Self-Limiting Epidemic Forwarding

Technical Report LCA-REPORT-2006-126

Alaeddine El Fawal, Jean-Yves Le Boudec, and Kave Salamatian

EPFL, I&C
CH-1015 Lausanne, Switzerland
{alaeddine.elfawal,jean-yves.leboudec,kave.salamatian}@epfl.ch

Abstract— **We define a self-limiting epidemic service as a dissemination service for ad-hoc environments that is broadcast in nature, but is limited to a local scope around each source. Example applications are chatting or bulletin boards in a traffic jam, in an instant crowd in a campus or, in contrast, along a desert highway. Our goal is to support such a service across a wide range of conditions (dense or sparse). The main problems are to adaptively control scoping and traffic rates to avoid congestion. We propose a system design with the following elements: (1) manipulation of TTL by adaptive aging mechanisms; (2) control of forwarding factor by self-inhibition and inter-inhibition and (3) control of rate of injection by sources. We validate the design by an implementation in Java and analyze it using both simulation and ordinary differential equations. We show how it can be tuned to achieve an appropriate balance between limitation of scope and rate of information. Our design is entirely self-organized, and is free of any form of clustering or leader election.**

I. Introduction

A. Self Limiting Epidemic Service

We define the self-limiting epidemic service as a broadcast dissemination service for short messages in ad-hoc environments that is limited to a local scope around each source. A typical application is chatting or a bulletin board among users in a traffic jam, in a railway hall, on a campus, or, in contrast, along a desert highway. In all these cases, the application is broadcast; if there would be no limits in network capacity, node storage or node processing, then an epidemic or flooding mechanism would be thinkable. However it may not be desirable, both from user and capacity viewpoints, to reach all users that can be reached by epidemic forwarding (on a busy hour this may be users spanning hundreds of kilometers along highways). Our goal is to support a self-limiting service across a wide range of conditions (dense or sparse). The scope limit should adapt to the density of node and number of active of users. In a dense environment with 100% active users (users chatting in traffic jam, crowd on campus), the information generated by one user would remain in close vicinity, perhaps reaching order of 100 nodes. In a similar environment but with only a small fraction of active users (i.e. few uses generate fresh information, but many relay), the spread could be much larger in kilometers. Note that in these dense cases, the problem is less a possible lack of connectivity than congestion and information overflow. At the other end of the spectrum, in a sparse environment (desert highway, or area with very small penetration of the system) the service should have a much wider spread. Last, transitions from a dense to a sparse environment may occur without user control, so we would like the service to smoothly adapt to such changes. A key difference with community networking proposals [3], [15] is that we deliberately accept non transitivity [1] as a feature of the service: a user has access to a local bulletin board of messages generated on the fly by those users who happen to be close. We would like the service to evolve smoothly in space, having no clusters, zones or boundaries.

B. Adaptive Use of TTL and Control of Forwarding by Inhibition

There are several proposals for flooding or epidemic services (seeSection V). On top of such services, the two main issues we need to solve are: control of scope and of traffic rate. This needs to be done such as to obtain the appropriate balance between spread, absence of congestion, and probability of delivery. A simple approach to control the scope is to use the TTL (“time to live”[2]) field in IPv4 packets. However, a plain use of this is not adaptive enough to support the various cases mentioned above, as the desired spread in hops depends on user activity (if few users generate traffic, the maximum hop count should be large). This is why we propose to manipulate the TTL field using a combination of three mechanisms, which we collectively call aging (Section II-C). The first mechanism (“adaptive age”) decreases the TTL such that the death rate of packets is roughly proportional to the rate of injection of new packets by the neighborhood (a fixed decrement takes place at every

[1]It may be that A can reach B, B can reach C but A cannot reach C, being too far away.

[2]We use the IPv4 denomination “TTL” even though it is classically used as a hop count.

packet reception). The second mechanism is the classical hop count, and is motivated as usual as a safety mechanism to avoid uncontrolled proliferation. Unlike classical hop count as in [16], it is decreased at every transmission if a packet is transmitted repeatedly by the same node. The third mechanism is the true time to live ("real time age"), introduced as an additional safety mechanism to account for cases of sporadic connectivity where packets can become very old in real time and become irrelevant before exhausting their hop count. In areas with many competing sources, the TTL decreases quickly by adaptive aging, which limits the spread well before either hop count or true time to live expire. Note that adaptive aging contributes both to limiting the spread and to congestion control. We show in Section III and Section IV that this combination is indeed adequate. Further, we show in Section III that even an adaptive mechanism for TTL adaptation is not sufficient. An example is the boundary between a dense, congested area (users chatting in a traffic jam) and a non congested area (users in a lane opposite to the traffic jam). An adaptive mechanism for TTL would tend to give short lifetimes to packets in the former area, in order to maintain useful rates to all users in the traffic jam; thus, a packet that manages to escape the traffic jam area and join the opposite lane would have a short lifetime; this is not desirable as such an escaping packets might have all chances of traveling far away on the opposite lane, assumed to be non congested. It would be better to have a mechanism that limits the spread of packets inside the congested area, while at the same time allowing escaping packets to travel far away. The complement of the age of a packet is carried in the TTL field of IPv4 headers, as explained in Section II-C. For controlling the traffic rate, we need to control both packet reproduction (i.e. the forwarding factor) and injection of fresh packets by sources. For the forwarding factor, we use a self-organized mechanism of inhibition inspired by [4]. Every packet at a node is associated with a "virtual rate", which decreases exponentially with the number of transmissions ("self-inhibition") or receptions ("inter-inhibition"). Self-inhibition avoids wasting transmission resources, while inter-inhibition acts as a global feedback, where a node interprets the reception of a duplicate as some form of acknowledgement. Injection of fresh packets by sources is controlled by a combination of the inhibition scheme, a scheduler and buffer management. Positive feedback to the source is obtained by reception of a packet duplicate, while negative feedback is provided by the inhibition mechanism, which is also applied to sources.

C. Self-Limiting Epidemic Forwarding

In the rest of this paper we present and analyze our proposed system design, called Self-Limiting Epidemic Forwarding (SLEF). The system is for implementation in ad-hoc nodes and does not require support from an infrastructure. The design is self organized, i.e. without central nodes, boundaries or clusters. The goal is to provide some limited spread of packets such that, with high probability, all nodes close to a source do receive all packets generated by this source, while allocating a minimum amount of transmission opportunities per user. The actual spread should depend on many factors, such as the capacity of the network; in order to quantify our design target, we use the spread factor M_s, defined as the total number of transmission events anywhere in the network, caused by a single original packet, in an hypothetical symmetric network with all users active (if a packet reaches order of 100 nodes, and is transmitted in average 3 times per node, then $M_s \approx 300$). M_s defines the load on the ad-hoc network that we accept per original packet. The goal is to maximize the number of nodes that receive a copy of an arbitrary packet, subject to the constraint that the spread factor is upper bounded by M_s.

We assume that packets are atomic application layer entities and that global ordering across sources is either unnecessary or maintained by methods outside the scope of this paper. We assume that nodes are mobile and contact times may be short, so we use a dissemination approach that does not require negotiation of contents. However, the concepts of aging and inhibition that we introduce could easily be extended later to dissemination methods that, like in [16], use explicit negotiations to control packet exchanges.

Our design is described in Section II. Its main features are the combination inhibition (to control the forwarding factor) and limitation of spread by aging. Our assumptions about the MAC layer (unreliable broadcast service) are explained in Section II-E. In Section III we analyze the system by a method of ordinary differential equation classically used for population dynamics models or particle systems (ODEs). We use analysis to tune the parameters, in order to achieve an appropriate balance between limitation of scope and rate of information. We implemented SLEF in Java; in Section IV we use the implementation over the Jist-Swans network simulator to assess the performance. Section V compares to existing work. The main contributions of the paper are :

1) the introduction of a new service semantics
2) a system design based on self-limiting epidemics mechanism (self-inhibition, inter-inhibition, competition based aging); it is self organized, all nodes need only local information, no server, clustering or zoning is required
3) implementation and validation by simulation and by a method of ordinary differential equation.

II. System Design

We first give an overview of SLEF and explain how these goals are achieved; then we describe the main components one by one. We propose values for all design parameters by a simple analysis, except for the inhibition constants a, b (Section II-B) and the adaptive aging constant K_1 (Section II-C), which requires a more in-depth analysis, given in Section IV.

A. Design Overview

Every node maintains one epidemic buffer, used to store received and fresh packets, with the following attributes:

- `sendCount` : how many times this packet was sent by this node.
- `rcvCount` : how many times this packet or a duplicate was received by this node.
- `vRate` ("virtual rate"): this attribute is derived from `sendCount` and `rcvCount` , using the method described in Section II-B. It is the rate at which this packet would be transmitted if it were alone in the epidemic buffer.
- `age` : combines hop count, real time age (true time to live) and adaptive age, which reflects the amount of competition this packet and its ancestors have encountered so far.
- `earliestSendTime` and `pendingSendConfirmation` : see Section II-E and Section II-F.

We call clone the set made of an original packet and its duplicates; all packets in the same clone have the same value for source address and SLEF identification field (Section II-H). When a packet is received, it is inserted into the epidemic buffer. If this is the first time a packet of this clone is seen by this node, a new entry is created, else if the existing entry is still present, it is overwritten (thus there is always at most one packet per clone in the epidemic buffer). The attributes are updated as explained in Algorithm 1. Creation of a new entry may require deletion of existing packets from the epidemic buffer (Section II-H).

The functions achieved by SLEF can be classified as (1) management of the spread of packets , (2) control of the rate of reproduction (forwarding factor), and (3) control of the rate of injection of new information by sources. Function 1 is achieved by the aging mechanism, described in Section II-C. Fresh packets created by the source at this node are also inserted into the epidemic buffer (Section II-G). These and other packets compete for transmission, with a fitness equal to their `vRate` (Section II-F). The `vRate` of a packet is, roughly speaking, the rate at which this packet would be transmitted or retransmitted if it were alone. It decreases exponentially for every transmission (self-inhibition) and every reception of a packet of the same clone (inter-inhibition) (Section II-B). This aims at avoiding unnecessary transmissions (Function 2). Congestion control is implemented by a combination of the decrease rules for `vRate` (Section II-B), of the rules governing the adaptive age and of the mechanism for limiting the generation of fresh packets described in Section II-G (Function 3).

B. Rate Adaptation

The `vRate` of a packet is set according to

$$\texttt{vRate} \leftarrow R_0 a^{\texttt{rcvCount}} \, b^{\texttt{sendCount}}$$

where R_0 is the nominal rate in packets per second of the MAC layer interface, a (inter-inhibition constant), b(self-inhibition constant) are positive less than 1, and `rcvCount` , `sendCount` are computed as in Algorithms 1 and2. The `vRate` thus decreases exponentially as more copies of the clone are received or sent. The values of a and b are tuned in Section IV, where we also show that both are necessary. The scheduler uses the inverse of `vRate` as a minimum time interval between transmissions (Section II-F).

C. Aging

The `age` attribute is inherited when a packet is received for the first time (Algorithm 1), and is equal to 0 for a newly created clone. The `age` stored in the epidemic buffer is a floating point number. It increases depending on events affecting the packet and the state of the epidemic buffer. There are three decreasing processes:

- (hop count): `age` is incremented by a constant amount $K_0 = 20$ whenever either this packet's `sendCount` or `rcvCount` is incremented. The value of K_0 is such that a packet can be transmitted in the very worst case at most ca. 25 times.
- (real time age) `age` increases at a constant rate $\gamma = 32h^{-1}$. We assume that nodes have free running clocks; there is no need for time synchronization. The constant γ is such that a packet lives at most 8 hours.
- (adaptive age) The `age` of all packets stored in the epidemic buffer increases by an amount K_1 every time a packet (of an existing or new clone) is received. The adaptive aging constant K_1 is a (possibly non integer) constant less than K_0; its value is determined by the analysis in Section IV.

When transmitting a packet, the complement to `maxTTL`(=255) of `age` , rounded to an integer, is written in the IPv4 TTL field [resp.IPv6 hop count].

A packet is killed whenever its age is too large to be sent, i.e. when `age` $\geq$ maxTTLL$+1$ (the $+1$ is due to rounding). It can also be killed due to lack of space in the epidemic buffer (Section II-H).

When a packet is received for a clone that is present in the epidemic buffer, the increase by K_0 is applied to the `age` of the packet already present in the epidemic buffer. This avoids that a packet gets killed by spurious receptions of “old” packets. Note that there is no risk of uncontrolled proliferation, as the age is incremented for every packet transmission.

D. Packet Receiving and Sending

Algorithm 1 and Algorithm 2 show the functions implemented when receiving and sending a packet. For brevity, error case handling is not shown. At line 7 of Algorithm 1, packet.TTL is the value read in the TTL field of the IP header in the received packet. In line 2 of Algorithm 2, $\lfloor$age$\rfloor$ is the largest integer$\leq$ age. The resulting TTL value is necessarily $\leq$maxTTL, as a packet is killed when its age reaches maxTTL$+1$ (Section II-C).

Algorithm 1 Receiving a Packet

```
 1: if clone was seen before but is not in epidemic buffer then
 2:     discard packet
 3:     leave
 4: else if clone seen for first time then
 5:     pass packet to application
 6:     rcvCount ← 0                                   ▷ initializations
 7:     age ← maxTTL − packet.TTL
 8: else if this packet belongs to a clone present in epidemic buffer then
                                   ▷ check if send confirmation is pending
                                                      ▷ (Section II-E)
 9:     if pendingSendConfirmation ==true then
10:         confirmSending(this packet)
11:     end if
12: end if
13: rcvCount ← rcvCount + 1
14: age ← age + $K_0$
15: update age of all packets (real time age, Section II-C)
16: if age ≥ maxTTL + 1 then discard packet
17: else
18:     update vRate (Section II-B)
19:     earliestSendTime ← now + $\frac{1}{\text{vRate}}$
20: end if
21: function CONFIRMSENDING(packet p)
                          ▷ called when we receive a copy of previously
                          ▷ sent packet for which it was dubious whether
                                  ▷ anyone received it (Section II-E)
22:     sendCount ← sendCount + 1
23:     age ← age + $K_0$
24:     update vRate (Section II-B)
25:     earliestSendTime ← now + $\frac{1}{\text{vRate}}$
26:     pendingSendConfirmation ←false
27: end function
```

E. MAC Layer Issues

We assume that nodes have a MAC layer capable of receiving and sending packets in broadcast mode, at a rate that depends on the network conditions (and is likely to be much less than the peak transmission rate R_0 used above). In practice, if we use the IEEE 802.11 MAC broadcast, there is a performance issue, as it does not use the RTS/CTS exchange and collisions during transmission go undetected. To avoid this issue, we use the pseudo-broadcast mode proposed in [10], by which a packet is sent to the MAC address of a neighbor (withRTS/CTS), but can be promiscuously copied

Algorithm 2 Sending a Packet

```
 1: when scheduler decides to send this packet
 2: set packet’s TTL field in IP header to maxTTL−⌊age⌋
 3: submit packet to MAC layer and wait for return code from MAC layer
 4: if return code==“packet was sent in pseudo-broadcast mode, or in
    broadcast with presence indicator” (Section II-E) then
 5:     sendCount ← sendCount + 1
 6:     age ← age + $K_0$
 7:     update vRate (Section II-B)
 8:     pendingSendConfirmation ←false
 9: else if return code==“packet was sent in broadcast mode without
    presence indicator” then
10:     pendingSendConfirmation ←true
11: end if
12: earliestSendTime ← now + $\frac{1}{\text{vRate}}$
13: update age of all packets (real time age Section II-C)
```

by all systems within range. This effectively solves much of the performance issue, but may not always be applicable to our case, since we do not want nodes to spend time discovering their neighbors’ MAC addresses.

Therefore, we use the following method. The MAC layer has a node global MAC state information that says whether the next packet will be sent in pseudo-broadcast, and if so, to which MAC address, or in broadcast mode. The destination MAC address in the pseudo-broadcast mode is the source MAC address of the last received packet.

As soon as the node receives one packet, the MAC state is set to pseudo-broadcast. The next packet is thus sent with an RTS. If no CTS is received in response, the MAC layer backs off for a random time (this is the standard operation of 802.11). If during the back-off time a packet is received, the packet is retransmitted (after expiration of the back-off timer) in pseudo-broadcast mode to the MAC address of the newly received packet. Else the MAC state moves to broadcast, and the packet is re-transmitted in broadcast mode.

There remains an issue, however, as a node does not know if a sent packet was received; this might become a problem in the desert highway scenario, where a node would repeatedly send a packet in the vacuum, until it ages out.

To avoid this, we use two heuristics when sending in broadcast mode: (1) indication of neighbor presence, and (2) implicit acknowledgment by reception of duplicate. (1) consists in building a function around the MAC layer that says whether, shortly before or after a packet transmission in broadcast mode, the carrier is sensed busy (line 4 of Algorithm 2) or not (line 9). If a packet is sent in the former case, or in pseudo-broadcast mode (some neighbors are around) then sendCount is incremented and pendingSendConfirmation is set to false. Of course, there is no guarantee that a packets sent in these circumstances is actually received by any one, but the rules for rate adaptation will make it likely for this packet to be retransmitted soon if no duplicate is received (in such a case `vRate` remains large). If in contrast a packet is sent in the latter case (presumably because there is no one around) the flag `pendingSendConfirmation` is set to true for this packet, and the packet is rescheduled at a later date with the same `vRate` . If a packet of the same clone is received while `pendingSendConfirmation` is true, the pending transmission is considered successful (lines 9 and 21 of Algorithm 1). The condition `pendingSendConfirmation` == true can be terminated either by reception of a duplicate or by a subsequent transmission that returns an indication of presence or is in pseudo-broadcast mode.

F. Scheduling of Packet Transmission

The packet scheduler decides which packet in the epidemic buffer is selected for transmission, i.e. for being passed to the MAC layer.

In order to ensure some level of fairness, the scheduler serves packets per source IP address, using a processor sharing approach. Furthermore, every packet should be served at a rate not exceeding its `vRate` (the `vRate` of a packet should be the rate at which it would be transmitted if it were the only one in the epidemic buffer –this is to avoid flooding the network at high rate if little new information is available). Note that the `vRate` of a packet may change with time. Last, the source at this node (the "self" source) should receive some minimum rate. Thus the scheduler should be similar to a practical implementation of weighted fair queuing , but with special requirements. We use the following method.

Packets with the same source IP address are linked in one FIFO per source. Each of these FIFOs has an attribute `sourceClaim` , which keeps track of how much this source can claim to be scheduled. It is initially 0 and is decremented by 1 when this source is selected for transmission by the scheduler. It is incremented by the scheduler as explained below.

Every packet in the epidemic buffer has a derived attribute `earliestSendTime` , equal to the last time at which the `vRate` of this packet was modified, plus $\frac{1}{\texttt{vRate}}$. At any time t, a packet is said to be "eligible" if it has `earliestSendTime` $\leq t$. The scheduler works as shown in Algorithm 3.

Algorithm 3 Scheduler Algorithm

```
1: when scheduler is ready to select a packet
                          ▷ send(packet) function to MAC has returned
2: for all FIFO $j$ do
3:     sourceClaim $(j) \leftarrow$ sourceClaim $(j) + \frac{1}{N}$
                                          ▷ $N$: number of FIFOs
4: end for
5: select the first FIFO $j_0$ by decreasing order of sourceClaim that
   has an eligible packet
6: decrement sourceClaim $(j_0)$ by 1
```

The scheduler issues a blocking send function to the MAC layer that returns whenever the packet is accepted by the MAC layer.

It can be seen that this algorithm allocates the transmission opportunities according to a water-filling algorithm, thus, it approximates an ideal fluid scheduler that would allocate rates to sources in a max-min fair way, subject to the constraint that the rate of a source does not exceed the sum of the `vRates` of the packets of this source.

G. Control of Injection of Fresh Packets

The packets generated by the node are injected in the epidemic buffer using a credit system that we now describe. The goal is to adapt the application rate to the traffic rate that the scheduler is able to allocate to self packets (Section II-F), while guaranteeing space for σ self packets in the epidemic buffer.

The self source has a credit counter CR initially equal to σ. A self packet from the application is accepted in the epidemic buffer if $CR > 0$. CR is decremented by 1 for every self packet injected in the epidemic buffer, and incremented by 1 when a self packet is removed from the epidemic buffer.

When the application has a packet to insert and $CR == 0$, we test if a self packet can be killed. A self packet can be killed if `rcvCount` ≥ 1. If at least one self packet can be killed, the packet with the largest `age` is chosen and removed from the epidemic buffer. Thus, the application is allowed to inject a fresh packet for every old packet for which a duplicate is received (we treat this as an implicit acknowledgement).

It may happen that the application is allowed to issue a fresh packet by this rule, but the epidemic buffer is full (due to non-self packets). In such cases, the non self-packet with the largest `age` is removed from the epidemic buffer.

H. Miscellaneous Issues

Space for new packets in the epidemic buffer is freed by the aging mechanism (Section II-C) and possibly by insertion of self packets (Section II-G). For nodes with very limited buffer size, this may not be sufficient. If an arriving packets requires space to be freed, the non-self packet with the largest `age` is deleted.

The source address used to identify a clone is the IP source; SLEF packets are sent to an IP multicast address reserved for SLEF. We assume that SLEF packets also contain an identification field, incremented by a source for every fresh packet it generates. The field should be large enough to avoid that two different clones have the same identification.

The epidemic buffer needs to keep track of clones that were seen before but may not be present anymore. In the current implementation we simply keep a record of all clone identifiers (IP address + packet identifier) for a duration equal to the maximum packet lifetime (i.e. $\frac{\texttt{maxTTL+1}}{\gamma}$). A more efficient implementation would be a Bloom filter.

III. Design Analysis

In this section we analyze the design of SLEF by an ordinary differential equation method. The goal is to obtain a first, approximate sizing of system parameters. As mentioned in the introduction, a detailed evaluation is done using an implementation in Java and simulations.

A. Derivation of an ODE model

ODE models are often used to study epidemic systems. They are usually derived from a simple inspection of the system. In our case, there are many state variables (the state of the counters associated with the messages at all nodes) and the inspection method seems difficult. We use instead a formal method, which consists in deriving the set of ODEs from a microscopic description of the system, expressed as a continuous time Markov process.

The starting point is the following well known result.

Theorem 1 (Forward Equation [18]): Consider a continuous time Markov Chain $\mathbf{S}(t)$ defined over a finite state space $\mathcal{S}$. For any function $f(\mathbf{s})$ of the state:

$$\frac{\partial \mathbb{E}\{f(\mathbf{S}(t))\}}{\partial t} = \mathbb{E}\{\sum_r h_r(\mathbf{S}(t))\left(f(\mathbf{S}(t)+\boldsymbol{\Delta}_r) - f(\mathbf{S}(t))\right)\}$$

where the summation is over all possible state transitions r, $h_r(\mathbf{s})$ is the rate of transition r when the state vector is $\mathbf{s}$; $\mathbf{s}+\boldsymbol{\Delta}_r$ is a symbolic notation for the state after transition r has occurred.

The term $\sum_r h_r(\mathbf{S}(t))(f(\mathbf{S}(t)+\boldsymbol{\Delta}_r) - f(\mathbf{S}(t)))$ is called the drift of function f. When applied to $f()=$ the indicator function of the singleton set $\{\mathbf{s_0}\}$, the forward equation gives the well known equation for the time dependent probability of being in state $\mathbf{s_0}$, which is often called the "master equation".

The ODE is then obtained by applying the forward equation to appropriate functions $f()$ of the state, and making the approximation that the right hand-side can be approximated by

$$\sum_r \mathbb{E}\{h_r(\mathbf{S}(t))\}\left(\mathbb{E}\{f(\mathbf{S}(t)+\boldsymbol{\Delta}_r)\} - \mathbb{E}\{f(\mathbf{S}(t))\}\right) \tag{1}$$

We model a network of N node $i \in \{1,\dots,N\}$ and M messages $k \in \{1,\dots,M\}$ as a continuous time Markov chain with state vector $\mathbf{s}$ given by the collection for i,k of $\mathbf{s}_{ik} = (X_{ik}, N_{ik}, M_{ik}, A_{ik})$. Here X_{ik} is a boolean equal to 1 if message k is present at node i, N_{ik} is the variable `recvCount`, *i.e.* the number of time a copy of the message k has been received by node i; M_{ik} is the `sendCount` for message k at node i; A_{ik} is the age of the message (assumed to be quantized in a finite set).

We are in this subsection interested in the transient behaviour (see Section III-C for a stationary behaviour), and assume that all messages to be transmitted are initially present at their source nodes. The source of message k is set to node i by letting $(X_{ik}(0)=1, M_{ik}(0)=0, N_{ik}(0)=0, A_{ik}(0)=0$.

We apply the forward equation to the collection of cases where $f()$ is a coordinate of the state (such as $f(\mathbf{s}) = N_{ik}$ for example) and obtain the following set of equations. We use the notation $\bar{X}_{ik} = \mathbb{E}\{X_{ik}\}, \bar{N}_{ik} = \mathbb{E}\{N_{ik}\}, \bar{M}_{ik} = \mathbb{E}\{M_{ik}\}, \bar{A}_{ik} = \mathbb{E}\{A_{ik}\}$ (note that $\bar{X}_{ik}(t) = \mathbb{E}\{X_{ik}(t)\}$ is simply the probability that message k is present at node k at time t):

$$\frac{\partial \bar{X}_{ik}(t)}{\partial t} = (1-\bar{X}_{ik}(t)) \sum_{j\in\mathcal{N}_i(t)} R^i_{jk}(t)(1-\bar{D}_{jk}(t))\bar{X}_{jk}(t) \tag{2}$$

$$\frac{\partial \bar{N}_{ik}(t)}{\partial t} = \sum_{j\in\mathcal{N}_i(t)} R^i_{jk}(t)(1-\bar{D}_{jk}(t))\bar{X}_{jk}(t) \tag{3}$$

$$\frac{\partial \bar{M}_{ik}(t)}{\partial t} = R_{ik}(t)(1-\bar{D}_{ik}(t))\bar{X}_{ik}(t) \tag{4}$$

$$\frac{\partial \bar{A}_{ik}(t)}{\partial t} = (1-\bar{X}_{ik}(t)) \sum_{j\in\mathcal{N}_i(t)} R^i_{jk}(t)(\bar{A}_{jk}(t)+K_0) + \bar{X}_{ik}(t)\left(K_1 \sum_{k=1}^{M} \sum_{j\in\mathcal{N}_i(t)} R^i_{jk}(t)\bar{X}_{jk}(t)(1-\bar{D}_{jk}(t))\bar{D}_{ik}(t)\right) \tag{5}$$

In the equations we used the notation $\bar{D}_{ik} = \mathbb{E}\{\mathbb{1}_{\{A_{ik}\geq \texttt{maxTTL+1}\}}\}$ (probabilities that message i was received is dead at node k). We also used the variables $R_{ik}(t)$ (rate of emission of message k by node i) and $R^j_{ik}(t)$ (rate at which message i is transmitted by node k and received by node j). These rates depend on specific MAC layer assumptions and are explained in detail in Section III-B. Also, we ignore the real time age constant γ as it its impact and dimensioning can be simple reasoning about maximum lifetime. The constants K_0 and K_1 are aging coefficients as defined in Section II; $\mathcal{N}_i(t)$ is the neighbourhood of node i, defined by the transmission and interference ranges (assumed in this analysis section to be identical).

For the variables $\bar{D}_{ik}$ we use the approximation

$$\bar{D}_{ik}(t) = \mathbb{1}_{\{\bar{A}_{ik}\geq \texttt{maxTTL+1}\}} \tag{6}$$

We solve the system of ODEs using the Runge-Kutta method implemented in matlab.

B. Rate Function

The rate of transmission of message k by node i (R^k_i) results from three effects : the virtual rate $\texttt{vRate}^k_i$, the scheduling mechanism implemented in the node and the MAC layer used. The virtual rate is defined as $\texttt{vRate}^k_i(t) = R_0 a^{N^k_{jk}(t)} b^{M^k_j(t)}$. The scheduler effect can be captured by the tentative sending rate $R^{k*}_i(t)$ of message k at node i:

$$R_i^{k*}(t) = \min\left(\mathtt{vRate}_i^k, R_0 \frac{\mathtt{vRate}_i^k}{\sum_{l\in\mathcal{M}_i} \mathtt{vRate}_i^l}\right)$$

where $\mathcal{M}_i$ is the set of messages in node i that are still alive. The transmission and reception rates are related to the overall transmission rate in the neighborhood of the receiver by :

$$R_i^k(t) = R_i^{k*}(t) g_1 \left(\sum_{i'\in\mathcal{N}_i(t),\, l\in\mathcal{M}_{i'}} R_{i'}^{l*}\right)$$

$$R_{ij}^k(t) = R_i^k g_2 \left(\sum_{j'\in\mathcal{N}_j(t),\, l\in\mathcal{M}_{j'}} R_{j'}^{l*}\right)$$

where the precise forms of the decreasing functions $g_1()$ and $g_2()$ depend on the MAC layer used. As we are assuming here an IEEE 802.11 MAC layer, we could use the approximation the broadcast mode of IEEE 802.11 protocol as a p-persistent CSMA mechanism proposed in [2].

The difference between the IEEE 802.11 protocol and the p-persistent resides in the selection of the back-off interval after a collision. Instead of binary exponential back-off as in IEEE 802.11, the p-persistent approach samples its back-off interval size from a geometric distribution with parameter p. Probability p depends on the mean size of contention window. In [2] a formula for ρ, the capacity of an IEEE 802.11 network, as a function of p (persistence probability), the number of nodes M in the environment and message size is derived (formula 8 in [2]). This formula is validated with simulations and shown to be a good approximation of the real behavior of IEEE 802.11. Moreover a procedure is presented in the same reference to derive the value of the p parameter as a function of the number of sources and a formula is provided for the probability of collision $\mathbb{P}\mathrm{rob}\{coll\}$ (in lemma 3) as a function of M and $p(M)$. Using these results we take for g_1 and g_2 :

$$g_1(x) = \frac{g_2(x)}{1 - \mathbb{P}\mathrm{rob}\{\mathrm{coll}\}} \;,\; g_2(x) = \min\{1, \frac{\rho(p, M)}{x}\}$$

C. Steady state analysis

So far we can compute the transient behaviour of epidemic forwarding, using (2) to (5). In this subsection we extend the method to account for stationary behaviour, with sources that emit packets according to Poisson processes each of rate λ.

It is difficult to do an exact model, as we need to keep track of all packets previously transmitted by all sources. Therefore we use the following heuristic. We represent any source of rate λ as a collection of H virtual, parallel mini-sources, each with rate $\frac{\lambda}{H}$. We then consider the virtual system such that, when any of the H mini-sources emits a packet, all previous packets of this mini-source are instantly discarded from all nodes in the network. This is clearly an ideal system; however, if the original system is stable and H is large, the virtual system should be close to the real one, since packets have a finite lifetime. A criterion for determining if H is large enough is when increasing H does not modify the performance.

In this model, the message index k now takes the form $k = (i', h)$ where i' the index of the source node and $1 \le h \le H$ the index of the mini-source at i' that generated this message. The ODEs are obtained in the same way as before, with an additional term that accounts for transitions resulting from message generation. We obtain, for $k = (i', h), i' \neq i$:

$$\frac{\partial \bar{X}_{ik}(t)}{\partial t} = (1 - \bar{X}_{ik}(t)) \sum_{j\in\mathcal{N}_i(t)} R_{jk}^i(t)(1 - \bar{D}_{jk}(t))\bar{X}_{jk}(t) - \frac{\lambda}{H}\bar{X}_{ik}(t)$$

$$\frac{\partial \bar{N}_{ik}(t)}{\partial t} = \sum_{j\in\mathcal{N}_i(t)} R_{jk}^i(t)(1 - \bar{D}_{jk}(t))\bar{X}_{jk}(t) - \frac{\lambda}{H}\bar{N}_{ik}(t)$$

$$\frac{\partial \bar{M}_{ik}(t)}{\partial t} = R_{ik}(t)(1 - \bar{D}_{ik}(t))\bar{X}_{ik}(t) - \frac{\lambda}{H}\bar{M}_{ik}(t)$$

$$\frac{\partial \bar{A}_{ik}(t)}{\partial t} = (1 - \bar{X}_{ik}(t)) \sum_{j\in\mathcal{N}_i(t)} R_{jk}^i(t)(\bar{A}_{jk}(t) + K_0) + \bar{X}_{ik}(t)\left(K_1 \sum_{k=1}^{M} \sum_{j\in\mathcal{N}_i(t)} R_{jk}^i(t)\bar{X}_{jk}(t)(1 - \bar{D}_{jk}(t))\bar{D}_{ik}(t)\right) - \frac{\lambda}{H}\bar{A}_{ik}(t)$$

For $i' = i$ the equations are the same except the first which is replaced by $\bar{X}_{ik}(t) = 1$ for all t. We are interested in the steady state, which we conjecture must exist because we have made the system stable (by the heuristic of mini-sources, the counters are reset at a rate $\frac{\lambda}{H}$). Thus we equate the left-handside to 0 and obtain a system of equations (not shown here due to lack of space) for quantities such as $\bar{X}^*_{i,(i',h)} = \lim_{t\to\infty} \bar{X}_{i,(i',h)}(t)$. We solve this system numerically (by iteration, it is a fixed point problem) and increase H until the aggregate variables $\sum_{h=1}^{H} \bar{X}^*_{i,(i',h)}$ (number of messages emitted by source i' present at node i) do not vary significantly. Note that all variables such as $\bar{X}^*_{i,(i',h)}$ are independent of h (this is true in steady state, not otherwise), so the complexity of the numerical solution is independent of H.

IV. Simulation and Validation

We present in this section two scenarios to evaluate SLEF: A campus scenario and a vehicular scenario. Both scenarios are based on a Bulletin Board Network (BBN) application. Through these scenarios, we deal with different network conditions: Dense, sparse, congested and not congested. The evaluation is done through two methods: ODE analysis and an implementation of SLEF in JIST/SWANS [1] simulator. ODEs are solved in Matlab using the Runge-Kutta method.

By comparing the results of the two methods we also validate the ODE approach.

A. Campus Scenario

We assume that a set of 50 non-mobile users are uniformly distributed over a 100×100 m^2 campus. With an IEEE802.11 reception range of 50 m (indoor reception range of WIFI), this makes each node connected in average to 25 other nodes.

1) Transient Analysis: We assume that only 10 messages are sent by different users and they should be received by as many users as possible. In this context we could use the ODEs approach to predict the transient behavior of this system. After solving the set of $50 \times 10 \times 5$ differential equations (Eqs. (2)-(5)), we show in Fig. 1 the transient behavior of 20 arbitrary nodes out of the 50 nodes. This figure shows the delay of information spread as well as the lifetime of a packet. By changing the value of a and b and solving the

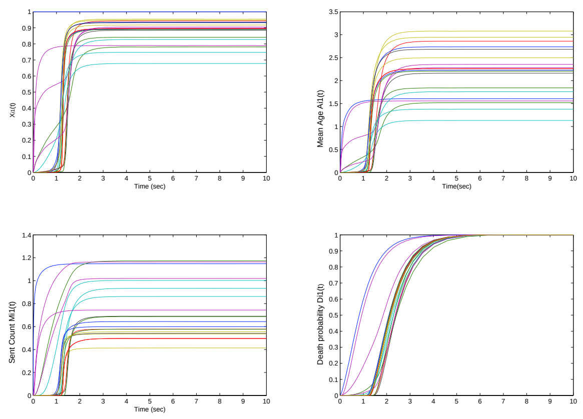

Fig. 1. Transient behavior of the campus scenario for 20 arbitrary nodes out of the 50 nodes. 10 nodes are sending a single message and $a = b = 0.15$, $K_1 = 0.1$, $K_0 = 20$, $\gamma = 32$ hour^{-1}

corresponding system of ODEs, one can obtain the transient and observe that the reception delay decreases with larger value of a and b as the transmission rate are increasing but this is achieved at the cost of larger number of sent clones. To have a broader perspective, for the same scenario (50 users in a campus) we have fixed the value of K_1 and K_0, and we have done a scan over the a and b values space. We compare in Fig. 2 the results obtained after the simulation of SLEF during 60 sec in the JIST/SWANS environment with what has been predicted by ODE. We first see for most values of a and b, a good adequacy of the ODE prediction with the results obtained from simulation. The main differences occurs for small and large values of a and b. The difference for small values of a and b could be explained by the fact that the ODE analysis is based on the assumption that a large number of events occurs in the network. Thus, when a and b are small the number of events occurring in the network

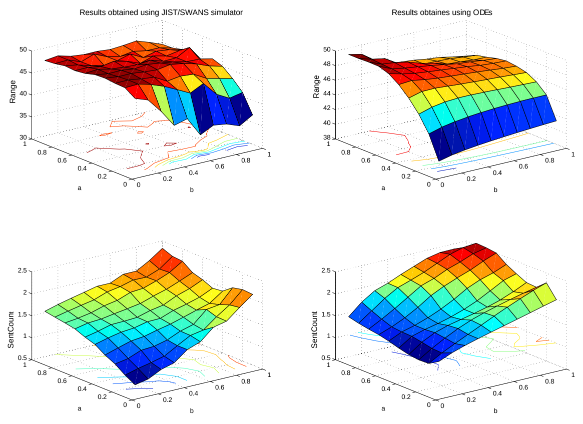

Fig. 2. Results of the browse of the a and b value space while fixing $K_1 = 0.1$, $K_0 = 20$. The left column contains results evaluated over JIST/SWANS simulation of SLEF. The right column contains results predicted by the ODE approach.

results in an imprecision of the ODE assumption. For large values of a and b, the amount of injected packet into the network goes higher and congestion builds up. For these values, our approximation of the IEEE802.11 behavior by the p-persistent is not tight and results in an overestimation of the congestion effect. First of all, it could be seen that for large enough values of a and b the mean number of nodes receiving a message is almost insensitive to the values of a and b. Increasing the values of a and b results in an increase of the number of sent copies within the same clone as we can see in the second row of Fig. 2. This means that increasing a and b will increase the number of sent clone copies (the number of received copies that is not shown here increases accordingly) and decrease the reception delay of messages without increasing the range of the diffusion.

Two points are of specific interest in this curve. First, the point $a = b = 1$ describes a system that is not using the inhibition mechanism and control the spread only through the killing mechanism. This point results in a large number of `sendCount` and a higher congestion without increasing the transmission range.

Another point of interest is the point $b = 0$. This point defines a system where any received packet is just forwarded at most one time. In such conditions the message reach the maximal range at a relatively low cost in term of sent packets. However, we need b to be strictly positive to support mobility; a node might move to another place where a packet is not sent before and thus it forwards it again. The curves also suggest a design heuristic for the SLEF system. The choice of a and b is made by choosing a value of `sendCount`. This results in a set of values a and b that achieves this `sendCount`. Now, a and b are set to the values maximizing the range of reception. Applying this heuristic, a good operating point is $a = b = 0.15$, which leads to $M_s = 27$ (meaning 27 copies of

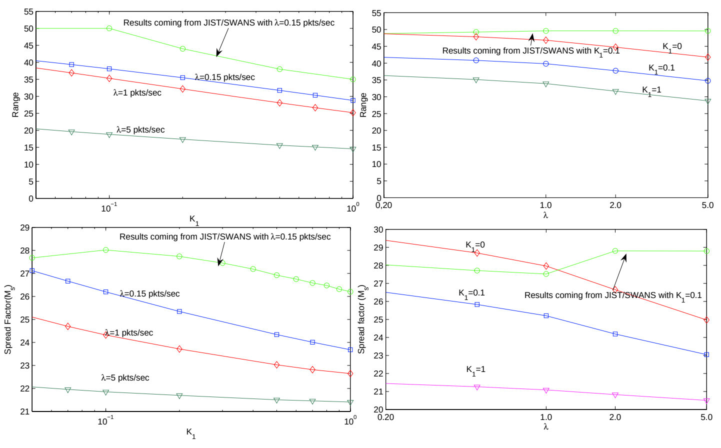

Fig. 3. Effect of K_1 and λ on the spread of messages in steady state for a 50 nodes scenario with 10 sources. The inhibition parameters used are $a = b = 0.15$

any clone are generated in the network). In the forthcoming, unless otherwise stated, we set a and b to these values and study the steady state behavior of the system.

2) Steady State Analysis: Now we assume that 10 nodes out of the 50 nodes inject with a rate λ a continuous flow of packets into the network. We will describe the steady state of this system and analyze the effect of K_1 coefficient as well as λ on the transmission range and on the number of sent clones. Intuitively, we expect that increasing λ will results in a larger amount of traffic in the network and in higher congestion and through the killing mechanism, the spread of the message will be reduced and less nodes would receive the messages. In the same direction, when we increase K_1 we expect that the sensitivity of the killing mechanism to packet density will increase and more packet will be killed, resulting in a decrease of the spread of packets.

Fig. 3 shows these effects. The first row contains the predicted range obtained through ODE analysis and compares it with that obtained by JIST/SWANS simulations (with a value of $K_1 = 0.1$ and $\lambda = 0.15\ pkts/sec$). It is noteworthy that for all the obtained curves a value of $h = 10$ was used to implement the heuristic developed to extend the transient behavior to the steady state. We see that JIST/SWANS predicts a larger range than the ODE analysis. This is because the SLEF implementation contains also a injection rate control mechanism that is not modeled by the ODEs. The second row in Fig. 3 shows the predicted results for the spread factor M_s. The spread factor describes the mean number of sent packets resulting from the transmission of each single new packet by any of the sources. The comparison of the ODE prediction with the value observed through implementation shows also a difference. This could be mainly explained through the same arguments as above. As the spread of clones is larger in the real implementation, the amount of sent packet whithin the

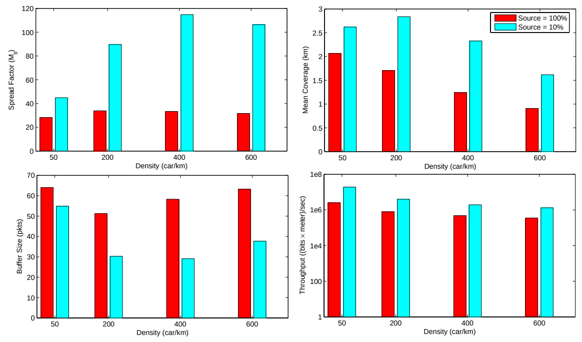

Fig. 4. Vehicular BBN scenario: Cars are on a highway with 2 lanes in each direction. $a = b = 0.15$, $K_0 = 20$, $K_1 = 0.1$ and $\sigma = 2$

same clone will also be larger. Nevertheless, we could see in the left column of graphs for large value of λ the slope of decrease of the range as well as the spread factor are captured well by the ODE analysis even if the predicted value are not completely accurate. This means that, qualitatively, the ODE analysis is correct.

B. Vehicular BBN

As a last scenario, we now illustrate the performance of SLEF in a BBN application running over a vehicular network. We consider that cars are in a two ways on a 4 lanes highway with different car densities. Cars are mobile and their speed is related to the car density according to the ”speed-density model” by Green shield, which is largely used in transportation engineering [6]. Cars are equipped with WIFI transceivers to communicate with each other. The transmission range is set to 250 m (outdoor WIFI transmission range). We consider that users are greedy, *i.e.* they have always fresh packets to send. We let SLEF to control the injection of information in the networks. For this scenario, we have set the value of SLEF parameters to $a = b = 0.15$, $K_0 = 20$, $K_1 = 0.1$ and $\sigma = 2$. We show two sets of results: When all the cars are sources and when only 10 of them are sources and others are acting as SLEF relays.

Fig. 4 shows the results of this scenario. The spread factor shows the efficiency of the killing mechanism; for the situation where all cars are sources, the spread factor is constant even when the density of transmitting node is multiplied by 3. This is confirmed when we observe the mean coverage of messages that show the mean distance where a message is received by all cars. It could be seen that increasing the density results in a lower coverage. This means that the scheme is able to self-adapt its spread to the density of nodes, *i.e.* to reduce its spread in high density situations as in a traffic jam and to increase its spread up to the hop count limit defined through K_0 and the lifetime limit in sparse situation as when one is driving through the Death valley. This last point is also confirmed by observing the throughput measured in term of (bits $\times$ meter)/sec. The well-known results for the capacity of Ad-hoc networks obtained in [7] states that the capacity in (bits $\times$ meter)/sec of a wireless network with arbitrary sources and destinations decreases as $1/sqrt(D)$ where D is the density of nodes. Here we see that SLEF scheme attains a capacity that is constant. This comes from the fact that, contrary to the capacity results in [7], with increasing density SLEF reduces the spread of packets to closer destinations. Therefore, one could expect SLEF to be a scalable solution. Another curve of interest is the mean buffer occupancy that is plotted in the lower left part. Interestingly, it shows that even for the very high density scenario there is no more than 70 different packets in the epidemic buffer (around 100 Kbytes) and, with this low number of packets in the buffer, coverage around 1 km is achieved.

V. State of the Art

Epidemic forwarding has been an active area of research during recent years. Most of the researchers in this area have identified the fact that limiting the spread and /or minimizing the redundancy are major challenges. Different solutions to deal with these problems have been proposed. In [16] the authors present a basic epidemic forwarding scheme that uses a fixed and predetermined value of TTL to limit the spread of diffusion. This scheme uses also content negotiation prior to information diffusion to ensure that redundant information is not sent. In comparison, SLEF proposes an adaptive spread control by combining TTL with the aging mechanism. Thus, the spread is adapted with the node density, mobility model and traffic load. Furthermore, SLEF avoids negotiation, which adds overhead and makes communication point-to-point. SLEF makes the most of the broadcast nature of the channel while maintaining a very low level of redundancy. In [14] the forwarding factor is reduced through defining localized dominating sets, which will ensure that messages are broadcast only when the list of neighbor that might need the message is not empty. However, this scheme assumes that any node has knowledge of the position of its neighboring nodes. In contrast, SLEF does not need about any topology or position knowledge. In [8] a probabilistic forwarding scheme named "Gossip-based" is presented. This scheme forwards a received packet with a fixed probability. In contrast, SLEF maintains an adaptive forwarding factor control based on the `vRate`. In [17] a broadcast mechanism is presented that use two neighborhood coverage conditions to decide if a node should forward a message. The neighborhood coverage conditions are based on k-hop connectivity information that is hard to gather in a mobile environment where SLEF would live. In [13], "spray" based routing methods are proposed. During the "spray" phase, L copies are forwarded to L distinct nodes in a binary fashion. This mechanism is similar to a fixed TTL with maxTTL $= log_2(L)$. "Spray" uses negotiation and forwards packets in a point-to-point way. Also, it includes a learning phase to adapt L to the network size but it does not take into account the traffic load as we do through aging. In [12], the authors propose an inhibition mechanism that prevents a packet from being forwarded if a fixed number of copy of this same packet have been received. In [11] an epidemiological model is applied to the study of a simple information diffusion mechanism. The approach followed in [11] is close to ours. However, we present a more methodological derivation of the ODEs that could be easily extended to other scenarios, where the approach followed in [11] is a direct fitting of an epidemiological model to a simple single source problem. In [19], an ODE based model was proposed for already existing epidemic forwarding methods. Although the basis of this model is similar to ours, our work is different in that (1) we propose a new complete epidemic method that includes new essential components such as spread control, and (2) these new components make our model more general.

VI. Conclusion

We presented a Self Limiting Epidemic Forwarding scheme, based on an inhibition mechanism to minimize redundancy, an aging mechanism to control the spread by adapting the number of copies of packets in the network and a injection-control mechanism to ensure that a source will not flood the network. We presented an analytical description of the proposed scheme using an ODE approach. The presented approach could be easily extended to other forwarding schemes and seems to be a promising modeling approach. Thereafter, we evaluated SLEF in two different scenarios and we show that SLEF could achieve its goals in both contexts. The obtained results validate SLEF as a good proposal for wide range of network settings (e.g. sparse, dense and different traffic loads). Globally this work opens a lot of perspective that might open new interesting research questions. In particular integration of mechanism such as content negotiation could improve the performance of SLEF. These points are also left to future study.

References

[1] Java in simulation time / scalable wireless ad hoc network simulator , jist/swans, http://jist.ece.cornell.edu/.

[2] F. Cali, M. Conti, and E. Gregori. Dynamic tuning of the ieee 802.11 protocol to achieve a theoretical throughput limit. *IEEE/ACM Trans. Netw.*, 8(6):785–799, 2000.

[3] C. Diot, J. Scott, E. Upton, and M. Liberatore. The Haggle architecture. Technical Report IRC-TR-04-016, Intel Research Cambridge, 2004.

[4] M. Durvy and P. Thiran. Reaction-Diffusion Based Transmission Patterns for Ad Hoc Networks. In *Infocom*, 2005.

[5] A. ElFawal, J. Leboudec, and S. K. A self limiting epidemic forwarding. Technical Report LCA-REPORT-2006-126, EPFL, 2006.

[6] B. Greenshield. A study of traffic capacity. *Highway Research Board Proceedings*, 14:448–477, 1935.

[7] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, IT-46, March 2000.

[8] Z. J. Haas, J. Y. Halpern, and L. Li. Gossip-based ad hoc routing. In *IEEE Infocom*, June 2002.

[9] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot. Pocket Switched Networks and the Consequence of Human Mobility in Conference Environments. In *Proceedings of the SIGCOMM 2005 Workshop on Delay Tolerant Networking*, 2005.

[10] S. Katti, D. Katabi, W. Hu, H. Rahul, and M. Medard. The importance of being opportunistic: Practical network coding for wireless environments. In *Allerton conference*, 2005.

[11] A. Khelil, C. Becker, J. Tian, and K. Rothermel. An epidemic model for information diffusion in manets. In ACM, editor, *Proceedings of the 5th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems*, pages 54–60, 2002.

[12] 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, August 15 - 19, 1999*, pages 151–162.

[13] T. Spyropoulos, K. Psounis, and C. Raghavendra. Spray and wait: An efficient routing scheme for intermittently connected mobile networks. In *ACM SIGCOMM workshop on Delay Tolerant Networking (WDTN-05)*.

[14] I. Stojmenovic, M. Seddigh, and J. Zunic. Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, Jan. 2002.

[15] P. Tennent, M. Hall, B. Brown, M. Chalmers, and S. Sherwood. Three applications for mobile epidemic algorithms. In *MobileHCI '05*.

[16] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical report, Duke University, April 2000.

[17] J. Wu and F. Dai. Broadcasting in ad hoc networks based on self-pruning. In *IEEE Infocom*, June 2003.

[18] B. Ycart. *Modèles et algorithmes markoviens*, volume 39. Springer Verlag, 2002.

[19] X. Zhang, G. Neglia, J. Kurose, and D. Towsle. Performance modeling of epidemic routing. In *IFIP Networking 2006*, 2006.