SOAMANET: A Tool for Evaluating Service-Oriented Architectures on Mobile Ad-hoc Networks

Himanshu Neema, Anand Kashyap, Robert Kereskenyi, Yuan Xue, Gabor Karsai
Institute for Software Integrated Systems
Vanderbilt University
Nashville, TN 37203
{himanshu, akashyap, roby, yxue, gabor}@isis.vanderbilt.edu

Abstract—**Service-Oriented Architectures (SOAs) are increasingly being used for designing and building large-scale networked and distributed systems. Catering to the complex and dynamically varying needs of business applications/clients, these systems must usually be realized by dynamically composing a variety of network-available services. Evaluation of large-scale SOAs, particularly on *dynamic* network platforms, such as Mobile Ad-hoc Networks (MANETs), is a non-trivial problem that requires not only a correct modeling of SOAs and the network platform, but also their relationships. This paper describes a new tool – SOAMANET – to design and rapidly synthesize simulations for the experimental evaluation of SOAs on MANET platforms. With its modeling techniques and analysis capabilities, SOAMANET allows simulation-based and system execution-based analysis of dynamic SOA and/or MANET designs and implementations.**

Keywords- service-oriented architectures; mobile ad-hoc networks; model-based integration; modeling and simulation; workflow modeling and execution; service discovery

I. Introduction

With the advances in networking technology and the availability of cheap and efficient mobile hardware, there is a growing trend of the use of networked and distributed mobile devices – which have now become a backbone for various strategic and tactical complex applications. These complex military applications are typical composed several different and collaborating smaller applications that are accessed as a service by authorized entities on an on-demand basis. We call such an approach as Service-Oriented Architectures (SOAs) – an architectural style of building large-scale business and military applications that run on networked and distributed platforms and are constructed by composing various constituent network-available and distributed services. Evaluation of such large-scale SOAs, particularly on *dynamic* network platforms, such as MANETs, is a non-trivial problem that requires not only a correct modeling of SOAs and the network platforms, but also the *relationship* between the two. For example, services from only those service providers must be accessed which are network-reachable both at the time of service invocation as well as when the service execution will finish. Even in this simple example one must consider the mobility patterns of the involved nodes and analyze where they will be at the time service execution will finish. Physical testing of such systems is not practical because of the limited resources. Also, theoretical analysis of these systems usually makes unrealistic simplified assumptions to accommodate their highly dynamic nature. Extensive modeling and analysis of these systems is essential to ensure that the models and the analysis support the software and systems engineering process. Therefore, a simulation tool that provides such high-fidelity modeling of SOAs, MANETs, and the dependencies between them is much needed. As further discussed in the related work below, there has been some effort in developing simulations for this purpose [1], [2], [3], [4], [5], but none of the existing tools can fully support this function.

Building a tool for simulation/evaluation for SOAs on MANETs is highly challenging and requires a different approach from existing tools such as traditional network simulators/emulators, application workload simulators, and MANET performance evaluators. Fig. 1 below shows the key requirements of such a tool. Firstly, a “software as a service” model must be supported by the tool so that smaller pieces of software are packaged as services that can be invoked by authorized nodes according to the needs of the application. Secondly, since the overall application is composed by combining these services in a meaningful way to perform its needed functions, the tool must be able to combine a related group of services into a larger service that accomplishes a higher-level objective. In addition, the binding between “service need” to “service provider” is dynamic, i.e., the exact service provider needed to satisfy a service need of an application is only determined at run-time. This is because a number of service providers might be configured to provide the same service to enhance application efficiency and reliability and the appropriate service provider can be determined (among the ones that are network-reachable at the time of the need) only at runtime. This may require optimization analysis to choose the best provider depending on the service requirements, time available for the service execution, and current and future layout of the mobile nodes for maximum chances of connectivity during the service invocation and completion. Further, realistic SOAs often involve parts that are data driven – a dataflow network may trigger part of the SOA or a service may fork a dataflow process and wait for it to finish.

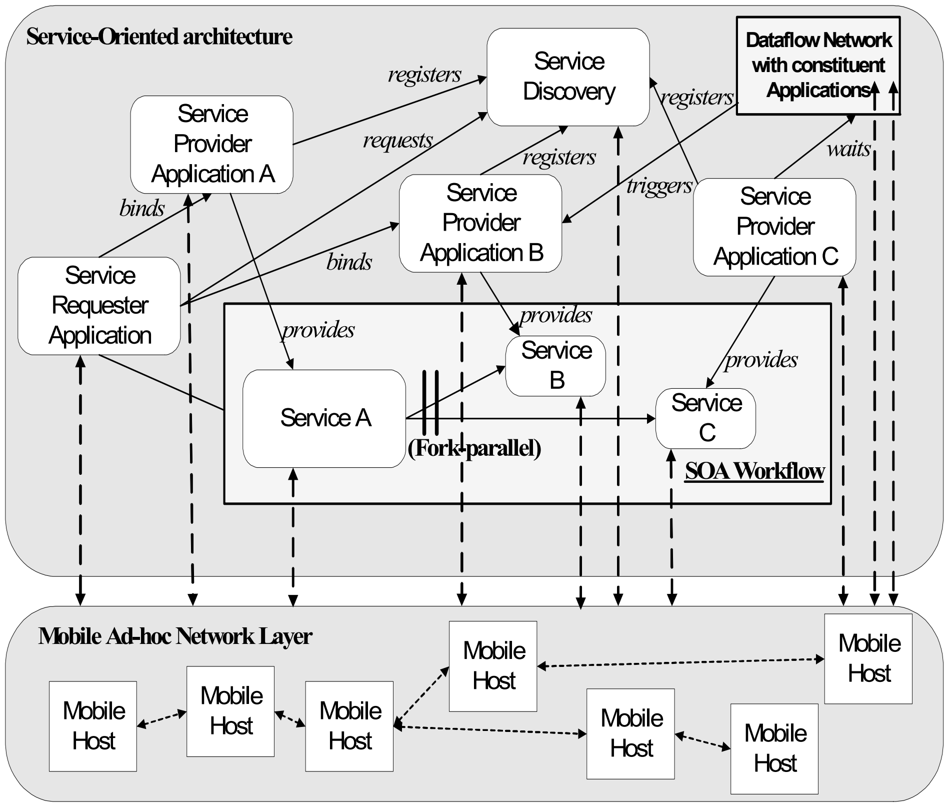
Service-Oriented architecture
Service Discovery
Dataflow Network with constituent Applications
registers
requests
registers
registers
triggers
waits
binds
binds
Service Provider Application A
Service Provider Application B
Service Provider Application C
Service Requester Application
provides
provides
provides
Service A
Service B
Service C
(Fork-parallel)
SOA Workflow
Mobile Ad-hoc Network Layer
Mobile Host

Figure 1: Tool requirements for evaluation of SOAs on MANETs

Additionally, in a mobile ad-hoc network, the network nodes constantly move causing the network topology to change dynamically and unpredictably (e.g., Command and Control (C2) applications with several kinds of mobile nodes and mobile ad-hoc networks, and IP-gateways connecting air-assets with Inter-flight data-links to networked ground vehicles), thereby requiring the SOA modeling and simulation tool to faithfully model and simulate the network topology changes and its effects on the SOA execution. For example, parts of SOA application may be running on multiple geographically separated MANETs that are connected to each-other only via satellite links. Further, a variety of complex workflows [6], including a hierarchy of parallel and sequential patterns, are often inseparable parts of business and military applications – which must be supported by the SOA modeling and simulation tool. This again becomes challenging by the introduction node mobility in MANETs because the assumptions about fixed networks and ever- reachable SOA applications no longer remain valid. For example, a service provider may become separated while executing a service and the workflow engine must be able to detect this situation and find another service provider for executing the failed service. Moreover, the SOA modeling and simulation tool must be integrated with a network simulation tool where data transfer and routing services are provided.

Thus, in order to build a simulation/evaluation tool that provides dedicated support for dynamic service composition of general-purpose SOAs on mobile ad-hoc platforms, extensive modeling and analysis techniques are essential for SOAs, MANETs, as well as the dependencies of SOAs on MANETs. Owing to the complex requirements presented above, an iterative approach (e.g., model-based system design, analysis, and synthesis) must be used to build, evaluate, and adapt these applications on an on-going basis. A model-based construction goes in lockstep with analysis-based validation, i.e. as the system design evolves, models are created, refined, and immediately validated. This is essential to enable system design and evaluation before the prototype is built, and to facilitate the system development.

There has been some effort in developing simulations to cater to this requirement. However, most of the tools generally found provide simulation support for either service-oriented architectures (SOA research) [7], [8], [9], or mobile ad-hoc networks (networking research) [10], [11]. To our best knowledge, a tool that provides dedicated support for combined simulation of SOA over MANET is still available and is much needed.

In this paper, we present SOAMANET (Service-Oriented Architectures on Mobile Ad-hoc NETworks) [12] – a novel tool that can provide various reusable modeling and analysis capabilities that facilitate the model-based construction of these systems and be used effectively for the evaluation of large-scale and highly dynamic service-oriented architectures on MANET platforms. It uses Model-Integrated Computing (MIC) technology [13], [14], [15], [16] for building models and synthesizing simulation artifacts. Also, it provides a library of reusable general-purpose SOA applications that can be customized for specific system requirements by configuring their behavioral properties. MIC allows it to capture SOA requirements, system architecture, and the environment of the system in the form of high-level models. For example, a workflow and dataflow modeling language supports specification of SOA requirements. Also, the binding between services and application to network nodes as well as the network properties, such as the ad-hoc routing protocol and its parameters, can be specified in the models. Furthermore, based on the SOA on MANET requirements, its re-usable SOA applications can be configured to update these bindings at run-time.

The rest of the paper is organized as follows. In Section 2, we consider some of the existing tools in this area and describe some of their strengths and weaknesses. In Section 3, we provide the architecture and design of SOAMANET in detail. In Section 4, we present an overview of the modeling and evaluation process using SOAMANET. In Section 5, we present two experimental case studies and discuss the results of our evaluation. Finally, Section 6 concludes the paper.

II. RELATED WORK

There are several existing works that define and support evaluation of service-oriented architectures. For example, [7], [8] consider SOAs as a uniform means to offer, discover, interact with and use capabilities of loosely coupled and interoperable services. However, these are primarily focused on the use of Web Services as the underlying platform [9]. These make an assumption that the network and the SOA applications, such as service providers and discovery, are always available. However, in MANETs, nodes frequently disconnect and reconnect and even entire sub-networks may become separated. Thus, these tools and techniques are not suitable for systems involving SOAs over MANETs.

Also, mobile ad-hoc networks have now become an active area of research. However, research in this area is mainly focused on the design and evaluation of MANET routing protocols and transport protocols [10], [11]. These techniques do not address the dynamic composition of services that are needed for SOAs and, more importantly, the dynamic nature of the dependency of SOAs on MANETs.

Finally, some research has been done lately in combining service-oriented architectures on mobile ad-hoc networks. For example, in [1] a method is presented for migrating existing desktop applications to mobile environments using XML compression techniques and web-services. Reference [2] focuses on policies of service composition and their effects on the SOA performance. In [3], an agile computing based approach is presented for migrating existing application to MANET environments. Also, [17], [18] present a comprehensive summary of discovery protocols that can be used for SOA on MANETs. In [4] a cross-layer design approach is presented for SOAs on MANETs and tested on real network and network devices (viz. Nokia internet tablets). Reference [5] provides a similar approach in the Vehicular Ad-hoc Networks (VANETs) and presents its preliminary prototype experimental evaluation. In contrast to current efforts, we present a dedicated tool for seamless evaluation of SOA and/or MANET designs in various ways such as complete simulation in the network simulator, simulation using externally running real applications. We present a general-purpose SOA simulation using a library of custom developed application, device, mobility, and routing modules. Moreover, our model-based approach enables rapid synthesis of SOAs on MANET simulations, which further facilitates development of such systems.

III. SOMANET DESIGN

A. Architecture Overview

The SOAMANET tool provides a rich set of modeling techniques and analysis capabilities for the evaluation of these complex and highly dynamic SOA and/or MANET designs and implementations. Fig. 2 shows the high-level of architecture of the SOAMANET tool. In the figure, the non-shaded parts represent the key components of the SOAMANET tool, whereas the lightly shaded box represents the external simulation at run time.

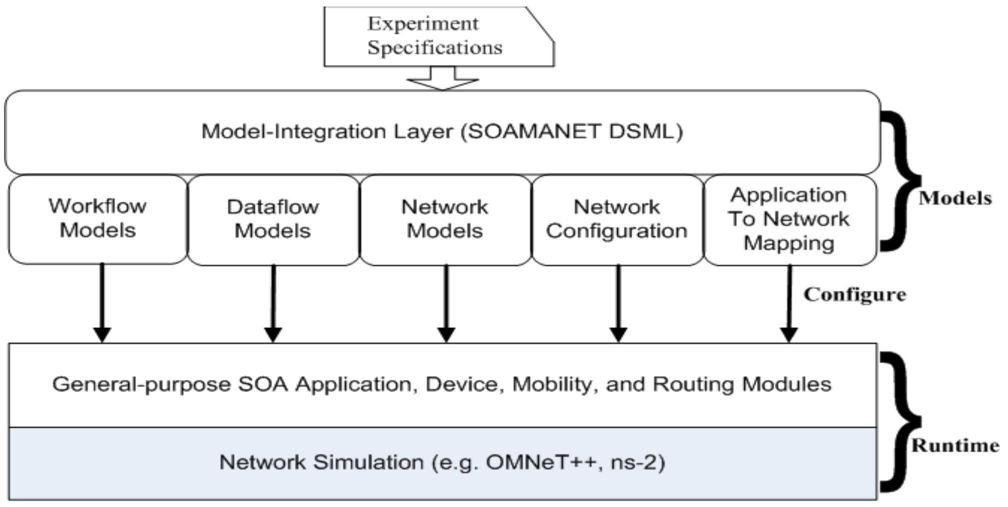
Experiment Specifications
Model-Integration Layer (SOAMANET DSML)
Workflow Models
Dataflow Models
Network Models
Network Configuration
Application To Network Mapping
Models
Configure
General-purpose SOA Application, Device, Mobility, and Routing Modules
Network Simulation (e.g. OMNeT++, ns-2)
Runtime

Figure 2: SOAMANET Architecture

The experiment specification provides a description of the system to be evaluated. This is used to create a domain-specific SOA modeling language (DSML) that is used to build models for SOA modeling and simulation. These models are used next to automatically configure the simulation at run-time. The run-time simulation platform consists of general purpose SOA application, device, mobility, and routing modules that run on an off-the-shelf network simulator that has support for mobile ad-hoc routing such as OMNeT++ [19] or ns-2 [20]. In this paper, we provide implementation details on how SOAMANET is integrated with OMNeT++.

Below we describe each of these key components of SOAMANET in detail.

1) Model Integration Layer: In SOAMANET, we use our existing metamodel-based infrastructure, called the Model-Integrated Computing (MIC) tool-suite [13], [14], [15]. In particular, we use the Generic Modeling Environment (GME) [15] meta-programmable MIC toolkit and the Universal Data Model (UDM) [15] MIC component. Using GME's metamodeling language, we compose a domain-specific modeling language [16] (i.e., SOAMANET DSML) based on experiment specification for a specific SOA on MANET simulation. In GME, this is automatically instantiated into a modeling tool that understands and enforces the modeling paradigm defined by the metamodels. Further, we use UDM's meta-programmable interface, to attain uniform access the SOAMANET models to configure our generic SOA application models.

2) Workflow Models: SOAMANET DSML supports modeling of a variety of workflow patterns [6]. From basic, sequential/parallel execution of services, the modeling language supports hierarchical construction of workflow models using the concept of Subworkflows.

3) Dataflow Models: Realistic SOAs often involve parts that are data driven. As opposed to Workflows, Dataflows involve data-oriented applications that depend heavily on the available of data tokens in the requisite quantities. For example, a LocationBasedSensor may sense situational awareness (SA) data in a given area and report SA information to an aggregator node, which aggregates SA information from different sensors according to domain rules of composition and freshness of data, and further disseminates it to the subscriber agents.

4) Network Models: In SOAMANET, most of the network modeling is done using the graphical modeling environments of the network simulation tool used in the runtime environment, such as OMNeT++ [19] or ns-2 [20]. However, as explained below, application deployment on network nodes is supported.

5) Network Configuration: In addition to linking the application models to the corresponding network nodes, SOAMANET also supports modeling time configuration of some of the key aspects of the network, such as the MANET routing protocols used and their parameters, and the changes in communication links among network nodes based on application requirements. For example, in a scenario DYMO [25] MANET routing protocol and logical communication channels from one group to another can be specified.

6) Application to Network Mapping: In SOAMANET, the mapping of SOA applications to the network nodes can be specified in the models. This is highly useful for rapid synthesis of SOA simulation/experiments. Further, the type of applications can be configured based on the models used.

7) General-purpose SOA modules: SOAMANET provides a library of general-purpose SOA modules. Some Workflow oriented application modules include Service Provider, Service Discovery, WorkflowInvoker, and WorkflowEngine. These applications were developed based on their use-case analysis to fulfill general SOA simulation requirements. Also, Some Dataflow oriented application modules include Sensor, Data fuser, Publisher, Subscriber, StreamingEndPoint, CellCallManager, etc. Additionally, SOAMANET provides several device models needed to support complex requirements of SOA over MANETs, such as StandardHost with full-duplex Ethernet cards, MANETHost with multiple WLAN cards. Also, support for system specific mobility modules exists to move network nodes according to a configured mobility path in a trace file, or with generic mobility patterns (e.g. CircleMobility or MassMobility) that restrict node movements to within a partial region of the overall geographical area simulated. Further, several routing modules needed for SOA modeling and simulation are supported, such as DynamicIPRouting for nodes that move from one MANET to the other.

8) Runtime Configuration: The runtime environment involves a network simulation tool. We currently support OMNeT++ - which can be easily integrated within the SOAMANET DSML to apply and configure its generic SOA application, device, mobility, and routing modules. The runtime configuration is responsible to parse domain models created using the DSML and translate them into SOA simulation configuration artifacts and/or interpret them at runtime to simulate the designed SOA models, such as Workflows, Dataflows, and their inter-dependencies.

B. Implementation/Design Details

SOAMANET is a SOA application simulator that supports modeling of network/topology changes of MANETs and can be configured to run on different network simulation tools that support MANET routing, such as OMNeT++ [19].

1) Model-based approach

SOAMANET uses a model-based approach to compose a Domain-Specific Modeling Language (DSML) [16] dedicated to and designed for the specific needs of SOAs and MANETs. The SOAMANET DSML is designed using the Model-Integrated Computing (MIC) [13], [14], [15] tool-suite and represents a family of models that can be created using it.

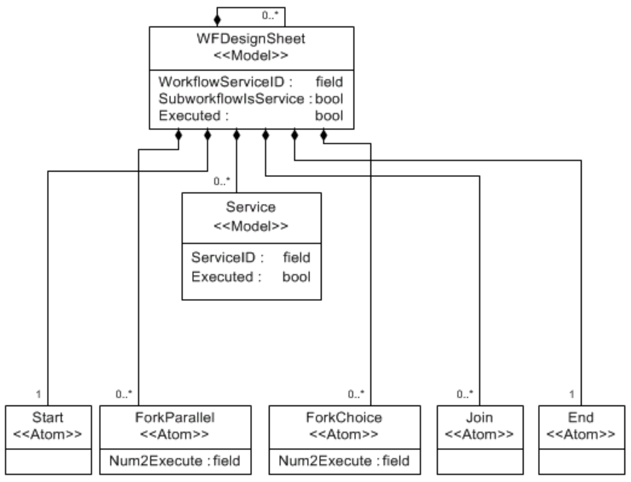

Figure 3: SOAMANET Metamodel: Workflows

We leverage the Model-Integrated Computing (MIC) [13], [14], [15] tool-suite. We use the Generic Modeling Environment (GME) [15] meta-programmable MIC toolkit to create the SOAMANET DSML that captures key SOA requirements, system architecture, and the environment of the SOA/MANET system in the form of high-level models. These models are that used to configure SOAMANET's generic application models using another MIC component called Universal Data Model (UDM) [15]. As an example, we will describe the Workflow and Dataflow parts of the SOAMANET metamodel. The idea here is to compose complex functions from individual services that are executed on service provider nodes on a network. *Workflows* are sequential/parallel graph of tasks that are invoked by a workflow engine in sequence and services are executed using a service registration/discovery capability. Fig. 3 shows the containment class diagram for workflow modeling elements. The root level *Workflow* (WFDesignSheet) has a *Start* and an *End* element and may contain any number of *Service*, *ForkParallel*, *ForkChoice*, *Join*, and *Subworkflow* elements. ForkParallel creates branches in the Workflow that must be executed in parallel and among these *Num2Execute* number of branches must finish for the entire ForkParallel to finish. On the other hand, a ForkChoice creates branches in the Workflow among which only *Num2Execute* number of branches must be executed that are chosen based on a probability assigned to each of the branches such that the sum of probabilities of all branches equals to 1.

In addition to containment, the language must also specify rules of association among contained elements. These include connections that originate from each of the contained elements. For space limitation, in Fig. 4, we show only the connections that originate from Service elements. As can be seen, a connection from a *Service* can lead to another Service (modeled using *Sv2SvCon*), a *ForkParallel*, a *ForkChoice*, *Start* of another Workflow, *End* of the current Workflow, and end of a fork (*Join*). Additionally, a *Service* may 'trigger' a Dataflow network (modeled using *Sv2SourceEntityCon*) or 'wait' for a Dataflow network to finish execution (modeled using *SinkEntity2SvCon*).

Real-world SOA applications are often driven by events generated from the sensing and fusion processes. For example, a collection of sensors may generate data packets that are sent through several processing stages that perform signal processing or data fusion.

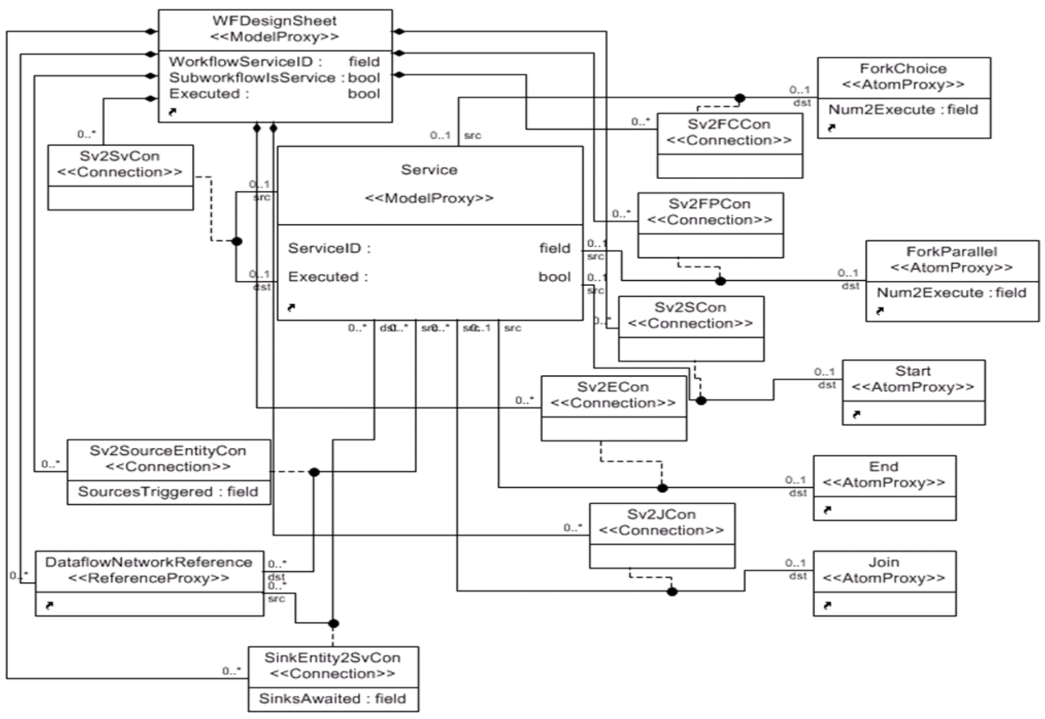

Figure 4: Connections origination from "Service" workflow elements

This model is different from the workflow execution model: it is driven by data, not by a centralized workflow engine. Interested reader is referred to SOAMANET website [12] for further description of other parts of the SOAMANET DSML, such as Dataflow networks and their dependency with Workflows.

The SOAMANET DSML is used to model SOA architectures and processes. This includes the workflows in the SOA application (e.g., service A and B must be finished before service C can be started), dataflows in the SOA application (e.g., sensors on aircraft A and B periodically output situational awareness (SA) data to the data fusion application on another aircraft, which further triggers a specific workflow based on the type of SA data that it had received), and experiment configurations for the analysis of various SOA and/or MANET designs and implementations. For example, Fig. 5 shows an example of a Workflow model built using the language. The Workflow involves execution of a Service, followed by a parallel execution of three branches, one of which is a sub-workflow. Also, Fig. 6 shows an example of a SOA application model that includes both Workflow and Dataflow elements. The execution begins with service *SV1*, which on finishing triggers two instances of the shown Dataflow network. The execution of the next service *SV2* does not start until the Dataflow network instance *DataflowNetwork1Ref2* has finished. *SV3* get enabled as soon as *SV2* finishes. However, *SV3* waits for Dataflow network instance *DataflowNetwork1Ref1* to finish before starting its execution. The entire Workflow is considered finished when the service *SV3* is finished.

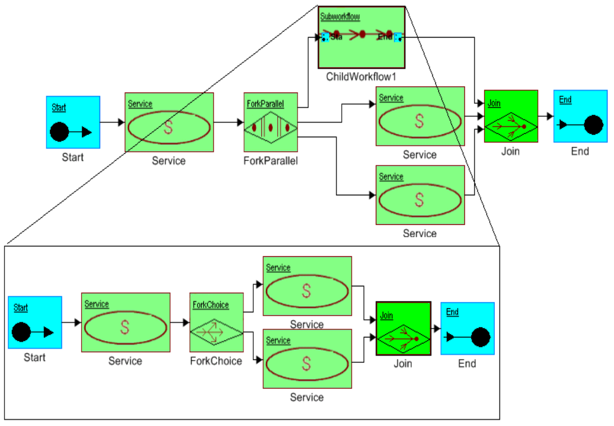

Figure 5: Example Workflow model

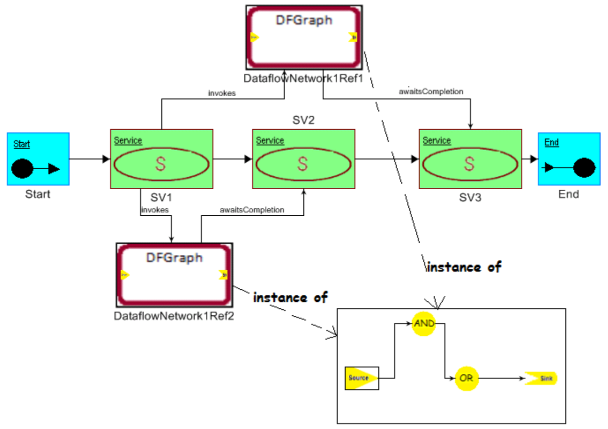

Figure 6: SOA application: Mixed Workflow and Dataflow

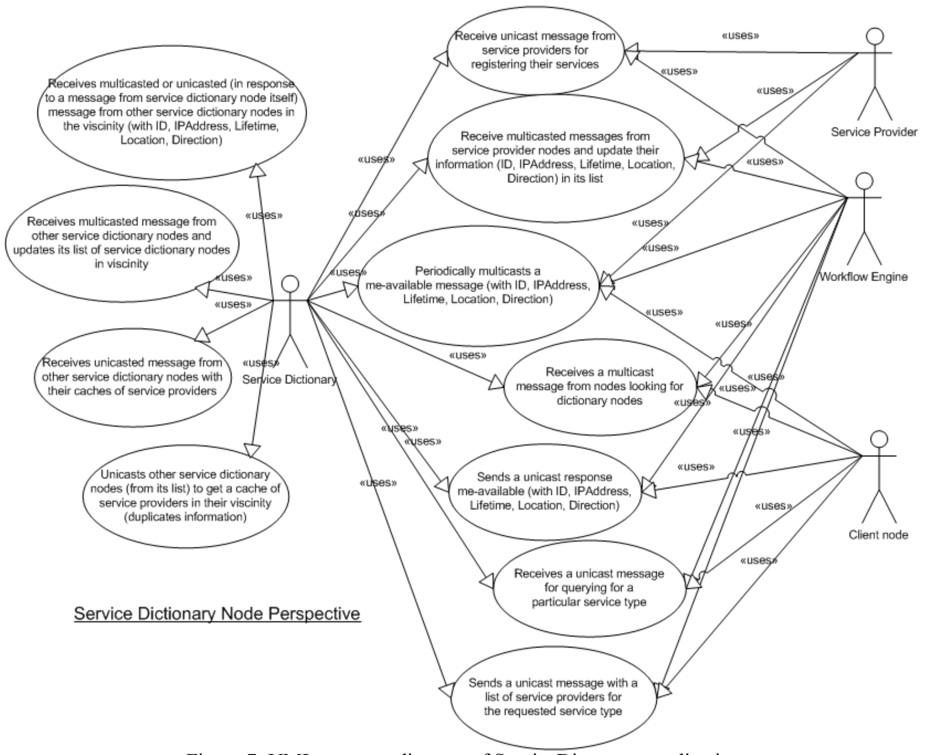
Receives multicasted or unicasted (in response to a message from service dictionary node itself) message from other service dictionary nodes in the viscinity (with ID, IPAddress, Lifetime, Location, Direction)
Receives multicasted message from other service dictionary nodes and updates its list of service dictionary nodes in viscinity
Receives unicasted message from other service dictionary nodes with their caches of service providers
Unicasts other service dictionary nodes (from its list) to get a cache of service providers in their viscinity (duplicates information)
Service Dictionary
Receive unicast message from service providers for registering their services
Receive multicasted messages from service provider nodes and update their information (ID, IPAddress, Lifetime, Location, Direction) in its list
Periodically multicasts a me-available message (with ID, IPAddress, Lifetime, Location, Direction)
Receives a multicast message from nodes looking for dictionary nodes
Sends a unicast response me-available (with ID, IPAddress, Lifetime, Location, Direction)
Receives a unicast message for querying for a particular service type
Sends a unicast message with a list of service providers for the requested service type
Service Provider
Workflow Engine
Client node
«uses»
Service Dictionary Node Perspective

Figure 7: UML use-case diagram of ServiceDiscovery application

2) Application Design

Fig. 7 shows a simplified UML use-case diagram [21] of one such SOA application – the ServiceDiscovery. This use-case diagram covers different functional requirements of a ServiceDiscovery application. For example, it interacts with ServiceProviders to register services, with WorkflowEngine to provide list of applicable ServiceProviders that provide a service requested by a client.

3) Networking and Simulation

SOAMANET current supports a discrete-event simulator, OMNeT++ [19] for network modeling and simulation. OMNeT++ is open-source software. It has a highly modular & flexible architecture that is well-suited to augment it for the development of generic SOA software models. As previously discussed, SOAMANET contains a library of SOA applications. These applications are built in C++ in a modular and reusable manner. These are currently configured as UDP applications in OMNeT++ language.

SOAMANET uses a library called INETMANET [22] that supplies models for the whole network stack including MANET routing protocols [23] that are also supported by SOAMANET for SOA application configuration, such as AODV [24] and DYMO [25]. SOAMANET further extends INETMANET's routing support by providing modules such as Dynamic IP Routing for dynamically assigning IP addresses to nodes that move from one network domain to other, bridging modules for connecting geographically separated MANETs and streaming-based communication.

SOAMANET provides not only custom mobility modules such as Region Mobility that supports MassMobility pattern restricted inside a given region, but also supports any custom mobility pattern by including support for configuring a trace file for the mobility path. In addition, SOAMANET also extends a number of INETMANET's device modules such as StandardHost, WirelessHost, Router, ManetHost, etc. in order to support SOA simulations that involve off-network functionality. In scenarios, such as an aircraft striking a mobile hostile ground vehicle, not only the target strike can be modeled instead of pre-configuring it, but also the confidence of the hit can be modeled based on the mobility of the aircraft and the hostile ground vehicle. We do not support, however, terrain (3D) modeling of objects, mobility, and communications.

4) SOAMANET Extensions

Thus far we have only discussed the application of SOAMANET as a simulation tool for SOA over MANETs (say Mode 1). However, with its highly re-usable modeling techniques and analysis capabilities, it can be used to support a variety of applications. For example, SOAMANET supports running simulations completely external to a network simulator, i.e. completely on real network nodes (Mode 2). Additionally, external/real applications can run externally while using OMNeT++ based network simulator as the network between them. This is accomplished by either establishing socket-based communication between corresponding proxy nodes inside the network simulator (Mode 3) or with the use the Command and Control (C2) Windtunnel (C2WT) [26] simulation integration framework (Mode 4). SOAMANET's Mode 1 and Mode 4 are highly useful when building a realistic network or running real applications becomes practically infeasible due to cost, security, or safety concerns. Additionally, while Mode 2 represents the "most realistic" simulation using all real entities, it often is not practical in the SOA/MANET evaluation context, particularly because dynamic networking platforms such as MANETs are used. This is because space and cost limitations constrains the use of real mobile devices and also the real hosts such as tanks and aircrafts are usually unavailable for testing purposes. In such cases, Mode 3 & 4 represents a good alternative that can provide safe, secure, confined, and efficient solution that also performs high-fidelity simulations by moving the networking and mobility inside SOAMANET simulator. Discussion of modes 2, 3, and 4 is beyond of the scope of this paper due to the space limit.

IV. MODELING & EVALUATION PROCESS

Evaluation of complex service-oriented architectures on MANET platforms requires extensive modeling and analysis of these systems to ensure that the models and the analysis support the software and systems engineering process. For this reason, as shown in the Fig. 8, we use models throughout the development lifecycle of SOA simulations. High-level, coarse-grained models are used initially, which are then gradually refined into finer-grained models and/or implementations as the design process proceeds.

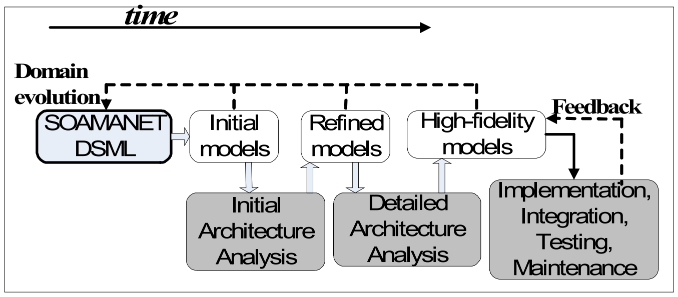
time
Domain evolution
SOAMANET DSML
Initial models
Refined models
High-fidelity models
Feedback
Initial Architecture Analysis
Detailed Architecture Analysis
Implementation, Integration, Testing, Maintenance

Figure 8: Use of models in SOAMANET

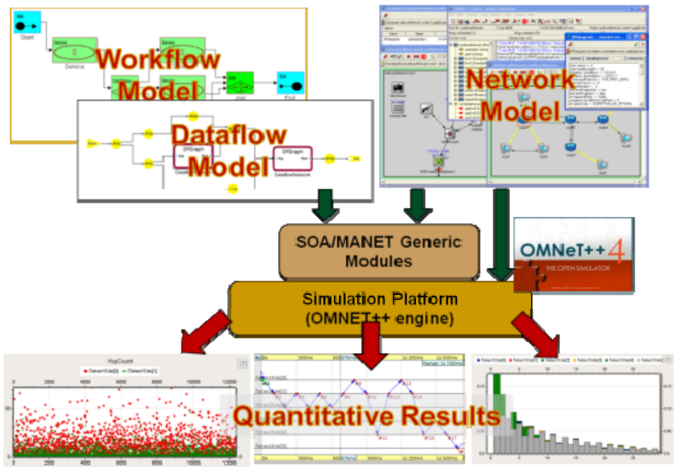

Figure 9: SOAMANET Experimentation Framework

Fig. 9 illustrates the overall SOAMANET approach for the evaluation of SOA and/or MANET designs and implementations. First the SOA Workflows and Dataflows are modeled using the graphical modeling environment of the SOAMANET DSML. Network platform models are captured using a standard textual/graphical modeling tool from OMNeT++ and represent the network topology, mobility patterns, performance properties, network stack composition, MANET routing protocols, and routing tables. The models are then used to rapidly configure generic simulation modules and the experimental execution environment. These generic SOAMANET modules are used to simulate the application software and network services, protocols, and layers. Once the modeling and configuration is done, the OMNeT++'s discrete-event simulation engine executes the high-fidelity SOA/MANET simulations. The simulations can be run either graphically and debugged or they can be run in a batch mode on a command-line to perform large scale experiments. The experimental data is then collected and analyzed to evaluate performance of the modeled SOA architecture in terms of the chosen Workflows, Dataflows, Workflow-Dataflow connections, Network designs, and MANET protocols with respect to the configured behavior of SOA applications such as Service Provider, Service Discovery, Workflow Engine, etc.

V. Experimental Results

Using SOAMANET's graphical modeling and synthesis environment and its number of reusable application, device, mobility, and routing modules, we have conducted a large number of realistic SOA/MANET experiments [12]. In this section, we present results briefly from two case studies.

A. *Scenario 1: Topology Resilience*

This scenario aims to study the execution of a SOA workflow in a MANET environment when the network topology changes drastically. In the scenario, as illustrated in Fig. 10, initially all nodes are connected in a **star** topology such that every node can talk to any other node. As the simulation progresses, the network topology changes to take the shape of **string-of-pearls** (all nodes in a straight line) representing a worst-case scenario for the mobile ad-hoc networking because most of the communication will need multiple hops, messages can get lost easily, and the performance degrades drastically. In this experiment, we used a mobility model order that takes timed coordinates of nodes in a trace file and moves the nodes according to the given update intervals (10 seconds in this experiment).

The MANET protocol used in this experiment is DYMO. Here, we included 6 service providers, a workflow engine, a workflow invoker, and a discovery node all of which are mobile. As shown in Fig. 11, a small workflow of 2 execution steps is used, where each of these execution steps represents a sub-workflow of 3 services, viz. SV1, SV2, and, SV3. We used 2 service providers for each of these services.

The key aim of the experiments was to study how SOA workflows execute during drastic changes in network topology. The parameters dealing with network setup, service binding, the workflow, and configuration of various SOA modules (e.g. workflow engine and discovery modules) were kept constant. We designed the experiment such that we overload the system in two ways - (a) the topology changes from star to string-of-pearls, and (b) the load on the system increases by more overlapping workflows executing simultaneously. For this, a total of 10 runs are executed and in each run, multiple copies of two-step workflow described above are invoked at times 10, 15, 20, 25, 40, 45, 50, 53, 56, 59, 60, 65, 70, and 72 seconds. The service execution time was fixed for all service providers to 3 seconds. The time the workflow engine waits for querying service providers from discovery node was 5 seconds. All nodes were configured to move using a custom mobility model that uses a trace file for desired coordinates of these nodes at given times.

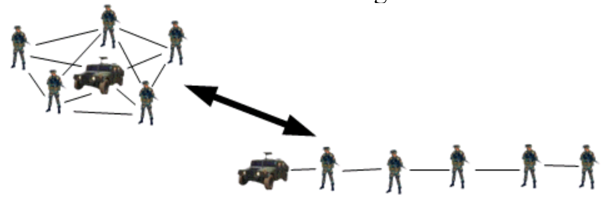
Figure 10: Topology Resilience: Overview

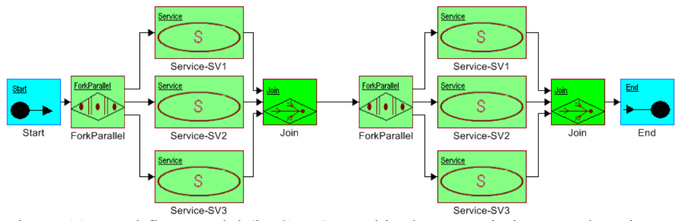

Figure 11: Workflow model (in GME), used in the scenario in several copies

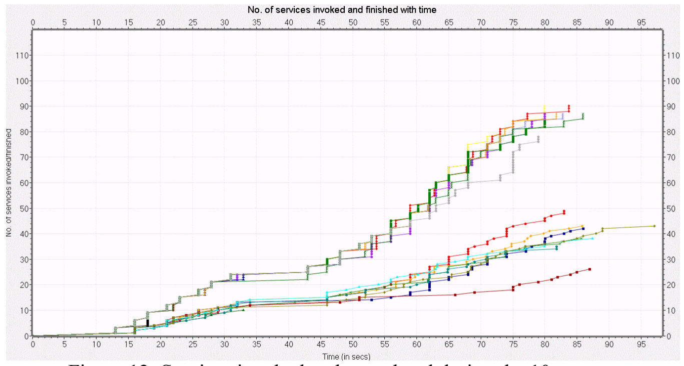

Figure 12: Services invoked and completed during the 10 runs

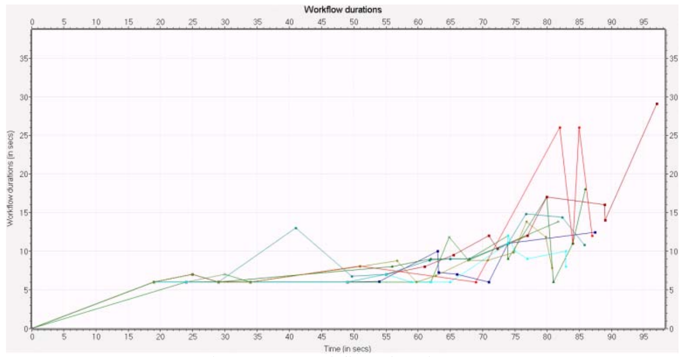

Figure 13: Workflow durations

Fig. 12 shows the number of services invoked and executed with time for the 10 runs. The top lines represent service invocation times and the bottom lines represent service completion times. As we can see, several services are dropped as the workflow executes. This is primarily due to network topology changes and increased load on the system due to multiple overlapped workflows. Also, Fig. 13 shows the durations of these workflows (difference between workflow invocation and completion times). Again, there is a marked increase in the duration due to drastic network topology changes and increased overlaps in the workflows.

From this experiment, it can be clearly seen that the execution of SOA workflows on a mobile ad-hoc network can be severely affected if the network topology changes drastically. As a result, the workflows take much longer to complete and several services get dropped. This shows that for better performance MANET networks should involve smaller and gradual changes in network topologies and should try to keep communicating nodes as close to each other as possible even as they move.

B. *Scenario II: Mission Planning*

This scenario aims to study the latency of SOA messages and information confidence in a MANET environment when the network mesh can get fragmented in time and recombine later. A subset of nodes represents the mission information providers generating mission information periodically. These nodes distribute the mission information via publisher nodes. End users can subscribe to these publisher nodes to receive the mission information updates. For this scenario, as shown in Fig. 14, we created a simulated area with four logically separated regions in it. All of the nodes on the field belong to the same ad-hoc wireless network and hence any of the nodes can talk to any other node if there is a multi-hop route between them. The MANET protocol used is DYMO. In each area a local mission information source is included that provides different types of mission information (SA data) with some periodicity and sends it to the local publisher. Only local publisher node distributes SA data to the local subscribers – which are interested in SA data of all 4 regions. To be able to distribute each type of SA data to all the subscribers in all the regions, publisher nodes can synchronize among themselves periodically with a given frequency. All the nodes on the field move only in their local region following a mass mobility pattern.

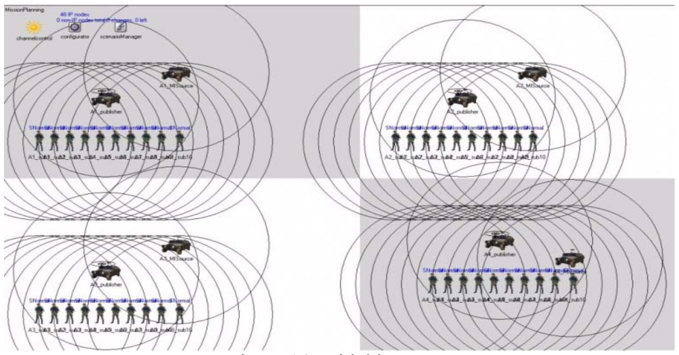

Figure 14: Initial layout

We conducted two experiments with this scenario. With a periodicity of 30 seconds each, the four sensors start generating SA data at time 2, 4, 6, and 8 seconds respectively. The publishers synchronize at every 10 seconds. Periodicity of mission information sources is set to 5, 10, and 30 seconds respectively in the three runs used in both experiments. In first experiment, we varied the synchronization frequency of publishers as 5, 10, and 20 seconds. We calculated the subscribers' confidence of mission information for periods of 90 seconds during the simulation. The confidence in a period is the number of received messages in that period divided by the number of all the generated messages during that period. For each period, we took the average of all the 40 nodes' confidence value – this is shown below in Fig. 15. In second experiment, the synchronization period was kept constant at 10 seconds, while 2 runs were executed – with and without reliable message passing. Reliable message passing is turned on by enabling UDP application-level handshaking and configuring timeout after which messages are re-sent if an ACK is not received (to a maximum of 5 attempts in this experiment). Fig. 16 below shows how the subscriber confidence increases simply by turning on the reliable communication.

From this experiment, we learnt that when mission information is generated frequently or synchronization period of publishers is increased, subscribers have to work more with stale information. Also, the shorter the time-period within which subscribers must be kept up-to-date, the greater is the chance that the subscribers will be working on stale information. Furthermore, use of reliable communication dramatically increases the nodes' confidence.

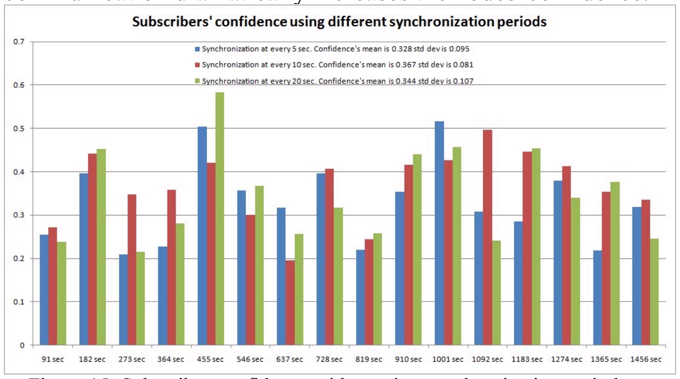

Figure 15: Subscriber confidence with varying synchronization periods

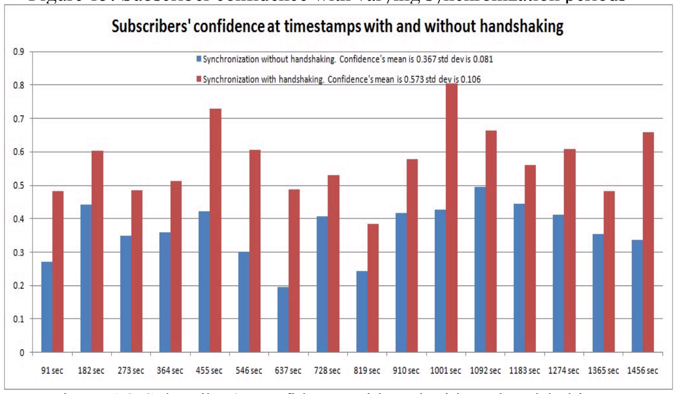

Figure 16: Subscriber's confidence with and without handshaking

VI. CONCLUSION & FUTURE DIRECTIONS

Evaluation of general-purpose service -oriented architectures on mobile ad-hoc networks involves a variety of challenges. In this paper, we presented SOAMANET [12] – a novel tool for rapid design and evaluation of SOAs on MANETs. We showed that simulations for the experimental evaluation of SOAs on MANET platforms can be *rapidly* synthesized using the MIC technology [13], [14], [15] and a library of custom developed application, device, mobility, and routing modules (e.g., the complete topology resilience experiment was designed, executed, and analyzed in less than an hour! We also presented two case studies for the experiments conducted using SOAMANET and showed that with its modeling techniques, analysis capabilities and an intuitive user interface, it provides a powerful tool for designing, developing, and analyzing dynamic SOA and/or MANET designs and implementations. SOAMANET's has a highly modular and flexible architecture which allows it to not only extend its application in several modes, but also for any extensions that may be needed by any specific domain.

We are currently working on enhancing SOAMANET's capabilities to support SOA/MANET scenarios with other Command and Control (C2) Architecture elements as well as to apply its generic application modules for in-the-field evaluation of SOA/MANET designs – network protocols, anticipated workflows, dataflows, and mobility patterns. In the future, we also expect to extend SOAMANET to support integration with additional simulation engines to increase its functionality in terms of the supported SOAs, Network topologies and protocols, and MANET routing protocols.

ACKNOWLEDGMENT

This research was sponsored by the U.S. Army Research Laboratory and was accomplished under Cooperative Agreement W911NF-07-2-0020. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation heron. The authors would also like to thank David Claypool, former research engineer at BAE Systems, for his guidance on drafting realistic experiment scenarios.

REFERENCES

[1] Yuri Natchetoi, Huaigu Wu, Yi Zheng, "Service-Oriented Mobile Applications for Ad-Hoc networks," scc, vol. 2, pp.405-412, 2008 IEEE International Conference on Services Computing Vol. 2, 2008.

[2] Dipanjan Chakraborty, Anupam Joshi, Tim Finin, Yelena Yesha, "Service Composition for Mobile Environments," Journal of Mobile Networks and Applications, Springer Netherlands, 2005, pp 435-451.

[3] N. Suri, et al., "An Adaptive and Efficient Peer-to-Peer Service-Oriented Architecture for MANET Environments with Agile Computing," in Proc. of ACNM 2008, Brazil, pp 364 – 371.

[4] T. Halonen, and T. Ojala, "Cross-layer design for providing service oriented architecture in a mobile Ad Hoc network," In Proc. Of MUM '06, vol. 193. ACM, New York, NY, 2006.

[5] Gonçalves, J. F., Esteves, E. F., Rossetti, R. J., and Oliveira, E, "Simulating Communication in a Service-Oriented Architecture for V2V Networks," In Progress in Artificial intelligence, LNAI, vol. 5816. Springer-Verlag, 15-26.

[6] W.M.P. van der Aalst and A.H.M. ter Hofstede, "YAWL:Yet Another Workflow Language," Information Systems, 30(4):245-275, 2005.

[7] R. Bruni, A. Lluch Lafuente, U. Montanari, E. Tuosto, "Service oriented architectural design," In Proc. of the 3rd International Symposium on Trustworthy Global Computing, 2007, LNCS.

[8] MacKenzie CM, Laskey KJ, McCabe F, Brown PF, and Metz R, "OASIS Standard Reference Model for Service Oriented Architecture 1.0," October 2006.

[9] The World Wide Web Consortium (W3C), Web Services Architecture Working Group, July 30, 2010, http://www.w3.org/2002/ws/arch/.

[10] J. Broch, D.A. Maltz, D.B. Johnson, Y.C. Hu, and J. Jetcheva, "Performance comparison of multi-hop wireless ad hoc network routing protocols," In Proc. of the 4th annual ACM/IEEE int. conf. on Mobile computing and networking, 1998, pages 85-97.

[11] G. Holland and N. Vaidya, "Analysis of TCP Performance over Mobile Ad Hoc Networks," In Proc. of the 5th annual ACM/IEEE int. conf. on Mobile computing and networking, 1999, pages 219-230.

[12] SOAMANET: Model-based tools for Service Oriented Architectures (SOAs) over Mobile Ad-hoc Networks (MANETs), July 30, 2010, https://wiki.isis.vanderbilt.edu/SOAMANET/index.php/Main_Page.

[13] G. Karsai, A. Ledeczi, S. Neema, and J. Sztipanovits, "The Model-Integrated Computing Toolsuite: Metaprogrammable Tools for Embedded Control System Design," IEEE CACSD, 2006.

[14] G. Karsai, J. Sztipanovits, A. Ledeczi, and T. Bapty, "Model-integrated Development of Embedded Software," In Proc. of the IEEE 91(1): 145-164 (2003).

[15] Escher Research Institute, MIC Toolsuite, July 30, 2010. http://www.escherinstitute.org/Plone/tools/suites/mic.

[16] A. Ledeczi, A. Bakay, M. Maroti, P. Volgyesi, G. Nordstrom, J. Sprinkle, and G. Karsai, "Composing Domain-specific Design Environments," IEEE Computer, Nov. 2001, Page(s): 44-51.

[17] C. Cho and D. Lee, "Survey of Service Discovery Architectures for Mobile Ad hoc Networks," Term Paper. Department of CICE, University of Florida, Fall 2005.

[18] Tyan, J. and Mahmoud, Q. H., "A comprehensive service discovery solution for mobile ad hoc networks," Mob. Netw. Appl. 10, 4, 423-434, 2005.

[19] OMNeT++ Community Site, OMNeT++, July 30, 2010, http://www.omnetpp.org.

[20] Information Sciences Institute, The Network Simulator - ns2, July 30, 2010, http://www.isi.edu/nsnam/ns.

[21] Object Management Group, Unified Modeling Langauge, July 30, 2010, http://www.uml.org.

[22] INETMANET: INET Framework for OMNEST/OMNeT++ 4.0, July 30, 2010, http://github.com/inetmanet/inetmanet.

[23] Internet Engineering Task Force (IETF), MANET charter, July 30, 2010, http://www.ietf.org/proceedings/53/179.htm.

[24] C. Perkins, E. Royer, and S. Das, "Ad-hoc On-demand Distance Vector (AODV) routing Internet-Draft," July 30, 2010, http://datatracker.ietf.org/doc/draft-ietf-manet-aodv.

[25] I. Chakeres and C. Perkins, "DYnamic MANET On-demand (DYMO) Routing Internet Draft," July 30, 2010, http://datatracker.ietf.org/doc/draft-ietf-manet-dymo.

[26] The Command and Control (C2) Windtunnel (C2WT), July 30, 2010, https://wiki.isis.vanderbilt.edu/OpenC2WT/index.php/Main_Page.