Empirical Performance Evaluation of Data Dissemination Mechanisms for Spot Applications

Alaeddine El Fawal
EPFL Middle East

Jean-Yves Le Boudec and Adel Aziz
School of Computer and Communication Sciences

Ecole Polytechnique Fédérale de Lausanne (EPFL)
{alaeddine.elfawal, jean-yves.leboudec, adel.aziz}@epfl.ch

ABSTRACT

We evaluate, by measurements on a real testbed, the performance of networking services required for spot applications. We call a "spot application" an application for smartphones (such as "Ad-hoc Flash Sales") that disseminates information in a local neighborhood through hop-by-hop wireless forwarding. The dissemination environment is challenging. It is open-ended, and highly dynamic with very limiting resource constraints. We define five success criteria that are required by a spot application to ensure a sustainable dissemination. They cover issues such as adaptability, resource conserving and co-existence with TCP applications. We evaluate a package of mechanisms and we identify how those, through their interaction, can altogether meet these criteria. This package includes flow control, forwarding factor control and buffer management. Our measurements are carried out on a realistic testbed composed of 50 wireless devices. Our metrics include spread, application rate, forwarding factor and delay. Some findings are: blind mechanisms can interact among each other to meet the success criteria; aging entails intolerable processing complexity for smartphones; and TTL-based buffer management performs poorly with large buffer size.

1. INTRODUCTION

We call "spot applications" a family of mobile applications that interact with their surrounding and disseminate information locally using hop-by-hop forwarding. Hence, a spot application makes a spot in the network, which corresponds to all wireless devices that receive its information. The nodes within an application spot are either the final destination or they may act as relays to deliver the information later to its final destination when they meet it. With very limited ad-hoc networks, e.g. in one building, the spot could be the entire network. In contrast, the spot is a very small part of a network in an open ended-environment such as the population in town or on the highway. Spot-applications could be a free alternative to costly cellular services. Typical applications are traffic information dissemination and social interactive applications (similar to Mobiclick [1] and Friends [3]).

In this paper, we consider "Ad-hoc Flash Sales", a typical spot application that we implemented for smartphones (Windows Mobile) as well as for laptops. This is a way to promote commercial products, usually joined with discounted pricing. A vendor uses this application to inject its flashes that describe its products. The potential customers receive the flashes on their smartphones, which semalessly contribute in the dissemination through hop-by-hop forwarding of the flashes .

Ad-hoc Flash Sales has the same spirit of "Mobile Concierge" by Cisco and IBM [2], with the difference that this later requires an infrastructure and flashes are limited to in-store customers. Ad-hoc Flash Sales is infrastructureless with open-ended dissemination environment. The spot is intended to cover, but it is not limited to, the surrounding geographical area. The dissemination environment shows new characteristics and challenges. Because of the open-ended networks, flashes have the potential to travel with people across cities. According to the spread-rate trade-off explained in [5], this too wide dissemination implies a very low application rate. Therefore, the dissemination should be controlled. The network is highly dynamic; It is formed of people in constant movement with very short contact time. The network might be shaped into disconnected islands. Inter-islands dissemination occurs with people moving among them. Such an application should take into account the limited resources of smartphones and their short battery lifetime. Adding to all these challenges, the high competition among the vendors who flood the network by flashes promoting their products.

We define five success criteria for a sustainable dissemination. They are adaptability, dissemination control, resource conserving, co-existence with TCP applications and fairness. Our goal is to come up with a package of mechanisms that, through their interaction, can altogether meet these criteria. We consider only blind mechanisms: they do not need or exploit topology information and do not implement handshaking. A node controls data dissemination based merely on its local information and observation of the network. The rational behind this blind assumption is to avoid control messages and handshaking overhead. Evaluating "non-blind" mechanisms is out of the scope of this paper. We consider broadcast-based communication because flashes are destined to all surrounding nodes. We survey

all relevant mechanisms that exist in the literature and meet the aforementioned blind assumption. These mechanisms cover functions such as flow control, forwarding factor control and buffer management. We evaluate their performance through real measurements carried out on a realistic testbed. It is composed of 50 mobile wireless devices. The scenarios that we measured include highly dynamic with short contact time, disconnected islands, congested, and scenarios where nodes apply power saving mechanisms. We indicate the mechanisms that are required to meet the success criteria. We explain how to tune them and we propose improvement to maintain sustainable performance in all circumstances. Surprisingly, a package of blind mechanisms adapts well to all considered scenarios and meet the success criteria thanks to their interaction among each other; the behavior of one mechanism is constrained by the use of the other mechanism. Our metrics are detailed in Sect. 4.4. Some examples are the forwarding factor, which is the average number of times a node forwards a packet; the spread, which is the average number of nodes that receive a copy of a given packet; and the application rate and the packet arrival delay.

2. SUCCESS CRITERIA

In this section, we define the success criteria that deployed mechanisms should meet in order to maintain a sustainable dissemination of Ad-hoc Flash Sales.

2.1 Adaptability

Ad-hoc Flash Sales faces a diversity of scenarios. Therefore, used mechanisms should be adaptive in order to ensure good dissemination in all circumstances. In particular, we consider the following:

Highly dynamic: Nodes are moving and constantly making new encounters. They should be able to forward flashes several times (to new encounters) while avoiding redundancy.

Short contact time: Nodes might meet for a short contact time. Therefore, they need to react quickly upon new encounters so that they do not miss this dissemination opportunity.

Disconnected islands: Nodes moving among disconnected islands should be able to carry packets from one island to another.

2.2 Dissemination Control

This is due to the aforementioned open-ended environment. Increasing the application rate occurs at the expense of the spread [5]. We recall that the spread is the average number of nodes that receive a copy of a given packet. On one hand, a node should adapt its application rate to the forwarding capacity of the network. Otherwise, nodes will not be able to forward its packets and they will drop them. On the other hand, the spread should be controlled in order to keep a sustainable application rate. This suggests a self-limiting dissemination that adapts its limit according to the network conditions.

2.3 Resource Conserving

Power saving: Smartphones might apply power saving mechanisms on the WIFI interface to extend the battery lifetime. When the wireless interface is ON, a node should be able to get the missing flashes that circulated when it was OFF.

Bandwidth saving: With Ad-hoc Flash Sales, the medium might be shared among tens of nodes. Therefore, redundant transmission should be avoided.

Processing capacity saving: Ad-hoc Flash Sales targets smartphones. They are well-known for their limited processing capacity. Therefore, mechanisms that require extensive processing should be avoided.

2.4 Co-existence with TCP Mobile Applications

While using Ad-hoc Flash Sales, people should be able to use other network services such as web surfing and email checking. However, Ad-hoc Flash Sales uses either UDP or raw sockets (see Sect. 4.1). Therefore, if its rate is not controlled, it kills all TCP traffic, including web surfing and email check.

2.5 Fairness

Arbitrating among competing packets for transmission at a given node should give the same chance to all existing flows in a network.

3. BUILDING BLOCKS

In this section, we describe the building blocks that are needed to meet the success criteria. Each block consists of one or several alternative mechanisms that we consider throughout our evaluation. A success criterion might be achieved through the interaction of several blocks (see Table 1).

	Adaptability	Dissemination control	Resource conserving	Co-existence with other applications	Fairness
Forwarding factor control	X	X	X	X	
Flow control	X	X		X	X
Buffer management	X	X	X		

Table 1: Building blocks interact among each other to achieve the success criteria.

3.1 Forwarding Factor Control

Recall that the forwarding factor is the average number of times a packet is forwarded by a given node. The forwarding factor control aims at saving bandwidth. It avoids redundant transmissions. Different approaches to forwarding factor control are proposed in the literature. They differ in being adaptive or not, and in the information they need

about the topology and neighbors. In this paper, we do not consider mechanisms that require handshaking or that are not adaptive.

In the literature, we find only two mechanisms that are blind and adaptive: counter based [7] and virtual-rate (VR) [5] based forwarding factor control. We implemented only the latter as it is a generalized version of the former, and it allows for multi-forwarding of a given packet. Further, we designed a new mechanism, that we call New Encounter Detection. When combined with the VR based forwarding factor control, it results in considerable gain.

Virtual-rate (VR) based forwarding factor control: With VR-based forwarding factor control, a packet in the dissemination buffer is retransmitted with a probability that depends on its "virtual rate" (`vRate`); it is equal to $c_0 a^{\texttt{rcvCount}}\ b^{\texttt{sendCount}}$ where c_0 is a constant (inverse of a time), `rcvCount` [resp. `sendCount`] is the number of times this packet or a duplicate was received [resp. sent] and a and b are unit-less constants less than 1. Thus, the virtual rate of a packet decreases exponentially with any send/receive event on the same packet. Upon any send/receive event on a given packet, its `vRate` is updated and its `earliestSendTime` is now equal to: `currentTime` $+ 1/$`vRate` . A packet becomes "eligible" for transmission when `currentTime` $\leq$ `earliestSendTime` . Hence, a packet in the dissemination buffer, which has seen many send/receive events, is scheduled at a very low rate and it is more likely that it will be dropped by the buffer management before being transmitted (see Sect. 3.3). The constant c_0 is equal to a fraction of the nominal rate of the MAC layer in [packets/s].

New encounter detection: We designed this mechanism to improve the performance of VR based forwarding factor control. The contact time is too short in highly dynamic networks. When a node meets a new neighbor, it should take advantage of this new encounter and schedules all packets in the dissemination buffer for transmission. Otherwise, it might loose this dissemination opportunity. However, the virtual-rate based forwarding factor control does not detect new encounters, although this information is available using local observation of the network. Hence, we propose the following New Encounter Detection (NED) mechanism. A node implements a FIFO list where it inserts the new node Ids of packets it receives. Upon receiving a new node Id, a node considers it as a new encounter and it schedules all packets in the dissemination buffer for transmission. In order to avoid bursts and to minimize redundancy, the `earliestSendTime` of the newly scheduled packets is computed as follows:
`earliestSendTime` $=$ `currentTime` $+ U$, where U is a random variable uniformly distributed over an interval of $10s$. A duplicate of a scheduled packet might be received before its transmission because a neighbor has detected the new encounter and scheduled it for transmission before. After transmitting a scheduled packet or upon receiving a corresponding duplicate, `earliestSendTime` follows again the usual VR process. Note that the NED mechanism has two parameters to optimize: the FIFO size and the distribution of U. This optimization is kept for future work. Nevertheless, this simple heuristic shows dramatic improvement in performance (see Sect. 5.1.1).

3.2 Flow Control

It deals with the spread-rate trade-off. It controls the application rate in order to ensure wide (multi-hop) dissemination. Further, the flow control should ensure some level of fairness among the competitors. We implement two alternatives:

MAC-based flow control: This mechanism adapts the application rate to the MAC layer rate. At the beginning, an application injects only one packet, which is placed in the dissemination buffer. The packet is "eligible" for transmission and competes with packets received from other nodes. The application is allowed to inject a new packet only when its previous packet is transmitted. Injecting a new packet by the application entails removing its previous one from the dissemination buffer.

Implicit acknowledgment-based flow control: This alternative is described in [5]. It is the only flow control mechanism that is explicitly defined in the literature. It is proposed for epidemic forwarding and it seems to be in complete harmony with the Ad-Hoc Flash Sales context. The packets generated by the application at a given node are placed in the dissemination buffer, where they compete with the other packets for transmission. The application rate is controlled by a windowing system : The number of outstanding packets the application is allowed to have in the dissemination buffer at this node is limited to -at most- 2; a packet is allowed to be deleted from the dissemination buffer when a duplicate is received, which serves as an implicit acknowledgment (Ack). Otherwise, it is deleted after its third transmission to avoid blocking the application.

3.3 Buffer Management

The buffer management drops packets according to predefined criteria in order to keep space for new incoming packets. It plays a main role in limiting the dissemination. Note that the buffer management concerns only packet received from other nodes. A node own packets are treated by the flow control (see Sect. 3.2). We consider three alternative mechanisms:

Time-to-live (TTL): When a packet is created by a source and placed into the dissemination buffer, it receives a TTL value equal to some positive constant "`maxTTL` ". When the packet is accepted for transmission by the MAC layer, the TTL field of the *transmitted* packet is equal to the value of the TTL field in the packet in the dissemination buffer, minus 1. The TTL field of the packet stored in the dissemination

buffer is unchanged.

When a packet created by some other node is received for the first time at this node, the packet is delivered to the application, and the value of the TTL is screened. If it is equal to 0, it cannot be retransmitted and the packet is discarded. Else (TTL$\geq$ 1), the packet is stored in the dissemination buffer, with TTL equal to the value present in the received packet. When and if the packet is later accepted for transmission by the MAC layer, the transmitted TTL field is equal to the stored TTL minus 1, and the stored TTL remains unchanged.

When the dissemination buffer becomes full, the packet with the smallest TTL is dropped to keep space for new incoming packets.

TTL combined with head-drop (TTL_HD): With TTL_HD-based buffer management, the dissemination buffer is assimilated to a queue where new incomers are pushed to the tail. When the queue is full, packets in the head are dropped first. Also, a packet is dropped when its TTL expires. The TTL manipulation is the same as with TTL-based buffer management.

Aging: This mechanism is inspired from [5], although it is proposed in a different context. Every packet in the dissemination buffer has an "`age` " field, which is a fixed decimal positive number less than 256. When a packet is created at a given node, its age is set to 0. When *any* packet is received, the stored age of *all* packets in the dissemination buffer is incremented by K: `age` $\leftarrow$ `age` $+K$. Upon transmission, the age of a packet is rounded and copied to the corresponding header field. When a packet, created by some other node, is received by this node for the first time, its `age` is copied from the corresponding header field. A packet is dropped when its `age` exceeds `maxAge` , a predefined constant.

4. EXPERIMENT FRAMEWORK

In order to evaluate the data dissemination mechanisms, we built a measurement framework. It consists of a modular implementation that groups all the building blocks described in Sect. 3, and a realistic testbed formed of resource constrained wireless devices.

4.1 Modular Implementation

Our implementation is in C++ and it targets Linux like platforms. It consists of two parts: a simple application and a modular static library that delivers a dissemination service to the application. The static library is composed of modules that are mapped in a one-to-one way to the building blocks described in Sect. 3. The combination of mechanisms used in a given experiment is set through a configuration file.

The application creates new packets and delivers them to the static library for dissemination. Also, it receives from the static library new packets received from the network. The maximum application rate is set through a configuration file.

The static library interfaces with the MAC layer through raw sockets. As a source unique Id, it considers the MAC address of the wireless interface. The smallest entity in the dissemination buffer is a "record". It contains a packet and its attributes, which are the following: `sendCount` , `rcvCount` , `age` and `earliestSendTime` .

4.2 Realistic Testbed

In order to create a realistic evaluation environment, we built an experimental testbed for wireless ad-hoc networks. It consists of 50 ASUS WL-500G Premium v1 wireless devices. These devices are very resource constrained. Each device involves a flash memory of 8MB, a RAM of 32MB and a processor running with a clock of 266MHz. We flashed these devices with OpenWrt [4], a Linux-like firmware. Mobility is ensured through adding to a device a plumb battery that lasts for more than four hours when transmitting at full rate and max power.

4.3 Measurement Settings

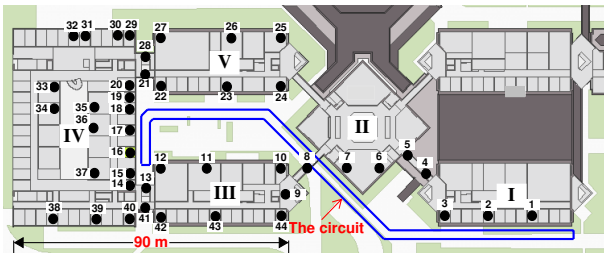


Figure 1: Wireless nodes are distributed over 5 buildings. Each dot corresponds to a node.

The wireless nodes of the testbed are deployed over five buildings, as depicted in Fig. 1: we refer to these buildings as Block I to Block V. The node distribution exhibits different node degrees: the lowest being in Block I with a degree of 2. The highest is in Block IV with a degree around 20. Unless explicitly indicated, the basic parameters of the experiments are as follows: Frame size is of 260 bytes; according to our implementation for a smartphone, this size is enough to describe the product, the brand, to carry comments and ranking. MAC nominal rate is of $1Mbps$, $c_0 = 500\,packets/s$ (almost nominal MAC rate in $packets/s$), $a = 0.1$, $b = 0.01$, `maxAge` = `maxTTL` = 255, $K = 0.1$. The values of c_0, a and b are taken from [5]. All devices are in ad-hoc mode and uses channel 12, which is booked for our experiments to avoid interference. An experiment duration is three hours, only results during the second hour are considered in order to eliminate the transient phase effect. As to the dissemination buffer size, only TTL and TTL_HD are concerned (see Sect. 5.3). We consider small size ($1000packets$) and large size ($10000packets$). We add the suffix SB [resp. LB] to indicate the small [resp. large] size. For instance, TTL_SB indicates TTL with small buffer size.

Unless explicitly indicated, we use the following default mechanisms: TTL_SB, the virtual-rate based forwarding factor and the implicit-Ack based flow control.

4.4 Metrics

- **Forwarding Factor (FF):** the average number of times a node forwards a given packet.
- **Spread:** the average number of nodes that receive a given packet.
- **Received-to-injected packet ratio from node i to node j:** the fraction of packets injected by the application at node i and received by node j.
- **Delay:** the average arrival delay of packets.
- **Application injection rate.**
- **Application Receive rate from node i at node j:** the application receive rate of packets injected by node i, at node j. Note that only packets received for the first time are delivered to the application at the corresponding node.
- **TCP throughput.**

5. MEASUREMENT RESULTS

We evaluate the performance of the building blocks according to the five success criteria.

5.1 Adaptability

Adaptability requires an adequate policy to make forwarding decisions. These decisions are taken mainly by the forwarding factor control module, but they are also infuenced by the deployed buffer management mechanism, as we will see later. In order to assess the adaptability ability of our module, we consider two challenging scenarios: highly dynamic and disconnected islands. A congested scenario will be studied in Sect. 5.3.

5.1.1 Highly Dynamic Networks with Short Contact Time

We want to study the case of highly dynamic networks where nodes join for a short time and leave (see Sect. 2.1). In order to emulate thousands of nodes, we let each node (except the sources) repeatedly start instances of the application for a short time and with a different unique Id each. The duration of one instance is randomly distributed according to a shifted exponential distribution: $SExp(\mu) = \mu + Exp(\mu)$, where μ is set to $60s$. At a given time, a node runs one and only one instance of the application. When an instance returns, the corresponding node creates a new one with a different unique Id. Hence, each instance emulates the presence of a new customer for two minutes on average in the surrounding of a store and his departure later. We refer to an instance as a virtual node. During our experiment, we consider four sources, nodes 5, 43, 36 and 26. Their application rate is fixed to $1packet/minute$.

Fig. 2 shows the spread with three buffer management mechanisms (aging, TTL_SB and TTL_RA_SB) with and without the NED mechanism described in Sect. 3.1. Without the NED mechanism, the three buffer management mechanisms show the same performance. The spread is nearly

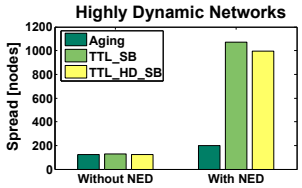
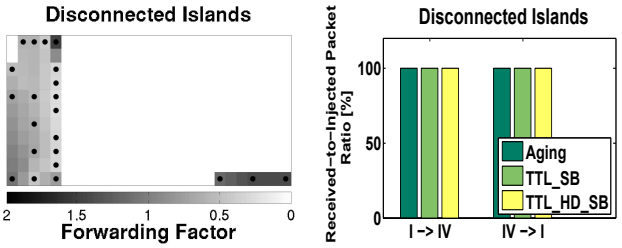


Figure 2: **The spread is averaged over the 4 sources: 5, 26, 36 and 43. Their application rate is fixed to $1packet/minute$.**

triple the number of physical nodes in the network. We repeated the same experiment using the NED mechanism. The NED mechanism increases considerably the spread while controlling redundancy. With TTL_SB and TTL_RA_SB, the spread is increased by 800%, when they exhibit a forwarding factor average of only 2.5. With aging, the spread is increased by only 50%. This is because the aging parameter, K, is large enough to make packet dropping start long time before the dissemination buffer is filled up. Hence, packets live a shorter time and have less of a chance to be disseminated. Note that the obtained large spread with TTL_SB and TTL_RA_SB is still much smaller than the overall number of virtual nodes that appeared in the network. This is because of the spread limiting capacity of the buffer management: when the buffer is filled up, packet drop mechanism starts and dropped packet cannot be disseminated anymore.

5.1.2 Disconnected Islands

In this scenario, we create the "disconnected islands" environment described in Sect. 2. We deploy two disconnected networks: the first is formed of Block I nodes and the second is of Block IV nodes. Then, we let ten mobile nodes move between these two networks according to the circuit depicted in Fig. 1. Each mobile node consists of a student riding a bike and carrying in the back bag a wireless device with a battery. Students were moving in a completely random way. Fixed nodes are injecting packets at a rate of 1 packet/minute. Mobile nodes are acting merely as relays.



(a) (b)

Figure 3: **The application rate of sources is fixed to $1packet/minute$. (a): it shows only active nodes according to the map in Fig. 1. The more dark it is, the larger the forwarding factor is. (b) shows the received-to-injected packet ratio from Block I to Block IV (and vice versa) averaged over all nodes.**

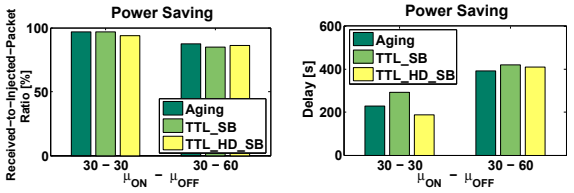
Fig. 3-(b) shows the receive-to-injected packet ratio from one Block to another. With the three buffer management mechanisms, both Blocks were fully connected in time; all injected packets in one block have reached the other one.

Note that, mobile nodes keep the same identities during the whole experiment, which is not realistic. But this does not affect our results, as the delivery delay of a packet from one network to another corresponds to the moving time between the networks.

5.2 Ressource Conserving

5.2.1 Power Saving

In this scenario, we consider only four sources (5, 26, 36 and 43), and other nodes are relays. The sources are assimilated to laptops in stores. Therefore, they do not exhibit power consumption constraints. As to the relays, they can be assimilated to smartphones with power consumption constraints. Therefore, they are running a power saving mechanism: the sleep [resp. awake] intervals follow a shifted-exponential distribution $SExp(\mu_{OFF})$ [resp. $SExp(\mu_{ON})$], already described in Sect. 5.1.1.



(a) (b)

Figure 4: **Nodes 5, 26, 36 and 43 are sources (no power saving) and others are relays. The source application rate is fixed to $1packet/minute$. (a) shows the average of the received-to-injected packet ratios from the 4 sources to all nodes. (b) shows the delay averaged over all packets.**

Fig. 4-(a) shows the received-to-injected packet ratio according to μ_{OFF}. The parameter of the distribution function of the awake interval is fixed to $30s$. The delay is depicted in Fig. 4-(b). As we can see, this power saving mechanism increases the battery life time by 300% (nodes are active one third only of the time), whereas it maintains an acceptable delay.

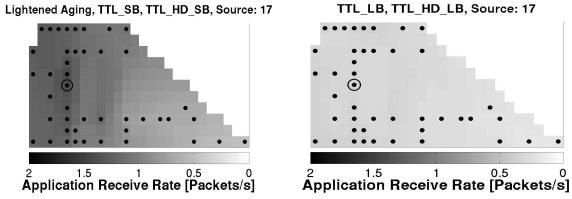
5.2.2 Bandwidth Saving

We consider the scenario of Sect. 5.1.2. Fig. 3-(a) shows the FF and stresses the adaptation ability of the virtual-rate based control: The higher the density is, the smaller the FF is. Although, this forwarding factor control allows multiple transmissions of a given packet, it keeps a very small average of the FF (0.6) while it is able to connect both islands. Nodes in the middle (nodes 14-20) show FF values less than 0.3, due to the high density. This indicates a much better performance compared to flooding.

5.2.3 Processing Capacity Saving

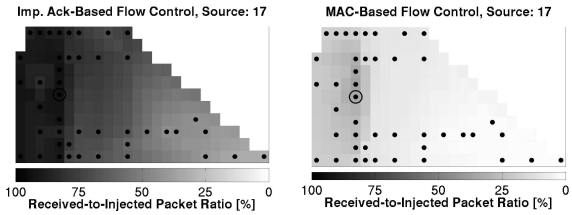
Aging in its original version, increases the `age` of the packets in the dissemination buffer by K at each reception. This entails a processing complexity of $O(n)$. To minimize this complexity, we update the `age` of packets on the basis of 100-packets reception by incrementing them by $100 * K$. The lightened version increases the application injection rate by more than 600%; it passes from 0.2 $packets/s$ with the original aging to 1.3 $packets/s$ with the lightened one. Indeed, the receive socket is served much slower with the original aging due to the processing complexity. Therefore, the receive-socket buffer is always saturated, which results in dropping received frames. Hence, when a node injects a new packet, it does not receive an implicit Ack either because this latter is dropped by the receive socket of the node itself, or because the newly injected packet is dropped by the neighbors at their receive sockets. This blocks the application and decreases its injection rate. With lightened aging, the receive socket is served much faster. Therefore, the receive socket-buffer is less saturated, which minimizes dropping received frames. Only lightened aging is considered in all other experiments; we refer to it simply by aging.

5.3 Dissemination Control



(a) (b)

Figure 5: **In both figures, the dots corresponds to the active nodes according to Fig. 1. Each node runs a greedy application. They show the application receive rate from node 17 to all other nodes. (a) corresponds to lightened aging, TTL_SB and TTL_HD_SB. (b) corresponds to TTL_LB and TTL_HD_LB.**



(a) (b)

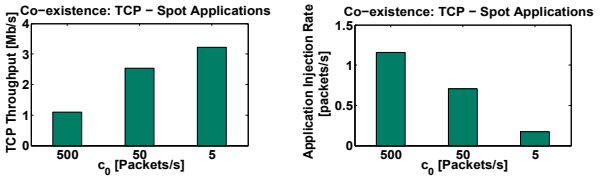
Figure 6: **Both figures show the received-to-injected packet ratio from node 17 to all other nodes. All nodes are active with greedy applications.**

The spread-rate trade-off is controlled through three elements. The first is the virtual rate parameters that control the forwarding capacity of the network (see Sect. 5.4). The second is the dissemination buffer size, which plays an important role in the dissemination performance. With aging, the maximum buffer size is set indirectly: it is equal to $\frac{\texttt{maxAge}}{K}$ [5]. In order to find an adequate buffer size with TTL and TTL_HD, we run experiments with aging with all considered experiments and we adopt the largest buffer occupancy (1000 packets) as the buffer size, which corresponds to TTL_SB and TTL_HD_SB (see Sect. 4.3). Figs. 2, 3, 4 and 5-(a) show that aging, TTL_SB and TTL_HD_SB exhibit the same performance except in highly dynamic scenario for the reason explained in Sect. 5.1.1. The results de-

picted in Fig. 5-(b) met our expectation, increasing the buffer size reduces dramatically the rate.

The third is the flow control. In Fig. 6, we compare the performance of MAC-based flow control to the implicit-Ack based one. The latter adapts the application rate to the forwarding capacity of the network. Therefore, it ensures a larger spread compared to the former: with the former, the dissemination hardly goes beyond the first hop.

5.4 Co-existence with TCP Mobile Applications



(a) (b)

Figure 7: TCP traffic is established only between nodes 35 and 36. (a): shows the TCP throughput. (b) the average of the application injection rates. All nodes are greedy.

In order to enable the user to surf on the Internet while running a spot application, the share of the latter of the bandwidth should be limited. To this end, we proceed as follows: We decrease a node capacity of forwarding spot-application packets and then, we use the implicit-Ack based flow-control to adapt the spot-application rate to the forwarding capacity. According to the virtual rate formula, the maximum forwarding rate of a given packet is $c_0 a$; it corresponds to `rcvCount` $= 1$ and `sendCount` $= 0$. Therefore, the forwarding capacity decreases with decreasing c_0. Fig. 7 shows that the TCP throughput between two nodes exceeds slightly 1Mbps when $c_0 = 500 packets/s$ (nominal MAC rate). While decreasing c_0, TCP throughput increases at the expense of the spot application rate.

5.5 Fairness

Using implicit-Ack based flow control delivers a high Jain's fairness index. It is larger than 0.7.

6. STATE OF THE ART

This is the first paper that considers the specific needs of Ad-hoc Flash Sales and try to fulfill all its success criteria. We surveyed the literature for relevant mechanisms. We consider only environment-oblivious mechanisms [5, 6, 7], as they meet our blind assumption. In [6], a node decides with a fixed probability on the forwarding of a packet that it has received. In [5, 7], the proposed forwarding-factor control mechanisms are adaptive. We consider only the mechanism in [5], which is based on the virtual rate, as it is a generalization of the one in [7], but it allows for transmitting packets several times if needed. Further, the work in [5] is the only one that addresses environment oblivious flow control and buffer management. The difference between their work and ours is threefold:(1) their evaluation method is based on simulation, whereas ours is based on measurements with real implementation that we exported to smartphones and it shows the same performance. (2) They do not consider the success criteria of Ad-hoc Flash Sales, such as power saving and processing capacity limitation. This difference leads to unexpected results: Aging behaves badly on resource constrained devices and it is outperformed by simple mechanisms such as TTL. (3) We proposed new mechanisms in order to improve the performance. Some proposals are the new encounter detection mechanism, TTL and lightened-aging based buffer management.

7. CONCLUSIONS

We evaluate, by measurements on a real testbed, the performance of networking services required for "Ad-Hoc Flash Sales", a typical spot application. We define the success criteria for a such application and we show the building blocks that are required to meet these criteria. We consider only blind mechanisms. We assess the performance of relevant mechanisms and then, we come up with a package of mechanisms that , through their interaction, meet altogether the success criteria. These mechanisms cover functions such as flow control, forwarding factor control and buffer management. Our measurements are carried out on a realistic testbed that we built for this purpose. It is composed of 50 wireless nodes. We consider key network settings such as dense congested as well as sparse scenarios, highly dynamic environments with mobility, and intermittent connectivity. We show that the combination of implicit-Ack based flow-control, virtual rate based forwarding factor control and a simple buffer management, ensures a good balance between the spread and the application rate. Other findings are the following: blind mechanisms get along with adaptability; with TTL based buffer management, a large size of the dissemination buffer harms the performance; aging performs poorly because of its processing complexity; and wireless nodes can apply power saving mechanism while still getting disseminated flashes.

8. REFERENCES

[1] MobiClique. http://www.thlab.net/ apietila/mobiclique/.

[2] Mobile Concierge. http://www.cisco.com/web/partners/pr67/pr30/pr302/mcdemo.html.

[3] Nokia Friend View. http://betalabs.nokia.com/apps/nokia-friend-view.

[4] OpenWrt. http://openwrt.org/.

[5] A. El Fawal, J.-Y. Le Boudec, and K. Salamatian. Multi-hop Broadcast from Theory to Reality: Practical Design for Ad Hoc Networks. In *Autonomics*, Rome - Italy, 2007.

[6] Z. J. Haas, J. Y. Halpern, and L. Li. Gossip-based ad hoc routing. In *Infocom*, pages 1707–1716, 2002.

[7] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu. The broadcast storm problem in a mobile ad hoc network. In *Mobicom, Seattle, Washington, United States, 1999*.