

# A Distributed Reinforcement Learning Approach to Mission Survivability in Tactical MANETs

[Extended Abstract]

Marco Carvalho, Ph.D.  
Institute for Human and Machine Cognition  
40 S. Alcaniz St.  
Pensacola, FL  
mcarvalho@ihmc.us

## ABSTRACT

In this paper we present an ongoing research to develop a distributed reinforcement learning approach for mission survivability that combines two basic strategies for mission resilience: a) mission decomposition and distribution with replication of critical components, and b) differential task allocation based on estimated level of threat. Level of threat is estimated from a locally perceived attack, or the possibility of an attack, based on threat information that is shared between similar nodes.

## Categories and Subject Descriptors

I.2.6 [Computing Methodologies]: Artificial Intelligence—*Learning*

## Keywords

Mission Survivability, Reinforcement Learning

## 1. INTRODUCTION

Tactical mobile ad hoc networks are of great importance for complex environments such as those often encountered in disaster relief operations and military missions. Tactical MANETs enable the communication and computational infrastructures for mission-critical applications and, in addition to conventional security mechanisms, they should also provide intrinsic support for the survivability of the mission.

While there is an expected correlation between the survivability of the system and the survivability of the mission, there is a conceptual difference between the two. To protect a mission means to ensure that its execution is carried out completely and successfully, with no concerns on the state of the underlying computational and communications infrastructure.

It is important to note, however, that an underlying system degradation or compromise will eventually lead to mis-

sion failure - or to the inability to support a mission. Thus mission survivability includes system survivability at an equivalent level of importance, but with lower priority than mission support. From this perspective, the notion of mission survivability requires two core capabilities in face of an attack or local failure:

- A short term adaptive response, re-allocating services and resources to ensure mission continuity while the attacked area is being addressed.
- A parallel defense mechanism that seeks to identify and isolate the attack, while learning attack patterns to estimate the level of vulnerability of other nodes.

Conventional strategies for survivable systems tend to rely on redundancy and service replication, often from an application perspective with little (if any) feedback from the dynamics of the network, ongoing attacks, or changes in system threats.

In this class of strategies, the assumption of equal probability of failure between nodes leads to a likelihood of survivability of a critical component that is directly proportional to the number of replicas. However, in for most scenarios the likelihood of failure and vulnerability is far from uniform across the system and, in fact, may change as a function of the mission being executed.

In this paper we introduce a new approach to the problem, leveraging from mission decomposition and adaptive task replication and distribution. Through a fully distributed reinforcement learning [5] (RL) algorithm, mission tasks will be distributed across the network to improve survivability requirements and global cost constraints, jointly addressing computational and communication costs.

## 2. MISSION DECOMPOSITION

In the context of this work a mission is represented as a set of workflows, each of them composed as an ordered set of tasks. Our proposed algorithm and representation can be easily generalized to accommodate more complex workflows with multiple parallel paths, having for instance the critical path as reference for task prioritization. However, for simplicity we will consider the special case where missions are composed of strict sequences of workflows, which are themselves composed as strict sequences of tasks, as illustrated in figure 2. A task consists on some atomic action that can be taken at any node, at a given cost. The set of possible tasks is fixed and well known to all nodes in the network, but not

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CSIIRW '09, April 13-15 Oak Ridge, Tennessee, TN  
Copyright 2009 ACM 978-1-60558-518-5 ...\$5.00.

all nodes are necessarily capable to perform all tasks and the cost for executing each task is node dependent.

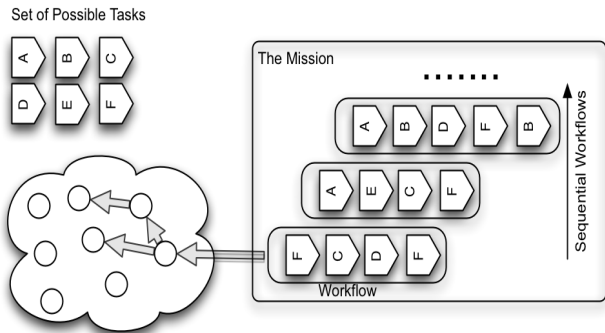


Figure 1: Mission represented as a set of workflows.

The total cost for processing a workflow is the cumulative cost of processing each task in sequence plus the cost of transmitting the workflow from one node to another.

There is a maximum number of times that a workflow can be transmitted while in execution, this TTL-like parameter prevents infinite loops and reduces the complexity of the problem.

The process of mission decomposition an allocation essentially consists on breaking each workflow into task-sets that will then be allocated to different nodes in the network. In a resource allocation problem, the goal is to determine an optimal mapping between tasks and resources in the network for the set of workflows.

Given the dynamic nature of the network, pre-defined and fixed allocation strategies are generally unfeasible. Solutions must be able to dynamically adapt to the overall constraints of the system and mission demands. Similar approaches have been proposed for routing [2], and a more comprehensive variation of this problem, refereed to as *in-stream data processing*, was introduced in [3].

In this paper, our focus will be on a robust algorithm for workflow allocation. The goal is to support a more complex objective function that takes into account the reliability of a node for task execution (proportional to an estimate of its likelihood of failure, or susceptibility to attacks), and redundant task allocation to improve the robustness of the workflow execution.

### 3. MISSION SURVIVABILITY

As illustrated in 2, a mission is defined as a large set of workflows to be executed one at a time.

We will define survivability of a mission as the ratio of successfully completed workflows. We will disregard the issue of differential priority between workflows for now, and simplify the discussion by assuming (without loss of generality) that all workflows in the mission have equal value and the same relevance to the mission.

To better constrain the problem, we will define our objective function as a ratio between the mission survivability and the overall execution costs.

Our goal is to approximately maximize that function using reinforcement learning strategies, while taking node vulnerability and threat expectations into account.

### 4. DISTRIBUTED TASK ALLOCATION

In addition to workflow decomposition, the proposed allocation would also take into account a criteria for task replication and positioning that includes the likelihood of failure of each node. Rather than focusing on a game theoretical approach to the resource coordination problem, our formulation is based on reinforcement learning strategies applied against a changing environment. Essentially, each node that receives a workflow makes a local decision about a) performing the next workflow task, and b) who to forward the workflow to for subsequent processing (if not completed).

Figure 2 illustrates a complete processing of a workflow through a small network. In the left side of the image, the flow is processed serially, one task at a time. In the right side, there is a duplication of the flow and parallel task processing on nodes 2 and 3. Since all workflow tasks must be processed in order, the duplication of workflows for parallel processing is done only for redundancy - not performance.

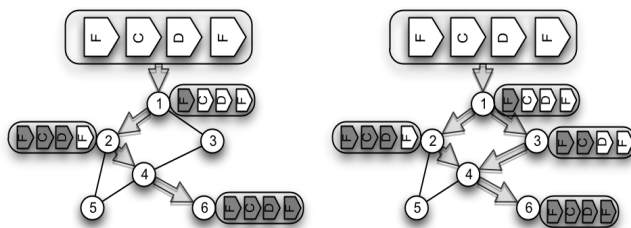


Figure 2: Distribution of workflow tasks without replication (left), and with replication (right).

At each node, using an RL-approach, a processing/forwarding strategy is learned for different states and set of neighbors. The learning process uses a basic reward function (4.1) of each workflow. The goal of an node is to minimize a cost objective function<sup>1</sup> for a given workflow in some state. Consider, for instance a node  $N_1$  that receives a particular workflow  $W = \{A_0, D, C, F\}$  in some state. By workflow state we refer to the set of workflow tasks that are marked as completed, the subindex '0' of a task represents that the task has been completed. In this example, workflow  $W$  is composed by the ordered tasks A, D, C and then F, with A already completed.

At this point, the state of node  $N_1$  is defined by the tuple  $S = \{T_{set}, N_{set}\}$ , where  $T_{set}$  is the set of all pending tasks, and  $N_{set}$  is the set of 1 hop neighbors to whom  $N_1$  can forward the workflow<sup>2</sup>.

In this example, assuming the workflow will not be duplicated, and because tasks must be executed in sequence, the number of actions available to node  $N_1$  is given by  $\|1 + T_{set}\| * \|N_{set}\|$ . Each of these options are, at first, equally acceptable to  $N_1$ . However, once a choice is made for one particular flow, the downstream node that received the workflow will reply with an estimate of cost. After processing a few (similar) workflows, each node starts building a distribution of preferred actions for a given state (i.e. a pol-

<sup>1</sup>Or maximize a function such as survivability/cost.

<sup>2</sup>The notion of topology here is also flexible. In most cases the topology is defined by the communications range between nodes, however, other interpretations based, for instance, in overlay networks or domains of servers are just as valid.

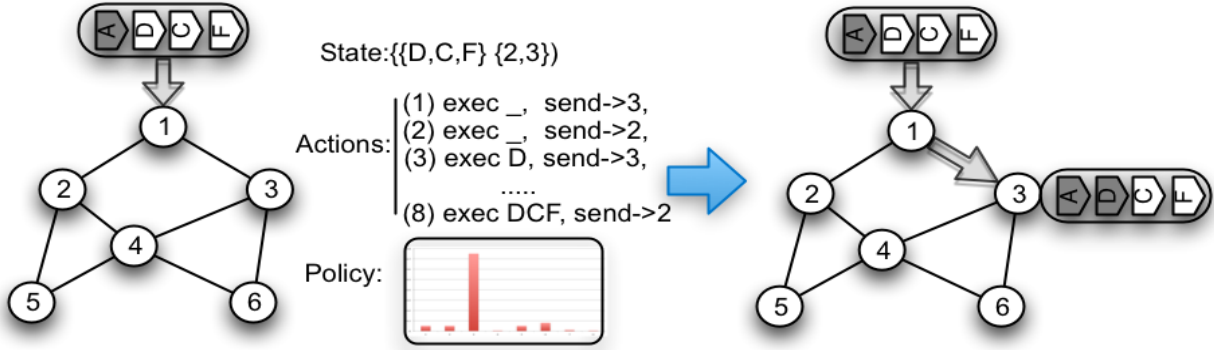


Figure 3: Distribution of workflow Tasks without replication (left), and with replication (right).

icy). The process is illustrated in Figure 3, for a single node making one decision about a partially processed workflow.

Nodes at the end of the processing chain will converge first and, as their cost estimates become more accurate, upstream nodes will quickly converge on the best policies for a global 'near-optimal' solution. In fact it can be shown that in a static environment the solution is asymptotically optimal [1]. The asymptotic nature of the inherited from the well-known exploitation versus exploration trade-off in reinforcement learning. Essentially, as nodes become more confident of good policies for a given state, less incentive they have in exploring new options, which leads to slower convergence.

#### 4.1 Reward Function

The reward function essentially consists in on a immediate feedback from the downstream neighbor that received the workflow. The feedback is the node's *estimate* of the cost for completing the workflow. Every time a node completes a workflow it broadcasts the actual accumulated costs (which is maintained as part of the workflow message). That information is received by all neighbors and used as information to adjust their policy distributions.

For new workflows and topologies, the cost estimates provided by a node (which are used as part of the reward function of its neighbors) is initially coarse because little information is available about the node actions. As local cost estimates become more accurate and consistent, the node stabilizes itself and the effect propagates upstream, leading to a global stable solution.

In addition to cost, another variables considered as part of the reward function is the potential for failure (or risk) associated with a node. Intuitively, a node that is known to be vulnerable should increase its own costs for carrying out workflow tasks, becoming less likely to be chosen as part of the workflow execution.

Combined with a mechanism capable to estimate the likelihood of failure of vulnerability, this approach allows the allocation of tasks to proactively drift away from nodes that are likely to fail or suffer from an attack - increasing the overall survivability of the mission.

#### 4.2 Workflow Task Duplication

One of the contributions of this paper is the addition of task replication as part of the resource allocation problem. The goal of task duplication is to provide some level of ro-

bustness to the overall workflow by duplicating the execution of critical tasks. Because of our initial assumption that all tasks as strictly sequential in a workflow, duplication brings no gains in performance. It actually induces and overhead both for the communications and computational resources.

In the proposed algorithm, however, we can take advantage of the need for replication to increase the exploration of new strategies at each node.

Our current implementation of the algorithm includes two options for exploration strategy, *e-greedy* and *softmax*. Depending on the strategy adopted, the approach for task replication will actually match the requirements for exploration. In *e-greedy* algorithms, for instance, where a periodic exploration strategy is adopted, the approach for data replication would follow the unknown space. That is, at the beginning, while few actions are actually known and characterized there would be a great incentive for task duplication. The heuristics are reasonable since little knowledge about the strength of our actions increase their risk of failure. As the policy becomes more defined, the task duplication is maintained with a distribution that continues to favor the unknown action space. The approach would be similar, in principle, for a softmax exploration strategy, but in that case favor the tails of the Boltzman distribution.

The approach greatly improves the convergence to a stable solution, and also provides a mechanism to ensure that the volume of task duplication reduces, as the confidence of the system increases in known allocations, and expected behavior. Furthermore, the approach allows for a self-tuning mechanism that changes the exploration strategy used for the replicated task if the 'primary' tasks fail to perform. This self-tuning mechanism (which is part of our future work) would allow for the exploration to work as a second best backup policy that will kick-in immediately if the first policy applications start to underperform.

#### 4.3 Failure and Vulnerability Expectations

A second contribution of this paper is to bring the notion of vulnerability (or failure) expectation as a weight for policy allocation. The estimation of risk is not a trivial task in general. Our approach for risk estimation consists on first identifying nodes that are functionally (and capability-wise) 'similar'. Our current metrics for node similarity are based in a software profile (i.e. portfolio of running services). Nodes are first clustered based on their similarity metrics

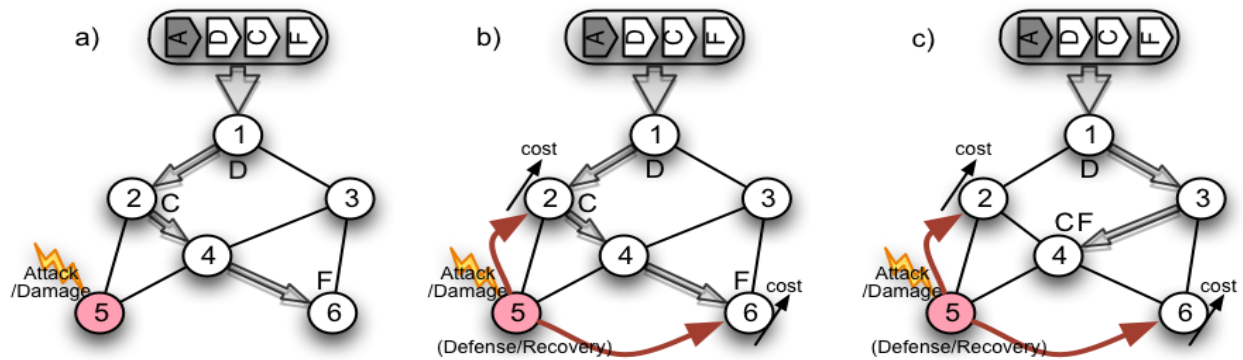


Figure 4: Dynamic adaptation of task distribution based on risk estimates.

and cluster membership is shared across all members.

As illustrated in 4, when a node comes under attack or experiences some kind of failure, it makes itself ineligible for task execution and immediately notifies its cluster members. Based on that information, the other members increase their cost functions to become less desirable for resource allocation.

The assumption is that similar nodes are likely vulnerable to similar threats. If an attack is detected, proactively moving similar nodes out of the critical execution points will help avoid disruption occurring from subsequent attacks.

## 5. EXPERIMENTAL RESULTS

While this is a work in progress, we have modeled and implemented a simple simulation of the approaches described above. Rather than collecting and analyzing rigorous experimental results, our current focus is on an empirical verification of the feasibility of the proposed approach. Figure 5 shows a simple example (NS-2 simulation, [4]) of the overall cost curve associated with the resource allocation of a series of workflows in a small (7 node) scenario.

In this example, the response of the e-greedy exploration strategy is shown for a sudden event (a change in topology caused by the removal of an important node). In response to the event, the algorithm searches for a new stability point and finds a new solution that is more expensive than the previous (due to the limitations of the new topology), but is the optimal solution for the current topology.

The time (or more importantly, the workflows) necessary to re-learn a new stable strategy was approximately 60 workflows, which is a reasonable number for many applications, including data-stream scenarios. It is also important to note that during this period of adaptation, some workflows were indeed lost (represented by the circles on top and bottom of the graph area) but most were successfully executed, although with a sub-optimal allocation.

## 6. CONCLUSIONS

In this work we have formulated a resource allocation problem for tactical MANETs based on Reinforcement Learning. The focus of our strategy is to improve the levels of mission robustness by intelligently allocating and replicating tasks across the network and by proactively moving resource from nodes that may be at a higher risk of failure or attack. We have conducted several tests for resource allocation using

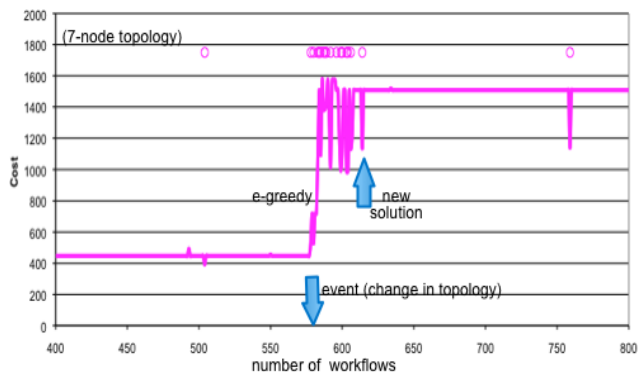


Figure 5: Comparing exploration strategies for a small 4-node network

a simplified version of the proposed task duplication strategy. While largely qualitative at this point, our initial simulation results seem to indicate that the approach is feasible and highly adaptive to unforeseen changes in resources.

## 7. REFERENCES

- [1] P. Auer, U. D. Milano, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *36th Annual Symposium on Foundations of Computer Science*, 1998.
- [2] J. A. Boyan and M. L. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In *Advances in Neural Information Processing Systems 6*, pages 671–678. Morgan Kaufmann, 1994.
- [3] M. Carvalho. In-stream data processing for tactical environments. *International Journal of Electronic Government Research*, 4(1):49–67, Jan. 2008.
- [4] F. Shull, M. Mendonça, V. Basili, J. Carver, J. C. Maldonado, G. H. Travassos, and M. C. Ferreira. The network simulator - ns2 homepage, <http://www.isi.edu/nsnam/ns/> [accessed july 31. *Empirical Software Engineering*, page 2, 2002.
- [5] R. S. Sutton, P. P. Chubak, D. Architecture, and D. Architecture. Reinforcement learning, 1998.