there were various composite services with same number of component services (such as 1, 4, 5 etc.), therefore, we see repeated X-axis labels in plot 7, but they are actually distinct configured services.

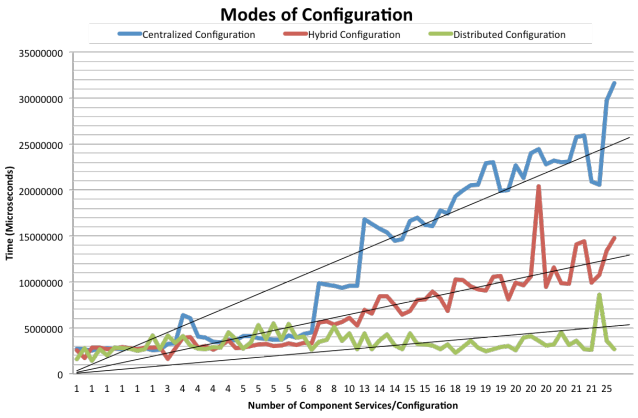

Fig. 7. Configuration times in various modes

As the composite graph gets bigger, the configuration times in *Distributed* mode slightly increases but not as much as in the other modes. The behavior is in agreement with our earlier discussion (in section 4.4) on advantages of *Distributed Mode*. Each service is hosted on a separate node and individually configures itself without depending on a centralized node. This is why services can configure themselves in parallel as long as they do not depend on the output of others. Even those dependent on the output of others configure themselves quicker than in *Centralized* or *Hybrid* mode because all services start configuring themselves as soon as the request is received. Moreover, this enables distributed mode of configuration to scale well with increasing number of services, as there is minimal or no dependency on other services and nodes. As discussed in section 4.4, the *Hybrid Mode* performs midway between *Centralized* and *Distributed*. In *Centralized Mode*, services are configured by a centralized node in *Bottom-up* approach and as the number of services in a composite graph increase the execution time increases as configurations are composed in serial fashion. In *Hybrid Mode*, services are divided in different groups which are configured by nodes responsible for their configuration, therefore a group of services is configured in parallel decreasing the overall configuration time.

To decrease the overall configuration time in *Centralized and Hybrid Modes*, we have incorporated *service caching* mechanism as described earlier. This saves time by storing frequently used services or services configured in recent past, in the registry database. These services act as pre-configured part of other complex services thereby saving time which will otherwise be spent on their configuration. We have noticed in our experiments that *caching services* reduces configuration time but it depends on the granularity of the cached services, i.e., component services with larger number of services (larger part of the service graph) save more time than caching non-composite single services or in our case the source services.

5.2 Results of Cost Based Model (CBM)

We evaluated the *Cost Based Model (CBM)* for relevancy and compared it with *SRM*. Figure 8 shows the comparison of prices of configured services in *CBM* and *SRM*. As seen in the plots the *Price of Relevancy* is lower in *Cost Based Model (CBM)*; lower price means that by using *CBM* the user is getting more relevant information (or value from service) for the payment made. In other words, *CBM* provides more relevant information for the cost incurred as compared to *SRM*.

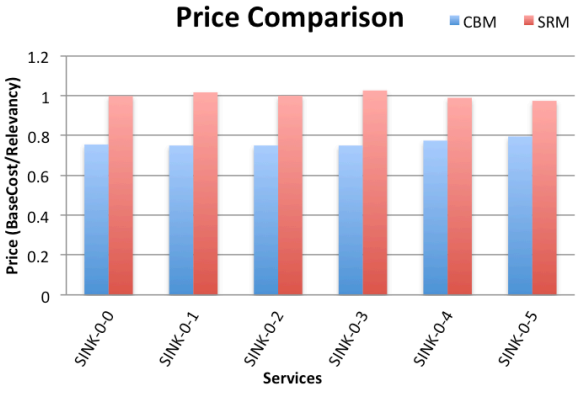

Fig. 8. Prices in CBM

We further analyzed the relevancies yielded by both models and compared the increase in "relevancy" and decrease in "prices" to judge the advantage that our *CBM* has over the *SRM* model. The gains are defined by Eqs. (8-9). The Γ_{CBM}, Γ_{SRM} represent relevancies and P_{CBM}, P_{SRM} represent prices in *CBM* and *SRM* configurations, respectively.

$$Increase\ in\ Relevancy = 100 \times \frac{\Gamma_{CBM} - \Gamma_{SRM}}{\Gamma_{SRM}} \quad (8)$$

$$Decrease\ in\ Price = 100 \times \frac{P_{SRM} - P_{CBM}}{P_{CBM}} \quad (9)$$

Fig. 9. Price and Relevancy change in CBM

Figure 9 shows that *CBM* configured services produce up to 50% more relevant information than those of *SRM*. Moreover, the *CBM* configurations incur on average 30% lower prices than *SRM* configurations. These results demonstrate that our *Cost Based Model* leads to low prices and configures highly relevant services as compared to services that are configured just based on the minimizing the base costs.

5.3 Results of Cost Based Model (CBM) vs. Gain Based Model (GBM)

We evaluated *GBM* and compared it with both *CBM* and *SRM* approaches. Figure 10 shows price comparison among models being compared. We see that both *GBM* and *CBM* incur low prices, the prices incurred by *CBM* are slightly lower than those of *GBM* but both of these approaches lead to highly relevant information for the cost incurred as opposed to *SRM* approach which minimizes base cost but leads to low relevancy as well.

Fig. 10. Price comparison in two models

Figure 11 compares the relevancy gains that the two models *CBM and GBM* yield compared to *SRM* and shows that the two models are complementary. We can see that despite slightly higher prices *GBM* produces up-to 250% more relevant information compared to 50% of *CBM* which meets the purpose of the *GBM* model. *GBM* is designed to deliver high relevancy services and optimize services such that the configured services yield high gains. Please note that the formulation of *GBM* is oriented towards gain achieved without putting any limit on the base cost therefore for this model, we expect the base cost to go very high.

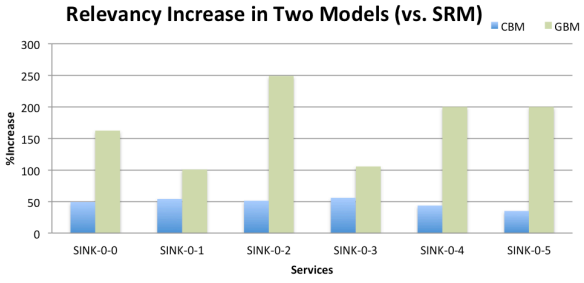

Fig. 11. Comparison of Relevancy in two models

Figure 12 shows additive base cost and price comparison of three models being compared. It is important to notice that the three models behave as expected. Consider the *SRM* approach, it is designed to incur low additive cost without optimizing for relevancy of individual services and the composite services, so it yields the lowest additive base cost but at the same time incurs very high prices due to low relevancy. The *GBM* on the other hand is designed to deliver highly relevant services which it achieves but at the same time, it incurs high base cost. This is because while selecting services for configuration the model does not explicitly take into account the base cost of the component services, it only optimizes gain based on the relevancy that a service brings, which is why it selects highly relevant services even if they have high base cost. In contrast the *CBM* optimizes a generic cost that is composite of both base cost as well as the cost of irrelevancy of a service. This results in selecting services that are both relevant and cost effective. That is why we see the additive base cost in *CBM* is slightly higher than *SRM* but is much lower than *GBM*.

The results show that both models, *CBM* and *GBM*, effectively capture the concept of relevancy and configure services with high relevancy to the area of interest to the user. The results also show that without optimizing the relevancy of services during the configuration process results in configurations with poor relevancy. Therefore, such service configuration will result in delivery of information that is not relevant to the user's interest. On the other hand, our relevancy aware service configuration models capture the concept of relevancy while fulfilling the service configuration requests. However, as we observed in the results, different relevancy models are appropriate in different application scenarios. When highly relevant

information is needed and there is no limit on budget or the *BaseCost* of alternative services do not vary much, *GBM* is a good model to use. Nevertheless, if there are budget restrictions or the *BaseCost* variation is high among services of different relevancies, the *CBM* is the way to go, because it strikes a good balance between additive base cost and overall relevancy of the configured service.

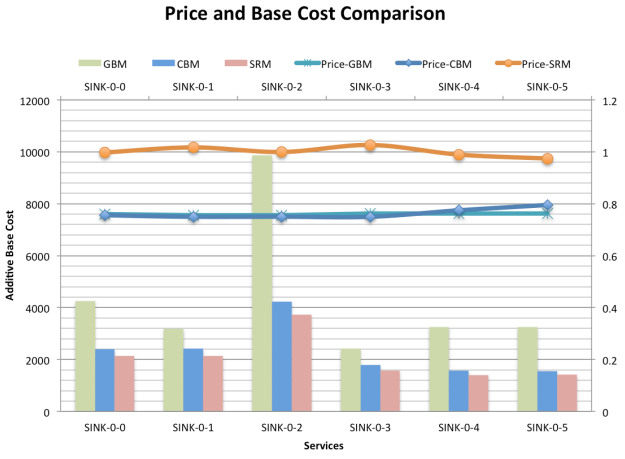

Fig. 12. Comparison of two models

5.4 ROI Based Service Configuration with Budget Constraints

We conducted simulation to evaluate the *ROI* based approach for a given constraint. We fixed the budget to 1000 and tried to efficiently select services that can be configured within the budget limits. Figure 13 shows the gain achieved and the number of services used in a configuration. It is clear from the figure that the number of services in a configuration and the total gain achieved from the configuration increases as the budget limit increases. However, services are added only when they lead to high return relative to their cost, this rule causes empty pockets in the graph where no additional services are added, i.e., between $92 - 199$, $201 - 223$. In our simulation, we select a service if it can be fully obtained within the available budget limitations. The gain shown in the Figure 13, is the total gain from all services in a configuration and is calculated using Eq. (10). As already explained through examples in the section 3.4, this mechanism selects services based on the "ROI" and optimizes the overall return on investment.

$$Total\ Gain = \sum_{x=1}^{n} (ROI_x \times cost_x) \qquad (10)$$

6 Related Work

The service oriented architecture and service composition on resource rich platforms has been well studied

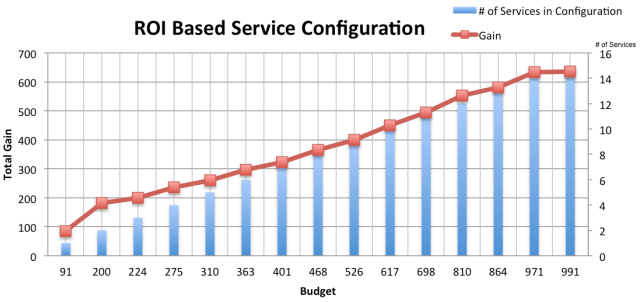

Fig. 13. ROI based service configuration for a given budget

and has been present in the literature for quite some time. However, the area of service oriented WSNs is getting significant attention only recently. Researchers are already investigating SOA for sensor networks but still the area is largely unexplored. Various aspects of the service oriented WSNs have been most studied so far include service composition and service management but not spatial relevancy. In this section we describe the most relevant work that has been done in the areas relevant to this paper.

Service composition for web service have been explored in the past [10], [11], [12]. Reference [13] provides an overview of techniques used for web service composition. In [14], the authors have proposed cost effective algorithms for mapping services in clouds. They have addressed the service mapping problem as a service composition problem. In the researched approaches a service is composed from various other web services hosted on servers in different locations. Such approaches are suitable for world wide web but WSNs are different in terms of network structure, computing resources and computing models. The web service compositions are designed based on some salient requirements, their functionalities are predefined and their composition relies mostly on a static graph of different functionalities. Even though the functionalities can be dynamically binded, the overall functional capability of the system is restricted. Researchers have also explored workflows for service composition [15], [16], [17]. Reference [18], also utilizes workflow representation for service compositions and presents an approach that is similar to web service composition. Workflows are typical example for static composition of services and are very useful in situations where the composition graph doesn't change during service execution. The nesC [19] was an effort to make component based system for mobile devices by connecting various reusable components. The approach is similar to CORBA [20] that was proposed for distributed system. However, in nesC the graphs and trees that are used to bind different components of the system are statically defined before the system can start working. This is why the nesC is missing dynamic reconfiguration capabilities which

which are one of the main features proposed in our system. Model for mobile services proposed in [21] suggests connecting services that are semantically and syntactically similar. The model is oriented around mobile users. Change in location of a mobile user initiates connection/disconnection with a server and triggers composition of services which may not be the case in WSN system. The approach does not compare costs or attempt any spatial relevancy check to find out most appropriate services from a bag of available services. In our approach, the system takes into account the cost and relevancy of services. Moreover, the composite service graph is dynamic and automatically reconfigures in case any changes occur in the network.

A closely relevant work in [1] proposes dynamic composition of service. The work focuses on composing services from various other services but the work does not consider spatial relevancy of the services during composition. The authors assume in their work that all the services that provide certain output have the same relevancy. They do not consider user's desired area of interest in their composition algorithm, the composition is done based on functional capabilities of the services. In contrast, in our work the spatial relevancy has a major role in the resultant composite service. Moreover, the authors have tested their centralized composition approach in a simulated environment and have provided theoretical results for the distributed version. We provide fully emulated centralized, distributed and hybrid sensor service configuration with full awareness of the geospatial locality of the service. To the best of our knowledge, this problem with all the aspects that we explore in this paper has not been investigated in the past. Service composition for MANETs has been researched in [22], however the work does not investigate the aspects we are looking at, such as cost, service relevancy, service configuration management and emulation.

In [2], the authors have considered problem of information provider selection and have presented numerical results. However, the authors do not consider the relationship among the providers and do not address the problem of sensor service composition and management. In their work, the authors has focused on only on selection of services based on their Quality of Information (QoI). The authors do not address the relevancy of composite services in a sensor service graph which we have addressed in this paper in service configuration process. In addition, we provide different models for capturing the relevancy in hierarchical services. These models enable users to achieve the suitable balance of cost and relevancy for their applications.

Researchers have also explored the coverage problem from perspective of sensor deployment and sensing algorithms. In [23], a good survey, of work on coverage in WSNs, is presented. The literature that focuses on coverage provided by the complete WSN and tries to cover maximum possible area with the sensors. In our work, we focus on dynamically configuring complex services; these complex service in our case might use only few of the sensors (that are valuable to user's request) of the complete WSN to provide relevant information about a small part of the complete area covered by a WSN.

Most of wireless sensor network research (e.g., [24], [25], [26], [27]) has historically been simulated or experimented either using TinyOs or some specialized hardware software stack. We employ a novel technique of emulating WSNs applications. Our testbed not only enables network emulation but also enables integration of emulated network with real time network.

7 Conclusion

In this paper, we demonstrate service configuration with relevancy constraints using a state-of-the-art service oriented system for WSNs. We show how services can be composed together to configure complex services that are relevant to the area of interest of the user. In our system, a service is configured in such a way that it produces information that is relevant to the user's interests. In this paper we have focused on geospatial relevancy of services but the techniques and models presented in this paper are applicable to user defined relevancy models. We have proposed two models namely "Cost Based Model (CBM) and "Gain Based Model (GBM) for capturing relevancy of services. The "Cost Based Model (CBM)" uses both cost and relevancy of the service to optimize the configured service, whereas the "Gain Based Model (GBM)" focuses more on the gain that services bring in terms of relevancy to the configured service. We also present "Return On Investment (ROI)" based service configuration which optimizes service selection for a given budget based on "ROI". We have shown that these models produce services that are highly relevant to the area of interest of the user as compared to models that do not consider the spatial relevancy of the services in configuration process. Moreover, we have demonstrated that the architecture of our service oriented system is self-recovering and capable of performing configuration in various modes. The system is highly extendable and allows new models and services to be integrated in the system.

Acknowledgments

Dr. Chatschik Bisdikian (IBM T.J. Watson Research Center) participated in the research that shaped the foundation of this work, thus, the authors wish to acknowledged his contributions to the earliest stages of this research.

Research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was

accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

REFERENCES

[1] S. Geyik, B. Szymanski, and P. Zerfos, "Robust dynamic service composition in sensor networks," *Services Computing, IEEE Transactions on*, vol. 6, no. 4, pp. 560–572, Oct 2013.

[2] G. Tychogiorgos and C. Bisdikian, "Selecting relevant sensor providers for meeting your quality information needs," in *2011 12th IEEE International Conference on Mobile Data Management (MDM)*, vol. 1, 2011, pp. 200–205.

[3] J. Wright, C. Gibson, F. Bergamaschi, K. Marcus, R. Pressley, G. Verma, and G. Whipps, "A dynamic infrastructure for interconnecting disparate isr/istar assets (the ita sensor fabric)," in *12th International Conference on Information Fusion, 2009. FUSION'09.* IEEE, 2009, pp. 1393–1400.

[4] G. Bent, P. Dantressangle, D. Vyvyan, A. Mowshowitz, and V. Mitsou. (2008, Sep.) A dynamic distributed federated database. [Online]. Available: https://www.usukitacs.com/papers

[5] J. Ahrenholz, "Comparison of core network emulation platforms," in *MILITARY COMMUNICATIONS CONFERENCE, 2010-MILCOM 2010.* IEEE, 2010, pp. 166–171.

[6] [Online]. Available: http://labs.cengen.com/emane/index.html

[7] R. Dilmaghani, S. Geyik, K. Grueneberg, J. Lobo, S. Y. Shah, B. K. Szymanski, and P. Zerfos, "Policy-aware service composition in sensor networks," in *2012 IEEE Ninth International Conference on Services Computing (SCC).* IEEE, 2012, pp. 186–193.

[8] C. Sakama and K. Inoue, "Negotiation by abduction and relaxation," in *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '07. ACM, 2007, pp. 242:1–242:8.

[9] P. D. Stone, P. Dantressangle, G. Bent, A. Mowshowitz, A. Toce, and B. Szymanski, "Query propagation behaviour in gaian database networks."

[10] L. Zeng, B. Benatallah, A. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *Software Engineering, IEEE Transactions on*, vol. 30, no. 5, pp. 311–327, 2004.

[11] S. Narayanan and S. A. McIlraith, "Simulation, verification and automated composition of web services," in *Proceedings of the 11th international conference on World Wide Web.* ACM, 2002, pp. 77–88.

[12] M. Alrifai and T. Risse, "Combining global optimization with local selection for efficient qos-aware service composition," in *Proceedings of the 18th international conference on World wide web.* ACM, 2009, pp. 881–890.

[13] N. Milanovic and M. Malek, "Current solutions for web service composition," *IEEE Internet Computing*, vol. 8, no. 6, pp. 51–59, 2004.

[14] K.-T. Tran, N. Agoulmine, and Y. Iraqi, "Cost-effective complex service mapping in cloud infrastructures," in *Network Operations and Management Symposium (NOMS), 2012 IEEE.* IEEE, 2012, pp. 1–8.

[15] D. Jordan, J. Evdemon, A. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. Ford, Y. Goland *et al.*, "Web services business process execution language version 2.0," *OASIS standard*, vol. 11, p. 11, 2007.

[16] Microsoft. (2014, Mar.) Workflow manager. [Online]. Available: http://msdn.microsoft.com/en-us/vstudio/jj684582.aspx

[17] A. Biem, E. Bouillet, H. Feng, A. Ranganathan, A. Riabov, O. Verscheure, H. Koutsopoulos, and C. Moran, "Ibm infosphere streams for scalable, real-time, intelligent transportation services," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data.* ACM, 2010, pp. 1093–1104.

[18] F. Rosenberg, P. Leitner, A. Michlmayr, P. Celikovic, and S. Dustdar, "Towards composition as a service - a quality of service driven approach," in *Data Engineering, 2009. ICDE '09. IEEE 25th International Conference on*, March 2009, pp. 1733–1740.

[19] D. Gay, P. Levis, R. Von Behren, M. Welsh, E. Brewer, and D. Culler, "The nesc language: A holistic approach to networked embedded systems," in *Acm Sigplan Notices*, vol. 38, no. 5. ACM, 2003, pp. 1–11.

[20] S. Vinoski, "Corba: Integrating diverse applications within distributed heterogeneous environments," *Communications Magazine, IEEE*, vol. 35, no. 2, pp. 46–55, 1997.

[21] I. Jorstad and T. Do van, "A service-oriented architecture framework for mobile services," in *Telecommunications, 2005. advanced industrial conference on telecommunications/service assurance with partial and intermittent resources conference/e-learning on telecommunications workshop. aict/sapir/elete 2005. proceedings*. IEEE, 2005, pp. 65–70.

[22] D. Chakraborty, A. Joshi, T. Finin, and Y. Yesha, "Service composition for mobile environments," *Mobile Networks and Applications*, vol. 10, no. 4, pp. 435–451, 2005.

[23] R. Mulligan and H. M. Ammari, "Coverage in wireless sensor networks: A survey." *Network Protocols & Algorithms*, vol. 2, no. 2, 2010.

[24] I. Leontiadis, C. Efstratiou, C. Mascolo, and J. Crowcroft, "Senshare: Transforming sensor networks into multi-application sensing infrastructures," *Wireless Sensor Networks*, pp. 65–81, 2012.

[25] D. Georgoulas and K. Blow, "In-motes: An intelligent agent based middleware for wireless sensor networks," in *Proceedings of the 5th WSEAS International Conference on Application of Electrical Engineering*, 2006, pp. 225–231.

[26] R. Smith, "Spotworld and the sun spot," in *Proceedings of the 6th international conference on Information processing in sensor networks.* ACM, 2007, pp. 565–566.

[27] P. Levis and D. Culler, "Maté: A tiny virtual machine for sensor networks," in *ACM Sigplan Notices*, vol. 37, no. 10. ACM, 2002, pp. 85–95.

S. Yousaf Shah S. Yousaf Shah is presently a PhD candidate at the Computer Science department at Rensselaer Polytechnic Institute (RPI), Troy NY. He received his MS (2011) degree in Computer Science from Rensselaer Polytechnic Institute (RPI), Troy NY and Bachelors in I.T (2005) degree from National University of Sciences and Technology (NUST), Islamabad Pakistan.

Boleslaw K. Szymanski Dr. Szymanski is the Claire and Roland Schmitt Distinguished Professor of Computer Science and the founding director of the Network Science and Technology Center at Rensselaer Polytechnic Institute (RPI), Troy, NY. He is also director of the Social Cognitive Networks Academic Research Center (SCNARC). Dr. Szymanski received his Ph.D. degree in computer science from the National Academy of Sciences, Warsaw, Poland, in 1976. His interests include network science, and social and computer networks.

Petros Zerfos Dr. Zerfos is a Research Staff Member and Manager of the Next Generation Applications group at Thomas J. Watson Research Center. He received his PhD (2005) and MSc (2002) in Computer Science from the University of California, Los Angeles (UCLA) and his B.Eng. from the National Technical University of Athens, Greece in 1999. His research interests primarily lie in the areas of cloud computing, BigData analytics services in the cloud, predictive analytics for network and service management, design of wireless network gateways and performance analysis of network protocols. He has (co-) authored more than 50 papers in technical journals and conferences and filed more than 30 patents. Dr. Zerfos served as associate editor of ACM/Springer Wireless Networks Journal and as guest-editor of the ACM/Springer Mobile Networks and Applications (MONET) Journal.

Christopher Gibson Christopher Gibson is technologist and manager at the Emerging Technologies at IBM Hursely UK.