Using Triggers for Emulation of Opportunistic Networking

Per Hurtig
Department of Computer
Science
Karlstad University, Sweden
per.hurtig@kau.se

Tanguy Pérennou
LAAS-CNRS & ISAE
Université de Toulouse,
France
tanguy.perennou@laas.fr

Johan Garcia
Department of Computer
Science
Karlstad University, Sweden
johan.garcia@kau.se

Anna Brunstrom
Department of Computer
Science
Karlstad University, Sweden
anna.brunstrom@kau.se

ABSTRACT

Opportunistic networks do not require the availability of an end-to-end path, but may instead take advantage of temporary connectivity opportunities. Opportunistic networks pose a challenge for network emulation as the traditional emulation setup where application/transport endpoints send/receive packets from the network following a black box approach is no longer applicable. Instead opportunistic networking protocols and applications need to react to the dynamics of the underlying network beyond what is conveyed through the exchange of packets. In order to support emulation evaluations for such challenging applications we in this paper introduce the concept of emulation triggers that can emulate arbitrary cross-layer feedback and that are synchronized with the emulated scenario. The design and implementation of triggers in the KauNet emulator are described. The use of triggers in the context of opportunistic networking is briefly sketched.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Modeling Techniques; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*

General Terms

Performance, Experimentation

1. INTRODUCTION

Opportunistic networks are a form of networks that opportunistically use network or other resources as they become available. In contrast to traditional networks, opportunistic networks do not require an end-to-end path to be available between the communicating application end points. It may instead rely on intermittent connectivity where some form of mobility is the typical cause of the availability/unavailability of communication opportunities. One of the most well known examples of opportunistic networks is Delay/Disruption Tolerant Networks (DTN) [4, 1]. The DTN architecture [1] defines a message-based overlay, the bundle layer, that can operate over a collection of networks of different types and which can each use its own protocol stack internally. DTNs may be characterized by occasional connectivity, high and variable delays and asymmetric data rates. Oppnets [7] is another example of opportunistic networks, in which a small seed network is deployed and then opportunistically expands itself to include additional nodes and resources as needed. Oppnets thus do not only use opportunistic communication opportunities, but the network is also opportunistically enlarged in order to acquire the resources necessary to carry out a specific application task. Even if an end-to-end path exists, additional network resources may also be opportunistically used to enhance application performance. A cellular user may for instance opportunistically take advantage of temporarily available WLAN networks to boast communication performance [8].

Evaluating the performance of any communication system is challenging due to the complexities and number of variables involved. Performance can be evaluated by several metrics and at several levels of abstraction ranging from analytical evaluation, via simulation, experiments in an emulated environment, up to full scale live experiments. As opposed to analytical modeling and simulation, the emulation approach uses a mixture of real entities and abstractions. For many performance evaluation tasks, IP-level emulation is an attractive approach, since it allows parts of the evaluated system to be real entities capturing all inherent complexity in them, while other parts of the system may be abstracted to a degree, and the behavior of these abstracted parts are emulated. In comparison to real live tests, emulation is typically less expensive to perform and produce more easily reproducible results. Due to these advantages, well known IP-level emulators such as Dummynet [10] or NetEm [6] have been extensively used by the networking research community. Several emulators specifically targeted for wireless or mobile systems such as MobiNet [9] and NCTUns [12] have also been developed.

Examining the literature on opportunistic networking re-

veals that very few studies in this field are based on IP-level emulation. (One exception would be the evaluation in [3].) Opportunistic networks pose a difficult challenge for IP-level emulation, as the traditional end-to-end communication path is no longer present and the nodes in the network need to react to dynamically appearing communication opportunities. Hence, the typical network black box approach used in IP-level emulation, where application/transport endpoints interact with the network only by sending and receiving packets, may no longer be applicable. Instead many opportunistic networking protocols and applications need to react to the dynamics of the underlying network beyond what is conveyed through the exchange of packets. In this work we examine emulation support for opportunistic networking configurations that are dependent on a cross-layer information flow. We have extended the KauNet emulator [5] with support for so called triggers that can emulate arbitrary cross-layer feedback and that is synchronized with the emulated scenario. We describe the architecture and implementation of triggers and sketch their use in opportunistic networking scenarios.

2. EMULATING CROSS-LAYER INFORMATION EXCHANGE

This section describes the KauNet network emulator, and the new trigger functionality that enables KauNet to emulate e.g. cross-layer information for evaluation of opportunistic network scenarios.

2.1 KauNet Overview

KauNet is an extension to the well-known Dummynet emulator [10]. By using high-resolution patterns, KauNet enables deterministic and fully repeatable emulation of effects like packet loss, bit-errors, bandwidth changes, and delay changes. The KauNet patterns can be used to emulate a certain effect either in a time-driven or a data-driven mode. In the time-driven mode, emulation effects can be applied with millisecond granularity. Alternatively, in the data-driven mode effects can be applied with packet granularity.

Using the FreeBSD `ipfw` command, so called pipes can be configured to carry specific flows. KauNet patterns, which are created ahead-of-time, can then be assigned to a specific pipe in order to apply the desired emulation effects on the pipe's traffic. KauNet achieves this by traversing the patterns as experimental traffic enters the corresponding pipe (data-driven mode) or as time goes by (time-driven mode). If multiple effects are to be emulated at the same time it is necessary to provide one pattern for each type of emulation effect, i.e. packet loss, bit-errors, bandwidth change, and delay change. To simplify emulation of multiple interrelated effects, several patterns can be combined into one emulation scenario. A simple experimental setup, involving KauNet, is shown in Figure 1.

By providing this fine-grained control over emulation effects, KauNet enables fully reproducible and deterministic emulation-based experiments. The KauNet system is also very flexible with regards to the origin of emulation patterns, which can be created from analytical expressions, collected traces, or even simulations. Further details on KauNet is available in [5].

2.2 Triggers

Triggers in KauNet can be seen as a general information passing functionality that can be used to deliver precisely positioned control information to applications or protocols during emulation run-time. As for the general patterns, the trigger patterns can be either data- or time-driven, according to what is being emulated.

While the trigger mechanism is not tied to any particular type of control information, it is reasonable to assume that some types of information will be more prevalent in emulation scenarios involving opportunistic networking. One such example is upward flowing cross-layer information. For example, triggers allow emulation of both link conditions as well as cross-layer information in cases where connectivity is intermittent and the link layer has the ability to inform upper layers about the presence or absence of connectivity. In such a case, bandwidth patterns can be used to control connectivity and trigger patterns, synchronized with the bandwidth patterns, are used to generate the upwards flowing connectivity information that for a real link would come from the link layer.

A general view of a simple emulation setup using triggers is shown Figure 2. In this setup two hosts are connected using a KauNet-enabled host. The KauNet host emulates the conditions of the particular link or network that is being emulated by means of bandwidth change patterns, packet loss patterns or others, as appropriate. These patterns control the behavior of the KauNet host only. The trigger pattern is, just as any pattern, located at the KauNet host. In contrast to other pattern types, however, triggers are often relevant to other hosts than the KauNet host. Triggers might, for instance, signal connectivity information that should be available to a protocol or an application at Host A. Thus, in addition to trigger patterns, a trigger communication module and an adaptation layer is needed to convey and use trigger information accordingly.

2.3 Trigger Patterns

Since KauNet already provides a pattern framework, it is natural to reuse it for implementing and encoding the semantics of triggers. The KauNet pattern framework provides the means to create pattern files of arbitrary length. The actual pattern data is represented by compressed short values (i.e. $0 - 65535$). For triggers the implication is that no more than 65535 mutually exclusive trigger values can exist. As the trigger values that are inserted in a pattern are under the control of the user, it is possible to generate triggers leading to arbitrary complex behavior.

As mentioned, the KauNet pattern framework allows patterns to be either time-driven or data-driven. In time-driven mode, the emulation effect of a pattern is applied on a per millisecond basis. Similarly, with data-driven patterns emulation effects can be applied on a per packet basis. The maximum resolution of a trigger pattern is therefore one millisecond or one packet.

2.4 Trigger Communication

Since KauNet is implemented in the FreeBSD kernel, triggers must be conveyed to external processes to be useful. Otherwise, only kernel space processes running locally on the KauNet host would be able to benefit from the trigger functionality. Thus, we need a mechanism to transfer the trigger value of a fired trigger to an arbitrary receiver. By
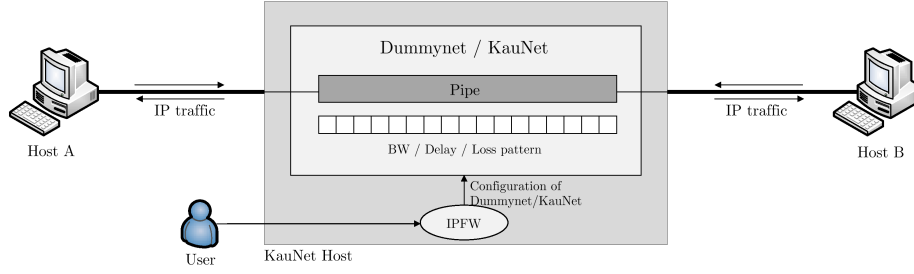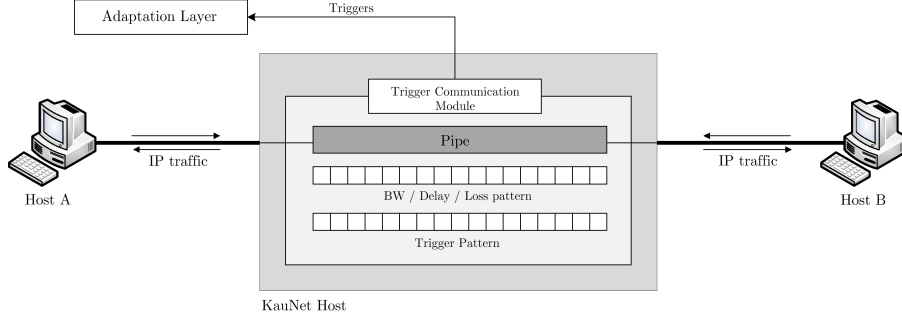
Figure 1: Simple KauNet Setup.

Figure 2: KauNet Trigger Overview.

arbitrary, we mean that the recipient of a trigger should be able to reside in either user space or kernel space, locally or on another host.

This functionality is implemented by the trigger communication module, shown in Figure 2. This module has a number of responsibilities. First, to enable both local and non-local communication the module uses an UDP interface. Using this interface, adaptation layers can register themselves to receive triggers from a certain pipe. The trigger communication module keeps a list of all registered adaptation layers to enable multiple adaptation layers to subscribe to the same trigger. Second, whenever a trigger is fired within KauNet the module is invoked to transmit this information to the registered adaptation layers. The communication of triggers is also done using UDP. Finally, the trigger communication module also provides the means for adaptation layers to unregister themselves.

It is important, for an experimenter, to make sure that the trigger communication and experimental traffic is separated, e.g. by using different network interfaces. Otherwise, experimental results might be affected by the trigger communication, or triggers might be lost as they are transported using UDP. Thus, it is important to provide a separate control network for the triggers, to guarantee a reliable and timely operation. Our initial experiences show that the trigger passing is reliable and delays are negligible.

2.5 Adaptation Layer

In general, an adaptation layer is a process, or part of a process, that is able to interface with the trigger communication module and thereby receive triggers. In the context of opportunistic networking, triggers are likely to contain some sort of cross-layer information. The role of the adaptation layer is then to work as an interface between the trigger com-

munication module and the upper layer consumer of cross-layer information. In case the emulation is used to examine an application or protocol implementation that is already using cross-layer information, it is the role of the adaptation layer to reshape the semantics-free triggers embedded in the trigger pattern into the specific type of cross-layer information that is used by the application or protocol that is evaluated. One example could be an ad hoc routing protocol that uses RSSI (Received Signal Strength Indication) as one factor to make the routing decisions. For such a case, the adaptation layer would take the role of translating the trigger values and hooking into the mechanism that reports RSSI to the application (for example by controlling part of the proc file-system if that is the mechanism used to get RSSI information).

3. EMULATION OF OPPORTUNISTIC NETWORKS

In this section we discuss two examples of how to emulate opportunistic networks using triggers. In the first example, experimentation nodes effectively run the Bundle Protocol [11], for instance the DTN2 reference implementation [3, 2]. One of the most distinctive features of DTNs is the intermittent connectivity between nodes. Over time, contact opportunities arise and allow the forwarding of data bundles towards their final destination. Those opportunities are mainly characterized by a time window and the contactable peer ID. Opportunities may be predictable or not, according to the considered application. In some cases several links allow for the contact between two peers, in which case there are several contact opportunities, one per available link for the contact. Each node running the DTN2 implementation has a component that manages contact opportunities, as well

as optional components providing contact discovery.

The solution proposed here mainly consists in using triggers to emulate contact opportunities. It implies the development of a new Announce/Discovery mechanism and subclasses for so-called Opportunistic Links in the DTN2 reference implementation. On each DTN2 node, these classes would be in charge of detecting contact opportunities by connecting to the KauNet trigger communication module and extracting contact information from the triggered data, thus constituting the trigger adaptation layer. Bundle forwarding would be unchanged, and carried out over Ethernet on the experimental network according to the announced opportunities and the local DTN2 node configuration. The implementation is considered for future work. Whether contact opportunities are predictable or not, hooking into the contact manager of DTN2 using triggers and a custom Announce/Discovery adaptation layer allows to emulate many different flavors of DTNs; the only difference lies in the content of the contact triggers. Thus, using KauNet triggers to emulate such scenarios combines the use of real implementations and the full control over network characteristics, a combination that is impossible to achieve using simulation or live experiments.

It is also possible to emulate scenarios where multiple contact opportunities are available simultaneously. For instance, consider a mobile node (MN) equipped with both 3G and IEEE 802.11 interfaces. Further, suppose the MN is communicating with a corresponding node using its 3G interface. If the MN moves into a WLAN access network (AN), it could opportunistically switch to its IEEE 802.11 interface for increased performance.

This scenario could easily be modeled using two time-driven bandwidth patterns: one for the 3G AN and one for the WLAN AN. Let us suppose that the bandwidth provided by the 3G AN is constant over time. The bandwidth available through the WLAN AN, however, increases as the MN approaches the access point, and then decreases when the MN moves away. By synchronizing these bandwidth patterns with a trigger pattern that is designed to send triggers informing the MN of the AN with most available bandwidth at all times, opportunistic handovers could easily be investigated. This setup allows e.g. application performance to be evaluated over different link characteristics, as modeled by the emulation patterns.

4. CONCLUSIONS

By extending the pattern-based KauNet emulation system with pattern-driven triggers it is possible to emulate opportunistic communication scenarios that depend on cross-layer information. The triggers can be tightly controlled and accurately synchronized with other emulation effects. The triggers are distributed to local and remote processes by the trigger communication module, where the adaptation layer is responsible for translating the triggers into application specific semantics and actions. Two usage scenarios, DTNs and opportunistic handovers, were discussed to highlight the applicability of the triggering mechanism. It is our hope that the proposed triggers will allow researchers to perform accurate and reproducible evaluations of application/protocol performance over opportunistic networks in a cost and time efficient way, and thereby contribute to the further advancement of the field.

6. REFERENCES

[1] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. Delay-Tolerant Networking Architecture. RFC 4838 (Informational), Apr. 2007.

[2] Delay Tolerant Networking Research Group. DTN Reference Implementation. Available at: `http://dtnrg.org/wiki/Code`.

[3] M. Demmer, E. Brewer, K. Fall, S. Jain, M. Ho, and R. Patra. Implementing delay tolerant networking. Technical Report IRB-TR-04-020, Intel Research, December 2004.

[4] K. Fall. A delay-tolerant network architecture for challenged internets. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–34, New York, NY, USA, 2003. ACM.

[5] J. Garcia, E. Conchon, T. Pérennou, and A. Brunstrom. KauNet: improving reproducibility for wireless and mobile research. In *MobiEval '07: Proceedings of the 1st international workshop on System evaluation for mobile platforms*, pages 21–26, San Juan, Puerto Rico, 2007.

[6] S. Hemminger. Network Emulation with NetEm. In *Proc. of the Linux Australia Conference (linux.conf.au 2005)*, Canberra, Autralia, April 2005.

[7] L. Lilien, Z. H. Kamal, and A. Gupta. Opportunistic networks for emergency applications and their standard implementation framework. In *Proceedings of 17th International Conference on Database and Expert Systems Applications (DEXA '06)*, 2006.

[8] L. Ma, F. Yu, V. Leung, and T. Randhawa. A new method to support vertical handover between WLAN and UMTS networks using SCTP. In *Proceedings of VTC03*, Orlando, FL, USA, 2003.

[9] P. Mahadevan, A. Rodriguez, D. Becker, and A. Vahdat. MobiNet: A Scalable Emulation Infrastructure for Ad-Hoc and Wireless. Technical report, UCSD, June 2004.

[10] L. Rizzo. Dummynet: a simple approach to the evaluation of network protocols. *SIGCOMM Comput. Commun. Rev.*, 27(1):31–41, 1997.

[11] K. Scott and S. Burleigh. Bundle protocol specification. RFC 5050, Internet Engineering Task Force, November 2007.

[12] S. Y. Wang and Y. B. Lin. NCTUns Network Simulation and Emulation for Wireless Resource Management. *Wireless Communications and Mobile Computing*, 5(8):899–916, December 2005.