Multi-Hop Broadcast from Theory to Reality: Practical Design for Ad Hoc Networks

Alaeddine EL-FAWAL
Joint work with
Jean-Yves Le Boudec, Kavé Salamatian

Autonomics 28-30 October 2007, Rome Italy
OUTLINE

- Introduction
- SLEF Middleware: Self Limiting Epidemic Forwarding
- Performance Validation
- SLEF vs. K-Hop broadcast
- Conclusions
Why Multi-Hop Broadcast?

**Information dissemination:**

- Opportunistic communications
- High mobility where classical methods based on distribution trees fail.
- Support routing, resource discovery, bootstrapping phases for application layer.

**Broadcast application:**

- Disseminating traffic info in vehicular networks.
- Chat on the highway / in a crowd / in the train...
- Ad applications
Real Situation: Variation and Diversity of Scenarios

- Density: average
- Few sources: little new injected traffic
Real Situation: Variation and Diversity of Scenarios

- Density: average
- Almost all are sources: a lot of new injected traffic
Real Situation: Variation and Diversity of Scenarios

✓ Very high density (traffic jam)
✓ *almost all* are sources: *a huge amount* of new injected traffic
✓ One hop: + 200 neighbors
An autonomic mechanism for multi-hop broadcast that adapts to this diversity in scenarios is a must, otherwise network failure.

- Density: very sparse.
- No communication without *mobility* (opportunistic communication)
OUTLINE

- Introduction
- **SLEF Middleware : Self Limiting Epidemic Forwarding**
- Performance Validation
- SLEF vs. K-Hop broadcast
- Conclusions
We propose a Multi-hop broadcast *Middleware* for *Ad Hoc* networks.

We call it SLEF: Self Limiting Epidemic Forwarding

- Nodes forward each packet they receive with a given probability called *Forwarding Factor*.
- Packets are forwarded within a *limited* hop-count.

**Features:**

- **Autonomic:** Adapts itself to any change in the network.
- **Complete middleware**
- Does not need / exchange *any topology information*. Uses only local information to the node (very short contact time).

*SLEF is designed to hold in all scenarios, in particular in very dense and very sparse ones*
SLEF implements 6 essential functions needed with any multi-hop broadcast

1. **Congestion control**: first mechanism proposed for broadcast in ad hoc networks

2. **Efficient use of MAC broadcast**: 
   - 802.11 broadcast does not implement any exclusion mechanism (RTS/CTS) and it performs poorly (similar to Aloha). We replace it by a new scheme that we call **pseudo-broadcast**.
   - 802.11 broadcast does not implement Ack. Pkts might be transmitted in the vacuum. We implement a **presence indicator** that does not need any message exchange.

3. **Scheduler / fairness**: A **scheduler** is needed to decide which packet to serve. It is based on Source ID to ensure some level of **fairness**.

4. **Buffer management**: responsible of cleaning the buffer in order to keep space for new incoming pkts.

5. **Spread control**

6. **Inhibition**
**Spread Control: Adaptive TTL**

**Why:** Trade-Off Spread vs. Application Rate  
Spread: number of nodes that receive a packet.

\[ \lambda = \frac{R_0}{1 + FF \times N} \]

- \( N \): Spread
- \( FF \): Forwarding Factor
- \( R_0 \): Nominal Rate of MAC Layer
- \( \lambda \): Application Rate

**How:**
- We use an *Aging* mechanism
- **Adaptive TTL:** Aging *adapts locally* the TTL to the different network setting, based on the *send/receive events.*

- The idea is as follows:
  - Density/rate \( \uparrow \) => TTL \( \uparrow \) with fixed TTL: Rate \( \downarrow \)
  - Density/rate \( \uparrow \) => TTL \( \downarrow \): In a traffic jam: TTL = 1
  - Density/rate \( \downarrow \) => TTL \( \uparrow \): In a very sparse network: TTL = 10

\[\text{TTL} = 2\]
Aging

- New created pkt: Age = 0
- Pkt received for the same time: \( Age = 255 - TTL \)
- Age manipulated locally.
- when transmitting: \( TTL = 255 - Age \)

\[ \begin{align*}
\text{hop count} & : \text{Send/receive the } \text{same pkt} \\
& : \text{Age} = \text{Age} + \text{K0} \\
\text{adaptive Age} & : \text{receive } \text{any pkt} \\
& : \text{Age} = \text{Age} + \text{K1} \\
\text{Real time Age} & : \text{Constant increase by time: 8h} \rightarrow 255 \\
\end{align*} \]

- \( K0 = 25 \)
- \( K1 = 0.1 \)

\( \text{Age} > 255 \)
- Drop packet

- Hop count: plays the role of a fixed TTL (=255/K0) if the network is not congested.
- Adaptive Age: adapts the TTL to the network activity, which reflects the density and the traffic load
- Real time Age: A pkt lives at most 8 hours (work cycle).
**Inhibition**

**Why:** To Minimize redundancy (save resources)

**How:**
- Inhibit nodes from transmitting over sent/received pkts.
- We compute a virtual rate for each packet based on the send/receive events.
- Adaptive Inhibition: the virtual rate adapts locally the Forwarding Factor to the different network settings.

Transmission would be redundant

One send/receive event

Two send/receive events:
- The green nodes are inhibited: smaller forwarding factor
Virtual Rate

**Virtual Rate**: The max rate a packet is transmitted with

For each send / receive event on a given pkt:

1- The virtual rate is computed as follows:

\[ vRate \leftarrow R_0 a^{rcvCount} b^{sendCount} \]

- \( R_0 \): nominal MAC rate [pkts/s]
- \( RcvCount \): number of times the pkt is received
- \( SendCount \): number of times the pkt is sent
- \( a \) and \( b \): are coefficient less than 1: \( a=0.1, \ b=0.01 \)

2- \( vRate \) decreases exponentially with send/receive events.

3- The pkt is allowed to be transmitted only after: current time + \( \frac{1}{vRate} \)

The smaller the \( vRate \) is, the longer the time a pkt has to wait is:
The pkt might be dropped before being transmitted

Unlike other inhibition mechanisms, the virtual rate based inhibition allows transmitting the pkt *multiple times* if needed.


**Setting**

**Scenarios:**

- Vehicular networks.
- Different network settings: node density, traffic load...

**Network Simulator:** JIST-SWANS, A JAVA simulator for Ad Hoc networks

**Vehicular Mobility Simulator:** STRAW, an extension of JIST-SWANS. It provides a mobility model based on the operation of the real vehicular traffic.

**Topo:** 2-lanes road, speed limit 80Km/h

**MAC:** 802.11/b

**Channel:** Fading

**Range:** 300 m in average
Adaptation of the Spread to the Rate

\[ \lambda = \frac{R_0}{1 + FF \times N} \]

Density: 12 vehicles/Km

Rate [packets/s]

Spread [vehicles]

Adaptive TTL
Adaptation of the Forwarding Factor to the Density

- Very dense (Traffic jam) +200 neighbors
- Very sparse (Death Valley)
Importance of the Pseudo-Broadcast

Very dense (Traffic jam): +200 neighbors

<table>
<thead>
<tr>
<th></th>
<th>Normal broadcast</th>
<th>Pseudo-broadcast</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel utilization</td>
<td>0.02</td>
<td>0.7</td>
</tr>
</tbody>
</table>

Table 1: Channel utilization in a traffic jam.

Idea: implement a mutual exclusion mechanism for broadcast in order to avoid collision
OUTLINE

- Introduction
- SLEF Middleware: Self Limiting Epidemic Forwarding
- Performance Validation
- SLEF vs. K-Hop broadcast
- Conclusions
K-Hop Broadcast fails if not Adjusted

Very dense (traffic jam): +200 neighbors

TTL = 1
Self adjusted

With SLEF, the rate is 100 times larger than with K-hop broadcast

TTL = K
Congestion collapse
K-Hop Broadcast Needs to Adjust $K$ for each Scenario

**SLEF**

- 2 parameters to adjust: $K_0$, $K_1$
- Once Adjusted, they work well with all settings

**K-Hop Broadcast**

- 1 parameters to adjust: $K$
- Needs to adjust whenever the network setting (density, traffic load, mobility,…) changes

Adjusting the *spread-rate balance* according to the application needs

Default values for $K_0$ and $K_1$ are computed in the paper

\[ \lambda = \frac{R_0}{1 + FF \times N} \]
OUTLINE

- Introduction
- SLEF Middleware: Self Limiting Epidemic Forwarding
- Performance Validation
- SLEF vs. K-Hop broadcast
- Conclusions
Conclusions

We propose SLEF: a Multi-hop broadcast middleware for ad hoc networks

SLEF Features:

✓ **Autonomic**: Adapts itself to any change in the network.

✓ **Complete middleware**

✓ Does not need/exchange *any topology information*. Uses only local information to the node.

✓ Works even in extreme scenarios (very dense/sparse...)

SLEF addresses *new issues* and solves them, such issues are *congestion control in broadcast mode in ad hoc network* and *spread-rate balance*.

*SLEF is a replacement of K-hop broadcast, SLEF works well in all circumstances*
The End

MERCI DE VOTRE ATTENTION