INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

# *Hash-Based Dynamic Source Routing (HB-DSR)*

Claude Castelluccia

**N° 4784**

Mars 2003

THÈME 1

Rapport de recherche

ISSN 0249-6399

# Hash-Based Dynamic Source Routing (HB-DSR)

Claude Castelluccia*

Thème 1 — Réseaux et systèmes
Projets Planete

Rapport de recherche n° 4784 — Mars 2003 — 18 pages

**Abstract:** This paper presents and evaluates *Hash-Based DSR*, an extension of the $DSR$ protocol. This protocol reduces the per-packet control overhead of DSR by compressing the source-route with a Bloom filter. Simulations on large networks show that $HB-DSR$ increases the network capacity by a factor of up to 15. $HB-DSR$ is an attractive alternative to DSR for large ad-hoc networks.

Another important property of $HB-DSR$ is that, as opposed to $DSR$, its performance is similar for IPv4 and IPv6. While IPv6 large addresses is a show-stopper for $DSR$, we show by simulations that $HB-DSR$ performs as well for both IP versions. This is important contribution considering the growing interest of the wireless network community for IPv6.

**Key-words:** Ad-hoc network routing protocol, DSR, routing, Bloom filters

* INRIA Rhône-Alpes, PLANETE group

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38330 Montbonnot-St-Martin (France)
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

# Hash-Based Dynamic Source Routing (HB-DSR)

**Résumé :** Ce rapport présente et évalue *Hash-Based DSR*, une extension au protocole de routage pour les réseaux ad-hoc *DSR*. *Hash-Based DSR* réduit le coût de signalisation du protocole *DSR* en compressant les adresses contenus dans les paquets DSR avec un filtre de Bloom. Les résultats de simulation sur de grands réseaux montrent que notre approche augmente considérablement la capacité du réseau.

**Mots-clés :** MANET, DSR, routage

# 1 Introduction

In an *ad hoc wireless network*, mobile nodes communicate with each other using multi-hop wireless links. Each node is also a router and is therefore part of the routing infrastructure. The bandwidth and the nodes' capacity (processing and power) of such networks are usually very limited. Therefore one of the biggest scientific challenges of this area is to design routing protocols that can efficiently find the routes between any two communicating nodes. These protocols must minimize the control overhead (i.e. the bandwidth overhead used to establish and maintain the routes) as much as possible.

Several protocols have been proposed recently. One of them is the *Dynamic Source Routing* (DSR) protocol [1]. The main characteristic of DSR is that it uses source-routing to route packets from the source to the destination. *DSR* is probably one of the most efficient protocols. However *DSR* has two important limitations:

- *Scalability limitation*: *DSR* uses source-routing and, as a result, does not scale to large networks.
- *IPv6 unfriendly*: IPv6's large address space limits considerably the performance of DSR. This is a important drawback of DSR considering the growing interest of the wireless community for IPv6.

Our goal is to propose an extension to DSR that overcomes these two problems, while still preserving its current features. We propose *Hash-Based DSR*, a protocol that compresses the list of addresses in the source-route using a Bloom filter [2]. Instead of inserting a source-routing option in each packets as in DSR, the source inserts the corresponding Bloom filter.

The rest of the paper is organized as follows. Section 2 reviews Mobile Ad hoc routing protocols and focuses on two proposals: the *Ad Hoc On-Demand Distance Vector* (AODV) and *Dynamic Source Routing* (DSR) protocols. Section 3 presents the proposed *Hash-Based DSR* protocol. Section 4 discusses some implementation issues, such as the packet formats. In Section 5, some simulation results are presented and analyzed. Finally, the last Section concludes the paper and presents some future work.

# 2 Mobile Ad hoc Routing Protocols

The existing Mobile Ad hoc Routing protocols may be categorized as *Proactive* or *Reactive* [3]. *Proactive* protocols attempt to maintain an up-to-date routing information between any two nodes of the networks. The protocols require each node to maintain a routing table and to respond to network topology changes by propagating routing updates. In contrast, *Reactive* protocols create routes only when desired by the source node. When a source desires to establish a route to a destination, it initiates a *route discovery* process. Once the route is established it is maintained by a *route maintenance* procedure as long as the source needs it. Previous studies [4, 5, 6] showed that *proactive* protocols have higher routing load since they constantly propagate routing information in the network. This feature makes *proactive* protocols not well adapted to large and constrained networks. In contrast, *Reactive* protocols only generates routing traffic when necessary and as a result has a much lower signaling load. *Reactive* approaches seems to be more appropriate to large networks.

The two most popular *Reactive* protocols are the *Dynamic Source Routing* protocol (DSR) [1] and the *Ad Hoc On-Demand Distance Vector* protocol (AODV) [7]. The main characteristic of DSR is that it uses source-routing to route packets from the source to the destination. The source knows the hop-by-hop route to the destination and stores it its *routing cache*. In contrast, AODV uses traditional routing tables that contain one entry (specifying the next hop) per destination. Both protocols use a similar route discovery procedure. When a source does not know the route to a destination, it broadcasts a *Route Request* (RREQ) message. Each node receiving it rebroadcasts it, unless it knows a route to the destination or it is the destination. In DSR, a node that forwards a RREQ adds its address in it. When a RREQ has reached its destination or has reached a node that knows a route to the destination, a *route reply* (RREP) message is sent back to the source via the backward path. This RREP is used in AODV to create in each intermediate node one entry towards the source and one entry towards the destination (AODV assumes bi-directional links). In contrast in DSR, this RREP does not create any state in the intermediate nodes but contains the list of addresses that was built by the RREQ. This list is used by the source to update its routing cache. Both protocols use a *Route Error* (RERR) packet to inform the source of a broken link. In DSR, a node that can not route a packet because its next hop is not available sends a RERR to the packet's source. The source removes the corresponding route and initiates a new route discovery process. In contrast, AODV maintains timer-based state in each node. A route that is not used for a given period of time expired. Furthermore, each routing entry maintains a list of predecessor nodes that indicates the set of neighbors which use that entry to route packets. These nodes are notified by a RERR message when the next-hop link break. The RERR is then propagated hop-by-hop until all the sources that use the broken link.

DSR has several attractive properties. For example, it is simpler than AODV. The intermediate nodes do not have to create and maintain a routing table. All the routing information is embedded in the packets. Only the sources are required to cache their routes to their destinations. Furthermore in DSR, the source can have several paths to the destination while in AODV it is limited to one path. This makes DSR more robust to failure since packets can be sent over several different paths. DSR gives more control to the source over which nodes can be used to route its packets. This is a nice property in term of robustness (some nodes might be more reliable than other) and security. Also in DSR, a source can potentially use different metrics (such as power, energy, bandwidth, number of hops, reliability) to select a path from a source to a destination. In [4], the authors showed that although DSR suffers from larger packets (due to the source routing), it generates less routing load (in bytes) than AODV. AODV control messages are smaller but more numerous. Furthermore in contrast to DSR, AODV uses periodic control messages even if the nodes are not moving. DSR is efficient in learning routes in terms of number of control packets used, and does not use periodic control messages. Most of the routing information is carried inside the packets. One important advantages of DSR is that using a single RREQ-RREP a source can learn the route to the destination but also to all the nodes on the path to the destination. Additionally any intermediate node that forwards a RREQ can also learn a route from itself to each of the nodes on the path from itself to the destination. In contract in AODV, route learning is limited only to the source of any packet being forwarded. This usually cause AODV to rely on a route discovery flood more often.

Although DSR is an attractive solution, it has two major limitations:

- DSR does not scale to large networks. In fact, source-routing becomes a severe limitation when the network's diameters increase. This is particularly true if the size of the data to carry is small compared to the size of the packet header.

- IPv6's large IP addresses (128 bits) is a show-stopper for DSR. This is a severe drawback considering the growing interest of the wireless community for IPv6.

The goal of our work is to develop an extension to DSR that overcomes these problems. This paper presents *Hash-Based DSR (HB-DSR)*, a variant of DSR, that can scale up to thousands of nodes.

# 3 Hash-Based DSR

One of the biggest limitations of DSR for large network is that it uses source-routing to route packets. A DSR packet must carry the IP addresses of all the nodes that are on the path from the source to the destination. The generated bandwidth overhead is significant when the source and the destination are far for each other.

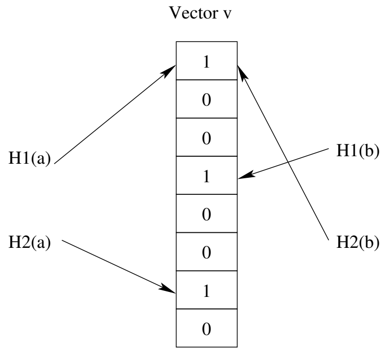
Indeed as we will show later in our simulations (see Figure 7), this control overhead can use up to 90% or 95% of the network bandwith when the path length is about 100 nodes. This overhead is not acceptable and limits the use of DSR to small networks.

We propose, in this paper, *Hash-Based DSR* (HB-DSR) an extension of DSR for large ad-hoc networks. The main idea of our scheme is to compress the list of addresses in the source-route using a Bloom filter. Instead of inserting a source-routing option in each packets as in DSR, the source inserts the corresponding Bloom filter.

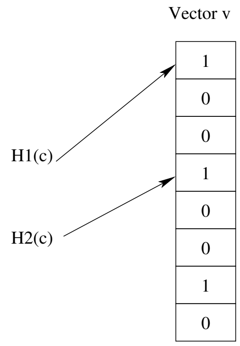
The resulting protocol is very similar to DSR and inherits from most of its features. Furthermore since the filter used in each packet is much smaller that the list of addresses, $HB-DSR$ is much more scalable than DSR for large networks.

## 3.1 Bloom Filter

A Bloom filter is a $m$-bit vector $v$ that codes the membership of a set $A = \{a_1, a_2, .., a_n\}$ of $n$ elements [2]. The idea is to allocate a vector $v$ of $m$ bits, initially all set to 0, and then to choose $k$ independent hash functions, $h_1$, $h_2$,...,$h_k$, each with a range $\{1, ..., m\}$. For each element $a$ of $A$, the $k$ bits at positions $h_1(a)$, $h_2(a)$, ...,$h_k(a)$ in $v$ are set to 1 (see Figure 1(a).




(a) Filter Construction




(b) False Positive

Figure 1: Bloom Filter

To verify whether a element $b$ is the set $A$, it is enough to verify if all the $k$ bits at positions $h_1(b)$, $h_2(b)$, ...,$h_k(b)$ in $v$ are set to 1. If any of them is set to 0, $b$ does clearly not belong to $A$. If all the bits are set to 1, then $b$ is probably a member of $A$, although there is some probably that $b$ is not a member of $A$. This is called a *false positive* (see Figure 1(b)).

The probability a *false positive* can be calculated in a straightforward fashion. After all the elements of $A$ are coded in the Bloom filter, the probability that a specific bit is still set to 0 is: $(1 - 1/m)^{kn}$ The probability of a false positive is then:

$$fp = (1 - (1 - 1/m)^{kn})^k \tag{1}$$

It can be shown that for a given $m$ and $n$, the optimal value of hash functions to used, $k$, is $k = ln2 \times m/n$. There is a clear trade-off between the size of the filter $m$ and the probability of a false positive. For a given $n$, $fp$ can be decreased but increasing $m$. However increasing $m$ reduces the compression rate of the Bloom filter. The optimal value of $m$ is application-specific. For some applications, the false-positive rate must be very small and as a result $m$ must be large. Other applications can tolerate higher false positive rates and therefore can use smaller values of $m$.

## 3.2 Protocol Overview

In $HB-DSR$, the $DSR$ source-routing option is *compressed* using a Bloom filter. The set $A$ is therefore composed of the nodes' IP addresses of the path from the source to the destination. $n$ is equal to the path length.

The proposed protocol is the following:

1. A source $S$ that wants to send packets to a destination $D$, invokes the *DSR route discovery* protocol. It then receives a *Route Reply* that contains the list of the addresses along the path from $S$ to $D$.
2. $S$ then computes from these addresses the corresponding Bloom filter as follows:

```
for (j=1;j<plen;j++) {
```

```
 for (i=0; i<k;i++)
  BF[ hash("i"| Addr_j).mod (m)] = 1;
}
```

where the Bloom filter, $BF$, is a bit-string of size $m$, $k$ is the number of hashes used and $Addr_j$ are the $(plen-1)$ addresses on the path from $S$ to $D$ [1].

3. When $S$ sends packets to $D$, it inserts a new hop-by-hop option that carries the bloom filter and the value of the parameter $k$ (the number of hashes to use).

4. Upon receiving a packet, a node verifies whether the destination address is one of its addresses. If this is the case, the packet has reached its destination. If destination address is not one of its addresses and the $TLL$ is zero, the packet is dropped, otherwise the node verifies if any of its neighbors' addresses (except the one it received the packet from) are contained in the Bloom filter carried in the packet using the following algorithm:

```
for (j=1;j<=R;j++) {
 IsMember[j]=1;
 for (i=0;i<k;i++)
  if (BF[hash("i"|addr_j).mod(m)] == 0)
   IsMember[j]=0;
 }
```

where $R$ is the number of neighbors $N_j$ defined by their address, $addr_j$.

If when the algorithm terminates, $IsMember[j]$ is set to 1 then the neighbor $N_j$ belongs to the filter and is therefore a node on the path from $S$ to $D$, otherwise $N_j$ does not belong to the filter.

If a neighbor belongs to the Bloom Filter, the packet's TTL is decremented and the packet is forwarded to that neighbor. If not, the packet is silently dropped (the packet was probably mis-routed to it after a false positive). If they are several neighbors that are contained in the packet's filter (this is the result of false positives), the packet is duplicated and forwarded to each of these neighbors.

As a result of this protocol, the packets are forwarded hop-by-hop until the destination.

It is clear from the protocol description that the filter's size is a crucial parameters. Indeed if the filter's size is too small, false positives will be frequent and the packets will be mis-routed. Although the packets will reach their destination, the bandwidth overhead will be quite large. On the other hand, if the filter's size is very large, false positives will be rare (and therefore no many packets will be mis-routed) but the size of the packets will be larger (since they carry the filter) and the benefit of the compression will be reduced.

In our scheme, the source computes the *optimal* filter size according some cost function. This cost estimates the filter's size that minimizes the total bandwidth overhead. The source then adjusts the size using some feedbacks from the networks. This two algorithms are described in the rest of this section.

## 3.3 Filter Size Computation

**Notation:**

In this section, we define and use the following notation:
*plen:* is the path length from source to destination.
*R:* is the average number of neighbors per intermediate node.
*fp:* is the false positive probability.

[1] The following sections describe how the parameters $m$ and $k$ are choosen.

*IP_addr_size*: is the size of an IP address.
*IP_head_size*: is the size of an IP header.
*IP_opt_size*: is the size of the IP option header that carries a Bloom filter.
*pkt_size:* is the total size of a packet including its IP header, options and its data.
*m*: is the size of the Bloom filter.

In DSR, the routing overhead resulting from the source routing is defined by:

$$dsr\_cost = plen \times [(plen - 1) \times IP\_addr\_size + IP\_opt\_size] \quad (2)$$

IP_addr_size is equal to 32 bits for IPv4 and 128 bits for IPv6.

In our scheme, the routing cost results partly from the false positives. If at a node, a false positive happens the packets will be sent to one or several wrong paths and these mis-routed packets will consume some extra bandwidth. Let's consider the scenario displayed in Figure 2 In this figure, $S$ is sending packets to $D$. The path length, *plen* is equal to 3.

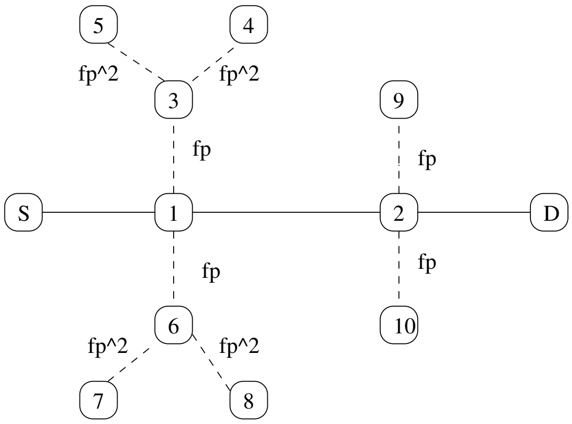



Figure 2: HB-DSR Routing

At the first node, *N1*, the probability that a false positive happens towards N3 (resp. N6) is defined by *fp*. The resulting cost is $fp \times pkt\_size$. The probability that the forwarded packet is forwarded to N4 (resp. N8) or to N5 (resp. N7) is $fp^2$. The induced cost is then $fp^2 \times pkt\_size$. The probability that the forwarded packets is forwarded by N4 or N5 is zero because the TTL has then reached zero. The total cost at node N1 is therefore:

$$Cost_{N1} = 2 \times pkt\_size \times [fp + (R - 1) \times fp^2] \quad (3)$$

Similarly at the second node, N2, the probability that a false positive happens towards N9 (resp. N10) is *fp*. The resulting cost is $fp \times pkt\_size$. The probability that this packet is forwarded by N9 (resp. N10) is zero because the TTL is then 0. The total cost at node *N2* is then:

$$Cost_{N2} = 2 \times pkt\_size \times fp \quad (4)$$

More generally, the cost at node *Nx* (with $0 < x < plen$) is defined by:

$$Cost_{Nx} = (R - 2) \times pkt\_size \times fp \times \sum_{i=0}^{plen-x-1} ((R - 1)^i \times fp^i) \quad (5)$$

As a result, the total cost resulting from the false positives is defined as follows:

$$\begin{aligned} fp\_cost &= \textstyle\sum_{j=1}^{plen-1} Cost_{Nj} \\ &= (R-2) \times pkt\_size \times fp \times [\textstyle\sum_{j=0}^{plen-2}((plen-j-1)\times(R-1)^j \times fp^j)] \end{aligned} \tag{6}$$

where *fp* is computed as follows:

$$fp = (1/2)^{ln(2) \times m/plen} \tag{7}$$

The total HB-DSR overhead is the cost resulting from the false positive and the cost of carrying the filter in each packet, i.e:. Therefore:

$$hbdsr\_cost = plen \times (IP\_opt\_size + m) + fp\_cost \tag{8}$$

Figure 3 plots the $hbdsr\_cost$ function according to $m$ for several path length ($plen$). This figure shows that, for a given path length, there is only one value of $m$ that minimizes the cost function.

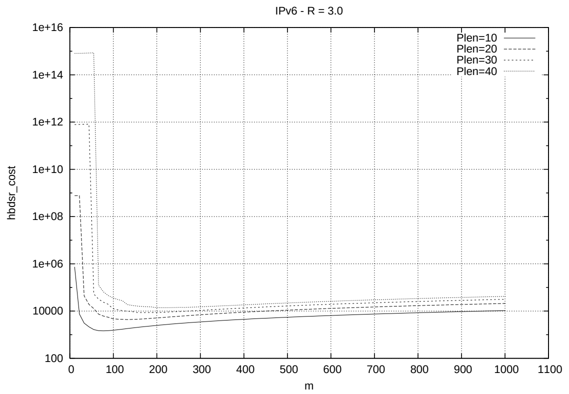



Figure 3: HB-DSR Cost vs m

In HB-DSR, when a source wants to send packet to a destination it performs a DSR route discovery. It then receives a Route Reply that contains the list of addresses along the path. It can deduce from this information the path length ($plen$) but still need to estimate the parameter $R$. We propose to extend the route request and reply messages with a *CN (Cumulative Neighbors) field*. The field is set to 0 in the the route request by the source. Each intermediate node that forwards it increments this CN field by the number of its neighbors. When the destination receives the route request, it copies the CN value in the route reply and send it to the source. The source can then get an estimate of the parameter $R$ by dividing the value $CN$ by the path length ($plen$).

Once the source has an estimate of $plen$ and $R$, it computes for several $m$ the value of the cost function $hbdsr\_cost$ and then uses the value of $m$ that minimizes it.

### 3.4 Closed-loop algorithm

In the previous section, the source computes the filter size using a cost function. However in some scenarios, the source can still suffer from the cost of false positives. An closed-loop algorithm is often more efficient and practical. We therefore propose to use some feedbacks from the network to tune the filter size.

We define, *FP_DUP*, a new error message. When a node, $N_i$, detect ts one or several false positives (i.e. there are more than one neighbor, let's say $r_i$ neighbors, in the filter), it returns a *FP_DUP* message to the source. This message contains the number of neighbors contained in the filter, i.e. $r_i$, at node $N_i$.

In order to avoid the explosion of *FP_DUP* messages and to avoid routing loop, we propose that nodes drop packets that have experienced two or more false positives. When a node detects a false positive upon reception of a packet, it sends a *FP_DUP* to the source and set a bit (the $D$ bit) in the packet to 1. If this bit is already set, the packet is dropped and, instead of a *FP_DUP* message, a *FP_DROP* message is sent to the source, otherwise the packet is forwarded (see Figure 4).

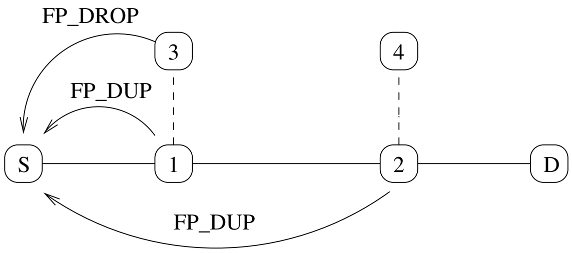



Figure 4: Closed-loop algorithm

The source executes the following algorithm:

- *Step1*: If the source receives one or several *FP_DROP* message, it increases the the filter size by one unit (i.e as shown in Section 4 by 64 bits for IPv6 or 32 bits for IPv4). The algorithm is then re-executed.
- *Step2*: If the source receives one or several *FP_DUP* (but no *FP_DROP* message) that specify $N$ false positives, the source evaluates whether it can live with the generated false positives or if it should increase the filter's size. The source computes *FP_cost*, the cost resulting from the $N$ false positives:

$$FP\_cost = N \times pkt\_size \tag{9}$$

It also computes, *Delta_cost*, the cost resulting from increasing the filter size by one unit, $BF\_unit$.

$$Delta\_cost = plen \times BF\_unit \tag{10}$$

If $Delta\_cost < FP\_cost$ then the filter's size is incremented by one unit and the algorithm is re-executed. Otherwise the filter size is kept to $m$ and the algorithm terminates here.

- *Step3*: If the source does not receive any *FP_DUP* nor any *FP_DROP* messages for a given value of $m$ and if this $m$ was not obtained from *Step2*, the source decrements $m$ by one unit and the algorithm is re-executed. Otherwise the algorithm terminates here.

# 4 Implementation Issues

This section presents some of the implementation issues of $HB-DSR$. It describes the new hop-by-hop option (for IPv6 and IPV4) and some possible optimizations.

## 4.1 Packet Format

$HB-DSR$ uses the same control messages, such as $RREQ$, $RREP$ or $RRER$, than $DSR$. However in contrast to $DSR$ that inserts in each packet a source-route, $HB-DSR$ carries in each packet a Bloom filter. This Bloom filter is generated from the source-route addresses that DSR uses. The Bloom filter is carried in a new hop-by-hop option, the *BF option*. Note that $HB-DSR$ can inter-operate and co-exist with DSR. When a node receives a packet to route, it looks at its options. If the options contain a source-route option, then regular DSR must be invoked. If the option contains a *BF option*, $HB-DSR$ must be invoked instead. The formats of this new option, for IPv6 and IPv4, are defined as follows:

### 4.1.1 IPv6 hop-by-hop BF option

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Hop-by-Hop=0  |  Header Len   |      Type???  |     l= 4+8.i  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|D|    k        |                 BF                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| ...                                                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+..
```

The field *Type* (8 bits) carries the BF option type (to be defined). The field $D$ (1 bit) is the the duplicated bit. This bit is set by any intermediate node that encounters a false positive and duplicates a packet. The field $k$ (7 bits) contains the number of hashes to use to verify the filter. The field *BF* (24 + 64.i bits) contains the Bloom filter.

Note that since the *Header length* field contains the length of the hop-by-hop option header in 8-octet units, not including the first 8 octets, we propose, in order to avoid padding, to use a bloom filter of size 24 + x.64. In IPv6, $IP\_opt\_size$ is therefore equal to 32 bits and $m$ to $24 + 64 \times i$, where $i$ is an integer.

### 4.1.2 IPv4 BF Option

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Option Type  |  Opt Data Len |D|     k       |       BF      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           ... BF                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

As for IPv6, the field *Option Type* contains the BF option type (to be defined). The field $D$ is the the duplicated bit. This bit is set by any intermediate node that encounters a false positive and duplicates a packet. The field $k$ contains the number of hashes to use to verify the filter. The field *BF* contains the Bloom filter.

Note that if any headers follow the DSR header in a packet, the total length of a DSR header, indicated by the *Payload Length* field in the DSR header must be a multiple of 4 octets. In this case, when building a DSR header in a packet, sufficient padding must be included in the Options field of the DSR header to make the total length a multiple of 4 octets. As a result, the BF length (i.e the option len) should be $16 + 4 \times i$ octets to avoid padding (i.e. 16, 48, 80,...) As a result, in IPv4, $IP\_option\_size$ is equal 48 bits and $m = 8 + 32 \times i$

## 4.2 Bloom Filter Caching

In $HB - DSR$, a node that receives a packet to be routed must compute for each of its neighbors their Bloom filter, ($NF_i$), and verify whether one of them is contained in the packet's filter ($PF$).

A neighbor whose filter is $NF_i$ is contained in the packet's filter $BF$ if all the $k$ bits set to 1 in $NF_i$ are also set to 1 in $BF$. For a given neighbor, defined by its IP address $Addr_i$, and a given filter size $m$, the $k$ bits, $nf_{i,j}$, set to 1 are computed as follows (for $j = 1, ...k$):

$$nf_{i,j} = hash("j"|addr_i)mod(m). \tag{11}$$

, where $m$ is the size of the filter.

If the node has $R$ neighbors and if the packet's filter uses $k$ hashes, the node needs to compute $R \times k$ hashes per packet. The resulting per-packet computing cost might overwhelming for some very small devices. However, a node can trade memory with computing processing. In fact, a node can pre-compute for several values of $k$, the hashes of its neighbors, $hash("j"|addr_i)$ and cache this information. As a result, when a node receives a packet, it only has to retrieve, for the packet's $k$ value, the corresponding hashes of its neighbors and compute the modulo $m$ operation to reconstruct the neighbors' Bloom filters.

If MD5 is used as the hash function, a node will need $16 \times R \times K$ bytes to store the $K$ filters corresponding to its $N$ neighbors. For example, if a node has $R = 4$ neighbors, and pre-computes the filters for $K = 5$ values of $k$, it will only need 320 bytes.

### 4.3 Route Error messages

In $DSR$, if any link on a source route is broken, the source node is notified using a *Route Error* (RERR) message. In $HB - DSR$, it is possible and highly probable that a node will not know how to route a packet that was generated by a previous node as the result of a false positive. In this case, the node must not send a $RRER$ to the source. In $HB - DSR$, a $RRER$ must only be sent if (1) a node does not know how to route a packet and if the $D$ bit is not set, (2) a node had a route but this route is not active anymore (i.e. the neighbor has moved away or is down).

### 4.4 Promiscuous listening

One important advantages of $DSR$ is that using a single $RREQ - RREP$ a source can learn the route to the destination but also to all the nodes on the path to the destination. Additionally any intermediate node that forwards a $RREQ$ can also learn a route from itself to each of the nodes on the path from itself to the destination. Since $HB - DSR$ uses the same route discovery process than $DSR$, $HB - DSR$ inherits this feature. Additionally in $DSR$, a node that listen promiscuously the transmissions can learn routes to each of the nodes on the source-route of the transmitted packets. Since in $HB - DSR$, the source-route is compressed, this promiscuous listening is not possible anymore. If that was a problem, a source can periodically switch to standard $DSR$ (for example once for every 100 packets) to transmit several packets.

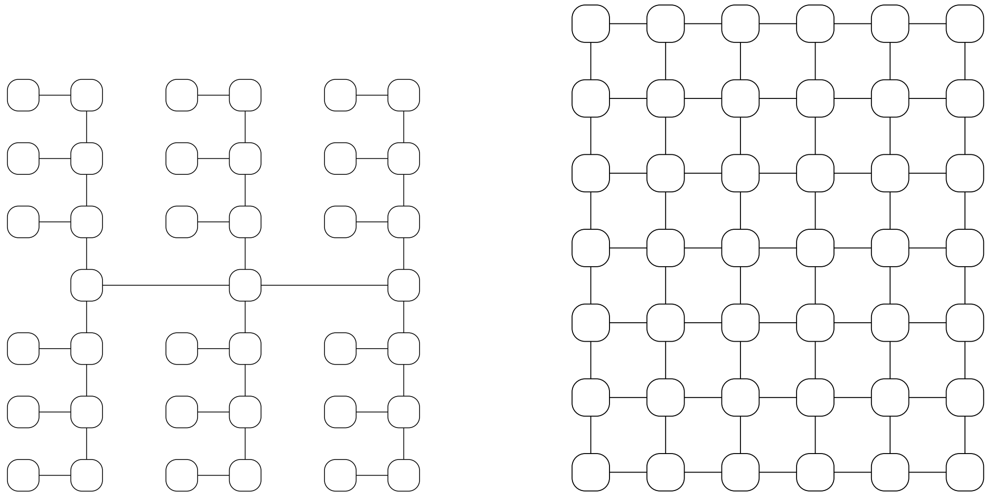
## 5 Simulation

### 5.1 Simulation Model

The main goal of our simulations is to evaluate the gain of our approach over $DSR$. We mainly focus our simulations on the per-packet bandwidth gain achieved by compressing the source-routes with Bloom filters. We do not consider the control messages (such as RREP, RREQ or RRER) overhead since they are identical in $DSR$ and $HB - DSR$. Furthermore for simplicity, we do not consider mobility in our simulations. The main contribution of our scheme is to reduce the per-packet control overhead. All the rest, including mobility management, is similar to DSR. We expect the mobility management performance of DSR and HB-DSR to be very similar.

We considered two different networks in our simulations (see Figure 5). Both of them contain 3600 nodes. The topology of the first one (MANET1) is a tree and as a result there is only one possible path from a given source to a given destination. The second one (MANET2) is highly connected (a node has on average 4 neighbors while in MANET1 a node has on average 2.8 neighbors) and there are several paths for a given source to a given destination.

For each of the network, we select randomly 2 nodes, the source and the destination. We then simulate the routing of four different packets from the source to the destination using $DSR$ and 3 variants of $HB - DSR$. The first variant, *HB-DSR*, use a filter with a fixed size (88 bits for IPv6). In the second version *AHB-DSR0* (*A* stands for *Adaptive*), the source computes the filter size using the cost function described in Section 3.3. The third



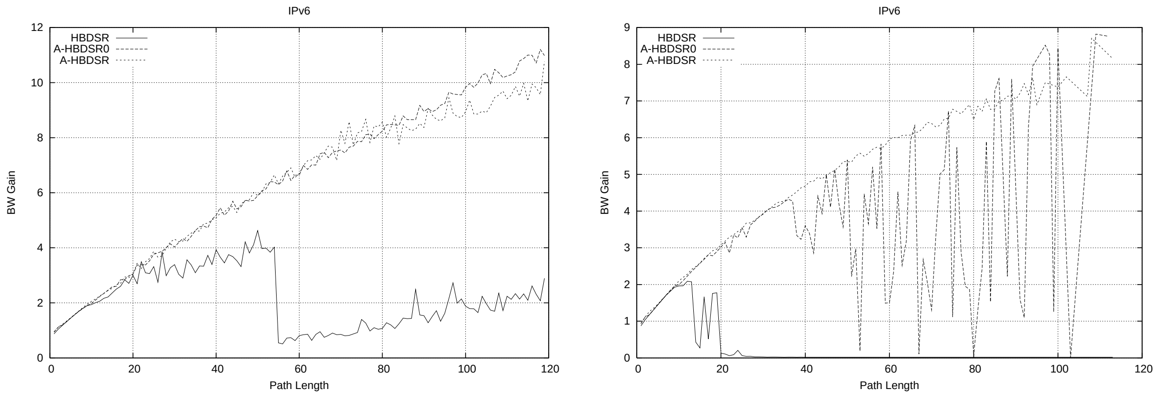
(a) MANET1 (b) MANET2

Figure 5: Network Topology

version, *AHB-DSR*, is similar the previous section but in addition to the cost function, the source adjusts the filter size using feedbacks from the network (The $FP\_DUP$ and $FP\_DROP$ messages) as described in Section 3.4.

Each of the 4 packets used has a different data size. The first one carries 8 bytes of data, the second 64 bytes, the third 256 bytes and the fourth 1024 bytes. We run these simulations 2500 times for IPv6 and IPv4.

## 5.2 Performance Results



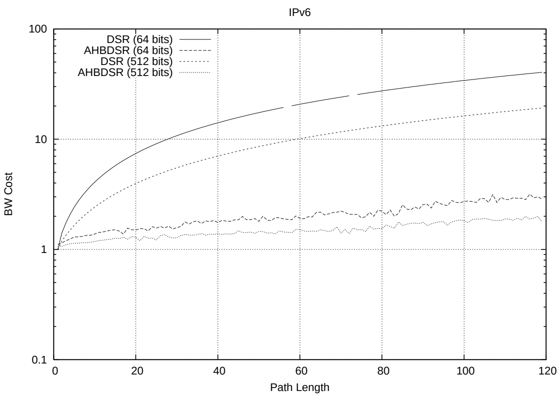

(a) MANET1 (b) MANET2

Figure 6: Bandwidth Gain (IPv6; data-size = 64 bytes)

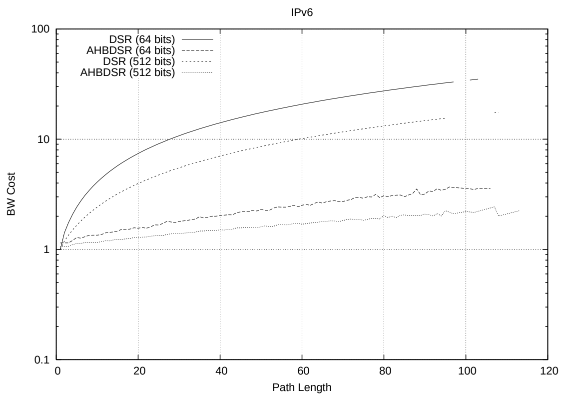
Figure 6 shows the *bandwidth gain* of $HB-DSR0$, $HB-DSR$, $AHB-DSR$ over $DSR$ for IPv6 according to the path length between the source and the destination. This gain of a proposal is defined by the bandwith, used by DSR to transmit a packet containing 64 bytes of data, divided by the bandwidth used by the proposal to transmit the same data to the destination. Each hop-wise transmission is a counted as one transmission. We assume here that the route discovery phase has been performed and that the source knows the route to the destination.

The results show that:

- $HB-DSR$: When $m$ is fixed and equal to 88, the gain decreases considerably with the path length. These results were expected because when the path length is large, the false positive rate is close to 1 and most of the transmitted packets are broadcast. For MANET1, the gain is always greater than one because the network is not highly connected and the broadcast cost is not too high (and a least lower than source-routing). In contrast, for MANET2, the gain converges to 0. MANET2 is highly connected and the cost of broadcasting a packet is very large. It is more efficient to use source-routing for this scenario.
- $AHB-DSR0$: The performance for MANET1 and MANET2 are very different. With MANET1 the gain is always greater than one and increases with the path length. When the path length is 100 nodes, the achieved gain is 10. This means that the network can accommodate 10 more connections than if DSR was used. With MANET2, $AHB-DSR0$ does not perform well and its performance fluctuates a lot. These results are explained by the different costs of a false positive in each networks. In MANET1, the cost of a false positive is not too high because the network is not very well connected and the mis-routed packets die out quickly. In MANET2, the false positive cost is quite high. In fact, since each node has a higher number of neighbors, the probability of a false positive is larger and as a result, false positives are more frequent. Furthermore, if a packet is mis-routed twice in a row, it can reach the correct path again and enter a routing loop.
- $AHB-DSR$: $AHB-DSR$ corrects the problems of the $AHB-DSR0$. By using feedbacks, the source can adjust the value of $m$ accurately and minimizes the number of mis-routed packets. The gain increases with the path length. The gain achieved for MANET1 is larger than the gain achieved with MANET2. Indeed, since the connectivity is smaller, fewer bits are required per bloom filters in MANET1.




(a) MANET1

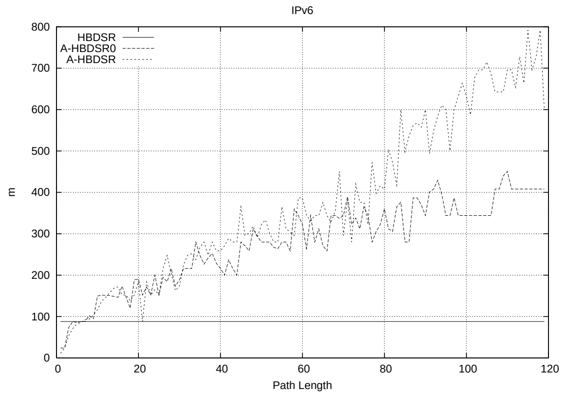



(b) MANET2

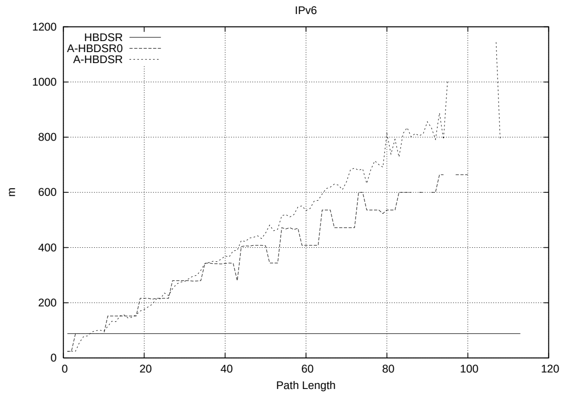
Figure 7: DSR and AH-BDSR Bandwidth Cost (IPv6)

Figure 7 presents the *bandwidth cost* of $AHB-DSR$ and $DSR$. The bandwidth cost of proposal is computed by dividing the number of bytes necessary to transmit a given piece of data from the source to the destination using the proposal by the the number of bytes necessary to transmit the same data with regular IP. This cost actually measures the control cost resulting from carrying in each packet a source-route or a Bloom Filter. We consider two data-sizes : 8 and 64 bytes.

The results show that DSR cost increases drastically with the path length. In fact since a packet carries the addresses of all the nodes from the source to the destination, the control cost increases with the path length. When the path length is 100, the cost of DSR for a data size of 64 bytes is 20. This means that the cost of sending 64 bytes from the source to the destination is 20 times larger that the cost needed to transport the same data in a regular IP packet (i.e. without the source-routing option). The cost to transport these data with $AHB-DSR$ goes down to 3. This is the result of compressing the source-address with a Bloom filter. As we will see later in this section, this cost is much lower when the data size is larger.

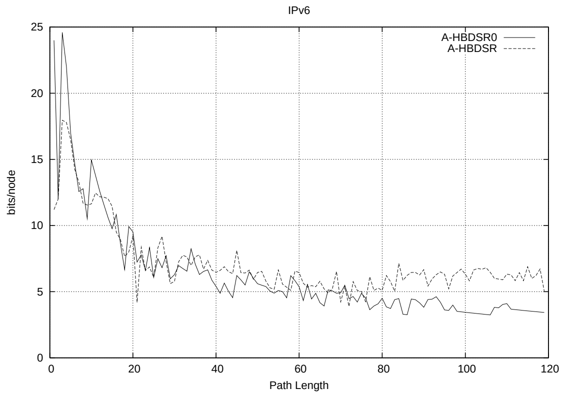



(a) MANET1

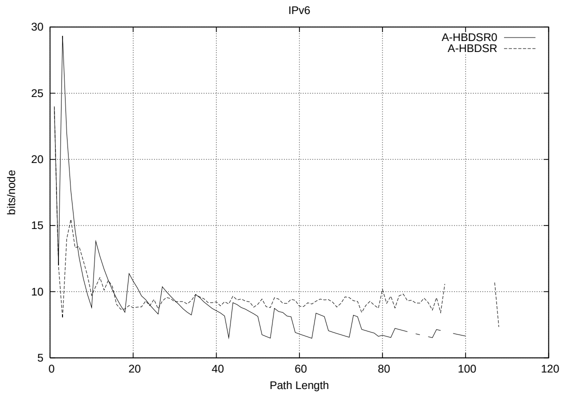



(b) MANET2

Figure 8: Bloom Filter size (IPv6)



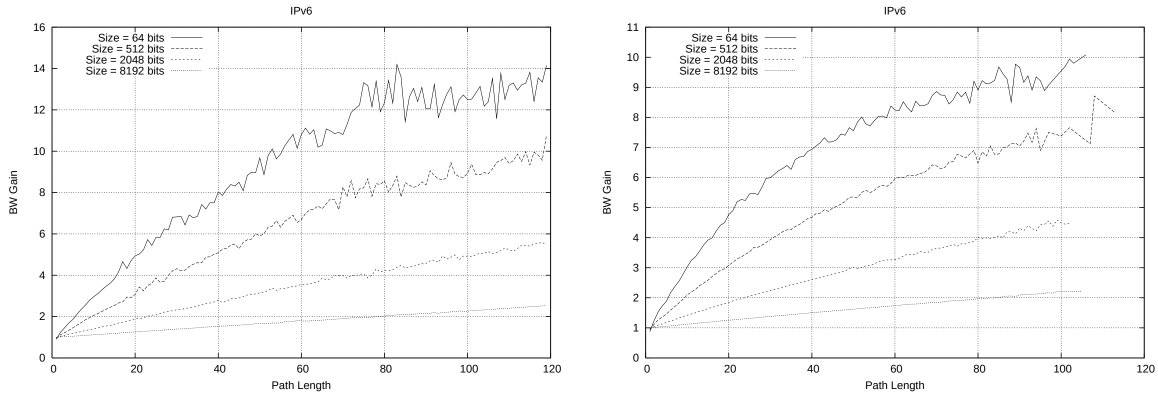

(a) MANET1




(b) MANET2

Figure 9: Number of bits per node (IPv6)

Figures 8(a) and 8(b) display the size of the bloom filter (i.e. the parameter $m$) used by our proposals. $AHB-DSR$ uses slightly larger filters than $AHB-DSR0$ to prevent false-positives and their resulting cost. Figures 9(a) and 9(b) display the equivalent number of bits used to code each node of the source-routing in the Bloom filter. These results summarize the compression rate of our proposal. For example, for MANET1, $AHB-DSR$ needs on average 6 bits per node when the path length is 100. This has to be compared with DSR that need 128 bits (i.e. an IPv6 address) per node. In other words, with $AHB-DSR$ we compress the per-packet source-routing bandwidth overhead of DSR by a factor larger than 20.




(a) MANET1 (b) MANET2

Figure 10: AH-BDSR Gain vs. data size (IPv6)

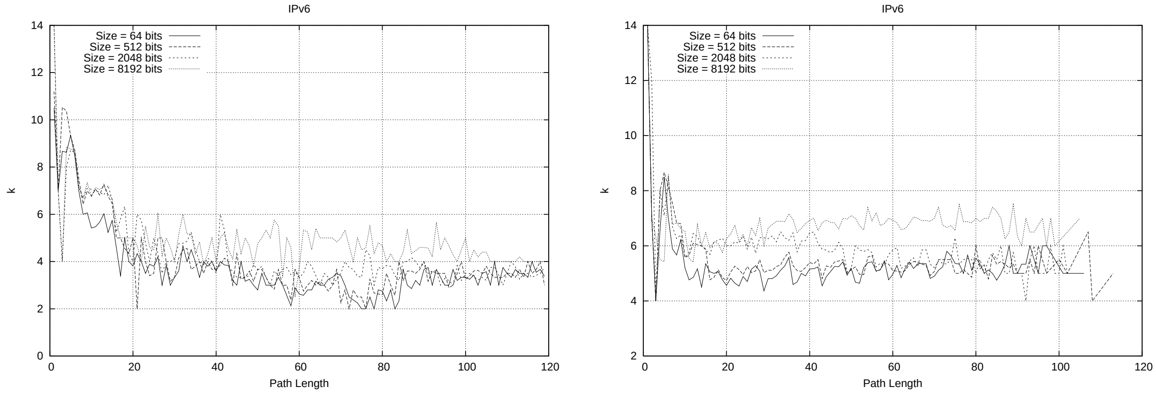
Figures 10 display the badwidth gain (compared to DSR) of $AHB-DSR$ for 4 different data-sizes (8, 64, 256 and 1024 bytes). These results show that the gain is larger when the data-size is smaller. The gain achieved with a data-size of 8 bytes is more than twice the gain with a data size of 256 bytes. When the data-size is small, the control overhead due to the source-routing is very large and we gain a lot by reducing it. Our proposal is more attractive when the path is large and data to transfer is relatively small. For example, for a data-size of 8 bytes and a path length of 100 the gain is about 13 for MANET1 and 10 for MANET1. This basically means that a network using $AHB-DSR$ as routing protocol can transport 13 (or 10) more packets than the same network using regular $DSR$.

Figures 11 show the number of hashes (i.e. the value of the parameter $k$) to compute and verify the membership according to the path length. These results illustrate the computing cost per packet of our proposal. They show that, if no caching is used, $AHB-DSR$ needs to compute 4-6 hashes per packet. The resulting cost is acceptable. If caching is used, the hashes can be pre-computed and cached, as explained in Section 4.

Figures 12 compare the performance obtained with IPv4 and IPv6. They display the bandwidth (in bytes) used by $DSR$ and $AHB-DSR$ to transmit 64 bytes of data from the source to the destination with IPv4 and IPv6. Each hop-wise transmission is counted as one transmission. These results show:

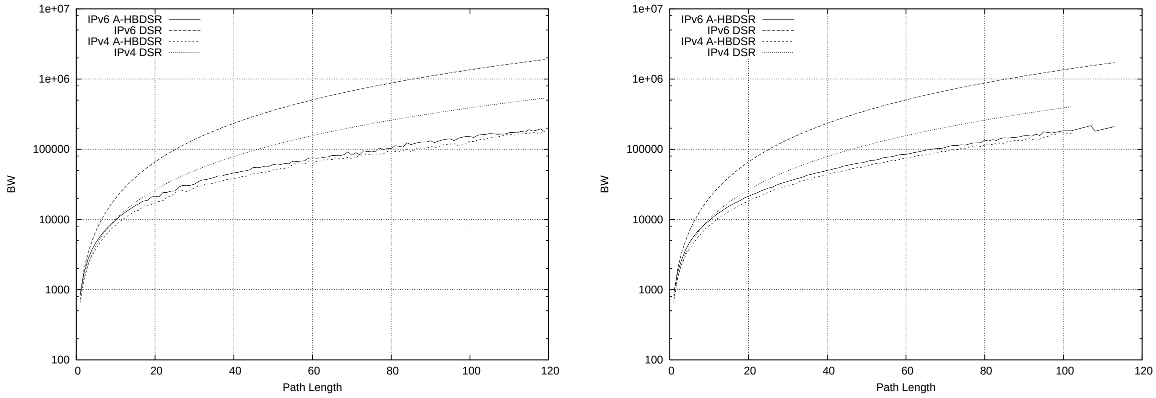
1. $DSR$ is more expensive in IPv6 than in IPv4. In fact since addresses are much larger in IPv6 than in IPv4, the source-routing overhead is much larger. As a result, IPv6-DSR is more expensive than IPv4-DSR in term of bandwidth.

2. IPv4 and IPv6 $AHB-DSR$ costs are very similar. The filter's size used by $AHB-DSR$ is independent from the IP version. The resulting cost is then identical despite IPv6 larger address space. This result is very encouraging. While IPv6 is not appropriate and inefficient in DSR, it becomes very attractive for $AHB-DSR$.




(a) MANET1 (b) MANET2

Figure 11: Number of hashes (IPv6)




(a) MANET1 (b) MANET2

Figure 12: IPv4 vs IPv6 Bandwidth (data-size= 64 bytes)

# 6 Conclusions

We have presented and evaluated *Hash-Based DSR*, an extension of the $DSR$ protocol. This protocol reduces the per-packet control overhead of DSR by compressing the source-route with a Bloom filter. The simulation results on large networks (diameter of 100 nodes) show that $HB-DSR$ increases the network capacity by a factor of up to 15.

Another important benefit of $HB-DSR$ over $DSR$ is that its performance is similar for IPv4 and IPv6. In fact while IPv6 large addresses is a show-stopper for $DSR$, we showed by simulations that $HB-DSR$ performs as well for both IP versions. We expect this result to generate a lot of interest in the IPv6 community.

Our future work will consider the following items:

- *group communications*: $HB - DSR$ can be extended to support *one-to-many* communication. In fact, if a source knows the source-routes to each of its destinations, it can build the Bloom filter that contains all the nodes of the delivery tree. By inserting such filter in a packet, the packet will be routed from the source to its destinations using the optimal delivery tree. We believe that this approach is promising for small groups.
- *Sensor-networking*: One potential application that deserves special attention for $HB - DSR$ is sensor-networking. Sensor networks are very large and have very limited capacities. $HB - DSR$ might be a solution for such networks. However, the *route discovery* mechanism of $DSR$ relies on flooding. We expect this mechanism to be overwhelming for large networks. An alternation route discovery protocol might be necessary for such networks. Future work will study this issue.

## References

[1] *The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks*, chapter 5, Addison-Wesley, 2001.

[2] C. Bloom, "Space/time Tradeoffs in Hash Coding with Allowable Errors," *CACM*, 1970.

[3] Elizabeth Royer and Chai-Keong Toh, "A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks," *IEEE Personal Communications*, 1999.

[4] Charles Perkins, Elizabeth Royer, Samir Ras, and Mahesh Marina, "Performance Comparison of Two On-Demand Routing Protocols for Ad Hoc Networks," *IEEE Personal Communications*, 2001.

[5] Per Johanson, Tony Larsson, and Nicklas Hedman, "Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks," .

[6] Josh Broch, David Maltz, David Johnson, Yih-Chun Hu, and Jorjeta Jetcheva, "A Performance Comparison of Multi-hop Wireless Ad Hoc Network Routing Protocols," .

[7] *The Ad Hoc On-Demand Distance Vector Protocol*, chapter 4, Addison-Wesley, 2001.

# Contents

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399