Trust-Based Task Assignment in Autonomous Service-Oriented Ad Hoc Networks

Yating Wang[†], Ing-Ray Chen,[†] Jin-Hee Cho*

[†]Virginia Tech
Department of Computer Science
{yatingw, irchen}@vt.edu

*U.S. Army Research Laboratory
Computational and Information Sciences Directorate
jinhee.cho@us.army.mil

***Abstract* — We propose and analyze a trust management protocol for autonomous service-oriented mobile ad hoc networks (MANETs) populated with service providers (SPs) and service requesters (SRs). We demonstrate the resiliency and convergence properties of our trust protocol design for service-oriented MANETs in the presence of malicious nodes performing opportunistic service attacks and slandering attacks. Further, we consider a situation in which a mission comprising dynamically arriving tasks must achieve multiple conflicting objectives, including maximizing the mission reliability, minimizing the utilization variance, and minimizing the delay to task completion. We devise a trust-based heuristic algorithm to solve this multi-objective optimization problem with a linear runtime complexity, thus allowing dynamic node-to-task assignment to be performed at runtime. Through extensive simulation, we demonstrate that our trust-based node-to-task assignment algorithm outperforms a non-trust-based counterpart using *blacklisting techniques* while performing close to the ideal solution quality with perfect knowledge of node reliability over a wide range of environmental conditions.**

Keywords—service-oriented mobile ad hoc networks, multi-objective optimization, task assignment, trust, performance analysis.

I. INTRODUCTION

In this paper, we are concerned with autonomous service-oriented mobile ad hoc networks (MANETs) populated with service providers (SPs) and service requesters (SRs). A realization of service-oriented MANETs is a web-based peer-to-peer service system with mobile nodes providing web services and users (also mobile) invoking web services. Unlike a web service system in which nodes are connected to the Internet, nodes in service-oriented MANETs are mobile and the communication between peers not within radio range is multi-hop with nodes in the system serving as routers. Service-oriented MANET applications are often realized in a coalition formation setting where a task must be accomplished and the SR, taking a role of the task leader, must perform node-to-task assignment (NTA) to assemble a team among SPs to accomplish the task.

We develop a trust management protocol specifically for autonomous service-oriented MANET applications. Previous work considered web service trust models [15, 16] and MANET trust models [5-8] separately without integrating unique MANET features with service-oriented applications. We demonstrate the *resiliency* and *convergence* properties of our trust protocol design for service-oriented MANETs in the presence of malicious nodes performing *opportunistic service attacks* and *slandering attacks*.

To demonstrate the applicability, we consider a coalition formation setting in which a mission must handle dynamically arriving tasks to achieve multiple objectives. The motivation is that real world service-oriented applications often have multiple objective optimization (MOO) requirements, where they often have conflicting goals. In this work, we consider three system objectives: (1) maximizing mission reliability based on task completion ratio; (2) minimizing utilization variance, leading to high load balance among all nodes; and (3) minimizing the delay to complete time-sensitive tasks, thus maximizing QoS. We note that objective of load balancing is in conflict with others since maximizing load balance may sacrifice task completion ratio and QoS.

The literature is rich in solution techniques and examples for solving a MOO problem, but is extremely unexplored in trust-based MOO solution techniques. Only [1-4] attempted using trust to solve MOO problems. Except [4], the environment is not MANETs.

The contributions of this work are in the following. First, to the best of our knowledge, this work is the first to solve a multiple objective optimization (MOO) problem dealing with

multiple, concurrent and dynamic task assignments with conflicting goals using trust in service-oriented MANETs. Our trust-based heuristic algorithm has a *linear runtime complexity*, thus allowing dynamic NTA to be performed at runtime. Second, this work proposes and analyzes a new design concept of trust-based MOO based on assessed trust levels to screen task team members for dynamic NTA. Lastly, we conduct a comparative analysis of our proposed trust-based heuristic member selection algorithm against the ideal solution with perfect knowledge of node status, demonstrating that our trust-based solution achieves solution efficiency without compromising solution optimality.

The rest of this paper is organized as follows. Section II describes our system model including the network model, attack model, trust protocol design, task model, and our MOO problem definition. Section III proposes a linear runtime complexity trust-based heuristic algorithm to solve the MOO problem. Section IV performs a comparative analysis of our proposed scheme against the ideal solution with perfect knowledge over node reliability as well as a non-trust baseline scheme and demonstrates that our trust-based scheme outperforms the non-trust-based counterpart using *blacklisting techniques* and performs close to the ideal solution quality. Section V concludes the paper and outlines some future research directions.

II. SYSTEM MODEL

A. Network Model

We consider a service-oriented MANET environment in which a node has two roles in executing operations: (1) as a service provider (SP) to support an operation; and (2) as a service requestor (SR) to request services in the process of initiating (and executing) a task. Nodes may be heterogeneous with vastly different functionalities and natures. For example, the entities may be sensors, robots, unmanned vehicles or other devices, dismounted soldiers or first response personnel carrying sensors or handheld devices, and manned vehicles with various types of equipment. We consider M ordered node types, $NT_1, \ldots, NT_M$, such that a higher node type has a higher capability than a lower node type. A node with a high node type also may involve a human operator and thus has additional trust dimensions pertaining to social trust [5, 6]. When mobile nodes are not involved in a task, they follow their routine-work mobility model. We use SWIM [10] in this paper.

We adopt a hierarchical structure to execute a mission consisting of dynamically arriving tasks. A commander node (CN) governs the mission team. Under the CN, multiple task leaders (TLs) lead task teams. The CN selects TLs at the beginning of network deployment based on the trustworthiness of nodes known to CN a priori and the TLs each recruit trustworthy SPs dynamically for executing the tasks assigned to them. A group key is used for communications among members to prevent outside attackers.

B. Attack Model

A node in the service-oriented MANET may be compromised, and exhibit the following malicious behaviors:

1. *Opportunistic service attacks*: a malicious node may collude with other malicious nodes to fail the task. It is opportunistic in the sense that a node would not perform attack when it does not see enough bad nodes around to perform a deadly attack at the expense of trust loss. Following the *Byzantine Failure* model [12], we assume that a task fails with at least 1/3 bad nodes executing the task.
2. *Bad-mouthing*: a malicious node may collude with other malicious nodes to ruin the reputation of a good node by providing bad recommendations against the good node so as to decrease the chance of the good node being selected for task execution.
3. *Ballot stuffing*: a malicious node may collude with other malicious nodes to boost the reputation of a bad node by providing good recommendations for the bad node so as to increase the chance of the bad node being selected for task execution.
4. *Packet dropping*: a malicious node may arbitrarily drop packets passing through it during packet routing to disrupt normal operations.

A malicious node could possibly perform self-promotional attack to boost its service quality information. However, we do not consider this attack because it can be easily detected, and accordingly a malicious node would expose itself as vulnerable, resulting in a low reputation. Also we do not consider malicious behaviors performing communication-level attacks such as data modification, Denial-of-Service or Sybil attacks [11]. We assume such behaviors are detected by network intrusion detection mechanisms [11, 14].

C. Trust Protocol

Our baseline trust protocol uses Beta (α, β) distribution [13] modeling a trust value in the range of [0, 1] as a random variable where α and β represent the number of positive evidence and negative evidence respectively, such that the estimated mean trust value of a node is $\alpha/(\alpha+\beta)$. When a task which a node participated in is executed successfully (unsuccessfully), this node's α is incremented by $\Delta\alpha$ (β is incremented by $\Delta\beta$ correspondingly). When we want to severely punish malicious behavior, we set $\Delta\beta >> \Delta\alpha$. In this paper, we propose a "penalty severity" parameter denoted by $\Delta\beta{:}\Delta\alpha$ to analyze its effect of trust penalty severity on our trust protocol performance. For all nodes, the initial α and β values are 1, representing ignorance.

Trust propagation and aggregation is performed periodically in every Δt interval. Trust propagation is done via recommendations. A trustor node evaluating a trustee node will select n_{rec} recommenders whom it trusts most to provide trust recommendations toward the trustee node. A recommender should only pass its direct interaction experience with the trustee node in terms of (α, β) as a

recommendation to avoid dependence and looping. After a task is completed, the TL can serve as a recommender toward the members in its team because it had gathered interaction experiences. For trust aggregation, each trustor aggregates trust evidence of its own (α, β) with a recommender's (α, β) toward the trustee node. Note that a recommender's (α, β) trust evidence is discounted based on the concept of *belief discounting* [13], such that the lesser the trustor node trusts the recommender, the more the recommendation is discounted. Because a bad node can perform bad-mouthing and ballot stuffing attacks, it can provide a bad recommendation with $\beta >> \alpha$ toward a good node and a good recommendation with $\alpha >> \beta$ toward a bad node, respectively. It can be shown that the beta reputation system is resilient to such attacks if the trustor node has a low trust value toward the bad recommender [13]. In this paper, we do not use trust to catch and evict bad nodes.

D. Task Model

Tasks arrive asynchronously and may start and end at different times. We denote the start time, end time and duration of task m by T_m^{start}, T_m^{end} and DT_m. Each task has unique properties:

- *Required node type* NT_m indicates the required functionality of nodes for executing task m. A node with a higher node type has a higher capability and, because of human involvement, also has social trust dimensions.
- *Required number of nodes* N_m refers to the number of nodes needed for execution of task m.
- *Importance* (I_m) refers to the impact of task failure on the mission with a higher value indicating higher importance.
- *Task execution Flow* (F_m) indicates the task structure (sequential, parallel or both) by which nodes coordinate with each other. For simplicity, we assume N_m nodes execute the task sequentially.
- *Task execution deadline* (T_m^{end}) specifies the deadline by which task m must complete, or it fails.

A task fails when the task execution time exceeds the task execution deadline, or when it suffers from opportunistic service attacks described in Section II.B which can result from malicious nodes purposely delaying task execution time to cause task failure. When a task fails, we assume that the TL can differentiate the guilty parties from lawful members and will apply a penalty to guilty parties by either blacklisting the guilty parties for the non-trust-based scheme, or applying a trust loss to the guilty parties in terms of $\Delta\beta$ for the trust-based scheme as discussed in Section II.C.

E. System Objectives

A CN aims to achieve the system goal in terms of three objectives: mission reliability (R), utilization variance (U), and delay to task completion (D). One can view minimizing utilization variance as maximizing load balance, and minimizing delay to task completion as maximizing QoS. These three objectives are defined below.

- **Mission Reliability (R):** This is the task reliability weighted by the task importance I_m, computed by:

$$R = \sum_{m \in L} R_m \frac{I_m}{\sum_{all} I_m} \quad (1)$$

L is the set of tasks in the mission. R_m, a binary number in $\{0, 1\}$, is the task reliability of task m. Following the Byzantine Failure model [12], we assume that a task fails when there are at least 1/3 bad nodes executing the task. Higher R is desirable. To achieve this objective, a TL should select highly trustworthy nodes.

- **Utilization Variance (U):** This measures the utilization variance of nodes and is defined by:

$$U = \frac{\sum_{i \in N}(|U_i - \tilde{U}|)}{|N|} \text{ where } U_i = \sum_{m \in L} U_{i,m} \quad (2)$$

N is the set of legitimate member nodes. $U_{i,m}$ is $DT_m / DT_{mission}$ if node i executes task m, and is zero otherwise, where DT_m is the task duration and $DT_{mission}$ is the mission duration. U_i is the overall utilization of node i. $\tilde{U}$ is the average utilization of all nodes. $|U_i - \tilde{U}|$ is the utilization variance of node i to the average. Lower U is desirable as it minimizes the utilization variance and achieves the load balance objective. To achieve this goal, a TL should select nodes with low utilization.

- **Delay to Task Completion (D):** This is the average delay to task completion over all tasks, defined by:

$$D = \frac{\sum_{m \in L} D_m}{|L|} \text{ where } D_m = T_m^{complete} - T_m^{start} \quad (3)$$

$T_m^{complete}$ is the actual completion time of task m. It is desirable to complete task m as early as possible with the drop-dead deadline of T_m^{end}. If task m is not completed by T_m^{end}, then task m fails and D_m is set to $DT_{mission}$. Lower D is desirable. To achieve this goal, a TL should select nodes with low execution time.

To formulate the MOO problem as a maximization problem, we first scale R, U and D into $\bar{R}$, $\bar{U}$ and $\bar{D}$ such that they each are in the range of [0, 1] and the higher the value to 1, the better the objective is achieved. Specifically, we scale R, U and D by [9]:

$$\bar{R} = \frac{R - R_{min}}{R_{max} - R_{min}}; \bar{U} = \frac{U_{max} - U}{U_{max} - U_{min}}; \bar{D} = \frac{D_{max} - D}{D_{max} - D_{min}} \quad (4)$$

Here R_{max} and R_{min}, U_{max} and U_{min}, and D_{max} and D_{min} are the maximum and minimum values of R, U, and D, respectively at the *mission* level. One can view $\bar{U}$ after scaling as "load balance" in the range of [0, 1], and $\bar{D}$ as "QoS" in the range of [0, 1]. Here we aim to solve the MOO problem by maximizing $\bar{R}$, $\bar{U}$ and $\bar{D}$, given node and task characteristics as input. We adopt the *weighted sum* form converting the MOO problem to a single-objective optimization problem. Specifically, we formulate the MOO problem as:

$$\text{Maximize } P_{MOO} = \omega_R \bar{R} + \omega_U \bar{U} + \omega_D \bar{D} \quad (5)$$

Here ω_R, ω_U and ω_D are the weights associated with $\bar{R}$, $\bar{U}$ and $\bar{D}$ with $\omega_R + \omega_U + \omega_D = 1$.

III. Trust-Based Dynamic Task Assignment Protocol

We have two layers of task assignment: by a CN to TLs and by each TL to nodes. For ease of exposition and due to space limitation, we assume that the CN-to-TL assignment has already been done based on prior trust profiles of the nodes. A TL is assigned to execute one task at a time, and a node can participate in only one task at a time, although it may participate in multiple tasks during its lifetime. TLs advertise tasks and free nodes respond as described next. Below we describe our heuristic-based dynamic task assignment protocol design based on auctioning with the objective to achieve MOO with a linear runtime complexity.

A. Advertisement of Task Specification

The task specification disseminated during the auction process includes a set of requirements for task execution specified by:

$$[ID_m, I_m, NT_m, F_m, (T_m^{start}, T_m^{end})] \quad (6)$$

ID_m is the identifier of task m.

B. Bidding a Task

When a node receives the task specification message by a TL, it makes a bidding decision on whether to bid the task or not. A node meeting the node type requirement NT_m is considered capable of handling the required work elements imposed by task m and will respond to the request with its node ID if it is free. To help the TL make an informed decision, node j sends its information to the TL as follows:

$$[ID_j, U_j, D_j, NT_j] \quad (7)$$

ID_j is the identifier of node j, U_j is the utilization of node j so far at the time of bidding, and D_j is the time required by node j to execute task m.

C. Member Selection

TLs implicitly seek to optimize the MOO function. However, to achieve run-time efficiency, they adopt heuristics to work independently of each other. The TL of task m ranks all bidding nodes (node j's) based on $\omega_R \bar{R_j} + \omega_U \bar{U_j} + \omega_D \bar{D_j}$ where $\bar{R_j}$, $\bar{U_j}$ and $\bar{D_j}$ are defined as:

$$\bar{R_j} = T_{TL,j}; \bar{U_j} = \frac{U_{max} - U_j}{U_{max} - U_{min}}; \bar{D_j} = \frac{D_{max} - D_j}{D_{max} - D_{min}} \quad (8)$$

Here a TL considers U_j (utilization of node j) instead of U (utilization variance of all nodes who have participated in at least one task) because it does not have information about the latter and picking nodes to minimize utilization variance essentially can be achieved by picking nodes with low U_j (equivalently with high $\bar{U_j}$ after scaling). Also a TL uses $\bar{R_j} = T_{TL,j}$ or its trust toward node j to predict task reliability if node j (a bidder) is selected for task execution. Top N_m nodes with the highest ranking scores are selected to execute task m. Here we note that the trust-based algorithm has a linear runtime complexity $O(N_B)$ where N_B is the number of bidders because it only needs to examine all bidders once and selects the top ranked N_m bidders.

D. Task Commitment by Nodes

A node may receive more than one offer from multiple TLs where tasks arrive concurrently. A TL sends out a winner notification with the full list of winners where the winners are potential members for the task. A node determines which task to join based on the expected payoff. For a good node, it selects the task of the highest importance to join because of high trust gain. For a malicious node, it does the same for high trust gain except when the Byzantine Failure condition is satisfied, i.e., at least 1/3 of the task members are malicious nodes in which case it selects the highest important task that is bound to fail to join to cause the greatest damage to the mission at the expense of trust loss.

TABLE I. Key parameters and default values

Parameter	Value	Parameter	Value
$(\|N\|, N_m)$	(20, 3), (60,9)	$\|L\|$	180 tasks
I_m	1-5	$\omega = (\omega_R, \omega_U, \omega_D)$	variable
D_j	U(1,5) min	P_b	10%-70%
$T_m^{end} - T_m^{start}$	$U(N_m, 5N_m)$min	n_{rec}	3
$\Delta\alpha$	I_m	$\Delta\beta: \Delta\alpha$	$0.1 - 10$

IV. Numerical Results and Analysis

In this section, we perform a comparative performance analysis of trust-based solutions against ideal solutions based on perfect knowledge, and non-trust-based solutions in terms of MOO performance with Matlab simulation.

Table I summarizes key parameter values used for this case study. Our example system considers $|N| = 20$, and 60 nodes for small, and large-sized problems, respectively. For both types of problems, there are $|L| = 180$ tasks arriving dynamically. For the small-sized problem, each task will need only $N_m = 3$ nodes, while for the large-sized problem, $N_m = 9$ nodes. A node's capability is specified by its node type, ranging from NT_1 to NT_4 equally divided among $|N|$ nodes. A node's service quality in terms of service time required (regardless of whether it is malicious) is specified by D_j which follows uniform distribution U(1, 5) min. Tasks with overlapping start and end times are grouped into a concurrent "chunk." Task importance is in the range from 1 to 5. We simulate a task's duration, that is, $T_m^{end} - T_m^{start}$ by $U(N_m, 5N_m)$min. This defines the task deadline by which a task must be completed, or it will fail. An effect of this is that nodes with a long execution time delay will not be selected

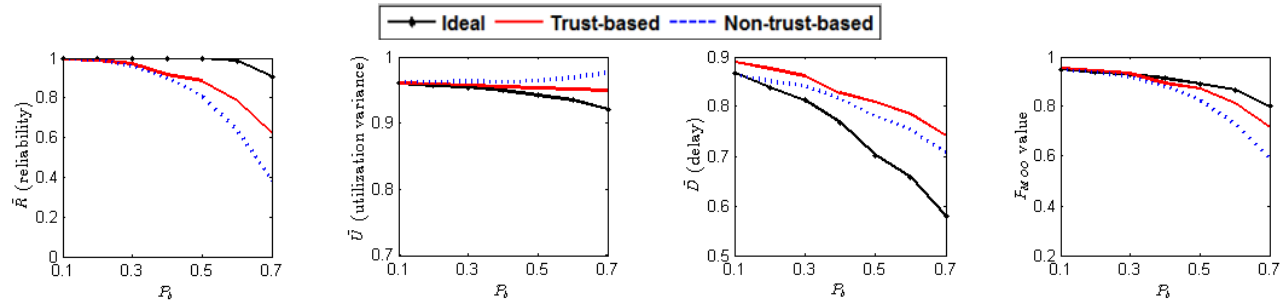

Fig. 1: Mission reliability ($\overline{R}$), Load Balance ($\overline{U}$), QoS ($\overline{D}$), and P_{MOO} vs. Bad Node Percentage P_b for a Small MOO Problem.

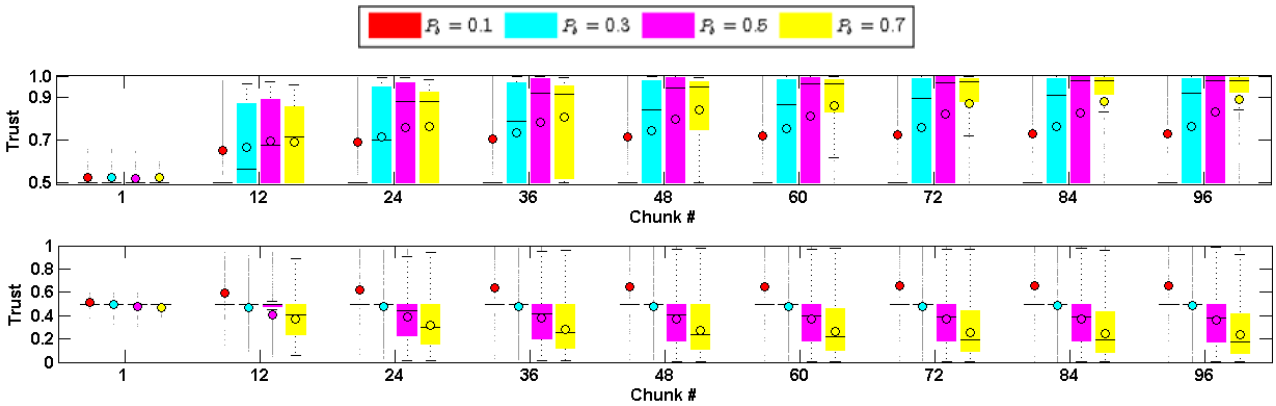

Fig. 2: Trust Values of Good Nodes (Top) and Bad Nodes (Bottom) over time in Boxplot Format.

for task execution by the TL of the task to prevent failure. A task's execution time is the sum of those of individual nodes selected for task execution since we consider sequential execution in this paper. The percentage of malicious nodes P_b ranges from 10% to 70% whose effect will be analyzed in this section. The weights associated with multiple objectives, i.e., $\overline{R}$, $\overline{U}$, and $\overline{D}$ in the MOO problem are $\omega = (\omega_R, \omega_U, \omega_D)$ which we vary to analyze its sensitivity.

A malicious node performs attacks as specified in the attack model in Section II. For the trust protocol, the number of recommenders n_{rec} is set to 3. The increment to positive evidence $\Delta\alpha$ is set to I_m, while the increment to negative evidence $\Delta\beta$ is set to I_m multiplied by the penalty severity parameter (i.e., $\Delta\beta$:$\Delta\alpha$) in the range of 0.1 to 10, with a larger number representing a more severe penalty to negative evidence. We will analyze the effect of severely punishing malicious behavior on MOO performance.

We consider two baseline algorithms against which our trust-based algorithm is compared in our performance analysis, ideal selection with perfect knowledge of node status vs. non-trust-based selection, as follows:

1. *Ideal selection*: The TL of task m ranks all bidding nodes in the same way as the trust-based algorithm described earlier except that it has perfect knowledge of node status, i.e., $\overline{R}_j = 1$ if node j is a good node, and $\overline{R}_j = 0$ if node j is malicious. The ideal solution is impossible to achieve; it is just used to predict the performance upper bound to the trust-based solution.
2. *Non-trust-based selection*: The TL of task m also ranks all bidding nodes in the same way as the trust-based algorithm except that $\overline{R}_j$ =0 if the bidding node is blacklisted; $\overline{R}_j$ =1 if the bidding node is not blacklisted and had participated in a successful task execution for which the TL was the task lead; and $\overline{R}_j$ =0.5 (no knowledge) otherwise. We assume intelligent behavior so that each TL can learn from experiences. If a TL experiences a task failure, it blacklists nodes participated in the task execution and excludes them from future NTA for which it is the TL. Top N_m ranked nodes are selected for executing task m.

Fig. 1 presents the solution quality in terms of the scaled mission reliability ($\overline{R}$), load balance ($\overline{U}$), QoS ($\overline{D}$), and P_{MOO} obtained by the trust-based solution against the ideal solution and the non-trust-based solution in the small-sized problem ($|N|$=20,$N_m = 3$) as a function of the percentage of malicious nodes in the range of 10% to 70% with $\Delta\beta{:}\,\Delta\alpha = 1{:}\,1$ and $\omega = (1/2{:}\,1/6{:}\,1/3)$ for a case in which reliability is more important than QoS and load balance. Note that $\overline{R}$, $\overline{U}$, $\overline{D}$ and P_{MOO} are all scaled in the range of [0, 1] with a higher number indicating a higher performance. Each result point indicates the average value of the metric based on 100 simulation runs, each of which has the same task arrival sequence with the D_j distribution randomized. Fig. 1 shows that the trust-based solution outperforms the non-trust-based solution and approaches the ideal solution, the effect of which is especially pronounced when P_b is high. There is an interesting tradeoff between the multiple objectives in terms

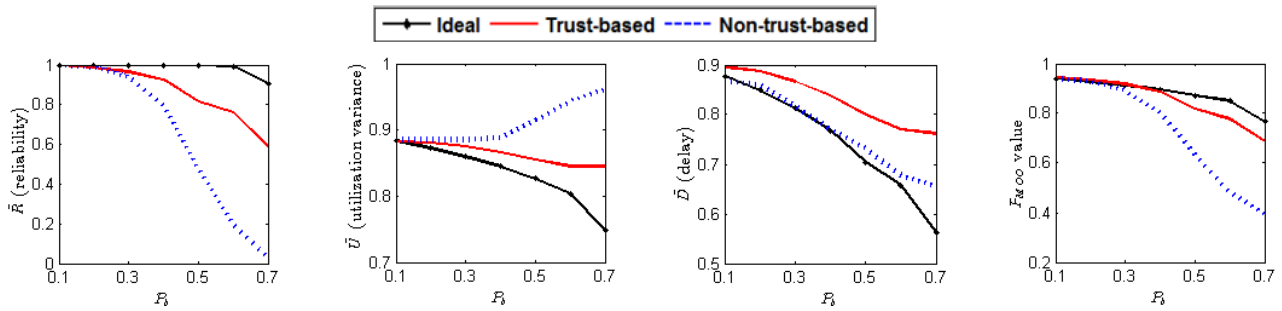

Fig. 3: Performance Comparison for a Large-sized MOO Problem.

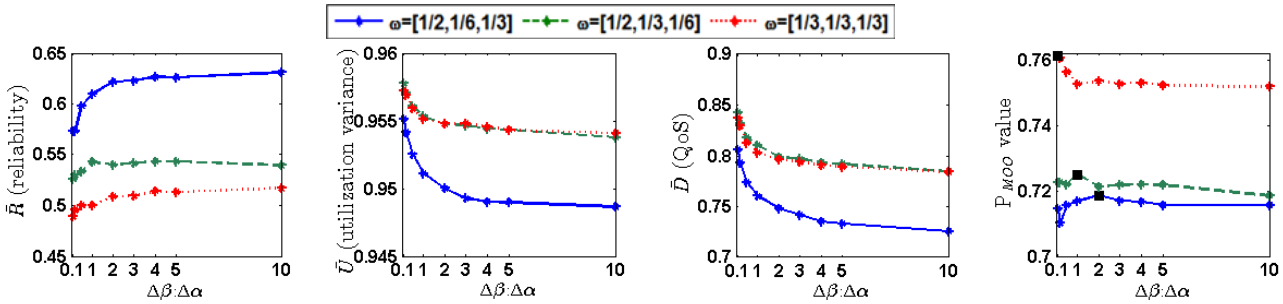

Fig. 4: Sensitivity Analysis of MOO with respect to $\Delta\beta{:}\,\Delta\alpha$ and $\omega = (\omega_R{:}\,\omega_U{:}\,\omega_D)$.

of $\bar{R}$, $\bar{U}$, and $\bar{D}$. The ideal solution attempts to maximize P_{MOO} in (5) by maximizing $\bar{R}$ because of its perfect knowledge of node reliability at the expense of $\bar{U}$ and $\bar{D}$. On the other hand, without having sufficient evidence to establish trust (at least initially), the trust-based solution attempts to maximize $\bar{D}$ without overly compromising $\bar{R}$ and $\bar{U}$. Finally, with only private blacklisting information kept by the TLs, the non-trust-based solution attempts to maximize $\bar{U}$ at the expense of $\bar{R}$ and $\bar{D}$. The ability for the TLs to differentiate good nodes from malicious nodes thus dictates how P_{MOO} in (5) is maximized.

Fig. 2 depicts the trust values of good nodes (top) and bad nodes (bottom) in *boxplot* format as a function of time (chunk #) in our trust protocol. A boxplot graphically depicts trust values through their quartiles without making any assumption about the distribution. In a boxplot, the bottom and top of a boxplot are the first and third quartiles, and the band inside the box is the second quartile (the median) with the ends of the whiskers showing the minimum and maximum of all of the trust values. We can see that for P_b = 10%, trust values are less dispersed, so the first and third quartiles are clustered into a thick dot. Further, the trust values of bad nodes are mostly above 0.5 because there are too few bad nodes in the system (2 out of 20) and the chance for them to be in the same task to perform opportunistic service attacks is low. In this case, bad nodes remain hidden and behave, with their trust values maintained above 0.5 to earn the trust reward. As P_b increases, the chance of performing opportunistic service attacks increases. As a result, the trust values of bad nodes are quickly updated to fall below 0.5 because of trust penalty. We also observe that trust convergence is achieved after 50 chunks (about 100 tasks). This is particularly the case when P_b is sufficiently high at which the medium trust value of bad nodes is sufficiently low and the medium trust value of good nodes is sufficiently high, so the system is able to differentiate good nodes from bad nodes for task execution. For example, the medium good node trust value is 0.75 and the medium bad node trust value is 0.35 when P_b is 50%. This explains why when P_b is 50% in Fig. 1, the trust-based solution outperforms the non-trust-based solution and approaches the ideal solution.

Fig. 3 compares $\bar{R}$, $\bar{U}$, $\bar{D}$, and P_{MOO} obtained by the trust-based solution against the ideal solution and the non-trust-based solution for the large-sized problem (($|N|$=60, N_m = 9), again as a function of the percentage of malicious nodes in the range of 10% to 70%. Fig. 3 is similar in trend as Fig. 1 with the trust-based scheme approaching the ideal scheme in the overall performance. Moreover the trust-based scheme significantly outperforms the non-trust-based scheme. We conclude that our heuristic trust-based solution with linear complexity $O(|N|)$ indeed can achieve solution efficiency without compromising solution optimality.

Fig. 4 tests the sensitivity of $\bar{R}$, $\bar{U}$, $\bar{D}$ and P_{MOO} obtained from our trust-based solution with respect to the ratio of the positive increment to the negative increment $\Delta\beta{:}\,\Delta\alpha$, and the weights associated with multiple objectives $\omega = (\omega_R{:}\,\omega_U{:}\,\omega_D)$ for the case in which P_b = 70% (picked to show area of interest) for the small sized problem. We see that in general $\bar{R}$ increases while $\bar{U}$ and $\bar{D}$ decrease as $\Delta\beta{:}\,\Delta\alpha$ increases because a larger ratio severely punishes bad nodes for performing attacks, making the bad nodes more distinguishable from good nodes. Selecting mostly good nodes for task execution, however, increases $\bar{R}$ but sacrifices

$\overline{U}$ because node selection tends to select mostly good nodes, and also sacrifices $\overline{D}$ because bad nodes with good service quality are not selected.

A striking observation is that the best $\Delta\beta{:}\,\Delta\alpha$ to maximize P_{MOO} is affected by the weights associated with multiple objectives, i.e., $\overline{R}$, $\overline{U}$ and $\overline{D}$ in the MOO problem. This is evident in the rightmost graph of Fig. 4 where we observe that the best $\Delta\beta{:}\,\Delta\alpha$ ratios are 0.1, 1, and 2, for $\omega = \left(\frac{1}{3}:\frac{1}{3}:\frac{1}{3}\right)$, $\left(\frac{1}{2}:\frac{1}{3}:\frac{1}{6}\right)$ and $\left(\frac{1}{2}:\frac{1}{6}:\frac{1}{3}\right)$, respectively, for maximizing P_{MOO} in (5). The reason is that a higher $\Delta\beta{:}\,\Delta\alpha$ increases $\overline{R}$ but sacrifices $\overline{U}$ and $\overline{D}$ as they are conflicting goals. Hence, under the equal weight scenario (the red line) when all objectives contribute equally, the best $\Delta\beta{:}\,\Delta\alpha$ value is small as so to best balance the gain of $\overline{R}$ vs. the loss of $\overline{U}$ and $\overline{D}$ for MOO.

Here we note that although the sensitivity analysis is demonstrated for the case in which P_b = 70%, the general behavior observed is true across. The only difference is the degree of sensitivity. For applicability, one can do static analysis as performed in this paper, collect the best trust protocol settings in terms of the best $\Delta\beta{:}\,\Delta\alpha$ ratio as a function ω and P_b, and then apply the best setting upon detection of dynamically changing environmental conditions at runtime to maximize protocol performance of the trust-based solution.

V. Conclusion

In this paper, we proposed a trust-based dynamic task assignment protocol for autonomous service-oriented MANETs where we are concerned with multi-objective optimization (MOO) for multiple objectives with conflicting goals. The results demonstrated that our trust-based solution has low complexity and yet can achieve performance comparable to that of the ideal solution with perfect knowledge of node reliability, and can significantly outperform the non-trust-based solution. We also provided insight of how MOO is achieved by the ideal, trust-based and non-trust-based solutions, and identified parameter settings under which the trust protocol performance in terms of MOO is optimized for the trust-based solution which can best balance multiple objectives with conflicting goals. The results obtained are useful for dynamic trust management to maximize application performance in terms of MOO.

In the future, we plan to refine our heuristic design for member bidding and selection strategies to further enhance MOO performance, possibly exploring game theory. We also plan to explore other forms of MOO formulation applicable to other autonomous service-oriented MANET scenarios.

Acknowledgments

This work is supported in part by the U. S. Army Research Laboratory and the U. S. Army Research Office under contract number W911NF-12-1-0445. This research was also partially supported by the Department of Defense (DoD) through the office of the Assistant Secretary of Defense for Research and Engineering (ASD (R&E)). The views and opinions of the author(s) do not reflect those of the DoD or ASD (R&E).

References

[1] A. Das, and M. M. Islam, "SecuredTrust: A dynamic trust computation model for secured communication in multiagent systems," *IEEE Trans. Dependable and Secure Computing*, vol. 9, no. 2, 2012, pp. 261-274.

[2] C. Dorn, et al., "Interaction mining and skill-dependent recommendations for multi-objective team composition," *Data Knowledge Engineering*, vol. 70, no. 10, 2011, pp. 866–891.

[3] L. Shen, L. Zhang, and D. Huang, "Trust-driven both-matched algorithm for grid task multi-objective scheduling," *2nd Conf. Information Science and Engineering*, 2010, pp. 1661-1664.

[4] M. Chang, J.H. Cho, I.R. Chen, K. Chan, and A. Swami, "Trust-based task assignment in military ad hoc networks," *17th Int'l Command and Control Research and Technology Symposium*, Fairfax, VA, June 2012.

[5] F. Bao, I.R. Chen, M. Chang, and J.H. Cho, "Trust-based intrusion detection in wireless sensor networks," *IEEE International Conference on Communications*, 2011, pp. 1-6.

[6] I.R. Chen, F. Bao, M. Chang, and J.H. Cho, "Dynamic Trust Management for Delay Tolerant Networks and Its Application to Secure Routing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 5, 2014, pp. 1200-1210.

[7] J.H. Cho, A. Swami, and I.R. Chen, "Modeling and analysis of trust management with trust chain optimization in mobile ad hoc networks," *Journal of Network and Computer Applications*, vol. 35, no. 3, pp. 1001-1012, May 2012.

[8] J.H. Cho, A. Swami, and I.R. Chen,, "Modeling and analysis of trust management for cognitive mission-driven group communication systems in mobile ad hoc networks," *IEEE International Conf. on Computational Science and Engineering*, pp. 641-650, 2009.

[9] L. Zeng, et al., "QoS-aware middleware for web services composition," *IEEE Transactions on Software Engineering*, vol. 30, no. 5, 2004, pp. 311-327.

[10] S. Kosta, A. Mei, and J. Stefa, "Small world in motion (SWIM): modeling communities in ad-hoc mobile networking," *7th IEEE Communications Society Conf. on Sensor, Mesh and Ad Hoc Communications and Networks*, Boston, MA, USA, June 2010.

[11] R. Mitchell and I. R. Chen, "Effect of intrusion detection and response on reliability of cyber physical systems," *IEEE Transactions on Reliability*, vol. 62, no. 1, 2013, pp. 199-210.

[12] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Trans. Program. Lang. Syst*., vol. 4, no. 3, pp. 382–401, 1982.

[13] A. Jøsang and R. Ismail, "The Beta reputation system," *Proc. 15th Bled Electronic Commerce Conf.*, 2002, pp. 1-14

[14] J.H. Cho, I.R. Chen, and P.G. Feng, "Effect of intrusion detection on reliability of mission-oriented mobile group systems in mobile ad hoc networks," *IEEE Transactions on Reliability*, vol. 59, no. 1, 2010, pp. 231-241.

[15] C.W. Hang and M.P. Singh, "Trustworthy service selection and composition," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 6, no. 1, 2011.

[16] Z. Malik and A. Bouguettaya, "RATEWeb: Reputation assessment for trust establishment among Web services," *The VLDB Journal.*, vol. 18, 2009, pp. 885-911.