A Framework for Configuration and Assembly of Routing Protocols for Wireless Ad Hoc Networks

Rafael Pereira Pires
rafaelpp@lisha.ufsc.br

Antônio Augusto Fröhlich
guto@lisha.ufsc.br

LISHA - Laboratory for Software and Hardware Integration
UFSC - Federal University of Santa Catarina
PO Box 476 – 88049-900 – Florianópolis, SC, Brazil
http://www.lisha.ufsc.br/

Abstract

A great number of routing algorithms for wireless ad hoc networks were proposed. Each one shows some advantages in specific situations. In this work, we present a system where it is possible to assemble a routing protocol by choosing and configuring some of the many proposed strategies. We drawn two distinct scenarios with different parameters and predefined requirements. Then we analyse the choice and configuration of the routing algorithm over the framework based on the expected results from the application's point of view. Our results show that our system and strategy bring benefits in the sense of producing personalized protocols, that adequately fits in the target deployment environment.

1 Introduction

Dozens of routing algorithms for ad hoc wireless networks were proposed. Usually, their efficiency is demonstrated through comparisons with previous approaches, where the protocols are simulated with a certain configuration parameters in a set of different scenarios. In this way, caused by great number of options, difficulties in the moment of defining the protocol for a certain application are generated. Targeting to ease the search for the protocol that best fits in the deployment scenario of a specific application and that have the best performance under the desired measures, we propose, in this work, a system where is possible to choose the strategies that potentially bring the best benefits from the application's point of view and then, to configure, to test and reconfigure them. This leads to the quick generation of routing protocol instances that can be compared and have their configuration parameters fine tuned until the creation of a customized protocol to an application.

Our system is built over a metaprogrammed framework, originally created to ease the selection of strategies and generate lightweight protocols instances for parallel computing. We have adapted the framework to the routing protocols for wireless ad hoc networks domain and factored some of their main strategies in order to generate complete protocols.

The idea is, based on the well defined deployment scenario and application's requirements and expected results, to select the different strategies of routing algorithms found in literature and start a process of configuring, measuring, comparing and adapting the protocol's parameters until getting the one that best fits to the user expectations. We describe two distinct scenarios and define, from the very beginning, their requirements and expected results. Then we select the strategies that would fit those requirements, based on previously known strategies characteristics. After that, we could run and compare, customize it and get to the final protocol.

The next section describes our system and presents some strategies used in routing algorithms for ad hoc wireless networks. Section 3 presents the scenarios used as base to our protocol adaptation to the target applications. The section 4 shows our achieved results. In section 5 we draw the conclusions and give our perspectives of continuing this work.

2 Aggregating Routing Strategies to form a Protocol

The metaprogrammed framework of lightweight protocols [1] combines a series of metaprograms based on generative programming [2]. It aggregates basic communication strategies, like: fragmentation, reliable delivery with confirmation and flow control. By simply selecting the desired strategies to a certain application, it generates the lightweight protocol ready to run on myrinet networks over the EPOS operating system [3]. Our system uses the infrastructure made available by that framework, although we work on a higher level, with more complex protocols, that have some data structures and whose main objective is to forward messages to the right destination.

The framework is compounded by:

- a set of strategies that will be joined to form a protocol;
- a baseline architecture or base-kernel, that corresponds to the device driver where some pointcuts are inserted in order to the strategies subroutines be called;
- a protocol generator responsible to combine the strategies and generate the composite protocol; and
- a configuration repository where the configuration parameters and strategies selection are stored.

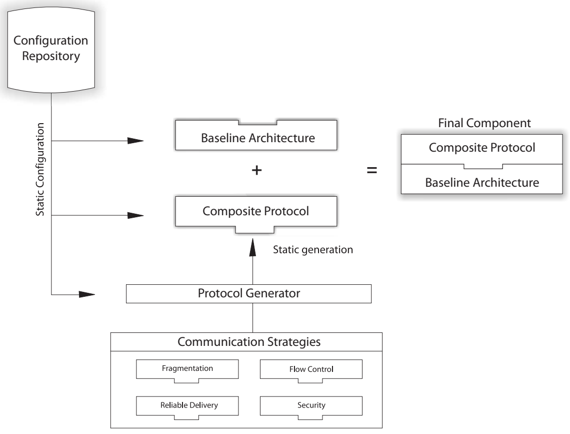

Figure 1. Metaprogrammed Framework and the relation between its structures

Figure 1 illustrates the relations between them. The key feature of being a metaprogrammed framework is that the final object code will contain only the select code, with the minimum required to the protocol to work as it is expected to. By static configuration and generation we mean that those steps are solved in compilation time, instead of carrying that overhead to running time. That feature is specially useful for dedicated applications, where the deployment scenario and the expected results are previously known.

The first routing algorithms used in the ARPANET, forerunner of the Internet, could be divided into two classes: *distance vector* and *link-state*. The former corresponds to the distributed Bellman-Ford algorithm [4, 5], where each node stores the next one that a message should be forwarded in order to reach a specific destination. The latter consists in each node having to create, based on the periodic advertisement from its neighbors, a complete map of the network and then, using some shortest path graph algorithm, they compute the best route to each destination.

Primarily, they both used to make periodic broadcasts in order to inform their neighbors what was their view of the network and to forward other nodes' view. To this strategy, known as proactiveness, a counterpart was proposed, known as reactiveness, where the routes are queried just when they are request by some node, avoiding the overhead of maintaining routes to all possible destinations. In the next subsections we describe the strategies we selected to compound the routing protocols through the metaprogrammed framework.

2.1 Flooding Strategy

Flooding is a routing technique where each node forward every message to all its reachable neighbors until it gets to the destination. In order to control the potentially infinite number of retransmissions, means of limiting it are necessary. One way could be the use of a time to live field, being decremented in every retransmission and, when it reaches zero, the message would no more be forwarded. Another way is to uniquely identify each flood message by putting in its header a sequence number that, together with the sender address, gives the flood id, allowing the message to be discarded by nodes that have already forwarded it. Those characteristics of flood control are configurable features that can be selected to compound the protocol.

Although it could waste resources because every reachable node on the network will receive every flood message, flooding is useful to compare with other strategies. It is also used as part of reactive protocols, when network wide queries are needed.

2.2 Proactive Strategy

The proactive strategy is responsible in making periodic advertisements and forwards of routing information, in a way that every node are able to find out the best route to all possible destinations, sometimes limited to a specific area. The advertisements interval is a configuration parameter.

Proactive protocols presents less delay when a route is requested, because they have already defined it previously. In contrast, there are a waste of resources in the advertisement messages. The overhead needed to maintain routes to all possible destinations may not be compensated when the nodes' application send messages to just a few destinations.

2.3 Reactive Strategy

In the reactive strategy a request is made when a route is solicited by the application. When the message gets to the destination or some other node that have a recent route to it, a route reply message is sent back to the original sender.

The delay when a route is requested is greater, because it takes some time to query the route. But the need of periodic transmissions is nonexistent, saving bandwidth.

2.4 Distance-Vector Strategy

The distance-vector strategy manages the routing table, keeping information about which node to forward when some message arrives destined to another one. The configuration parameters embraces the timeouts that control stale routes and which metrics to consider in order to compare the weight of routes.

Although every strategy can be turned on and off, there are some dependencies and incompatibilities that have to be observed. For example, when the Reactive Strategy is enabled, the Flooding Strategy has also to be, because the queries use flood messages. The Flooding Strategy works as a routing protocol by itself, as it delivers the message to the destination while the Distance-Vector Strategy have to be combined with the Proactive or Reactive Strategy.

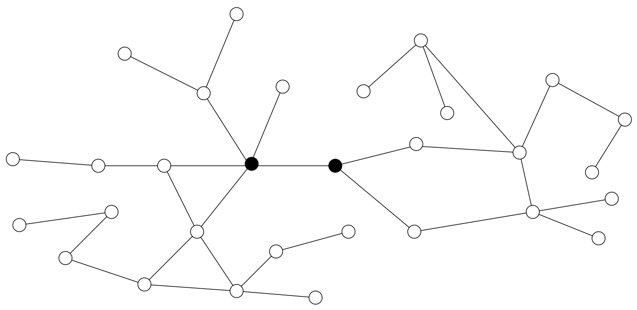

Figure 2. Scenario 1 Topology

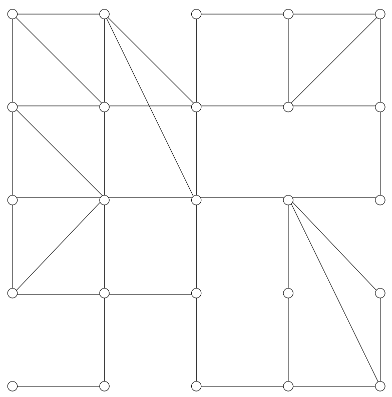

Figure 3. Scenario 2 Topology

3 Case Studies

In order to test our system we ran two distinct applications and scenarios. The first one would be a set of sensors scattered in a forest aiming to measure natural phenomena, even for environmental monitoring or forecasting. The sample rate of each sensor is around five times a minute and the messages are short. Their data have to be centralized in a server for future processing. A set of nodes acts as gateway between the wireless sensor network and the server. All measures should be sent to one of them. The nodes are battery driven and their lifetime should be maximized. Each node is geographically static. The network is composed by 30 nodes. Figure 2 shows the network topology.

The other case occurs in a meeting room. Each participant have its own portable device. And they should be able to get in touch with everyone else. Although pronunciation from one person for all another is possible, the major part of time they have discussions between groups, in order to deliberate some specific subject. The length and frequency of communications here are greater, since they can exchange files and multimedia data. Each device's radio range may not be symmetrically distributed in all directions. Figure 3 illustrates the network topology.

4 Evaluation

We implemented the Baseline Architecture (section 2) over unix sockets in order to simulate our system. Each process represents a node, listening on a different UDP port. The network topology was represented by its connectivity graph, with every outgoing message being able to be seen by all adjacent nodes, as would occur in a real single channel wireless environment.

The first application was implemented sending periodic messages to its gateway node. The time interval varied randomly within 9 and 12 seconds to simulate non synchronous behavior. The message size was of 100 bytes. The second one starts with a coordinator node sending a set of messages to every other node. After that, all nodes start to communicate bidirectionally among small groups. The initial message corresponds to 1024 messages of 1KB each, and the peer communications rounds of 10 messages of 500 bytes.

We reined the above described applications changing the selection of the routing protocol under it. The combinations were:

- Flooding Strategy alone;
- Reactive, Flooding and Distance-Vector Strategies combined, resulting in a similar protocol to AODV (Ad Hoc On demand Distance-Vector) [6]; and
- Proactive and Distance-Vector Strategies combined, giving a protocol like DSDV [7].

As it is an ongoing work, our prototype is still being developed. However, we could notice some patterns considering the strategy features and the applications behavior. In the first application, due to the fixed destinations - the gateways -, the routing table when using the second combinations of strategies was dramatically smaller than the proactive approach. The flooding strategy, as predicted, made a lot of unnecessary forwards, that could cause quick battery discharge. Such overhead would only be justified on applications where almost every communication is broadcast, like the first phase of the second application, or when mobility is so great that any protocol but flooding would be quick enough to deal with the topology changes.

From the second application we could see that routing tables of both proactive and reactive strategies were almost the same, as communication between every peer occurred. The differences in the delay after requesting a route on proactive or reactive strategies were not significant, because in both scenarios there was no topology changes and new route requests in reactive approach was not necessary.

5 Conclusions and Future Work

We presented here some insights of statically combine routing strategies for ad hoc wireless networks. This approach could bring benefits in getting the best protocol to a specific application.

From our experiment, we could get hints that the reactive strategy presented better performance for both application scenarios we choose. We also saw, from the second scenario, that flooding could not waste much resources when delivering broadcast messages, which lead us to also consider dynamic protocols, to adapt in applications with multiple behaviors.

We plan to give robustness to our implementation and add a lot more strategies, like hierarchical, geographic and multicast routing protocols, to cover a wider range of scenarios. Future simulations should also include scenarios and protocols that give support for mobile nodes.

References

[1] T. R. C. dos Santos and A. A. Fröhlich, “A customizable component for low-level communication software”, in *19th International Symposium on High Performance Computing Systems and Applications*, May 2005, pp. 58–64, Guelph, Canada. IEEE.

[2] K. Czarnecki and U. Eisenecker, *Generative Programming: Methods, Tools, and Applications*, Addison-Wesley, 2000.

[3] A. A. Fröhlich, *Application-Oriented Operating Systems*, GMD Research Series. GMD - Forschungszentrum Informationstechnik, Sankt Augustin, Germany, Aug. 2001.

[4] R. Bellman, *Dynamic Programming*, Princeton University Press, 1957.

[5] L. R. Ford and D. R. Fulkerson, *Flows in Networks*, Princeton University Press, 1962.

[6] C. E. Perkins and E. M. Royer, “Ad-hoc on-demand distance vector routing”, in *2nd Workshop on Mobile Computing Systems and Applications. WMCSA*, Feb. 1999, pp. 90–100, New Orleans, USA. IEEE Press.

[7] C. E. Perkins and P. Bhagwat, “Highly dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for mobile computers”, *SIGCOMM Computer Communication Review*, vol. 24, no. 4, pp. 234–244, 1994.