Applying Distributed Optimization for QoS-Security Tradeoff in a Distributed Information System*

(Extended Abstract)

Hala Mostafa, Nathaniel Soule,
Nicholas Hoff, Partha Pal
Raytheon BBN Tecchnologies
Cambridge, MA 02138
{hmostafa,nsoule,nhoff,ppal}@bbn.com

Patrick Hurley
Air Force Research Laboratory
Rome, NY 13441
Patrick.Hurley@rl.af.mil

ABSTRACT

In a distributed information system, Quality of Service (QoS) and Information Assurance (IA) compete for the same set of resources. This tension increases in the presence of cyber attacks. Previous work formulated the problem of trading off QoS against IA as a DCOP whose solution sets the local configuration at individual decision-making nodes to optimize overall levels of QoS and IA delivered by the system. In this paper, we report on the first implementation of max-sum in a realistic distributed system running on multiple machines. Sample results from mission-oriented scenarios based on published documentation and run in an emulated network show the advantage of tradeoff-driven adaptation in meeting QoS and IA requirements.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed AI

Keywords

DCOP; Implementation; QoS

1. INTRODUCTION

Information Assurance (IA) mechanisms such as firewalls and antivirus scanners use the same resources (e.g., CPU, network bandwidth) as the services they defend, leading to a tension between IA and Quality of Service (QoS). Currently, these tensions are resolved manually, with local decisions that are oblivious to system-wide consequences. We extend QIAAMU, a framework to continuously assess various IA/QoS quantities [4], to perform distributed tradeoffs between QoS and IA to best meet user requirements.

In QIAAMU, each node tries to meet the requirements of its users by manipulating **actuators** (e.g. setting a firewall policy, turning compression on/off). A QoS or IA **attribute** is an aspect of system performance (e.g., Availability of a service, Confidentiality of a link). Users specify the required level for each attribute, with preferences specifying relative interests in different attributes. For example, the system administrator may care more about the Integrity of a database, while a regular user cares more about the Timeliness of a service. Attribute levels depend on actuator settings and continuously monitored system conditions.

Nodes engage in distributed optimization to trade off QoS and IA to bring delivered attribute levels close to requirements. The result of a tradeoff is a configuration for actuators on all nodes. We formulate each tradeoff as a Distributed Constraint Optimization Problem (DCOP) and solve it using max-sum [2] and our value propagation scheme [4].

We report on deploying QIAAMU in an existing system where the factor graph is distributed across nodes. To the best of our knowledge, ours is the first application of distributed constraint optimization in the QoS/IA domain, and the first deployment of max-sum and value propagation on multiple machines (beyond the simple scenario in [3]).

Applying QIAAMU

Our distributed environment emulates a military scenario [1] where 4 clients (JTAC, JFO, CAS and UAV) communicate via a server accessible through a defensive layer consisting of 2 nodes, *lime* and *grape*, equipped with security mechanisms like firewalls and single packet authorization. An adversary launches cyber attacks affecting various system conditions. The nodes' responses to attacks depend on the state of the system and user requirements in the current mission epoch.

A node's configuration file specifies users, preferences, requirements, actuators, system conditions to assess on this node and how their values are obtained (e.g., from a SQL query). It also specifies a *cause-effect network* (Figure 1) describing how attributes on this node depend on actuators and system conditions, including those residing on other nodes. From these distributed cause-effect networks, the nodes construct the distributed factor graph and set up infrastructure for message passing across machines.

2. SAMPLE RESULTS

We present an experiment using scenarios based on published reports of close air support operations [1]. Figure 2 shows how the global penalty (differences between required and delivered attribute levels, summed over attributes and weighted by preferences) changes as the mission progresses through 4 epochs and an attack is injected.

*Supported by US Air Force Research Laboratory contract No. FA8750-08-C-0196. Approved for Public Release; Distribution Unlimited: 88ABW-2012-5734,26 Oct 2012.

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*, Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.

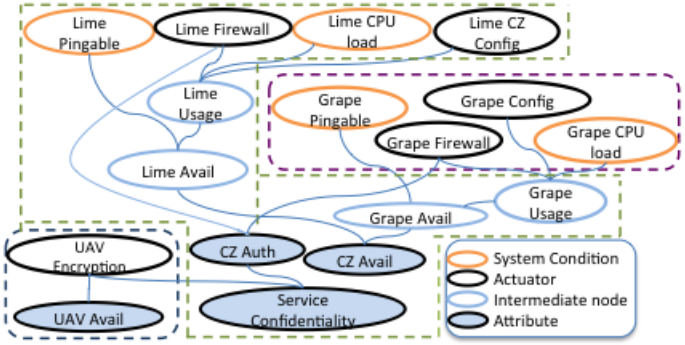

Figure 1: Cause-effect network distributed across *lime*, *grape* and UAV

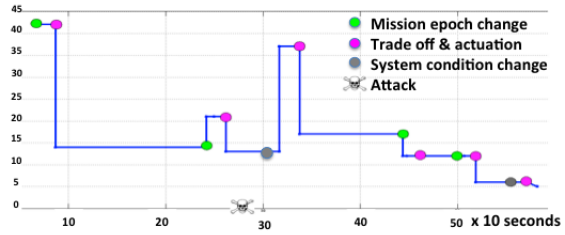

Figure 2: Global penalty over time

Initially, *lime* and *grape*'s firewalls are off and the clients use no encryption, which incurs a penalty of 42 because IA requirements are not met. The system performs a tradeoff (solves a DCOP) and the resulting configuration brings down the penalty to 14 by turning both firewalls on and turning on encryption on CAS.

In the second mission epoch, confidentiality requirements on JTAC and JFO increase, incurring a penalty of 21 and triggering a tradeoff which turns off the firewalls while turning on JTAC and JFO encryption. The adversary then compromises *lime*, which is picked up a system condition, triggering a tradeoff that enforces a strict firewall policy for *grape* to mitigate the security breach. Although this does not restore the penalty to its pre-attack value, it ameliorates most of the attack effects.

In the third epoch, this actuator configuration is still adequate. The fourth epoch lowers the confidentiality requirements of JTAC and JFO. The penalty of the current configuration is still 12, but a trade off reveals that the Availability of JTAC/JFO can be improved if their encryption is turned off, reducing the penalty to 6.

Handling bandwidth flood attack

In this scenario, the adversary floods the link between UAV and *lime*, reducing available bandwidth (BW in Table 1). Requirements place a strong emphasis on the Availability of *lime* and *grape* which depends on the security mechanisms they employ and the number of clients using them. Availability drops if more than 2 clients use the same node. The level of threat perceived by *grape* is derived from logs of various intrusion detection mechanisms.

Upon startup, all clients default to *grape*, firewall is off and client encryptions are on. This doesn't meet QoS and IA requirements, resulting in a high penalty of 51 and triggering tradeoff A in Table 1. The tradeoff evenly splits the 4 clients. Because *grape*'s threat level is low, its firewall remains off, but the clients' encryptions are on because each client's Confidentiality requirement is met when either its encryption is on or *grape*'s firewall is on. The attack then reduces the UAV-*lime* bandwidth. Separately, a system condition on *grape* detects invalid login attempts and sets threat level to High. This triggers another tradeoff which results in UAV moving to *grape* to avoid the flooded link. Because *grape*'s threat level is high, it turns its firewall on, which would reduce its Availability if it continues to serve 2 clients. Therefore the other clients switch to *lime*, with the desirable side-effect that clients turn their encryption off and still meet their Confidentiality requirements, which reduces their CPU load and improves their Availability.

Table 1: Snapshots from the flooding attack scenario

	BW	*grape* Threat	P_{bef}	Client Encr.	FW	Nodes used by clients	P_{aft}
A	Hi	Lo	51	On	Off	*lime*:UAV,CAS *grape*:JTAC,JFO	15
B	Lo	Hi	26	Off	On	*grape*:UAV *lime*:CAS,JTAC, JFO	13

The above examples show how nodes in the distributed system coordinate over their actuator configuration to optimize QoS/IA attribute levels as system conditions change.

3. DISCUSSION AND FUTURE WORK

We presented an implemented framework for the runtime assessment and adaptation in a distributed system to manage the tradeoff between QoS and IA. Each node in the system is configured with a complex cause-effect network describing the effects of system conditions and actuators on QoS/IA levels. We formulate the tradeoff problem as a DCOP and use message passing on the distributed factor graph to solve it. We provide the first implementation of max-sum and value propagation in a realistic distributed system running on multiple machines.

Areas of future work include improving knowledge elicitation from system users and administrators to build the cause-effect networks. We will also try to maintain a consistent user experience where nodes are discouraged from frequently changing their actuators if doing so does not decrease the penalty significantly.

4. REFERENCES

[1] J. Cleveland, J. Loyall, and J. Hanna. Fault tolerance requirements of tactical information management systems. In *Military Communications Conference*, 2012.

[2] A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *AAMAS*, pages 639–646, 2008.

[3] F. M. D. Fave, A. Rogers, Z. Xu, S. Sukkarieh, and N. Jennings. Deploying the max-sum algorithm for decentralised coordination and task allocation of unmanned aerial vehicles for live aerial imagery collection. In *IEEE ICRA*, 2012.

[4] H. Mostafa, P. Pal, and P. Hurley. Message passing for distributed QoS-security tradeoffs. In *Optimization in Multi-Agent Systems Workshop, AAMAS 2012*, 2012.