Using the HINT network emulator to develop opportunistic applications

Antoine Auger, Gwilherm Baudic, Victor Ramiro, Emmanuel Lochin

Demo: Using the HINT Network Emulator to Develop Opportunistic Applications

Antoine Auger, Gwilherm Baudic, Victor Ramiro, Emmanuel Lochin
Institut Supérieur de l'Aéronautique et de l'Espace (ISAE-SUPAERO)
Université de Toulouse, 31055 Toulouse Cedex 4, France
{first.second}@isae.fr

ABSTRACT

In this work, we show how to use HINT, a real-time event-driven network emulator, to support the development process of opportunistic applications. In this demo, we use this emulator in conjunction with an example Android chat application to demonstrate its features.

Keywords

DTN, Emulation, Opportunistic networks, Architecture

1. INTRODUCTION

Developers of applications running over opportunistic networks, must not only deal with network characterization, but also with its impact on their application. Indeed, the complexity to evaluate the performance before a real world deployment is still an open issue.

Simulations help to gain insight in the network behavior [2, 3, 4], but they do not support real nodes, and typically focus on network performance. Testbeds provide an almost real world feedback, but they are heavy and expensive to deploy. Currently there is no tool to help during the development process.

For this purpose, we propose HINT [1], a new hybrid emulation system for opportunistic networks, where nodes can be either real or virtual. Thanks to the support of Android devices as real nodes, it aims to help developers to conceive and test their opportunistic applications before moving to a real-world deployment.

In this demo, we show the development process of an opportunistic application from development, test and performance evaluation perpectives. We show three different use cases for HINT: How to connect an application to an opportunistic network, to test the application using the HINT emulator and finally to easily test different application scenarios using the HINT monitoring system.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

2. HINT IN A NUTSHELL

In this section, we present the main details of the HINT architecture. HINT is a lightweight real-time event-driven emulator meant to fit into existing development environments.

We define two interaction levels: the real world and the emulated world. In the real world, *real nodes* (i.e., Android mobile devices) run applications to be tested, while in the emulated world, both *virtual nodes* and *real nodes* interact in an opportunistic way. HINT defines nodes' interaction between all nodes at the emulated world, and applies changes in real-time according to contact opportunities. *Real nodes* can only communicate with the emulator (at the real world level), and not directly with each other. Hence, we ensure that all connections go through HINT. Several network topologies can be drawn, according to the considered user scenario.

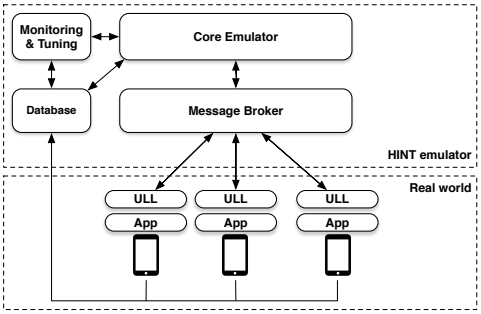
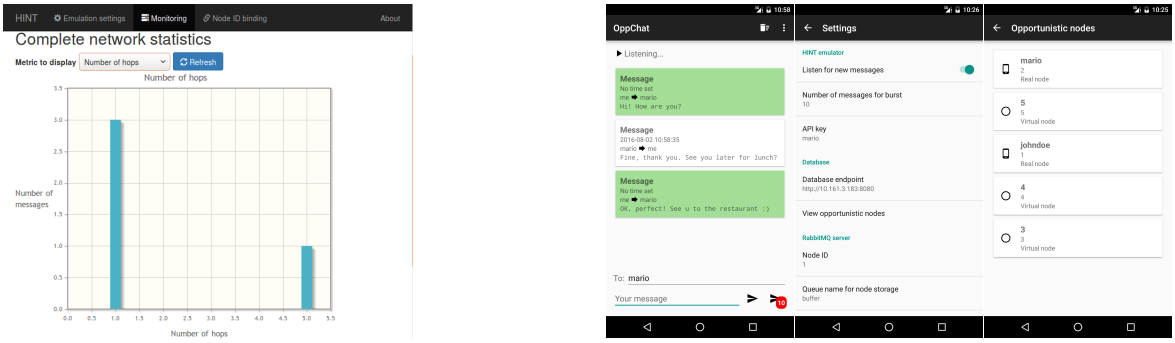


Figure 1: HINT network emulator architecture.

Figure 1 shows the main architecture. HINT is organized around the event-driven Core Emulator which runs the experiment scenario. Events define contact or intercontact durations and message management (creation, replication, forwarding, etc.). The Message Broker allows the communication between each pair of nodes (real or virtual). It is used to represent node buffers and store messages. The User Link Layer is deployed on the real nodes as an Android service, to act as an abstraction layer between the emulator and the application. This makes HINT transparent to the application being tested. A Cross-layer Monitoring and Tuning system, implemented with a web interface, allows to follow the experiment in real-time and adjust parameters. Finally, a Database stores the characteristics of each node, along with the data required for the Monitoring system.



(a) (b)

Figure 2: A set of screenshots: (a) shows the web-based monitoring interface for HINT; (b) shows different views of the OppChat Android application.

3. DEMO OVERVIEW

We now present the validation of the HINT emulator. Our scenario defines an opportunistic network of 50 nodes, split into 47 virtual nodes and 3 real nodes (Android devices). Pairwise contact and intercontact events are generated according to an homogeneous exponential distribution. All nodes route their messages using the forward protocol. Real nodes both deploy an opportunistic chat application and the HINT user link layer. The chat application can either broadcast messages to all connected nodes or deliver a private message to a specific node.

HINT application development: first, we demonstrate how to bind an existing application to the HINT emulator using the User Link Layer package. We develop OppChat, an Android opportunistic chat application. OppChat takes advantage of our User Link Layer service to seamlessly exchange messages with other nodes through the HINT Core Emulator. Indeed, to send a chat message the application just needs to call the appropriate User Link Layer method. To receive messages, the application declares a standard Broadcast Receiver subclass that appropriately filters and handles incoming chat messages. When messages are forwarded to a node within the HINT Core Emulator, they are placed into Message Broker queues. Then, the broker fires a notification to the User Link Layer of the corresponding real node. Finally, the message is consumed from the broker by the User Link Layer and passed to the application by means of a Broadcast Intent to be finally displayed in OppChat.

HINT validation: the goal of this experience is to test the correctness of the HINT emulated connections with both emulated and real nodes. We deploy OppChat on real nodes (see Figure 2b). When this chat application must broadcast one message to all other opportunistic nodes, one message is created for each node in the emulator. When a message is sent from OppChat, the User Link Layer transmits it to the broker using a TCP connection. We show that the network connection delay is negligible compared to the contact and intercontact durations tested. Messages are routed among nodes within HINT until their delivery to the final destination where they are displayed at the chat log window. In this special validation, we can send messages between two or more real nodes connected to HINT. Since there is no direct connection between the real nodes but rather an emulated connection with the HINT Core Emulator, once a message is sent it will traverse the emulated opportunistic network before arrival to all other real nodes. This allows developers to test the interaction of the application on different scenarios of emulated opportunistic networks.

HINT monitoring: finally, we demonstrate the use of the HINT Monitoring system. HINT provides a real-time monitoring interface for different levels of granularity: metrics for the whole emulated opportunistic network, pairwise-based metrics and a node status view. In this experiment, a real node sends a private message to another real node thanks to the OppChat application. We set nodes' buffer size limit to 20 bytes and tail drop buffer policy. To show the impact on node's buffer size and its drop ratio, we intentionally send "message bursts" (10 copies of the original message). We then monitor the global drop ratio, as well as the hop number distribution (see Figure 2a).

4. CONCLUSION

In this demo, we presented the use of HINT to develop, test and monitor a simple opportunistic application. The aim of HINT is to help bridge network characterization and real application development. Differently from a simulator, our proposal is designed to handle both real and virtual nodes in the same setup. Compared to testbeds, our system has low hardware requirements (one computer and some Android devices) and may be easily deployed in development environments.

Acknowledgments. This research was supported in part by the French Ministry of Defense through financial support of the Direction Générale de l'Armement (DGA).

5. REFERENCES

[1] G. Baudic, A. Auger, V. Ramiro, and E. Lochin. Using emulation to validate applications on opportunistic networks. Available: http://arxiv.org/abs/1606.06925, June 2016.

[2] X. Chang. Network simulations with OPNET. In *ACM WSC'99*, 1999.

[3] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena. Network simulations with the ns-3 simulator. *SIGCOMM demonstration*, 14, 2008.

[4] A. Keränen, J. Ott, and T. Kärkkäinen. The ONE simulator for DTN protocol evaluation. In *Proceedings of the 2nd international conference on simulation tools and techniques*. ICST, 2009.