Efficient Multihop Broadcast with Distributed Protocol Evolution

Bernát Wiandt, Vilmos Simon, Endre Sándor Varga

Efficient Multihop Broadcast with Distributed Protocol Evolution

Bernát Wiandt, Vilmos Simon, and Endre Sándor Varga

Budapest University of Technology and Economics, Department of Telecommunications
http://www.hit.bme.hu

Abstract. In this paper we describe an efficient way of implementing multi hop broadcast in ad hoc mobile networks with an online, distributed machine intelligence solution. In our solution not just the runtime parameters of predefined protocols are optimized, but the decision logic itself also emerges dynamically. The model is based on genetic programming and natural selection: sucessive generations of protocol instances are produced to approximate optimal performance by picking certain instances from the previous generation (natural selection) and combining them with each other and/or mutating (genetic operators) them. We implemented (i) a genetic programming language to describe protocols, and (ii) defined a distributed, communication-wise non-intensive, stigmergic feed-forward evaluation and selection mechanism over protocol instances, and (iii) a budget based fair execution model for competing protocols. The results indicate that online, autonomous protocol evolution outperforms traditional approaches, by adapting to the situation at hand, when used for the multi-hop broadcast problem in ad hoc mobile networks. The evolution also protected the system from the negative effects of initially present harmful protocols.

Keywords: multihop, distributed, evolution, genetic programming

1 Introduction

Choosing the right communication protocols for achieving efficient multi-hop broadcast in a mobile ad hoc network proved to be a complex problem. While too chatty protocols waste resources such as bandwidth and processing power, unnecessarily tight-lipped communication strategies can impede the effective operation of the system. Recent studies indicate that while there is no solution for the riddle in general, it makes sense to evaluate the goodness of communication protocols for a certain problem case [1,10,4,2]. The idea of protocol selection or protocol switching has been present for many years in other areas, such as cryptography. Our proposal goes one step further: we do not only select but create and shape the protocols, therefore they are not static, pre-deployed parts of the system but the protocol logic emerges dynamically and adapts online to the current network environment. In this article we will show that the application

of genetic programming for this task not only reduces costs, but with a suitable model, also guarantees the emergence of successful protocols in the end. In this article we are focusing on solving the multi-hop broadcast problem where the network must broadcast a message to all it's nodes while keeping resource usage as low as possible.

1.1 Multi-Hop Broadcast

It is a common task in mobile ad hoc networks to distribute messages globally to all, or almost all participants. By it's nature, this kind of service consumes a significant amount of resources (channel usage, collisions, messages sent multiple times), therefore finding an efficient solution to this problem is of high importance. Channel usage is just one of the difficulties that present themselves when one implements global scale broadcast protocols. One dangerous phenomenon is the so called Broadcast Storm [7] that happens when multiple nodes start forwarding a message simultaneously after receiving it from a common source node, leading to excessive collisions. To avoid this situation, protocols have means to de-correlate from the traffic of their neighbors, for example by waiting for a random time before forwarding the message (Random Assessment Delay). Multi-hop broadcast algorithms typically exploit the local broadcast channel to reduce channel usage and the number of collisions in the system. This way, as one transmission may be overheard by multiple devices, it is possible to drastically reduce the amount of transmissions. Efficient broadcast could be achieved by identifying a Minimal Connected Dominating Set (MCDS)[5,4] in the network, and broadcasting the message once per set. However, the identification of an MCDS is an NP-complete problem, but even if it was not, the network is distributed and too dynamic - changes may occur much faster than they can be discovered. In general a centralized MCDS solver is not feasible, so instead of tackling with real MCDSs, ad hoc broadcast protocols typically use some kind of approximation based on simple heuristics and local knowledge. These heuristics range in sophistication from simple counter based solutions to probabilistic methods and complex graph theoretic approximations[11,2].

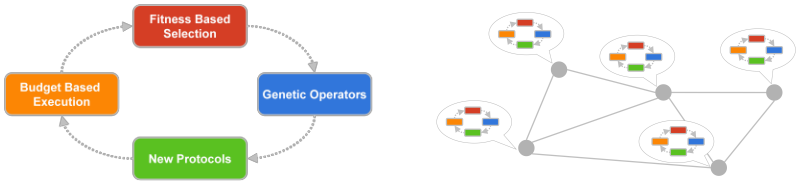
1.2 Natural Selection

Various literature sources investigate possible protocols for multi-hop broadcast and their performance characteristics, a few examples are [10,4,2,1]. Results suggest that there is no general winner; instead, the performance of a protocol heavily depends on volatile attributes of the environment. These attributes include mobility patterns, node speed, node density, transmission technology, and traffic models. Selecting the suitable protocol, therefore, requires deep and exact knowledge about the actual environment. However, that is generally hard to acquire, given the complex factors involved, such as human behavior influencing the mobility pattern and the load characteristics. Worse, the environment will change over time, through appearance and disappearance of nodes, technology turnovers, or changes in the usage practice, i.e. human habits; therefore any

static off-line design is just a compromise. The issues above raise the question whether an automated, online, adaptive approach could solve the problem of obtaining the best protocols for a given situation. The use of online, adaptive techniques for protocol optimization (i.e. fine tuning of operational parameters on-the-fly) is a known, but not widely used practice[9]. For protocols, even if machine learning is applied, this step typically happens during the manual design phase, and not as part of the operation of the actual system. An exception is [3], where authors used online machine learning to approximate the behavior of sophisticated broadcast algorithms and found that simple heuristics were able to approximate the sophisticated protocols with 87% accuracy. This result indicates that in practice small but powerful heuristics could provide good approximations instead of sophisticated calculations. Note that Colagrosso's work uses predefined (fixed) protocol bodies, and aims to optimize the runtime parameters of these protocols. Our approach goes one step further: in our work the protocol body itself is also an emergent, ever-changing element. In [8] we proposed stigmergic communication and natural selection for online, automatic protocol replacement.

2 Autonomous Online Protocol Evolution

2.1 The overall picture



(a) Main loop

(b) Fully distributed, autonomous system

Fig. 1: Overall picture

In our proposed model each node selects and generates protocols on its own agenda, therefore protocol evolution is a fully distributed, asynchronous mechanism. Each protocol candidate's performance is measured continuously by it's node's neighbors. The evaluated protocols undergo a selection step, deciding which protocols survive and which end their lifetime. In the next phase, genetic operators, i.e. crossover and mutation are used in order to introduce new protocols by combining and/or modifying selected instances. Then each of the newly created protocols is executed, using a budget-based execution scheme, giving equal opportunity to each individual of the generation to live. Finally, the loop starts over. Protocol evolution runs in each node of the network, in parallel, without any explicit synchronization with other nodes. No global clock or database

is assumed. Neighbors, as part of the inverted decision making mechanism, discover each other's protocols; thus, successful protocol instances can spread over the network.

2.2 Decision Inversion

Using natural selection means we need a reliable performance metric to rank the different protocols present in the system. Performance evaluation criterion in a multi-hop broadcast based ad hoc network needs to meet conflicting requirements (maximal coverage vs. minimal number of duplicate messages) as well as the problem of measurability. The factors we considered here are the following:

- Only the sender node is able to reliably measure the real cost of a successful message transmission.
- Lost messages (by definition) could not be seen by other nodes.
- Only the receiver nodes are able to reliably measure the number of duplicated messages.

Each node can measure the number of duplications they personally receive, but they can not measure the number of total duplications in the system. Collecting of measurement data in the network is not feasible because these messages would use the same channel as the useful data messages, furthermore they may get lost. These factors imply [8] that implementing a centralized (even locally centralized) protocol selection criterion is impractical, because the reliable collection of performance data is both technically challenging and wasteful in terms of channel usage. Instead, we propose a feed-forward selection method using stigmergy and natural selection, which is based on the idea of decision inversion.

Instead of evaluating protocols in the sender node we gave the ability of decision making to the receivers, because they are in an optimal position to observe the performance of a protocol. They do so by observing the received messages and extracting sender protocol information from them. In order for this to work, nodes in the system attach the code of their current protocol to every message they send. Every payload that is useful to the receiver node means a chance for the sender protocol to survive. Every unnecessary message (duplicate) means wasted resources to the sender protocol.

The main advantage of the inverted selection is that performance results don't need to get back to the sender: instead, the receiver will utilize them during the creation of its own next protocol generation (so, in the next round, the sender may meet the offsprings of its own good protocols). This way, the measurement overhead is minimal – result container messages are not needed–, and there is no need for synchronization of any kind. The fitness function evaluated in every node can be observed in figure 2.

2.3 Budget Based Protocol Execution Model

In order to keep the number of messages sent out by a protocol at bay - keeping them from flooding the channel - we assigned a cost to each transmission. We

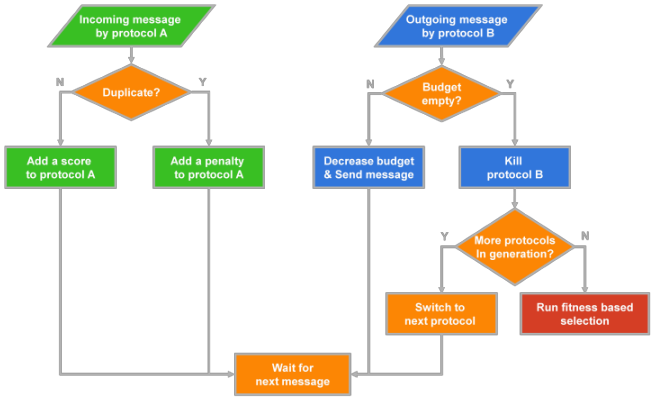


Fig. 2: Protocol fitness evaluation

wanted to keep the system fully distributed and avoid expensive measurement messages being sent, but still integrate the cost of sending messages into the final cost function. Therefore we adopted a stigmergic solution: by assigning a limited transmission budget to each protocol instance, which as a result forces protocols to make good use of channel resources. Any lost or duplicate message is a lost opportunity for reproduction, therefore transmission has an implicit cost function, even if it is not expressed directly. Similarly, every protocol is used for a limited amount of time, therefore they have only a small window of opportunity to spread in the network.

2.4 A Genetic Programming Language for Protocols

Natural selection implemented by decision inversion, along with the budget based execution model, answers the question: how protocols should be executed, evaluated and selected for survival in a distributed fashion. The only question remaining is how should we represent our protocols in order to use them in this system. As the protocols are no longer engineered by humans, a lightweight, flexible and robust formal description is needed which suits genetic operators. We propose the GPDISS language, which is based on PUSH but specializes in implementing multi hop broadcast algorithms. GPDISS is a stack based language, which means that every type in the system has an associated stack and every instruction gets it's arguments from the appropriate stacks and puts the results on these stacks too. In a GPDISS protocol definition one creates event handlers to handle the different messages that the node receives. During the crossover phase, only appropriate event handlers are mixed, essentially reducing the number of useless offsprings. GPDISS has another very important property: crossover and mutation instructions are guaranteed to produce a syntactically correct offspring, which of course does not mean that the protocol will be semantically correct too but contributes to reduce the number of useless instances in the system.

Selection Every protocol generation is created from the non-local protocols that were discovered in the previous round. We used SUS (Stochastic Universal Sampling) with a fitness function that gives priority to better performing individuals [6]. SUS provides zero bias and minimum spread, meaning that the actual and expected probabilities of selecting an individual are equal, and the range in the possible number of trials that an individual can achieve is minimal. SUS is a variant of the roulette wheel selection.

Crossover and Mutation A modified one-point crossover is used for combining two event handlers. Given that the protocol pair (A, B) is selected for crossover, the algorithm is the following:

1. Choose an event handler from A randomly. If B has no such event handler, then return.
2. Select a cutting point randomly in A's handler, and another point in B's handler.
3. Cut the handlers along the cutting points, resulting in four fragments: A-head, A-tail, B-head, and B-tail.
4. With 0.5 probability exchange the head and the tail fragment of the original handlers.
5. Glue fragments together forming two new handlers, an (A-head, B-tail) and a (B-head, A-tail).

To protect handlers from growing indefinitely, we limited the maximal size of event handlers (measured in instruction count); bodies above the limit were chunked. For the mutation part we use constant parameter mutation, meaning that instead of modifying instructions in the event handler body, the mutation affects the constants, i.e. the runtime parameters of the algorithm. For example such a runtime parameter is the message propagation probability in a standard Gossip protocol. When a parameterised instruction in the protocol code with current value x is mutated, the new value is chosen from the $(0, 2x]$ range with a Gaussian distribution, favoring fine-tuning but also allowing larger changes. It is important to note that only the instruction's parameter is changed during mutation.

3 Experimental Evaluation

Several simulations were executed to study our system's performance characteristics and prove that it can improve overall performance of the network. First we will describe the simulation environment, then we establish a baseline performance and in the third part we will analyze our system's performance in a situation where malicious attackers try to bring down the network with flooding. In the graphs below ever data point corresponds to a protocol instance in the system. The data points are ordered in time.

3.1 Simulation environment

Throughout the experiments nodes were moving in the simulation area according to the Random Direction Mobility Model. In this mobility model nodes randomly choose a direction and distance and move with $1^{m}/_{s}$ speed until they reach the desired distance. After reaching the destination, the process starts over. In the beginning of each simulation the nodes present were placed evenly distributed in the area by generating random starting coordinates for them. In the simulations nodes are totally independent of each other; there were no global database or control present. During the experiments new messages were broadcasted by the nodes periodically. The overall goal was to broadcast the payload messages with the possible highest total coverage and lowest duplicate count before the payload expires. There were a variable percentage of malicious nodes in the network for each simulation scenario. Their goal was to flood the network and decrease performance, possibly causing a broadcast storm. Another goal was to spread among the nodes and control the whole network. The attackers were running the Flood protocol.The general settings were the following:

- Node count:500
- Simulation time: $1000s$
- New message broadcast interval:$1..20s$, evenly distributed for every node
- Maximum age of broadcasted payload: $20s$
- Protocol time budget: $7s$
- Size of simulation area: $800m * 1200m$
- Wireless range: $50m$
- Interference range: $70m$

3.2 Used protocols

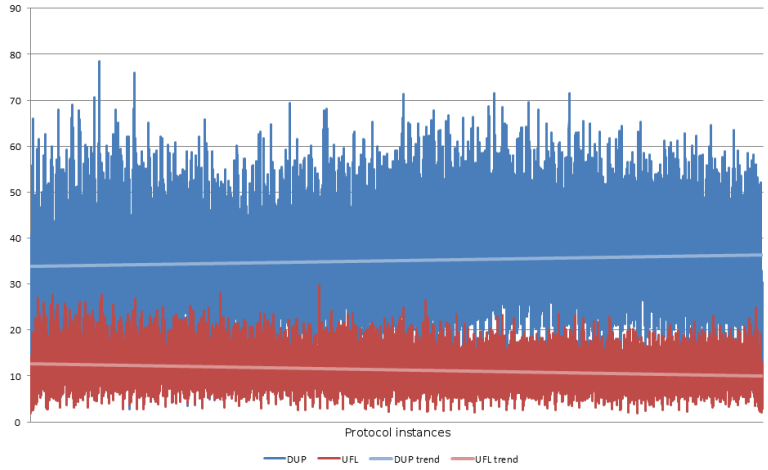
The initial population was selected from a small set of well-known protocols that are simple enough to be the starting point of evolution. For example Adaptive Periodic Flood (APF) as an optimization of blind flood: an APF node periodically transmits all the messages it possesses to all neighbors it encounters, after a random waiting period. However, when it detects that there is another node sending the same message, it increases the waiting period to reduce the total channel usage. Another well-known protocol is Gossiping (Gos): a gossiping node forwards the received message to its neighbors with a given probability. Gossiping is easy to analyze mathematically, as neighboring nodes have minimal effect each other's operation. The last simple heuristic used was density sensitive adaptive gossiping (AGos): in adaptive gossiping the probability of propagating the message depends on some condition. We used a density sensitive model, where the probability of forwarding the message decreases as the number of neighbors gets higher. We used one more protocol in the simulations to model a simple attack against the network. This is a simple flooding protocol (Flood): it sends out three copies of each message immediately after receiving it. This behavior increases the chances of causing a broadcast storm in the network.

3.3 Measurements without evolution in the system

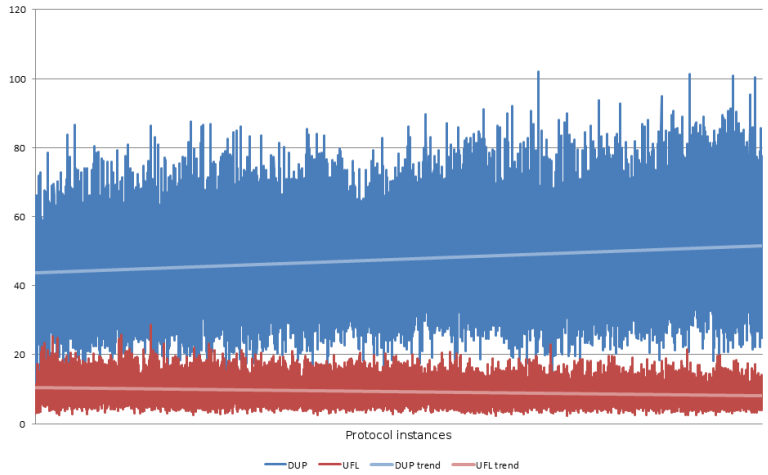
The first measurements serve as the baseline performance of the network. We simulate a normal mobile ad hoc network with different protocols in the nodes and varying proportions of attackers. Attackers are an important part of the experiment as we are interested in the effect they have on the overall performance. In some situations flooding can be the right strategy for a node so it's possible that flooding protocols will survive and there is a chance that they will spread in the network. Each run is carried out with 5, 30 and 70 percent of malicious nodes. Other nodes in the system have APF, GOSSIP or Adaptive GOSSIP as their initial protocol evenly distributed in the beginning of each simulation. Figure 3a shows a higher number of duplicate messages than useful ones. This is normal for our network as it is the nature of data dissemination in mobile ad hoc networks. We can observe the effect of flood protocols present in the system by comparing figure 3a and figure 3b. It is clear that flooding results in overall worse performance by increasing the number of duplicate messages in the network. In figure 3c we see that as we add more flooding nodes to the network, the efficiency of the network drops significantly. We can conclude from the above experiments that our approach is working: the more flooding nodes present in the network the lower the performance drops. We can also state that our network is stable throughout the simulation in a sense that performance figures do not vary apart from the increases and decreases resulting from the mobility and the ever changing architecture of the network.

3.4 Measurements with evolution

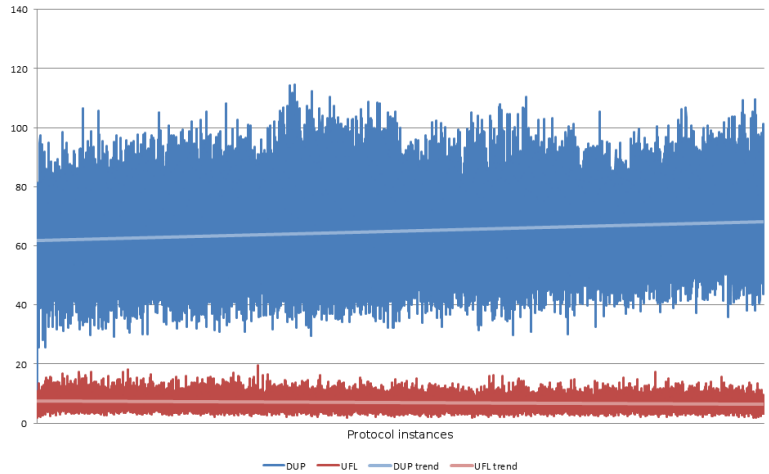
In this series of measurements we were interested in the effects of evolution in the network. Protocols are competing with each other, their properties are mixed, creating new protocols as time passes. Each node optimizes it's performance over time, therefore we are expecting that the overall performance of the network will improve too. In figure 4a the starting numbers of duplicate and useful messages compared to figure 3a are the same. Figure 4a shows that the number of duplicate messages are declining over time, while the number of useful messages are nearly the same. Generally it is not enough to just silence the inefficient nodes because that would eliminate a lot of useful messages too. In figure 4a useful messages are sent throughout the simulation, which is an indication that the network preserved it's capability to disseminate data and it's efficiency got better over time by eliminating duplicate messages. We see the same trend in figure 4b and 4c. That means that the system can eliminate malicious nodes even in the case when only a small minority (30%) starts the experiment with a non-flooding protocol. Figure 5a and 5b show the proportion of flood protocol in successive generations of new protocols created in the system. We can observe a clear decline in the proportion of flood but it's much less pronounced in figure 5b than in figure 5a. This phenomena shows that the protocol composition in the beginning can influence the optimal solution our system generates.



(a) Baseline results without evolution, 5 percent of attackers

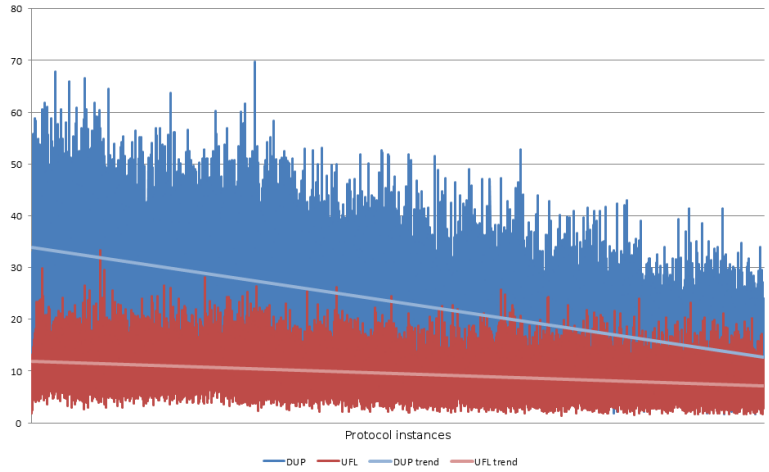


(b) Baseline results without evolution, 30 percent of attackers

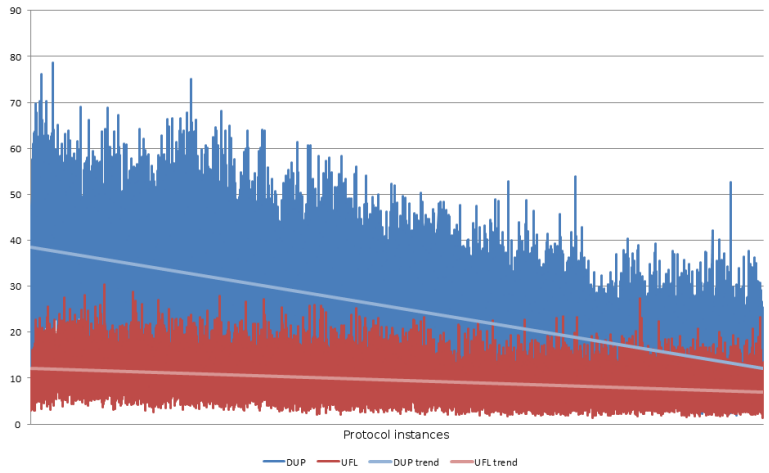


(c) Baseline results without evolution, 70 percent of attackers

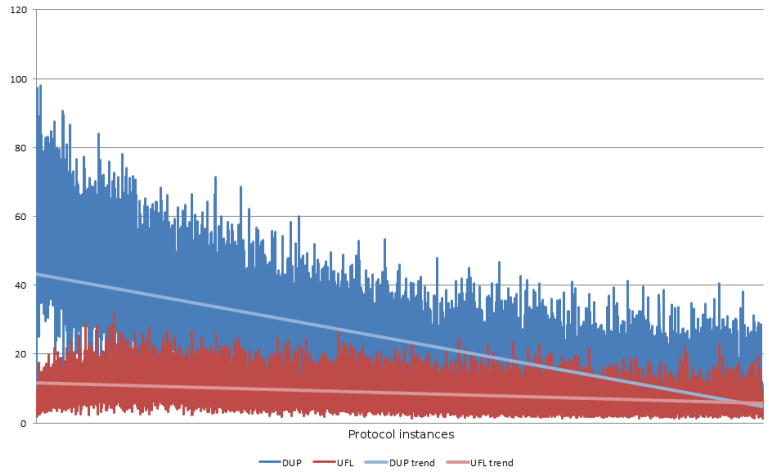
Fig. 3: Baseline performance metrics



(a) Useful and duplicate message counts, using evolution, 5 percent of attackers

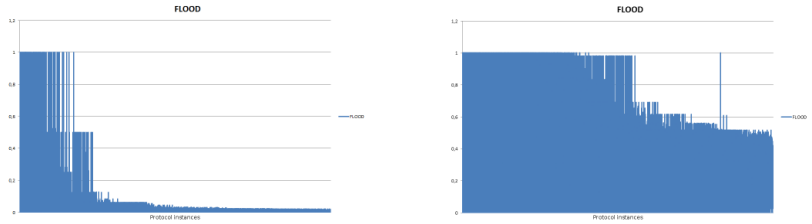


(b) Useful and duplicate message counts, using evolution, 30 percent of attackers



(c) Useful and duplicate message counts, using evolution, 70 percent of attackers

Fig. 4: Performance with evolution enabled



(a) Proportion of flooding protocol in other protocol instances over time, 5 percent of attackers

(b) Proportion of flooding protocol in other protocol instances over time, 70 percent of attackers

Fig. 5: Proportion of flooding protocols over time

4 Conclusion

Our simulations confirmed that the autonomous, online protocol evolution model is a working approach to optimize and self-adapt multi-hop broadcast networks. In our experiments, evolution reduced the number of sent duplicate messages while maintaining the number of useful messages in the network. Our model using evolution and natural selection was able to neutralize the negative effects of a malicious protocols present in the system (Flood). However, Flood was not simply eliminated from the system, but instead, parts of its code got incorporated into good-performing offsprings in some cases.

Our results affirm our belief, that the demands for the new forms of networking infrastructure can be effectively addressed by bio-inspired solutions. Our focus was on presenting an evolutionary framework for the family of multi-hop broadcast protocols in ad hoc networks, where it is usually impossible to find a single absolute candidate, as the optimal protocol choice always depends on the actual environment and application conditions. We introduced a novel idea in this field: instead of human engineered static protocols, autonomous evolutionary methods were applied to achieve dynamic emergence of new ones, driven by the current needs and environment of the communicating nodes. We showed that the proposed model of evolving protocols is applicable for the multi-hop broadcast problem in ad-hoc networks: with time, evolution results in better performance than that the initial, manually engineered, protocols could provide. The fitness function was defined so that it used only local and quasi-local input, resulting in a model that is applicable for fully distributed systems such as ad-hoc sensor networks. Also, the feed-forward nature of the evaluation and selection process eliminated most of the communication overhead needed for the calculation of fitness values. Additionally, the process was carried out in an online manner, that is, the evolution of protocols happened continuously during the normal operation of the system. An important limitation of the model is that being based on a quasi-random search, it cannot provide any quality guarantee on the short term; for example, we cannot claim that the next generation of protocols will always improve the current one. While guarantees do not exist for the quality of

protocol individuals, the overall performance of the system, especially for longer time windows, improves with high probability.

References

1. Ahmad Al Hanbali, Mouhamad Ibrahim, Vilmos Simon, Endre Varga, and Iacopo Carreras. A survey of message diffusion protocols in mobile ad hoc networks. In *Proceedings of the 3rd International Conference on Performance Evaluation Methodologies and Tools*, ValueTools '08, pages 82:1–82:16, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
2. Xiuzhen Cheng, Xiao Huang, Deying Li, and Ding zhu Du. Polynomial-time approximation scheme for minimum connected dominating set in ad hoc wireless networks. *Networks*, 42:202–208, 2003.
3. Michael D. Colagrosso. Intelligent broadcasting in mobile ad hoc networks: three classes of adaptive protocols. *EURASIP J. Wirel. Commun. Netw.*, 2007:25–25, January 2007.
4. Fei Dai and Jie Wu. Performance analysis of broadcast protocols in ad hoc networks based on self-pruning. *IEEE Trans. Parallel Distrib. Syst.*, 15:1027–1040, November 2004.
5. S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20:374–387, April 1998.
6. Tang K. S. & Kwong S. Man, K. F. Genetic algorithms: Concepts and applications (in engineering design). *IEEE Transactions on Industrial Electronics*, 43:519–534, 1996.
7. Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, MobiCom '99, pages 151–162, New York, NY, USA, 1999. ACM.
8. Vilmos Simon, Márton Bérces, Endre Varga, and László Bacsárdi. Natural selection of message forwarding algorithms in multihop wireless networks. In *Proceedings of the 7th international conference on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, WiOPT'09, pages 16–22, Piscataway, NJ, USA, 2009. IEEE Press.
9. E. S. Varga, B. Wiandt, B. K. Benko, and V. Simon. *Biologically Inspired Networking and Sensing: Algorithms and Architectures*. IGI Books, 2012.
10. Brad Williams and Tracy Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, MobiHoc '02, pages 194–205, New York, NY, USA, 2002. ACM.
11. Jie Wu and Hailan Li. On calculating connected dominating set for efficient routing in ad hoc wireless networks. In *Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications*, DIALM '99, pages 7–14, New York, NY, USA, 1999. ACM.