[1] S. Edelkamp. Symbolic exploration in two-player games: Preliminary results. In *International Conference on AI Planning and Scheduling International Conference on AI Planning and Scheduling (AIPS'02) Workshop on Model Checking*, 2002. [bib | .html]

[2] J. Feigenbaum, C. H. Papadimitriou, and S. Shenker. Sharing the cost of multicast transmissions. *Journal of Computer and System Sciences*, 63(1):21-41, August 2001. [bib | DOI]

We investigate cost-sharing algorithms for multicast transmission. Economic considerations point to two distinct mechanisms, marginal cost and Shapley value, as the two solutions most appropriate in this context. We prove that the former has a natural algorithm that uses only two messages per link of the multicast tree, while we give evidence that the latter requires a quadratic total number of messages. We also show that the welfare value achieved by an optimal multicast tree is NP-hard to approximate within any constant factor, even for bounded-degree networks. The lower-bound proof for the Shapley value uses a novel algebraic technique for bounding from below the number of messages exchanged in a distributed computation; this technique may prove useful in other contexts as well.

[3] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Experiences applying game theory to system design. In *PINS '04: Proceedings of the ACM SIGCOMM workshop on Practice and theory of incentives in networked systems*, pages 183-190. ACM Press, 2004. [bib | DOI]

[4] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166-196, 2001. [bib | DOI]

We consider algorithmic problems in a distributed setting where the participants cannot be assumed to follow the algorithm but rather their own self-interest. As such participants, termed agents, are capable of manipulating the algorithm, the algorithm designer should ensure in advance that the agents' interests are best served by behaving correctly. Following notions from the field of mechanism design, we suggest a framework for studying such algorithms. Our main technical contribution concerns the study of a representative task scheduling problem for which the standard mechanism design tools do not suffice.

[5] M. Palomino, N. Martí-Oliet, and A. Verdejo. Playing with Maude. In *5th International Workshop on Rule-Based Programming (RULE)*, 2004. [bib | .html]

[6] C. Papadimitriou. Algorithms, games, and the internet. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing (STOC)*, pages 749-753, New York, NY, USA, 2001. ACM Press. [bib | DOI]

[7] O. Sheyner. *Scenario Graphs and Attack Graphs*. PhD thesis, Carnegie Mellon University, April 2004. [bib | .pdf]

We develop formal techniques that give users flexibility in examining design errors discovered by automated analysis. We build our results using the model checking approach to verification. The two inputs to a model checker are a finite system model and a formal correctness property specifying acceptable behaviors. The correctness property induces a bipartition on the set of behaviors of the model: correct behaviors, which satisfy the property, and faulty behaviors, which violate the property. Traditional model checkers give users a single counterexample, chosen from the set of faulty behaviors. Giving the user access to the entire set, however, lets him have more control over the design refinement process. The focus of our work is on ways of generating, presenting, and analyzing faulty behavior sets.

We present our results in two parts. In Part I we introduce concepts that let us define faulty behavior sets as “failure scenario graphs.” We then describe algorithms for generating scenario graphs. The algorithms use model checking techniques to produce sound and complete faulty behavior sets.

In Part II we apply our formal concepts to the security domain. Building on the foundation established in Part I, we define and attack graphs, an application of scenario graphs to represent ways in which intruders attack computer networks. Attack graphs depict ways in which an adversary exploits system vulnerabilities to achieve a desired state. System administrators use attack graphs to determine how vulnerable their systems are and to determine what security measures to deploy to defend their systems. This application of formal analysis contributes to techniques and tools for strengthening network security.

[8] O. Sheyner and J. Wing. Tools for generating and analyzing attack graphs. In *Proceedings of Formal Methods for Components and Objects*, Lecture Notes in Computer Science, pages 344-371, 2004. [bib | .pdf]

[9] K. wei Lye and J. Wing. Game strategies in network security. *International Journal of Information Security*, 4(1-2):71-86, February 2005. [bib | DOI | .pdf]

This paper presents a game-theoretic method for analyzing the security of computer networks. We view the interactions between an attacker and the administrator as a two-player stochastic game and construct a model for the game. Using a nonlinear program, we compute Nash equilibria or best-response strategies for the players (attacker and administrator). We then explain why the strategies are realistic and how administrators can use these results to enhance the security of their network.

[10] J. M. Wing. Scenario graphs applied to security. In *Proceedings of Verification of Infinite State Systems with Applications to Security (VISSAS)*, Timisoara, Romania, March 2005. Summary paper. [bib | .pdf]
