Trust-Based Task Assignment with Multi-Objective Optimization in Service-Oriented Ad Hoc Networks

Yating Wang†, Ing-Ray Chen†, Jin-Hee Cho*, and Jeffrey J.P. Tsai‡

Abstract— **We propose and analyze a trust management protocol in service-oriented mobile ad hoc networks (MANETs) populated with service providers (SPs) and service requesters (SRs), and demonstrate the resiliency and convergence properties against bad-mouthing, ballot-stuffing, opportunistic service, and self-promotion attacks. To demonstrate the applicability, we consider a mission-driven service-oriented MANET that must handle dynamically arriving tasks to achieve multiple conflicting objectives. We devise a trust-based heuristic algorithm based on auctioning with local knowledge of node status to solve this node-to-task assignment problem with multi-objective optimization (MOO) requirements. Our trust-based heuristic algorithm has a polynomial runtime complexity, rather than an exponential runtime complexity as in existing work, thus allowing dynamic node-to-task assignment to be performed at runtime. It outperforms a non-trust-based counterpart using blacklisting techniques while performing close to the ideal solution quality with perfect knowledge of node status over a wide range of environmental conditions. We conduct extensive sensitivity analysis of the results with respect to key design parameters and alternative trust protocol designs. We also develop a table-lookup method to apply the best trust protocol parameter settings upon detection of dynamically changing environmental conditions to maximize MOO performance.**

Index Terms— **Trust management, multi-objective optimization, task assignment, service-oriented computing, mobile ad hoc networks, performance analysis.**

I. INTRODUCTION

With the proliferation of fairly powerful mobile devices and ubiquitous wireless technology, traditional mobile ad hoc networks (MANETs) now migrate into a new era of service-oriented MANETs wherein a node can provide and receive service from other nodes it encounters and interacts with. In this paper, we are concerned with autonomous service-oriented MANETs populated with service providers (SPs) and service requesters (SRs). One can view a service-oriented MANET as an instance of Internet of Things (IoT) systems with a wide range of mobile applications including smart city, smart tourism, smart car, smart environmental monitoring, and healthcare [7][19].

†*Yating Wang and Ing-Ray Chen are with the Department of Computer Science, Virginia Tech, Falls Church, VA 22043. E-mail: {yatingw, irchen}@vt.edu.*
**Jin-Hee Cho is with Computational and Information Sciences Directorate, U.S. Army Research Laboratory, Powder Mill Rd. Adelphi, MD 20783. E-mail: jinhee.cho@us.army.mil.*
‡*Jeffrey J.P. Tsai is with the Department of Bioinformatics and Biomedical Engineering, Asia University, Taichung, Taiwan 41354. Email: jjptsai@gmail.com.*

In this paper, we consider a mission-driven service-oriented MANET that must handle dynamically arriving tasks to achieve multiple system objectives. Each task requires an SR to be the task lead and to assemble a team among SPs to accomplish the task. For example, a service-oriented MANET may have the following 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 quality of service (QoS). We note that the objective of load balancing is in conflict with others since maximizing load balance may sacrifice task completion ratio and QoS. Nevertheless, all objectives are essential in order to maintain the "global welfare" of the service-oriented MANET. The problem we are interested in solving is dynamic node-to-task assignment (or *task assignment* for short) with multi-objective optimization (MOO) requirements, given that multiple objectives often have conflicting goals. This issue is further compounded by the fact that nodes may exhibit malicious behavior for "individual welfare" (explained later in Section III.B) and the information received is often erroneous, uncertain and incomplete in MANET environments [14] [15].

The literature is rich in solution techniques for solving task assignment MOO problems. Two major problems of existing solutions are (1) not considering the existence of malicious nodes acting for their own interest and colluding for individual welfare, and (2) solving the task assignment MOO problem in exponential time complexity, making it unsuitable for runtime deployment. In this paper, we develop trust-based solutions to mitigate these problems. In particular, we develop a trust-based algorithm to solve the task assignment MOO problem in polynomial time complexity, making it possible to perform dynamic node-to-task assignment at runtime. We demonstrate that our trust-based allocation protocol outperforms a non-trust-based counterpart using blacklisting techniques while performing close to the ideal solution quality with perfect knowledge of node status over a wide range of environmental conditions.

The contributions of this work are as follows:

1. We develop a trust management protocol specifically for autonomous service-oriented MANET applications and demonstrate the resiliency and convergence properties against bad-mouthing, ballot-stuffing, opportunistic service, and self-promotion attacks.
2. 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 polynomial runtime complexity, thus allowing dynamic node-to-task assignment to be performed at runtime. Our heuristic design is based on local knowledge of node status, so it can only produce a suboptimal solution. However, our heuristic design is able to achieve a solution quality approaching that of the ideal solution which has perfect knowledge of node status.
3. 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 node-to-task assignment.
4. 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.
5. We develop a table-lookup method to apply the best trust parameter settings upon detection of dynamically changing environmental conditions to maximize protocol performance.

The rest of this paper is organized as follows. Section II discusses related work, and contrasts our work with existing work. Section III describes our system model including the network model, attack/defense model, baseline trust protocol design, task model, and our MOO problem definition. Section IV proposes a polynomial runtime complexity trust-based heuristic algorithm to solve the MOO problem. Section V 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. We demonstrate that our trust-based scheme outperforms the non-trust-based counterpart using blacklisting techniques and performs close to the ideal solution quality. We conduct extensive sensitivity analysis of the results with respect to key design parameters and alternative trust protocol designs. Section VI discusses the applicability issues allowing one to make use of the analysis results obtained for maximizing protocol performance at runtime in response to dynamically changing environment conditions. Section VII concludes the paper and outlines some future research directions.

II. Related Work

Existing work on task assignment MOO can be categorized into two classes, depending on whether the work deals with system objectives for global welfare and/or individual objectives for individual welfare. Table I compares Class 1 and Class 2 solution techniques.

Class 1 represents the case in which there are multiple system objectives for global welfare, but there are no individual objectives. Class 2 represents the case in which there are multiple system objectives for global welfare but there are also individual objectives for individual welfare which may induce or hinder global welfare. Our work falls under Class 2 as we consider the presence of malicious nodes performing malicious attacks and colluding to monopoly service as the individual objectives for individual welfare.

Class 1 concerns solving a task assignment MOO problem to maximize *global welfare* but individual nodes do not have individual objectives. Balicki [5] studied a task assignment problem in a distributed environment based on a multi-objective quantum-based algorithm. The objectives are to maximize system reliability while minimizing workload and communication cost, all are global welfare. Chen et al. [12] solved a task assignment problem in a multi-robot environment consisting of heterogeneous mobile robots, given resource constraints associated with tasks. They proposed a heuristic leader-follower structure that identifies optimal solutions of the task allocation problems. Guo et al. [18] examined a task assignment problem using a particle swarm optimization technique that minimizes task execution time and cost for data transfer between processors in cloud computing environments. Xie and Qin [31] proposed an energy-efficient task assignment protocol based on the tradeoff between energy and delay to execute a task for collaborative networked embedded systems to minimize the length of schedules of task allocation and energy consumption. Shen et al. [26] develop a trust-based solution for task assignment in grid management with multiple system objectives including security, reliability, load balance, and throughput. Solutions fall under Class 1 assume no malicious entity in the system, which is not a valid assumption in a service-oriented MANET environment which very likely will be populated with malicious nodes acting for own interest and colluding for individual welfare. In our work, we develop a trust management protocol specifically for autonomous service-oriented MANET applications. We demonstrate the resiliency and convergence properties of our trust protocol design for service-oriented MANETs in the presence of malicious nodes performing bad-mouthing, ballot-stuffing, opportunistic service, and self-promotion attacks.

Table I: A Comparison of Class 1 and Class 2 Solution Techniques.

Reference	Individual objective	System objective	Solution technique	Problem to solve
Class 1				
Balicki [5]	NA	Minimize workload and cost while maximizing system reliability	Quantum-based evolutionary algorithm	Task assignment
Chen et al. [12]	NA	Maximize coalition utility while minimizing the number of robots involved in a task	Heuristic leader-follower based coalition algorithm	Task allocation
Guo et al. [18]	NA	Minimize task execution time while minimizing data transfer time	Particle swarm optimization	Task assignment
Shen et al. [26]	NA	Maximize security, reliability, load balance, and throughput	Trust-driven grid job scheduling	Resource allocation
Xie et al. [31]	NA	Maximize energy savings while minimizing the length of the schedule of task allocation	Energy-delay tradeoff algorithm	Task allocation
Class 2				
Anagnostopou-los et al. [4]	maximize use of individual skill set, minimize workload, and minimize communication overhead	Satisfy the team skill sets requirement, minimize communication overhead, and maximize load balance	Bi-criteria approximation algorithm	Team formation
Edalat et al. [16]	Minimize bid waiting time	Minimize energy consumption and delay	Reverse auction in cooperative game	Task allocation
Szabo et al. [27]	Minimize cost and delay	Minimize workflows, task completion time, and communication overhead	Evolutionary genetic algorithm	Task allocation
Tolmidis et al. [28]	Minimize energy consumption and task completion time while maximizing the degree of relevancy and task priority	Maximize the number of tasks completed while minimizing delay	Auction theory	Multi-robots task allocation
Wang et al. [30]	Minimize energy consumption while maximizing load balance	Maximize mission completion ratio based on the relevance of security and reliability for node-to-task assignment and minimize mission completion time	Trust-based max-min algorithm	Task scheduling

Class 2 concerns solving a task assignment MOO problem to maximize global welfare but nodes may have separate individual objectives for individual welfare. Anagnostopoulos et al. [4] explored a solution for a task assignment problem by matching a set of skilled people to each task. Their solution aims to minimize the communication overhead while balancing workloads by solving a single objective problem that considers multiple objectives. Edalat et al. [16] proposed an auction-based task assignment solution in wireless sensor networks with two global objectives: maximizing the overall network lifetime while satisfying application deadlines. An individual entity seeks to maximize its payoff by bidding on a task with low workload so as to consume less energy but have a high chance of being assigned to the task. Szabo and Kroeger [27] examined a task allocation problem in cloud computing using evolutionary genetic algorithms. This work has the system goals to minimize workflows, delay introduced by the task completion, and communication cost while each individual user wants to minimize cost and service delay. Tolmidis and Petrou [28] proposed an auction theoretic approach to solve a task allocation problem in multi-robot systems where each robot is able to perform several functions. An individual robot has the goals to minimize energy consumption and delay in task completion while maximizing the degree of relevancy and priority level to an assigned task. Similarly, the system aims to maximize the number of completed tasks and minimize delay introduced due to task assignment and completion. Wang et al. [30] proposed a trust-based task assignment technique for mobile ad hoc grid computing environments for maximizing mission completion ratio based on required levels of security and reliability in task assignment and minimizing delay to mission completion. The main drawback of the existing work cited above is that the worst case runtime complexity is exponential because the MOO problem to be solved is NP-complete [17], making it unsuitable to be deployed at runtime. Our work remedies this problem by devising trust-based heuristic solutions that incur only polynomial runtime complexity, and verifying that the performance of our trust-based solution approaches the ideal MOO performance with perfect knowledge of node status.

This work substantially extends [29] by (a) adding a new literature survey section comparing and contrasting our work with existing work in task assignment MOO problem

Table II: Acronyms and Symbols.

Acronym/Symbol	Meaning
MOO	Multi-objective optimization
SP	Service provider
SR	Service requester
CN	Commander node
TL	Task lead
NT	Node type
STO	Service trust only
SSTRT	Separating service trust from recommendation trust
ISTRT	Integrating service trust with recommendation trust
α, β	Amount of positive evidence, amount of negative evidence
$\mathcal{N}, \lvert\mathcal{N}\rvert$	Node set in the system, # of nodes in the system
$\mathcal{T}, \lvert\mathcal{T}\rvert$	Task set in the system, # of tasks in the system
$R,\ U,\ D$	Reliability, utilization variance, delay to task completion
$\bar{R},\ \bar{U},\ \bar{D}$	Scaled $R,\ U,\ D$
$\omega_R : \omega_U : \omega_D$	Weights associated with $\bar{R},\ \bar{U},\ \bar{D}$ for multi-objective optimization
P_{MOO}	$\omega_R \bar{R} + \omega_U \bar{U} + \omega_D \bar{D}$
$[I_m, NT_m, N_m, F_m, (T_m^{start}, T_m^{end})]$	Task *m*'s specification: importance, node type, number of nodes needed for task execution, task flow, start time, and end time (deadline)
N_B	Number of bidders in response to task *m*'s specification advertisement
$[U_j, D_j, NT_j]$	Node *j*'s specification: utilization, execution time, node type
$DT_m,\ DT_{mission}$	Task *m* execution duration, mission execution duration
$T_{i,j}$	Trust of node *i* has toward node *j*
n_{rec}	Maximum number of recommenders for trust propagation
P_b	Percentage of malicious nodes

solving; (b) adding a cost analysis of the runtime communication/memory overhead involved during trust protocol execution; (c) adding an extensive sensitivity analysis of the results with respect to key design parameters and alternative trust protocol designs, as well as providing insightful conclusions for designing the best trust protocol and trust-based task assignment algorithm in service-oriented MANETs, and (d) developing a novel method to apply the best trust parameter settings upon detection of dynamically changing environmental conditions to maximize MOO performance.

III. System Model

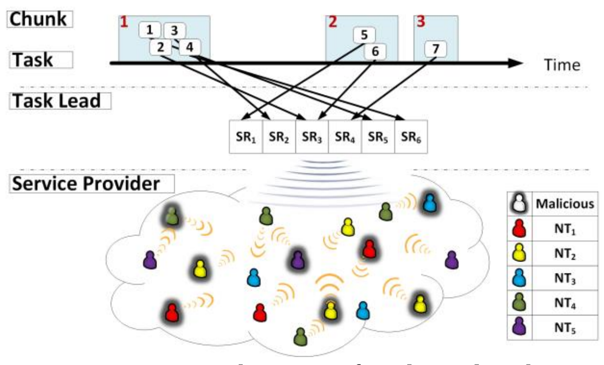


Fig. 1: System Architecture of Nodes and Tasks.

In this section, we discuss the system model including the network model, attack/defense model, trust protocol, task model, and system objectives. Table II summarizes the acronyms and symbols used in the paper.

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, *1, 2, ... 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 [6] [9] [10]. When mobile nodes are not involved in a task, they follow their routine-work mobility model. We use SWIM [21] in this paper to model social routine work mobility patterns of human operators. The implementation detail will be discussed later in Section V.

Fig. 1 illustrates the system architecture of nodes and tasks. We consider a mission-driven service-oriented MANET that must handle dynamically arriving tasks in a mission setting to achieve multiple system objectives. A commander node (CN) governs the mission team. Under the CN, multiple task leads (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 (acting as SRs) each recruit trustworthy SPs dynamically for executing the tasks assigned to them. Tasks that overlap in time are grouped into a chunk. In Fig. 1, tasks 1-4 are grouped into chunk 1 and tasks 5-6 are grouped into chunk 2. Tasks in the same chunk must compete with each other for services provided by qualified SPs as each SP can participate in only one task at a time. Here we make the assumption that the TLs selected by the CN are fully trusted. In military command and control environments, a hierarchical chain of command and control is a common network structure [2] wherein this assumption is justified. One example is a tactical convoy operation where a commander and multiple assistant commanders work together to control members to maximize communication and efficiency. Assistant commanders are selected a priori and the trust relationships between the commander and assistant commanders are assumed [1].

B. Attack/Defense Model

A node in the service-oriented MANET may exhibit the following malicious behaviors for individual welfare:

1. *Bad-mouthing attacks:* 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. Our trust protocol deals with bad-mouthing attacks by belief discounting [20] such that the lesser the trustor node trusts the recommender, the more the recommendation is discounted.
2. *Ballot-stuffing attacks*: 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. Our trust protocol deals with ballot-stuffing attacks also by belief discounting.
3. *Opportunistic service attacks*: a malicious node can provide good service to gain high reputation when it senses its reputation is low, and can provide bad service when it senses its reputation is high. Our trust protocol deals with opportunistic service attacks by severely punishing nodes that fail to provide the advertised service quality during task execution. Following the Byzantine Failure model [22], we assume that a task fails when at least 1/3 nodes providing bad service. Here we note that with good reputation, a malicious node can effectively collude with other bad nodes to perform bad-mouthing and ballot-stuffing attacks. Hence, a malicious node will provide good service (at its true service capability) most of the time in order to gain high reputation. However, a malicious node can opportunistically collude with other malicious nodes to fail a task, when it senses that there are enough bad nodes around (at least 1/3) at the expense of trust loss.
4. *Self-promotion attacks:* A malicious node can boost its service quality or lie about its utilization information so as to increase its chance of being selected as the SP. Our trust protocol deals with self-promotion attacks by severely punishing nodes that lie about their utilization or fail to provide the advertised service quality during task execution. In practice, self-promotion attacks can be easily detected, and as a result a malicious node would expose itself as vulnerable, resulting in a low reputation. This attack is less likely to be performed by a smart attacker.

Here, we note that our trust protocol design is to defend against inside attackers (who are legitimate members of the service-oriented MANET community), not outside attackers. Forgery of trust values can indeed happen during our protocol execution in the form of bad-mouthing and ballot-stuffing attacks by inside attackers. Opportunistic service attacks and self-promotion attacks are also performed by inside attackers. We assume that a MANET member key, as a symmetric key, is used for communications among members to prevent outside attackers. Public Key Infrastructure (PKI) can be used to uniquely identify each node and can be useful to defend against identity or Sybil attackers [24]. However, it is not required in our protocol execution. Defending against communication-level attacks such as Denial-of-Service, identity or Sybil attacks is outside the scope of this paper. We assume such behaviors are detected by intrusion detection mechanisms [3] [6] [13] [23] [25] [33].

A node is a legitimate node if it is a member of the mobile ad hoc group. Our trust-based protocol does not use blacklisting to identify "bad" legitimate nodes and prevent them from task bidding. Blacklisting is a non-trust-based baseline protocol against which our trust-based protocol is compared for comparative performance analysis. Our trust protocol simply assesses a trust value for each legitimate node but allows all legitimate nodes to bid for tasks. A legitimate node with a low trust value is automatically filtered out (i.e., not selected) during task allocation because they are not trustworthy to complete the task assignment. So in effect we filter out less trustworthy nodes, although we do not label these less trustworthy nodes as "bad" nodes. Note that to incorporate blacklisting into our trust-based protocol, we must define a minimum trust threshold below which a node is blacklisted as "bad" and therefore prevented from participating in task bidding. This approach requires the best minimum trust threshold value to be identified, which is an error-prone process. Our trust protocol does not classify nodes as "good" or "bad" but simply gives them each a trust value to assess their trustworthiness in task execution.

C. Trust Protocol and Cost Analysis

Our baseline trust protocol uses Beta (α, β) distribution [20] modeling a trust value in the range of [0, 1] as a random variable where α and β represent the amounts of positive service evidence and negative service evidence respectively, such that the estimated mean trust value of a node is $\alpha/(\alpha+\beta)$. A node uses the mean of Beta (α, β) distribution as the trust value it has toward another node. 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 $\Delta\beta \gg \Delta\alpha$. 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 with the initial trust value of 0.5.

All nodes can serve as recommenders based on self-observation experiences. To reduce message overhead especially for large MANETs, trust propagation is not by flooding. Instead, trust propagation is performed (via a recommendation message) only when a trustor node encounters a recommender node (not necessarily a TL). 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* [20], 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 $\beta \gg \alpha$) toward a good node and a good recommendation (with $\alpha \gg \beta$) 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 [20].

Below we do a cost analysis of the communication/memory/energy-consumption overhead involved. Based on our protocol design, a recommender propagates its (α, β) recommendations toward other $|\mathcal{N}|-2$ trustee nodes upon encountering a trustor node. The communication overhead per node in terms of bits/sec (from node i's perspective) is $\sum_{j=1}^{|\mathcal{N}|-1} \lambda_{ij}\,(|\mathcal{N}|-2)(b_I + b_o)$ where b_I is the information bits holding (α, β) of a trustee node, b_o is the encryption bits for secure communication, and λ_{ij} is the encountering rate of node i (the recommender) with node j (the trustor node) which can be derived by analyzing the encounter pattern, e.g., a power-law distribution, in a mobility model such as SWIM [21] and LSWTC [8]. Here we note that because trust propagation is encountered-based, trust convergence does not depend on the size of the network but depends on the encountering rate of node i (the recommender) with node j (the trustor node). Essentially upon encountering node j, node i exchanges trust information with node j in terms of the (α, β) value pairs of other nodes in the MANET community. In practice, the message overhead is lower because one can combine all required information into one message during transmission.

The energy-consumption overhead per node in terms of J/sec (from node i's perspective) is the sum of energy consumption rate for reception and energy consumption rate for transmission, i.e., $\sum_{j=1}^{|\mathcal{N}|-1} \lambda_{ij}\,(|\mathcal{N}|-2)(b_I + b_o)E_R + \sum_{j=1}^{|\mathcal{N}|-1} \lambda_{ij}\,(|\mathcal{N}|-2)(b_I + b_o)\left(1+\frac{1}{p}\right)E_T$ where E_R is the reception energy consumption rate (J/bit), E_T is the transmission energy consumption rate (J/bit), and $p = e^{-\sum_{j=1}^{|\mathcal{N}|-1} \lambda_{ij}(T_{RTS}+T_{CTS})}$ is the probability that other nodes within radio range are not transmitting during T_{RTS} + T_{CTS} and thus $1/p$ is the number of trials before node i clears the channel for transmission based on the RTS/CTS transmission protocol [11]. Here λ_{ij} is the encountering rate between node i and node j, accounting for the rate at which node j will transmit a recommendation packet to node i when they encounter during T_{RTS} + T_{CTS}, thus causing a collision and triggering a packet retransmission.

The memory overhead of our protocol is minimum. Each node only needs to allocate space to store its (α, β) value pairs toward other $|\mathcal{N}|-1$ nodes in the MANET community.

D. Task Model

Tasks arrive asynchronously and may start and complete at different times. Each task is characterized by the following unique properties:

- *Importance* (I_m) refers to the impact of task failure on the mission with a higher value indicating higher importance.
- *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*.
- *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 start time* (T_m^{start}) and *end time* (T_m^{end}).

A task fails when (a) the TL cannot recruit enough SPs to execute the task; (b) the task is not completed by the task end time (deadline), or (c) when it suffers from a Byzantine failure due to opportunistic service attacks as described in Section III.B which can result from malicious nodes purposely delaying task execution time to cause a task failure. For case (a), the TL will restart the bidding process until the deadline is reached before announcing failure. When a task fails due to (b) or (c), 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 III.C.

E. System Objectives

Without loss of generality, we consider three objectives, namely, mission reliability (*R*), utilization variance (*U*), and delay to task completion (*D*). We note that energy may also be an important objective for MANET applications which must run a long time without energy replenishment. In such environments, energy consumption minimization is another conflicting goal with reliability maximization and delay minimization. The trust-based algorithm design principle for maximizing task allocation MOO performance for the three objectives considered in this paper can still be applied if additional objectives (such as energy minimization) need to be considered.

One can view minimizing utilization variance as maximizing load balance, and minimizing delay to task completion as maximizing QoS. A TL tends to select high QoS nodes because they can reduce service delay. However, always selecting high QoS nodes will cause the utility of these high QoS nodes to be extremely high compared with other nodes, thus causing a high utilization variance. Similarly, a TL tends to select highly reliable nodes because they can increase task reliability. However, always selecting highly reliable nodes will cause the utility of these highly reliable nodes to be extremely high compared with other nodes, thus again causing a high utilization variance. Consequently, there is a tradeoff between load balancing, task delay, and task reliability. These three objectives are further defined below.

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

$$R = \sum_{m\in\mathcal{T}} R_m \frac{I_m}{\sum_{all} I_m} \qquad (1)$$

where $\mathcal{T}$ is the set of tasks in the mission, R_m is the reliability of task *m* such that $R_m = 1$ when task *m* succeeds; $R_m = 0$ otherwise, and I_m is task *m*'s importance. A task fails when (a) the TL cannot recruit enough SPs to execute the task; (b) the task is not completed by the task end time (deadline), or (c) when it suffers from a Byzantine failure due to opportunistic service attacks. 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:

$$U = \frac{\sum_{i\in\mathcal{N}}(|U_i-\widetilde{U}|)}{|\mathcal{N}|} \text{ where } U_i = \sum_{m\in\mathcal{T}} U_{i,m} \qquad (2)$$

where $\mathcal{N}$ is the set of legitimate member nodes, i.e., nodes that are recognized as legitimate members of the ad hoc group and can bid for tasks. $U_{i,m}$ 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*. $\widetilde{U}$ is the average utilization of all nodes. $|U_i - \widetilde{U}|$ 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:

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

where $\mathcal{T}$ is the set of tasks in the mission, $T_m^{complete}$ is the actual completion time of task *m*, and T_m^{start} is the start time of task *m*. It is desirable to complete task *m* as early as possible before the drop-dead end time T_m^{end}. If $T_m^{complete} > T_m^{end}$, 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 [32]:

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

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. These values define the maximum achievable and minimum tolerable solution quality, as determined by the user and system designer during the specification stage. 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:

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

```
# TL Execution in each task chunk:
1:  $\boldsymbol{if}$ a task is assigned $\boldsymbol{then}$
2:      $N_m$ ← number of nodes required for the task
3:      $taskExecuted \leftarrow false$
4:      $\boldsymbol{while}$ $not\ taskExecuted$ $\boldsymbol{do}$
5:          advertise task specification to all SPs
6:          $S \leftarrow$ all SPs that bid the task
7:          $\hat{S}$ ← SPs that are selected for task execution based on Eq. (8)
8:          send offers to $\hat{S}$
9:          $\hat{S}'$ ← SPs that commit to the task
10:         $N_m \leftarrow N_m - |\hat{S}'|$
11:         $\boldsymbol{if}$ $N_m == 0$ $\boldsymbol{then}$
12:             $taskExecution \leftarrow true$
13:             wait for task execution
14:             update and broadcast trust of participant SPs
15:         $\boldsymbol{end}$
16:     end
17: end

#  SP Execution in each task chunk:
18: $\boldsymbol{initialize}$ $committed \leftarrow false, P \leftarrow \emptyset, R \leftarrow \emptyset$
19: $\boldsymbol{if}$ task requests received $\boldsymbol{and}$ $not\ committed$ $\boldsymbol{then}$
20:     $P$ ← received task requests
22:     $NT_x$ ←SP's node type
21:     foreach $p\ in\ P$
22:         $NT_p$ ← required node type by $p$
23:         $\boldsymbol{if}$ $NT_x \geq NT_p$ $\boldsymbol{then}$
24:             send bidding information to the TL of $p$
25:         $\boldsymbol{end}$
26:     $\boldsymbol{end}$
27:     $R$ ← all tasks that make offers
28:     $\boldsymbol{if}$ $|R| \geq 1$ $\boldsymbol{then}$
29:         $r$ ← the task with the highest importance in $R$
31:         send task commitment to the TL of $r$
32:     $\boldsymbol{end}$
33:     wait for the TL decision
34:     $\boldsymbol{if}$ SP is selected $\boldsymbol{then}$
35:         $committed \leftarrow true$
36:         $\boldsymbol{if}$ SP is malicious $\boldsymbol{and}$ Byzantine failure condition meets $\boldsymbol{then}$
37:             fail $r$
38:         $\boldsymbol{else}$
39:             execute $r$
40:         $\boldsymbol{end}$
41:     $\boldsymbol{end}$
42: $\boldsymbol{end}$
```

Fig. 2: Actions taken by the SR and SPs during Auctioning for Task Assignment.

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

IV. Trust-Based Dynamic Task Assignment

We have two layers of task assignment: by a CN to TLs and by each TL to nodes. We assume that the TLs selected by the CN are fully trusted. A TL is assigned to execute one task at a time. 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 algorithm design based on auctioning with local knowledge of node status, with the objective to achieve MOO with a polynomial runtime complexity. Fig. 2 lists the pseudo code of the actions taken by the TL acting as the SR and the actions taken by well-behaved and malicious SPs during the execution of the trust-based task allocation algorithm.

A. Advertisement of Task Specification

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

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

where $ID_m, I_m, NT_m, F_m, T_m^{start}$ and T_m^{end} are task *m*'s identifier, importance, node type, number of nodes needed for task execution, task flow, start time, and end time (deadline), respectively.

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:

$$\left[ID_j, U_j, D_j, NT_j\right] \tag{7}$$

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

C. Member Selection

TLs implicitly seek to optimize the MOO function. However, to achieve run-time efficiency, they adopt heuristics with local knowledge of node status 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:

$$\bar{R}_j = T_{TL,j}; \bar{U}_j = \frac{U_{max} - U_j}{U_{max} - U_{min}}; \bar{D}_j = \frac{D_{max} - D_j}{D_{max} - D_{min}} \tag{8}$$

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 global knowledge about the latter and picking nodes to minimize utilization variance essentially can be achieved by picking bidding nodes with low U_j (equivalently with high $\bar{U}_j$ after scaling). A TL also does not know the reliability of node *j*. So the heuristics used is to relate the reliability of node *j* with the trust value the TL has toward node *j*, i.e., $\bar{R}_j = T_{TL,j}$ to predict the 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 polynomial runtime complexity $O(N_B log(N_m))$ where N_B is the number of bidders. This is so 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 that are selected by the TL to execute its task. A node determines which task to join based on the expected payoff. This depends on if the node is a good node or a bad node. For a good node, it selects the task of the highest importance to join so as obtain the highest trust gain. For a malicious node, it does the same in order to gain high trust except when the Byzantine failure [22] condition is satisfied, i.e., at least 1/3 of the task members are malicious nodes. In the latter case, the bad node 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. Here we note that ballot-stuffing will not allow a malicious node to maintain high trust while inflicting damage because (1) our protocol will severely punish the malicious node for failing the task by significantly reducing the malicious node's trust value; (2) our protocol will discount false recommendations from other malicious nodes performing ballot-stuffing attacks due to their low trust values.

Table III: Parameters Used in Simulation.

Parameter	Value	Parameter	Value
$(\lvert\mathcal{N}\rvert, N_m)$	(20, 3), (100,15), (1000, 150)	$\lvert\mathcal{T}\rvert$	180 tasks
I_m	1-5	$\omega_R : \omega_U : \omega_D$	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	$\Delta\beta$:$\Delta\alpha$	$0.1 - 10$
Slope of SWIM	1.45	Max pause time of SWIM	4 hours

V. NUMERICAL RESULTS AND ANALYSIS

In this section, we conduct 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. The mobility pattern of each node is modeled by the SWIM mobility pattern based on the C++ simulator in [21]. We also perform sensitivity analysis of the results with respect to key design parameters and alternative trust protocol designs.

Table III summarizes key parameter values used for this case study. Our example system considers $|\mathcal{N}|$ =20, 100, and 1000 nodes for small-sized, medium-sized, and large-sized problems, respectively. For the small-sized problem, each task will need only $N_m = 3$ nodes, for the medium-sized problem, $N_m = 15$ nodes, while for the large-sized problem, $N_m = 150$ nodes. For all problems, there are $|\mathcal{T}|$=180 tasks arriving dynamically. A node's capability is specified by its node type, ranging from 1 to 4 equally divided among $|\mathcal{N}|$ nodes. For simplicity, the required node type is 1 so all nodes are qualified for bidding. 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 task m's execution duration DT_m by $U(N_m, 5N_m)$ such that T_m^{end} is $T_m^{start} + DT_m$, defining the task end time (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 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.,$\bar{R}$, $\bar{U}$ and $\bar{D}$ in the MOO problem are specified by a weight ratio ω_R: ω_U: ω_D which we vary to analyze its sensitivity on MOO performance. The system designer must provide the weight ratio ω_R: ω_U: ω_D as input to reflect the relative importance of $\bar{R}$ vs. $\bar{U}$ and $\bar{D}$ as dictated by the application requirement.

We consider $|\mathcal{N}| = 20$, 100, and 1000 nodes for small-sized, medium-sized, and large-sized problems, respectively, moving according to the SWIM mobility model [21] modeling human social behaviors in an m×m=16×16 (4km×4km) operational region, with each region covering R=250m radio radius based on 802.11n. The operational area and the radio range remain the same for all problems. In SWIM, a node has a home location and a number of popular places. A node makes a move to one of the population places based on a prescribed pattern. The probability of a location being selected is higher if it is closer to the node's home location or if it has a higher *popularity* (visited by more nodes). Once a node has chosen its next destination, it moves towards the destination following a straight line and with a constant speed that equals the movement distance. When reaching the destination, the node pauses at the destination location for a period of time following a bounded power law distribution with the slope set to 1.45 (as in [21]) and the maximum pause time set to 4 hours. In addition, when a node is selected to execute a task, it moves to the location of the TL and follows the TL over the task execution duration. After completing a task, a node resumes its SWIM mobility pattern until it is selected again for executing another task. The movements and locations of $|\mathcal{N}|$ mobile nodes are simulated this way and then integrated with Matlab for evaluating trust-based, ideal, and non-trust-based solutions.

A malicious node performs attacks as specified in the attack model in Section III.B. 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 the performance analysis, namely, 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., $\bar{R}_j = 1$ if node j is a good node, and $\bar{R}_j = 0$ 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 $\bar{R}_j$ =0 if the bidding node is blacklisted; $\bar{R}_j$ =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 $\bar{R}_j$ =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 a future node-to-task assignment for which it is the TL. Top N_m ranked nodes are selected for executing task m.

A. *Comparative Performance Analysis*

Figs. 3, 4, and 5 present the solution quality in terms of the scaled mission reliability ($\bar{R}$), load balance ($\bar{U}$), QoS ($\bar{D}$), and P_{MOO} obtained by the trust-based solution against the ideal solution and the non-trust-based solution for the small-sized ($|\mathcal{N}| = 20$, $N_m = 3$), medium-sized ($|\mathcal{N}| = 100$, $N_m = 15$), and large-sized ($|\mathcal{N}| = 1000$, $N_m = 150$) problems, respectively, as a function of the percentage of malicious nodes in the range of 10% to 70%, with $\Delta\beta$: $\Delta\alpha = 1$: 1 and the weight ratio ω_R: ω_U: $\omega_D = \frac{1}{2}$: $\frac{1}{6}$: $\frac{1}{3}$ for a case in which reliability is more important than QoS and load balance.

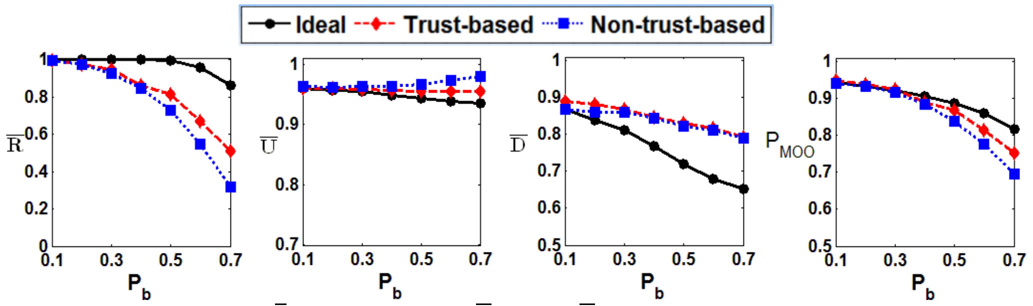


Fig. 3: Mission Reliability ($\bar{R}$), Load Balance ($\bar{U}$), QoS ($\bar{D}$), and P_{MOO} vs. Bad Node Percentage (P_b) for a Small-Sized MOO Problem.

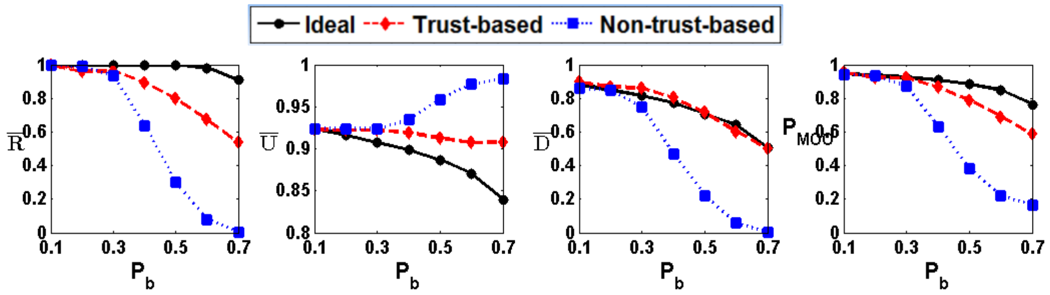


Fig. 4: Mission Reliability ($\bar{R}$), Load Balance ($\bar{U}$), QoS ($\bar{D}$), and P_{MOO} vs. Bad Node Percentage (P_b) for a Medium-Sized MOO Problem.

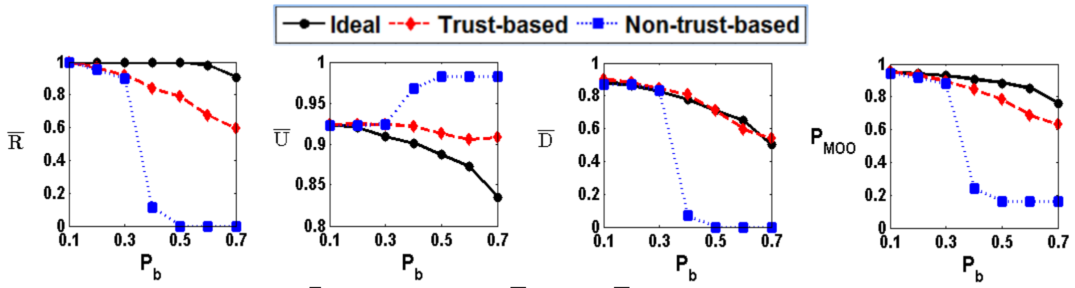


Fig. 5: Mission Reliability ($\bar{R}$), Load Balance ($\bar{U}$), QoS ($\bar{D}$), and P_{MOO} vs. Bad Node Percentage (P_b) for a Large-Sized MOO Problem.

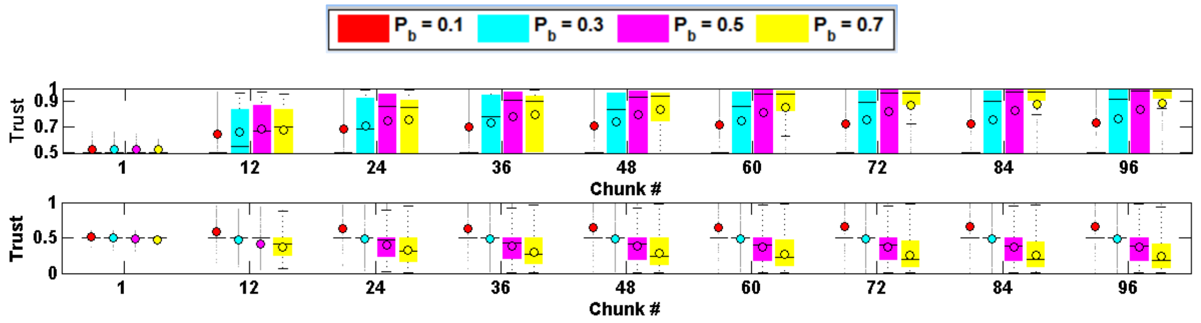


Fig. 6: Trust Value Distribution of Good Nodes (Top) and Bad Nodes (Bottom) in Boxplot Format.

Note that $\bar{R}$, $\bar{U}$, $\bar{D}$ 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. We observe that Figs. 3, 4, and 5 are similar in trend with the trust-based scheme approaching the ideal scheme in the overall performance. As the problem size increases, the performance gain of our trust-based scheme over the non-trust-based scheme is more pronounce because there are more qualified nodes to select from for task execution. Furthermore, the dominance is more manifested as the percentage of malicious nodes (P_b) increases.

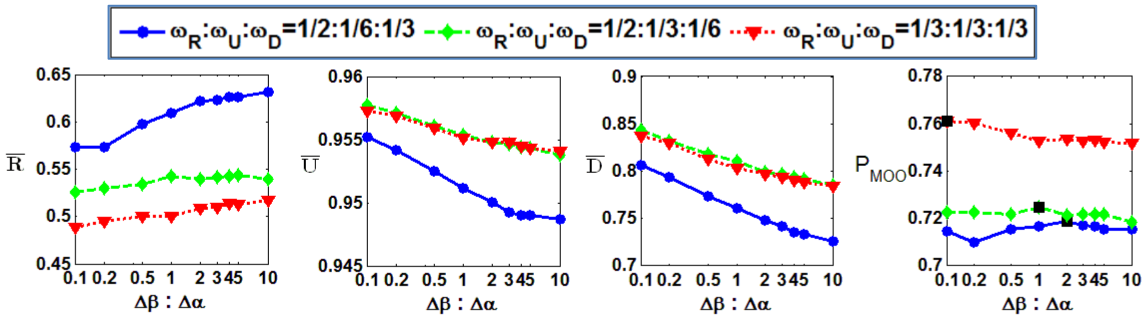


Fig. 7: Sensitivity Analysis of MOO with respect to $\Delta\beta: \Delta\alpha$ and $\omega_R: \omega_U: \omega_D$.

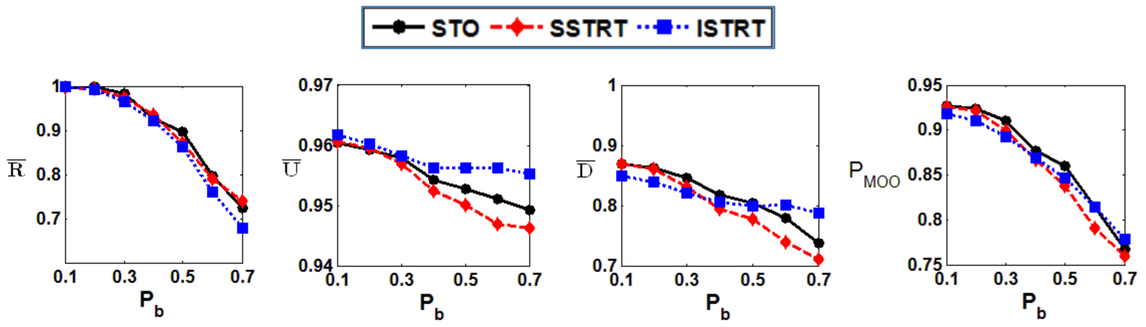
The ideal case has perfect knowledge over node status, i.e., it knows whether a node is malicious or not. Therefore, the ideal solution is perfect in maximizing mission reliability ($\bar{R}$). However, since mission reliability and load balance ($\bar{U}$) are conflicting objectives, the ideal solution can sacrifice load balance. This is indicated by the second subfigure in Figs. 3, 4 and 5 where we see both trust-based and non-trust-based solutions actually perform better than the ideal solution in load balance ($\bar{U}$). Lastly, the ideal solution, by selecting nodes to maximize mission reliability, does not necessarily guarantee maximizing QoS ($\bar{D}$). This is indicated by the third subfigure in Figs. 3, 4 and 5 where we see our trust-based solution actually performs comparably to or even better than the ideal solution in QoS ($\bar{D}$).

There is an interesting tradeoff between the multiple objectives in terms 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. We conclude that our heuristic trust-based solution with polynomial complexity indeed can achieve solution efficiency without compromising solution optimality.

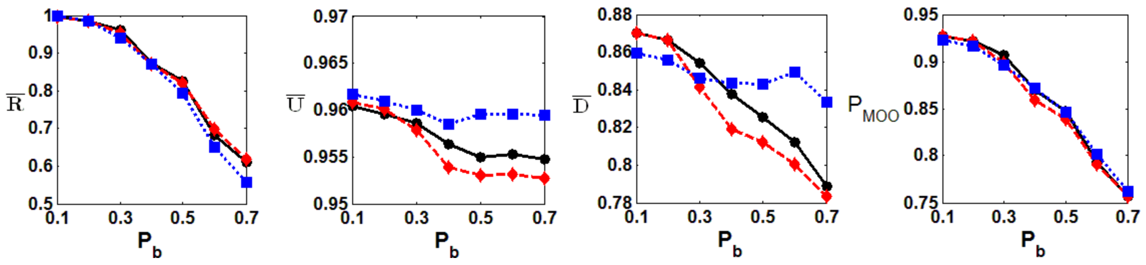
Fig. 6 depicts the trust value distribution of good nodes (top) and bad nodes (bottom) in boxplot format as a function of time (chunk #) in our trust protocol. This boxplot is the same for all problem sizes. A boxplot graphically depicts trust values through their quartiles without making any assumption about the probability 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. The trust value of a good node should be above 0.5 and approaching 1 (ground truth), while the trust value of a bad node should be blow 0.5 and approaching 0 (ground truth). 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 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 Figs. 3, 4, and 5, the trust-based solution outperforms the non-trust-based solution and approaches the ideal solution.

B. *Sensitivity Analysis of $\Delta\beta: \Delta\alpha$ and $\omega_R: \omega_U: \omega_D$*

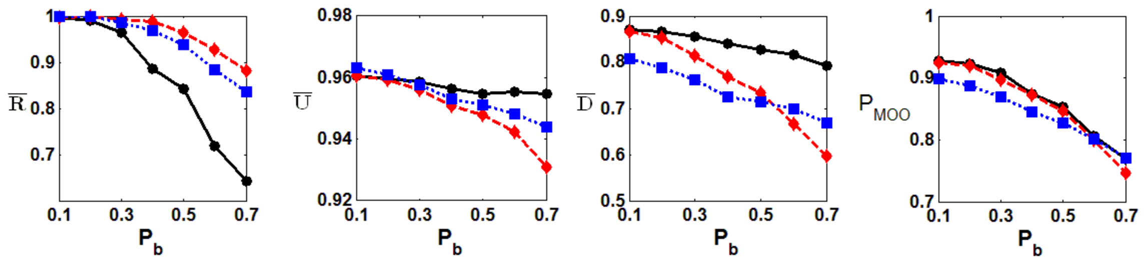
The observation that people hope to severely punish malicious behavior by having a large $\Delta\beta: \Delta\alpha$ ratio is true if mission reliability is the most important or is the sole objective especially for mission-critical applications. However, when there are multiple conflicting objectives such as mission reliability ($\bar{R}$), load balance ($\bar{U}$), and delay to task completion ($\bar{D}$) considered in this paper, this observation is not necessarily true. One contribution of this paper is that we identify the best $\Delta\beta: \Delta\alpha$ to use in order to maximize P_{MOO}, depending on the weights associated with multiple objectives, i.e., $\bar{R}$, $\bar{U}$ and $\bar{D}$ in the MOO problem.



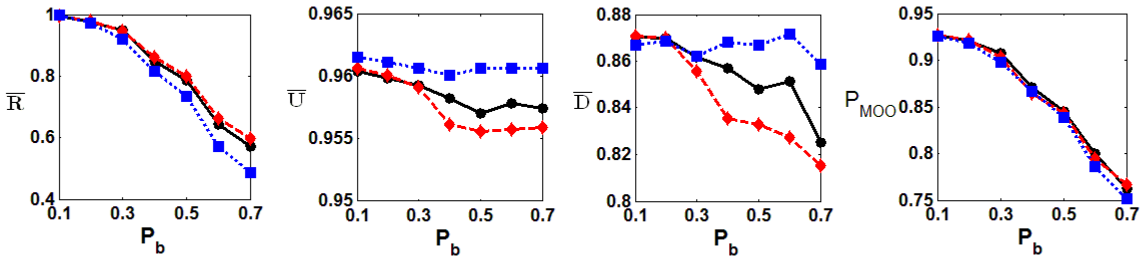
(a) $\omega_R : \omega_U : \omega_D = \frac{1}{3} : \frac{1}{3} : \frac{1}{3}$



(b) $\omega_R : \omega_U : \omega_D = \frac{1}{2} : \frac{1}{3} : \frac{1}{6}$



(c) $\omega_R : \omega_U : \omega_D = \frac{1}{2} : \frac{1}{6} : \frac{1}{3}$



(d) $\omega_R : \omega_U : \omega_D = \frac{1}{3} : \frac{1}{6} : \frac{1}{2}$

Fig. 8: Sensitivity Analysis of MOO with respect to Trust Protocol Design: STO vs. SSTRT and ISTRT.

Fig. 7 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 weight ratio of the weights associated with multiple objectives $\omega_R : \omega_U : \omega_D$ 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 $\bar{U}$ because node selection tends to select mostly good nodes, and also sacrifices $\bar{D}$ because bad nodes with good service quality are not selected.

We observe that the best $\Delta\beta : \Delta\alpha$ to maximize P_{MOO} is affected by the weights associated with multiple objectives, i.e., $\bar{R}$, $\bar{U}$ and $\bar{D}$ in the MOO problem. This is evident in the rightmost graph of Fig. 7 where we observe that the best $\Delta\beta : \Delta\alpha$ ratios are 0.1, 1, and 2 (labeled by black dots), for $\omega_R : \omega_U : \omega_D = \frac{1}{3} : \frac{1}{3} : \frac{1}{3}$, $\frac{1}{2} : \frac{1}{3} : \frac{1}{6}$, and $\frac{1}{2} : \frac{1}{6} : \frac{1}{3}$, respectively, for maximizing P_{MOO} in (5). The reason is that a higher $\Delta\beta : \Delta\alpha$

increases $\bar{R}$ but sacrifices $\bar{U}$ and $\bar{D}$ 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 $\bar{R}$ vs. the loss of $\bar{U}$ and $\bar{D}$ 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.

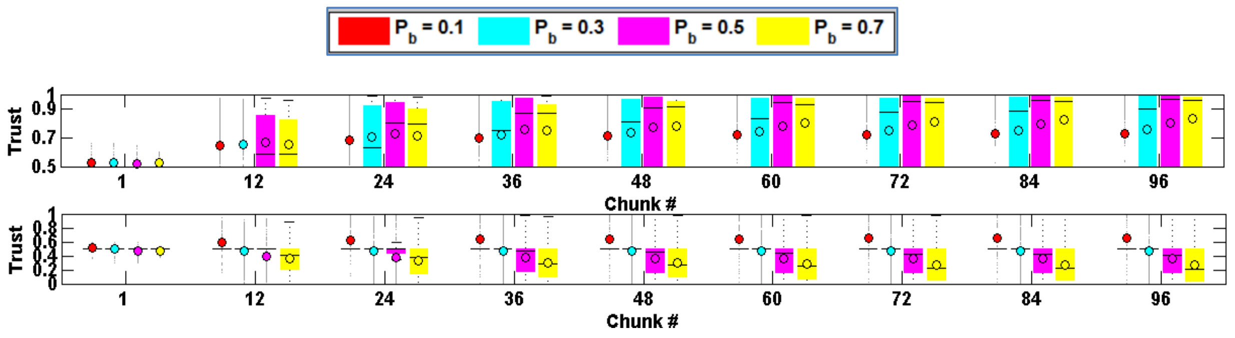
C. *Sensitivity Analysis of Trust Protocol Design*

In this section, we consider the effect of trust protocol design on trust-based MOO performance. Recall that the trust protocol design considered in Section III.C is based on a service trust value in the range of [0, 1] represented by α and β denoting the amount of positive evidence and negative evidence, respectively. We consider three trust protocol designs as follows:

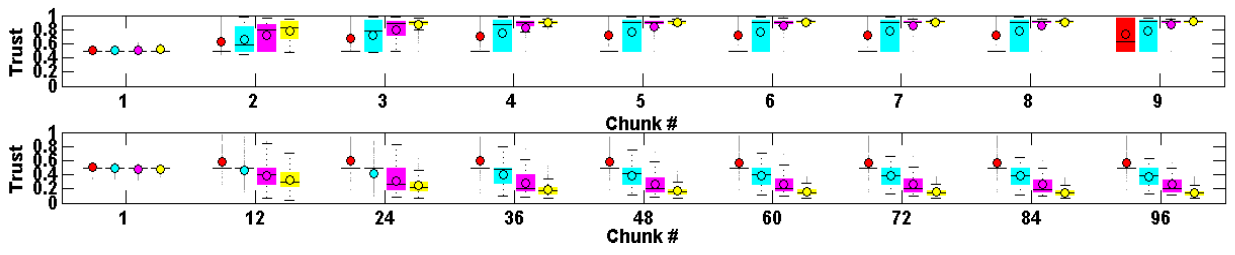
1. *Service Trust Only* (STO): this is the protocol described in Section III.C.
2. *Separating Service Trust from Recommendation Trust* (SSTRT): this protocol behaves the same as STO when updating the service trust. In addition, it maintains a separate trust value for rating a recommender. Specifically, there is a second pair of α and β denoting the amount of positive evidence and negative evidence respectively for rating a recommender. Upon receiving a recommendation from node *k* regarding node *j*, node *i* can compare its own service rating toward *j* with the recommendation received from *k* about *j*. If the difference deviates more than a percentage threshold (25% is considered in the paper), then node *i* views it as negative evidence against node *k* because of a possible bad-mouthing or ballot staffing attack by node *k*. Otherwise, node *i* views it as positive evidence for node *k*. When node *i* receives a recommendation from node *k*, node *i* uses the recommendation trust (instead of the service trust) it has toward *k* for trust merging. Here α and β for recommendation trust are set to 1 initially. The same $\Delta\beta{:}\Delta\alpha$ ratio applies.
3. *Integrating Service Trust with Recommendation Trust* (ISTRT): this protocol also considers positive/negative evidence of a recommender as SSTRT does. However, as STO, it only maintains a pair of α and β denoting the amount of positive evidence and negative evidence. Both recommendation quality evidence and service quality evidence collected are combined for updating α and β. Again α and β are set to 1 initially. The same $\Delta\beta{:}\Delta\alpha$ ratio applies.

The effect of trust protocol design on MOO performance is summarized in Figs. 8 and 9.

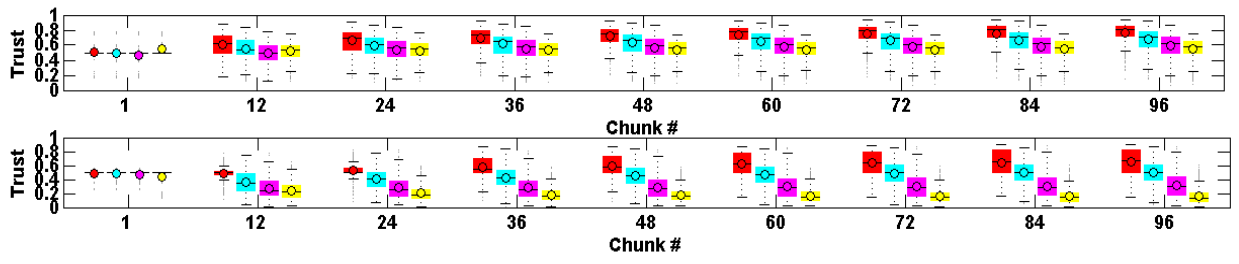
Figs. 8(a), 8(b), 8(c) and 8(d) compare $\bar{R}$, $\bar{U}$, $\bar{D}$ and P_{MOO} obtained by STO, SSTRT, and ISTRT as a function of the percentage of malicious nodes in the range of 10%-



(a) Trust Values of Good Nodes (top) and Bad Nodes (bottom) under STO.



(b) Trust Values of Good Nodes (top) and Bad Nodes (bottom) under SSTRT.



(c) Trust Values of Good Nodes (top) and Bad Nodes (bottom) under ISTRT.

Fig. 9: Trust Value Distribution of Good Nodes and Bad Nodes under STO vs. SSTRT and ISTRT.

70% for $\omega_R:\omega_U:\omega_D = \frac{1}{3}:\frac{1}{3}:\frac{1}{3}$, $\frac{1}{2}:\frac{1}{3}:\frac{1}{6}$, $\frac{1}{2}:\frac{1}{6}:\frac{1}{3}$, and $\frac{1}{3}:\frac{1}{6}:\frac{1}{2}$, respectively. While there is no clear winner among these three trust protocol designs, SSTRT appears to perform the best in terms of $\bar{R}$ over all weight ratio scenarios. Because of the tradeoff between the multiple goals between $\bar{R}$ vs. $\bar{U}$ and $\bar{D}$, maximizing $\bar{R}$ is often offset by sacrificing $\bar{U}$ and $\bar{D}$. In particular, we observe that under the weight ratio $\omega_R:\omega_U:\omega_D = \frac{1}{2}:\frac{1}{3}:\frac{1}{6}$, SSTRT outperforms STO and ISTRT in P_{MOO} since in this scenario SSTRT can best balance the gain of $\bar{R}$ against the combined loss of $\bar{U}$ and $\bar{D}$ compared with STO and ISTRT.

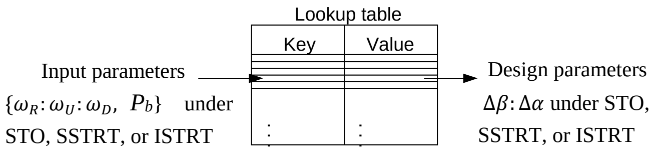
Figs. 9(a), 9(b), and 9(c) show the trust values of good nodes (top) and bad nodes (bottom) over time (chunk #) in boxplot format under STO, SSTRT and ISTRT, respectively, with the weight ratio $\omega_R:\omega_U:\omega_D = \frac{1}{2}:\frac{1}{3}:\frac{1}{6}$. We see clearly that SSTRT can best discern good nodes from bad nodes compared with STO and ISTRT. We attribute this to the ability of SSTRT to separate service trust from recommendation trust, which improves trust accuracy.

We conclude that the choice of the best trust protocol design is dictated by the relative importance of $\bar{R}$ vs. $\bar{U}$ and $\bar{D}$, i.e., it is highly sensitive to the weight ratio $\omega_R:\omega_U:\omega_D$ which in turn is dictated by the application requirement. The analysis performed here allows the system designer to choose the best trust protocol design for maximizing task assignment MOO performance.

VI. Applicability

The simulation results obtained reveal the best trust protocol settings in terms of the best $\Delta\beta:\Delta\alpha$ ratio to achieve MOO, given the relative importance of $\bar{R}$ vs. $\bar{U}$ and $\bar{D}$ (which determines $\omega_R:\omega_U:\omega_D$) and the hostility condition (which determines P_b) as input. More specifically, the simulation results obtained can be built into a lookup table, covering a conceivable range of $\omega_R:\omega_U:\omega_D$ and P_b values as input.

The lookup table as shown in Figure 10 would store key-value pairs where the "keys" are combinations of input parameter values, and the "values" are the best $\Delta\beta:\Delta\alpha$ ratio for maximizing MOO performance under the input parameter values. The *input parameters* on the left are input to the lookup table at runtime. The *design parameters* on the right are output as a result of a table lookup operation. Upon sensing the environment changes in terms of input parameter values, the system can perform a simple table lookup operation augmented with extrapolation or interpolation techniques to determine and apply the best $\Delta\beta:\Delta\alpha$ ratio in response to dynamically changing conditions. Depending on data granularity, a set of input parameter values may not directly map to a set of output parameter values. Extrapolation or interpolation techniques may be used to produce the matching output. The lookup time is *O(1)* and can be efficiently applied at runtime to determine the best trust protocol design as well as the best $\Delta\beta:\Delta\alpha$ ratio for maximizing task allocation MOO performance.



Lookup table
Key
Value
Input parameters
$\{\omega_R:\omega_U:\omega_D,\ P_b\}$ under STO, SSTRT, or ISTRT
Design parameters
$\Delta\beta:\Delta\alpha$ under STO, SSTRT, or ISTRT

Figure 10: Lookup Table Mechanism.

VII. Conclusion

In this paper, we proposed a trust-based dynamic task assignment algorithm for autonomous service-oriented MANETs where we are concerned with 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 the trust protocol parameter settings under which MOO performance is maximized for the trust-based solution which can best balance multiple objectives with conflicting goals. The results obtained are useful for dynamic trust protocol 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.

Acknowledgements

This work was 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 authors do not reflect those of the DoD or ASD (R&E).

References

[1] Air Land Sea Application Center, *Tactical Convoy Ops: Multi-Service Tactics, Techniques, and Procedures for Tactical Convoy Operations*, Langley AFB, VA, USA, March 2005.

[2] D.S. Alberts and R.E. Hayes, *Power to the Edge: Command and Control in the Information Age.* CCRP Publication Series, 2003.

[3] H. Al-Hamadi and I.R. Chen, "Adaptive Network Management for Countering Smart Attack and Selective Capture in Wireless Sensor Networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 3, 2015, pp. 451-466.

[4] A. Anagnostopoulos, L. Becchetti, C. Castillo, A. Gionis, and S. Leonardi, "Online Team Formation in Social Networks," in *21st Int.*

Conf. World Wide Web, 2012, pp. 839-848.

[5] J. Balicki, "An Adaptive Quantum-Based Multiobjective Evolutionary Algorithm," in *World Scientific and Engineering Academy and Society (WSEAS) 13th Int. Conf. Computers*, 2009, pp. 417-422.

[6] F. Bao, I.R. Chen, and J. Guo, "Scalable, Adaptive and Survivable Trust Management for Community of Interest Based Internet of Things Systems," in *11th IEEE International Symposium on Autonomous Decentralized Systems*, 2013, pp. 1-7.

[7] E. Borgia, "The Internet of Things Vision: Key Features, Applications and Open Issues," *Computer Communications*, vol. 54, pp. 1-31, December 2014.

[8] M.R. Brust, C. Ribeiro, D. Turgut, and S. Rothkugel, "LSWTC: A Local Small-World Topology Control Algorithm for Backbone-Assisted Mobile Ad hoc Networks," in *IEEE Conf. Local Computer Networks*, 2010, pp. 144-151.

[9] 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 Trans. Parallel and Distributed Systems*, vol. 25, no. 5, 2014, pp. 1200-1210.

[10] I.R. Chen, J. Guo, and F. Bao, "Trust Management for SOA-based IoT and Its Application to Service Composition," *IEEE Transactions on Services Computing*, vol. 9, no. 3, 2016, pp. 482-495.

[11] I.R. Chen and Y. Wang, "Reliability Analysis of Wireless Sensor Networks with Distributed Code Attestation," *IEEE Communications Letters*, vol. 16, no. 10, 2012, pp. 1640-1643.

[12] J. Chen, X. Yan, H. Chen, and D. Sun, "Resource Constrained Multirobot Task Allocation with A Leader-Follower Coalition Method," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2010, pp. 5093 - 5098.

[13] 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 Trans. Reliability*, vol. 59, no. 1, 2010, pp. 231-241.

[14] 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," in *Int. Conf. Computational Science and Engineering*, 2009, pp. 641-650.

[15] 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, 2012, pp. 1001-1012.

[16] E. Edalat, C. Than, and W. Xiao, "An Auction-Based Strategy for Distributed Task Allocation in Wireless Sensor Networks," *Computer Communications*, vol. 35, no. 8, 2012, pp. 916-928.

[17] M. R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.*: W.H. Freeman & Co., 1979.

[18] L. Guo, G. Shao, and S. Zhao, "Multi-Objective Task Assignment in Cloud Computing by Particle Swarm Optimization," in *8th Int. Conf. Wireless Communications, Networking and Mobile Computing*, 2012, pp. 1- 4.

[19] J. Guo, I.R. Chen, J.J.P. Tsai, and H. Al-Hamadi, "Trust-based IoT Participatory Sensing for Hazard Detection and Response," *1st International Workshop for IoT Systems Provisioning & Management in Cloud Computing*, Banff, Canada, Oct. 2016, pp. 1-6.

[20] A. Jøsang and R. Ismail, "The Beta Reputation System," in *15th Bled Electronic Commerce Conf.*, 2002, pp. 1-14.

[21] S. Kosta, A. Mei, and J. Stefa, "Large-Scale Synthetic Social Mobile Networks with SWIM," *IEEE Trans. Mobile Computing*, vol. 13, no. 1, 2014, pp. 116-129.

[22] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, 1982, pp. 382-401.

[23] R. Mitchell and I.R. Chen, "Behavior Rule Specification-based Intrusion Detection for Safety Critical Medical Cyber Physical Systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 1, 2015, pp. 16-30.

[24] R. Mitchell and I.R. Chen, "A Survey of Intrusion Detection in Wireless Network Applications," *Computer Communications*, vol. 42, 2014, pp. 1-23.

[25] R. Mitchell and I.R. Chen, "Adaptive Intrusion Detection of Malicious Unmanned Air Vehicles using Behavior Rule Specifications," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 44, no. 5, 2014, pp. 593-604.

[26] L. Shen, L. Zhang, and D. Huang, "Trust-Driven Both-Matched Algorithm for Grid Task Multi-Objective Scheduling," in *2nd Int. Conf. Information Science and Engineering*, 2010, pp. 1661–1664.

[27] C. Szabo and T. Kroeger, "Evolving multi-objective strategies for task allocation of scientific workflows on public clouds," in *IEEE World Congress on Computational Intelligence*, 2012, pp. 1-8.

[28] A. Tolmidis and L. Petron, "Multi-Objective Optimization for Dynamic Task Allocation in a Multi-Robot System," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 5-6, 2013, pp. 1458-1468.

[29] Y. Wang, I.R. Chen, J.H. Cho, K.S. Chan, and A. Swami, "Trust-Based Service Composition and Binding for Tactical Networks with Multiple Objectives," in *32th IEEE Military Commun. Conf.*, 2013.

[30] H. Wang, C. Li, C. Yan, Q. Li, and J. Li, "Ad Hoc Grid Task Scheduling Algorithm Considering Trust-Demand," in *2nd Int. Conf. Future Computer and Communication*, 2010, pp. 108-113.

[31] T. Xie and X. Qin, "An Energy-Delay Tunable Task Allocation Strategy for Collaborative Applications in Network Embedded Systems," *IEEE Trans. Comput.* , vol. 57, no. 3, 2008, pp. 329-343.

[32] L. Zeng et al., "QoS-Aware Middleware for Web Services Composition," *IEEE Trans. Software Engineering*, vol. 30, no. 5, 2004, pp. 311-327.

[33] H. Zhu, S. Du, Z. Gao, M. Dong, and Z. Cao, "A Probabilistic Misbehavior Detection Scheme Towards Efficient Trust Establishment in Delay-Tolerant Networks," *IEEE Trans. Parallel and Distributed Syst.*, vol. 25, no. 1, 2014, pp. 22-32.

AUTHOR BIOGRAPHIES

Yating Wang received her Bachelor degree from Hubei University of Technology, Wuhan, China in 2007. She received her PhD degree in Computer Science from Virginia Tech in 2016. She is currently a software engineer at Google. Her research interests include security, computer networks, wireless networks, mobile computing, trust management, and reliability and performance analysis.

Ing-Ray Chen received the BS degree from the National Taiwan University, and the MS and PhD degrees in computer science from the University of Houston. He is a professor in the Department of Computer Science at Virginia Tech. His research interests include mobile computing, wireless systems, security, trust management, and reliability and performance analysis. Dr. Chen currently serves as an editor for *IEEE Communications Letters, IEEE Transactions on Network and Service Management, The Computer Journal*, and *Security and Network Communications*. He is a recipient of the IEEE Communications Society William R. Bennett Prize in the field of Communications Networking.

Jin-Hee Cho received the BA from the Ewha Womans University, Seoul, Korea and the MS and PhD degrees in computer science from the Virginia Tech. She is currently a computer scientist at the U.S. Army Research Laboratory, Adelphi, Maryland. Her research interests include wireless mobile networks, mobile ad hoc networks, sensor networks, secure group communications, group key management, network security, intrusion detection, performance analysis, trust management, cognitive networks, social networks, dynamic networks, and resource allocation. She received the best paper awards in IEEE TrustCom09 and BRIMS13. She is a recipient of the IEEE Communications Society William R. Bennett Prize in the field of Communications Networking, and a recipient of the Presidential Early Career Awards for Scientists and Engineers. She is a senior member of the IEEE and a member of ACM.

Jeffrey J.P. Tsai received a Ph.D. degree in Computer Science from the Northwestern University, Evanston, Illinois. He is the President of Asia University, Taiwan, and a professor in the Department of Bioinformatics and Biomedical Engineering at Asia University. Dr. Tsai was a Professor of Computer Science at the University of Illinois, Chicago. His current research interests include bioinformatics, ubiquitous computing, services computing, intrusion detection, knowledge-based software engineering, formal modeling and verification, distributed real-time systems, and intelligent agents. He was an Associate Editor of the *IEEE Transactions on Knowledge and Data Engineering* and is currently an Associate Editor of the *IEEE Transactions on Services Computing*. He is currently the Co-Editor-in-Chief of the *International Journal on Artificial Intelligence Tools* and *Book Series on Health Informatics*. Dr. Tsai received an IEEE Technical Achievement Award and an IEEE Meritorious Service Award from the IEEE Computer Society. He is a Fellow of the AAAS, IEEE, and SDPS.