

# GP-Pro: Protocol Generator Based on User Specifications for QoS Routing in Mobile Ad Hoc Networks

Pedro E. Villanueva-Peña  
SITE, University of Ottawa  
Ottawa, ON  
K1N 6N5  
1-613-562-5800 ext. 2228  
pvillanu@site.uottawa.ca

Thomas Kunz  
SCE, Carleton University  
Ottawa, ON  
K1S 5B6  
1-613-520-5727  
tkunz@sce.carleton.ca

## ABSTRACT

Mobile Ad Hoc Networks (MANETs) are self-configuring networks where nodes are mainly mobile and communication is achieved wirelessly and in a multi-hop fashion. Many routing protocols for MANETs have been proposed but only a few of them have been reliably implemented for deployment. The mobile nature of nodes establishes constraints in terms of resource consumption (e.g. bandwidth, energy) and creates a highly dynamic network topology. Therefore, finding paths in MANETs is not a trivial task. In addition, the constantly increasing and scenario-dependant network requirements in terms of robustness, reliability and quality of service for multiple platforms demand fast development and implementation of bug-free routing protocols that satisfy specific user and network requirements. However, current practices for protocol development and implementation are costly and time-consuming, especially when existing implementations are not properly reused. This paper discusses the steps that should be followed in order to build the GP-Pro protocol generator, which is based on Generative Programming[9], for automatic generation of ad hoc routing protocols according to user requirements and also discusses how the generated protocols could support QoS. GP-Pro is designed to be extensible, with the explicit goal of generating a large number of different protocols by different component combinations.

## 1. INTRODUCTION

Mobile ad hoc networks (MANETs) are infrastructureless wireless networks where the topology changes constantly, due to node mobility and environmental conditions. Therefore, routing becomes a non-trivial task. Multiple ad hoc routing protocols have been proposed. However, only three of them have reached the RFC status (AODV[4], OLSR[2] and TBPRF[3]). Ad hoc routing protocols can be classified as proactive or reactive. Proactive (table-driven) protocols maintain routing tables, at all times, with paths toward every network destination. On the other hand, reactive (on-demand) routing protocols only attempt to discover routing paths when a destination node has to be reached. A complementary classification for ad-hoc routing protocols that considers node location in order to make routing decisions is position-based protocols. However, no position-based protocol has reached the RFC status.

Constantly increasing network requirements in terms of bandwidth, robustness, reliability and quality of service for a

broad variety of networking scenarios ask for the development of more robust routing protocols that satisfy more demanding user and network conditions. However, the process of designing, developing, testing, evaluating and implementing stable routing protocols is a very costly and time-consuming process that does not satisfy the constantly increasing demands, especially when no previous implementation solutions are reused. Therefore, development from scratch is not longer an option and new mechanisms to support faster development with reasonable performance are needed. Function libraries appear as a first alternative; however, usually they not provide specialized enough functions to satisfy new requirements. Protocol development frameworks (e.g. The X-kernel [6]) that define the main protocol architecture are another alternative. However, there is still a lot of work that has to be done by the protocol developer and also, the number of frameworks that have been especially developed for ad hoc routing protocols is limited (e.g.[7]). Component-based engineering is another alternative where fine-granularity components are used to build the protocols (e.g. [1, 8]). However, there still exists the need to select and interconnect the components. A more attractive alternative is Generative Programming (GP) [9], which makes use of components and is also powered with the knowledge to automatically assemble them. In addition, the selection of components is based on user specifications, which are expressed by means of a proprietary specification language or a Graphic User Interface (GUI). GP tremendously reduces the development time, and the built-in knowledge considerably decreases the probability of errors introduced by the protocol developers. GP strongly supports the concept of automatic generation given that a language to specify user requirements exists and that the protocol generator can understand it. Therefore, in this paper we discuss in detail the steps that should be followed to build the GP-Pro protocol generator, which is based on generative programming, to automatically generate ad hoc routing protocols based on user specifications and we also discuss how the generated protocols could support QoS by applying different routing metrics based on node and link state information. GP-Pro is designed to be extensible, in terms of being able to continuously add new components with the explicit goal of generating a large number of different protocols by different component combinations. GP-Pro envisions the automatic generation of proactive, reactive and position-based routing protocols ready for implementation. Finally, as far as we know this is the first time that the

generation of position-based protocols is addressed by any protocol generation tool.

This paper is organized as follows: First the architecture of GP-Pro is discussed. Next, the corresponding domain analysis is presented followed by an example of the QoS version of a well-known protocol that could be generated using our protocol generator. Finally, some discussion is elaborated and the future work is addressed.

## 2. GP-PRO

One of the main reasons to design GP-Pro is to speed up the protocol generation process based on user requirements while achieving reasonable protocol performance. Here, the meaning of speeding up the generation process is not only related to quickly generating the source code for the new protocol. It is also related to the mechanism to specify the required protocol. Therefore, an easy to use but powerful mechanism is required. Such a mechanism should be capable of processing the simplest specification, which is: “Generate any ad hoc routing protocol”, but also capable of processing a much more advanced specification. In terms of the protocol specification, GP-Pro will support two different mechanisms to provide the user specification. The first mechanism, which is more user-friendly, is a GUI (Graphic User Interface) that does not require any knowledge about any programming or specification language. Only the selection of protocol features by means of lists and check boxes is required. The second mechanism is a XML-style user specification based on a proprietary *Domain Specification Language (DSL)*. This alternative mechanism provides an opportunity for the more advanced user that wants to specify a more specialized protocol. The entire protocol generator will be built inside a Java application. Therefore, the application will provide all the required functionalities to handle both types of user specifications and to track the changes experienced by the user specification during the generation process. Even though the application is written in Java, the source code of the generated protocols will be C/C++ code for better protocol performance.

Figure 1 shows a feasible architecture and processing flow for the protocol generator. We decided to make use of a XML-based DSL for describing the protocol to be generated. During the entire protocol generation process the original user specification may experience changes. However, it will be available for the user at anytime in the XML-based format. This way the user can fully understand the entire generation process. In Figure 1, the processes *0.1-Domain Analysis*, *0.2-Domain Design* and *0.3-Components Implementation* (part of *Domain Implementation*) are not elements of the processing flow. These processes correspond to the three phases of Domain Engineering [10], which allows creating families of applications based on the commonalities of their members. Therefore, domain engineering will be used to generate the components for the routing protocols. Regarding the rest of the processes in the architecture, processes *1.0-User Specification* and *2.0-Specification Validation* represent the provision of a user specification using the proprietary XML-based *DSL* and the validation on the content and structure of it (according to a XML schema) respectively. A simple example of a protocol specification is next.

```

<protocol>
  <type=proactive/>
  <path determination>
    <metric>maximum link quality</metric>
  </path determination>
</protocol>

```

This example shows that each protocol will be described by several features with different levels of specificity (represented by indentation) and that the amount of chosen features might vary. Processes *3.0-GUI* and *4.0-Specification Generator* are to be performed when the GUI is utilized, process *3.0* represents the use of the GUI (powered by component configuration knowledge) to describe the required protocol by means of list boxes, check boxes and text fields; and process *4.0* is in charge of generating the corresponding specification in the format of the XML-based *DSL*. Process *5.0-Buildability checking* will make sure that the required protocol can be assembled (i.e. requested components do not overlap nor are in conflict). In case that additional protocol features or configuration parameters are required, process *6.0-Completing Specification* will do the job. Using the complete specification as a guideline, process *7.0-Component Selection* will select the components that will build the new protocol. Finally, process *8.0-Components Assembly* will generate the protocol source code by automatically assembling the components according to the protocol architecture and the component interconnection model. Process *9.0-Additional outputs* will allow the generation of additional outputs such as any kind

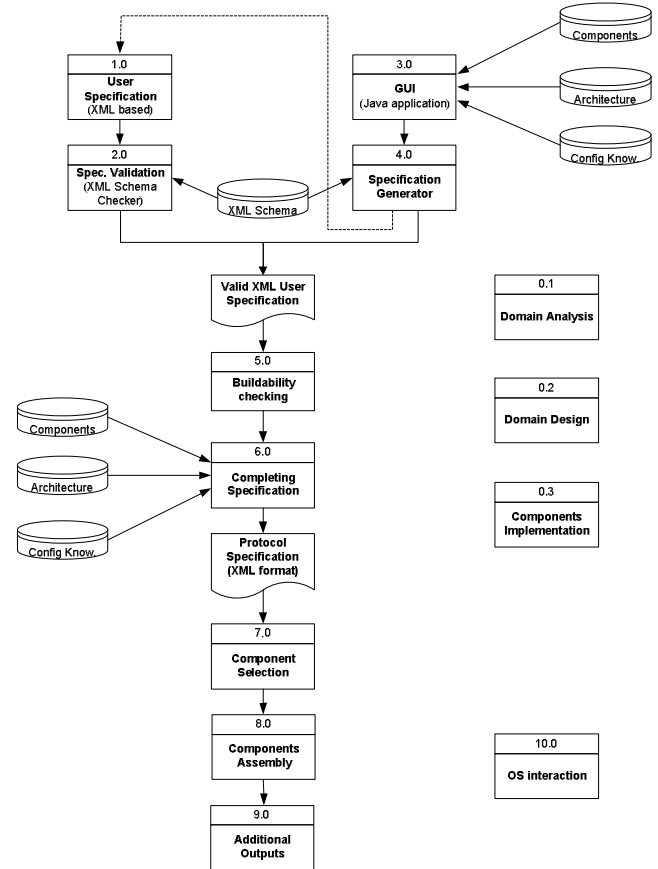


Figure 1. GP-Pro Architecture

of documentation about the architecture of the generated protocol. The last process, *10.0-OS Interaction*, which is not part of the processing flow either, provides an interaction layer with any chosen operating system. It provides platform independency.

### 3. DOMAIN ANALYSIS

The generator domain represents the common underlying structure of broadcasting and unicasting routing protocols for IP-based mobile ad hoc networks. GP-Pro envisions the generation of proactive, reactive and position-based protocols. The protocol generator domain is modelled by abstractions (that we refer to as components) for collecting, distributing, storing and processing routing information that will be utilized to determine the “best” (according to the chosen metric) existing path towards any routing destination. The set of main components, which is expected to satisfy every ad hoc routing protocol requirement, is identified by the corresponding Domain Analysis. Each of these main components might be constructed by  $n$  subcomponents ( $n \geq 0$ ), also called first-level subcomponents, and each subcomponent might be constructed by  $m$  sub-subcomponents ( $m \geq 0$ ) also called second-level subcomponents. There are no limitations on the number of subcomponent levels that each component can be broken into as long as they are compatible and satisfy the expected functionality of the main component.

*Process 0.1 - Domain Analysis* in Figure 1 is the first step in domain engineering. It identifies the systems that belong to the family and the commonalities and variabilities across the domain. The result of it is a feature diagram, which captures the important properties of the domain. It is at the feature level where decisions can be made to define particular members of the ad hoc routing protocols family. Figure 2 shows the corresponding feature diagram of GP-Pro. The root element of the diagram represents the domain or concept; the leaf nodes represent its features. The solid circles on top of the features indicate mandatory features; the empty circles indicate optional features. A filled arc connects or-features and open brackets indicate an open feature (which can be replaced without affecting any other feature). Each protocol feature, representing a component in the system, is described next.

The *MADINI* (*Manager for distribution of network information*) keeps control of the information that is distributed over the network (i.e. one-hop neighbours), how often and which node specific information is to be included. It is essential for proactive protocols. The *delivery mechanisms* (i.e. broadcasting) define the way to forward any control packet ready for transmission. The *CONI* (*Collector of network information on-demand*) resembles the route discovery process of reactive protocols and is responsible for obtaining any kind of information required by the protocol and that is expected to be available in the network. It is essential for reactive protocols. The *additional computations* feature represents any particular computation (i.e. MPR computation [2]) that might be essential for a particular protocol. The *OS Interface* provides an interface between the routing protocol and the operating system (i.e. Linux). The *type of addressing* defines the unique identification mechanism that will be used by the network nodes. The *path determination* feature computes the routing paths based on any chosen metric. The *RIR* (*Route Information Repository*) receives and stores routing

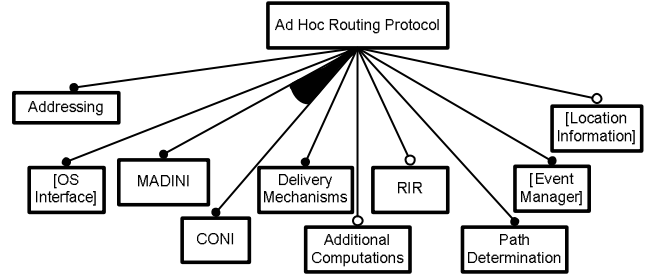


Figure 2. Feature Diagram

information coming from the routing protocol. Its structure is variable and depends on the protocol. The *Event manager* provides a mechanism for scheduling tasks that will be launched after a certain period of time (i.e. expiring processes). Finally, the *location information* feature is in charge of providing the location information (if needed) of the node itself (i.e. GPS).

If providing QoS is a protocol requirement, there are four components where the required support can be accommodated. In the *MADINI* node and link state information (e.g. battery power, bandwidth) to support QoS aware metrics for path determination can be proactively distributed by adding the corresponding subcomponents. However, if the QoS supportive information will be used on-demand, the *CONI* is the component where subcomponents should be added. To take advantage of the distributed or collected information, QoS aware metrics have to be provided as part of the *Path Determination* component. Finally, *additional computations* to support QoS could be added as well.

### 4. QoS PROTOCOL EXAMPLE

Many ad hoc routing protocols have been proposed in the literature. However, only three of them have reached the RFC status (AODV[4], OLSR[2] and TBPRF[3]). In most cases, to accomplish best-effort routing is the prime goal of routing protocols when they are originally proposed. However, sometimes extensions are proposed, later on, in order to support QoS. A clear example is the well-known OLSR protocol which is fully described in [2] and QoS extensions were later on proposed on [5]. OLSR is a table-driven, proactive, link-state routing protocol that periodically broadcasts topology information (TC messages). OLSR optimizes link advertisement and message broadcasting by relying on MPRs (Multipoint Relay nodes). Node neighbourhood information (Hello messages) is also exchanged to support the shortest path algorithm for path determination.

The authors in [5] propose a QoS aware version of OLSR (QoS OLSR) which selects routes of “best available bandwidth” instead of shortest path routes. Bandwidth is measured as the amount of time a node monitors an idle channel and is available to transmit (node’s idle time). In order to generate QoS OLSR, with GP-PRO, starting from an OLSR implementation, the following changes should be made. First, the two subcomponents in the *MADINI* that generate *Hello* and *TC messages* have to be replaced by similar components that also advertise node’s idle times. An *additional computation* to compute Bandwidth (BW) from idle times has to be added and the MPR Selection

subcomponent has to be replaced for a similar subcomponent that gives preference to links of better BW. Finally the *Path Determination* subcomponent has to be replaced for the Extended BF[5] algorithm that selects paths of best available bandwidth. Figure 3 shows the full configuration of QoS OLSR in GP-Pro (shaded components are not part of the configuration).

## 5. DISCUSSION

The process of designing, implementing and deploying routing protocols for ad hoc networks is non-trivial, costly, time-consuming, error-prone and constrains the development of routing protocols capable of satisfying a broad variety of user and network requirements for multiple scenarios and multiple platforms. Additionally, the implementation from scratch of every new protocol without significantly reusing existing implementations makes the process even longer and more complex. GP-Pro is presented as an automatic protocol generator that speeds up the protocol development process by reusing existing components. GP-Pro provides a flexible specification mechanism that supports the simplest and the more advanced user specifications while generating full protocol implementations ready for deployment. Such a mechanism has to be extensible in order to incorporate future user and network requirements. Therefore, any new requirement could be fulfilled by providing GP-PRO with the required new component variation. A clear example of new requirements was presented with QoS OLSR where the exchange/addition of a few subcomponents generates a completely different and QoS aware routing protocol. In terms of QoS, multiple protocols can be generated by using different metrics (i.e. node energy, connectivity, delay, throughput, security, jitter, loss rate, etc.) in order to determine the best routing paths.

## 6. FUTURE WORK

Currently we are working on the development of the first set of components that will allow the generation of the first ad hoc routing protocol with GP-PRO. From there we will continue with the development of the proprietary specification language and the GUI. Right after, we will focus on providing efficient solutions to perform each remaining process in the GP-Pro architecture until achieving the automatic generation of routing protocols. Additional components to increase the range of routing protocols that can be generated will be also created. Finally, what will constitute a success for GP-Pro will be the capability to generate a broad range of protocol variations, which are also competitive with hand-crafted protocol implementations. Competitive here does not necessarily mean that the generated protocols perform better than the hand-crafted protocols. However, we expect to achieve a reasonable trade-off between performance and generation-time. Obviously, the time to generate protocols by using GP-Pro will be minimal.

## 7. REFERENCES

[1] E. Kohler, R. Morris, B. Chen, J. Jannotti and M. F. Kaashoek. *The Click modular router*. ACM Transactions on Computer Systems, Vol.18, No.3, pp.263-297, August 2000.

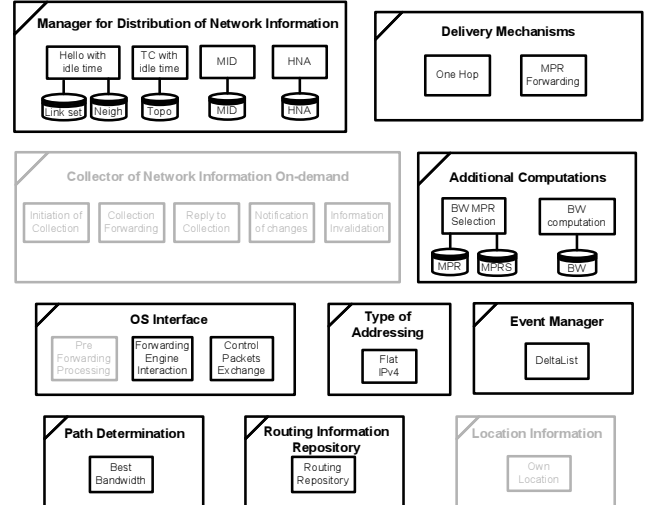


Figure 3. QoS OLSR Protocol Structure

- [2] C. Adjih, T. Clausen, P. Jacquet, A. Laouiti, et al. *Optimized Link State Routing Protocol*. IETF: The Internet Engineering Task Force, October 2003, RFC 3626.
- [3] R.G. Ogier, M. Lewis and F.L. Templin. *Topology Dissemination Based on Reverse-Path Forwarding (TBPRF)*. IETF: The Internet Engineering Task Force, February 2004, RFC 3684.
- [4] C.E. Perkins and E.M. Royer. *Ad hoc On-Demand Distance Vector Routing*. In Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA, February 1999, pp. 90-100.
- [5] Y.Ge, T.Kunz, L.Lamont, *Proactive QoS Routing in Ad-Hoc Networks*, 2nd International Conference on Ad-Hoc Networks and Wireless, Montreal, Canada, October 8-10, 2003.
- [6] N.C. Hutchinson and L.L. Peterson. *The x-kernel: architecture for implementing network protocols*. IEEE Trans. Software Engineering, Vol. 17, No. 1, pp. 64-76, January 1991. J.
- [7] Allard, P. Gonin, M. Singh and G.G. Richard III. *A User Level Framework for Ad hoc Routing*. In Proceedings of the 27th Annual IEEE Conference Local Computer Networks, pp. 13-19, Nov. 2002.
- [8] I. Wang, L. Benmohamed, S.D. Jones, C. Liu and T. Saadawi. *Component-Based analysis and design of routing protocols for mobile ad hoc networks*. In Proceedings of the 24<sup>th</sup> army science conference (ASC 2004), Orlando, Florida, USA, Dec 2004.
- [9] K. Czarnecki and U.W. Eisenecker. *Components and Generative Programming*. ACM SIGSOFT, 1999.
- [10] J.C. Cleaveland. *Program Generators with XML and Java*. Prentice-Hall, 2001.