# Abbreviated Dynamic Source Routing: Source Routing with Not-Unique Network Identifiers

Miguel A. Ortuño, Vicente Matellán, Luis Rodero and Gregorio Robles
Departamento de Informática, Estadística y Telemática
Universidad Rey Juan Carlos, Móstoles, Madrid, Spain
Email: mortuno,vmo,lrodero,grex@gsyc.escet.urjc.es

***Abstract*— Current ad-hoc network protocols are designed for hosts similar to those used in fixed networks. Those protocols are not adequate for some applications of ad-hoc networks, where used resources are very scarce. One example is the size of the network addresses, which may be a critical issue, specially with the use of IPv6 in DSR. This is due to the fact that this protocol uses source routing, so each datagram must carry the addresses of all the machines in its path. In this paper a new protocol named ADSR is proposed to solve this problem. This new protocol is a modified version of DSR based on using abbreviated addresses. The abbreviation procedure can lead to two different nodes having the same address, what we will named 'collision'. ADSR does not avoid but allow collisions, which is analyzed in this paper. Some results about this new protocol performance are shown. This results have been obtained by simulations implemented in ns-2 network simulator.**

## I. Ad-Hoc Networks

*Ad-hoc* networks are computer networks settled-up when needed, composed by the hosts that happen to be in a certain place at a certain time, without any fixed infrastructure like, for instance, the existing Internet. Wireless links, batteries power and the right algorithms free those networks from the need of wires, access points, routers or external power [1].

Those networks are made by similar nodes with no hierarchy. Each host is able to produce, route and consume data. It is also desirable not to need the assistance of any user or human administrator.

If the link layer allowed every node to reach any other, routing would not be an issue. But this is very unlikely: Most of the times it would lead to a great waste of energy (high transmission power) and a bandwidth fall (rise of hosts competing for the medium).

So, most of the times every node will be able only to reach closest neighbors, relaying in the net to reach more distant nodes. In good faith, all the nodes will act as packet data routers, producers and consumers. As every node can play all this roles, there is no single point of failure, which is a very interesting feature.

This kind of technologies can be useful in many situations: Conferencing, emergency situations where the infrastructures are lost, military scenarios where fast communications deploy is needed, etcetera.

### A. Routing protocols for Ad-Hoc networks

Classical routing protocols as those used in fixed networks (i.e. IP in Internet) are not well suited for Ad-Hoc networks, because their routing tables do not stabilize under frequent changes in connectivity due to the mobile nature of the nodes.

Several protocols for ad-hoc networks have been developed since middle of 90's. We can split them into two different groups: based on *proactive* or *reactive* routing.

Protocols based on proactive routing permanently update a table that allows them to route to *any* destination. This kind of protocols send lots of routing information to adapt the table to changes in the network connectivity. Examples of this category of protocols are: DSDV (*The Destination-Sequenced Distance-Vector Routing Protocol*) [2], CGSR (*Clusterhead Gateway Switch Routing*) [3] and WRP (*The Wireless Routing Protocol*) [4].

On the other hand, protocols based on *reactive* routing only store in their tables the routes that have been needed so far. When a packet addressed to an unknown destination is received by a robot, a route discovery process will be initiated on demand in order to learn such a new route. A route maintenance process is also needed to update the routes learned and to delete the unused ones. Some examples of this category are: AODV (*Ad Hoc On-Demand Distance Vector Routing*) [5], DSR (*Dynamic Source Routing*) [6], LMR (*Lightweight Mobile Routing*) [7] and TORA (*Temporary Ordered Routing Algorithm*) [8] .

Nowadays two protocols, both reactive, stand out: DSR (described in next section) and AODV. AODV is based on *Distance Vector* (DV), also known as DBF *Distributed-Bellman-Ford*, where each node knows the first hop to reach any host in the net and the length of the path.

Some studies [9] [10] show that DSR offers better routing overhead and better performance that AODV in scenarios with low or medium *stress* (data load, number of nodes and mobility), whereas AODV seems to be more suitable for more demanding environments.

### B. DSR Protocol

DSR is a *source routing* protocol that works completely on-demand, there is no activity unless it is required. When the transmitter moves or when the topology changes, the algorithm perceives those changes and adapts itself to them, but only for paths currently on use. Sometimes it is refereed as a *2.5 layer* protocol, as often can be found between a link layer and a network layer. DSR is based on two mechanisms that work coordinated: *Route Discovery* and *Route Maintenance*:

*1)* Route Discovery*:* Let's suppose a net as shown in fig 2. Node A is supplied with some wireless technology that allows it reach node B, but not further. Node B reaches A, C and E, and so on. Node A wants to send a packet to node D, but does not know any route, so begins a Route Discovery

- Node A broadcasts to all its neighbors a *Route Request* to D. If the receiver is not D, it forwards the request. That is the classical flooding algorithm [11]. To limit the overhead, each request has an unique identifier, so each host forwards it only once.
- At every hop in the route request, the node forwarding the request adds its own address to the datagram, so the path the datagram follows is logged.
- If the Request arrives to D, it gets the path in the datagram (*ABCD* ) and sends it back to A by a *Route Reply*. As usual in *source routing*, the reply follows the reverse path the request made.
- When A knows the route to D, it includes it in all the packets addressed to D, using source routing.
- Every node keeps a cache of known routes, so any node receiving a Route Request (and not only the addressee) can answer if it knows a intended path.

*2)* Route Maintenance*:* Every node is responsible for the packet it sends to reach next hop. In fig. 2, when B receives a packet from A, it must make sure that the packet arrives to C. If the underlying link protocol offers that service (as IEEE-802.11 does), this requisite means no additional effort.

So, if A sends data packets to D, and node C moves out of B's scope, then B will realize and will send to A a *Route Error* message, to report that the route is no longer valid.

- The protocol can be classified as *best effort*: if a route *falls*, the data is not sent again. If appropriate, resends will be done by upper layers of the communications stack.
- Next time node A has a packet to D, it will use an alternative path (if already known), or begin a new Route Request.

## II. Limited Resources Devices

Hardware is continuously improving its features and as its price gets lower, but even though there are always (and we dare to say that will always be) devices endowed with wireless communication capabilities, but with limited resources: Because of price, battery power or radio-electric spectrum restrictions: PDAs, electrical appliances, toys, sensor networks, industrial equipment, etc.

Let's see the equipment used by our students in the undergraduate robotics course: Lego Mindstorm RCX. It has a Hitachi H8/300 processor, 16k ROM, 32k RAM. We can compare its features with a micro-computer of the eighties, the Sinclair ZX-Spectrum. It has an infra-red port that allows it communicate with other RCX or with a PC. It communications protocol, LegOS, has a 256 bytes frame. Similar devices have this kind of frame or even smaller.

If we just try to port DSR in IPv4 over that sort of machine, a very significant part of the datagram would be occupied by the headers. Headers size is variable. Taking as a reference the DSR implementation available for the network simulator ns-2 [12], about 88 bytes would be necessary: one third of the whole frame.

If DSR was used under IPv6, about 288 bytes would be needed, it would be non-viable in the system we propose as reference. Under the same circumstances, the protocol we propose would need 46 and 65 bytes ( fig 1).

## III. Proposed Solution

### A. Collisions

As we saw, when using source routing in certain kind of architectures appears the necessity to reduce the header's size. One way of achieve that is reduce the size of the addresses, to reduce the size of the routes.

Making the addresses shorter in an arbitrary way leads to the possibility that two different nodes with different addresses have the same abbreviated address, which seems an undesirable event. Borrowing hashing vocabulary, we will call that event a ***collision***. Two different addresses sharing the same abbreviated address will be called ***synonyms***.

Probably the most intuitive approximation, (and our first idea), is to avoid collisions. But after some analysis, we found collision avoidance is clearly not suitable:

Making sure the absence of collision it is equivalent to *perfect hashing* [13]. A perfect hashing function applied to a set of keys will return unique *storage addresses*. But perfect hashing has a huge computational cost. And, what is worst, it demands to know all the keys before defining the function. In an ad-hoc network that means to know the addresses of all the hosts in the net before assigning them, which is completely opposite to the idea of an ad-hoc network. But even with perfect hashing, the maximum number of hosts in the net will be limited by the maximum number of different abbreviated addresses.

So, in this paper we propose *Abbreviated Dynamic Source Routing* (ADSR protocol), which applies hashing technique over the addresses, and whose more relevant characteristic is that it allows collision.

As far as we know, ADSR is the only protocol that does not rely on unique identifiers. Somehow, ADSR is analogous to lossy compression algorithms, as JPG or MP3. We can recall the Amoeba Operating System [14] where random addresses are assigned, so the collision is hypothetically possible, but the probability of a collision is almost null.

### B. Abbreviated Routes

Let R be an ordinary route as used in DSR, we can abbreviate it with any function $Abb()$ that satisfies:

1) Given any route
   $R1 = (D_1, D_2, ...D_n)$
   and its abbreviated route
   $Abb(R1) = (d1, d2, ..., d_n)$
   It must be satisfied
   $\forall i, 1 \leq i \leq n$
   $size(d_i) \leq size(D_i)$

where $size(d)$ is an address' size in bytes.

2) Given two ordinary routes
$R1 = (D_1, D_2, ...D_n)$
$R2 = (E_1, E_2, ...E_m)$
let these be their abbreviated routes
$Abb(R1) = (d1, d2, ...d_n)$
$Abb(R2) = (e1, e2, ...e_m)$
It must be satisfied: $d_i = e_j \wedge D_i \neq E_j \Rightarrow$
$i < n \wedge j < m$
So, if there is a collision between two addresses, they are not the last address in a route. That is equivalent to say that last address in a route is chosen in a way that guarantees the absence of collision. (Or its probability is almost null).

The second constrain avoids any host to receive some packet not addressed to it, but that is not the main purpose of this requirement (as higher layers in the communication stack could realize it and throw the packet). The main goal is that no host holds one route that seems to lead to some node (i.e. $D_n$), but indeed leads to some other node ($E_m$) synonym of the former.

For initial work and for the shake of simplicity , the $Abb()$ function chosen will be

- For $1 \leq i \leq n-1$, $d_i$ its the last byte in $D_i$
- For $i = n$ , $d_i = D_i$

Taken this premises, ADSR modifies DSR as little as possible to let it work with this kind of routes. We will enumerate these modifications later.

## IV. Classification of the collisions

The header size reduction shown in section II implies only one problem: Collisions.

An analysis and classification of the collision is shown below, describing ADSR behavior attending to the collision type.

In order to improve notation simplicity, in this section we will not refer a generic route like
$r_1 = (d_1, d_2, ..., d_{n-1}, d_n)$ instead we will use an example route, always the same: Node *a* searches a route to *d*. That route will be $r = a, b, c, d$.

The apostrophe $a$ , $a'$ means that nodes $a$ and $a'$ are synonyms ($d_i = e_j$ with $D_i \neq E_j$ ).

- Indifferent Collision
  Obviously, collisions in nodes not visible to *a*,*b*,*c* or *d*, do not affect them.
- Addressee Collision
  A state as shown in fig 3 will never happen, because of the second property in function $Abb()$.
- Distant Collision
  Figure 4 shows a harmless collision. When route request reaches *c'* it will be dropped, as it is not possible to reach *d* from *c'* (unless the Route Request went back by *a* but this is not allowed by flooding algorithm)
  When the packet is sent by *b* to *c*, *c'* does not receive it, so it does not affect. We call this event *distant collision* as the synonyms are distant in the ad-hoc network.
- Adjacent collision
  The only controversial event is the one shown in fig. 5. There is a *witness* node that can *see* both synonyms. This collision is analyzed in next section.

## V. Adjacent Collision

Fig. 5 shows adjacent synonyms. There are two possible routes, both legal. The algorithm will find both routes and the one that arrives first to *a* will be the one used.

- *a,b,c,c,d,* (that could be written as *a,b,c,c',d*)
  If this is the first route arriving to *a*, this node will include it in its data packet, when the packet arrives *b*, then *b* will send it to *c*, *c* and *c'* will get it as they are synonyms, both send it again to *c* ( *c* to *c'*, *c'* to *c*), so *d* will receive the packet duplicated, which is no severe problem.
- *a,b,c,d*
  If this is the route that *a* uses, it will be included in all the data packets; when the packet arrives *b*,then *b* will send it to *c*, (and also *c'*). The packet trough *c* will reach *d*, as it was intended.
  But the packet *c'* tries to send to *d*, will not reach next hop. As every node is responsible of data to get next hop, *c'* will retry and finally, desist. So, *c'* will send a *Route Error* to *a* ,as it perceives the route as broken. This Route Error will make *a* to stop using the route and start a new Route Discovery. That is the worst case, we say that *c' put c in the shade*. But meanwhile, some data packets were able to arrive its destination. That is unsuitable for a data stream, but can be enough in many situations as a sensor read, a event notification or any other discrete information.

As *a,b,c,d* is shorter than *a,b,c,c',d*, is more likely to arrive first to *a* and be the chosen route.

## VI. ADSR Features

ADSR is essentially equal to DSR, but in the features incompatible with abbreviated addresses, shown below:

**Route Building**

- DSR:
  In the *Basic DSR Route Discovery* stage, Route Request packet is distributed by the flooding algorithm, and logs the address of every node it traverses.
- ADSR: Route Request packet logs the abbreviated address of every hosts it traverses, taking into account that the last address must be collision free.

**Many addressees at the link layer**

- DSR: When some data packet with route $R1 = (D_1, D_2, ..., D_i, D_{i+1}, ...D_n)$ $D_{i+1}$ arrives to $D_i$ , reaching $D_{i+1}$ is immediate:
  Under DSR there will be a link layer that probably uses a different address scheme, but each network address will have a unique link address that can be resolved by technics such as ARP.
- ADSR: Given a route $r_1 =$
  $(d1, d2, ..., d_i, d_{i+1}, ..., d_n)$ datagram must be transmitted

from $d_i$ to $d_{i+1}$, but $d_{i+1}$ does not indentify a unique node, so this packet must be received by any node synonym of $d_{i+1}$, *iff* it is visible from $d_i$.
Thus, from the point of view of the link layer, this becomes a *multicast*. Most of the times multicast is not anticipated by the link layer, so there are two workarounds:

- Several *unicasts*, this demands to know the complete address of all the adressees.
- A *Broadcast*. Data is sent to every neighbour, every node receiving the packet drops it if its abbreviated address does not match the one in the packet. We must notice that the use of broadcast is incomptaible with data acknowledge. We will recall this issue later.

**Partial Routes**

- DSR: A route $R_1 = (D_1, D_2, ...D_n)$ shows how to reach $D_n$, but also can be used to route packets to $D_2, ...D_{n-1}$.
- ADSR: A route $r_1 =$ $(d_1, d_2, ...d_{n-1}, d_n)$ can be only used to reach $d_n$. To send a datagram to $D_i$ (where $i < n$) it is necessary a new route request, as $d_i$ could lead to $D_i$ or to any other $D_i$ synonym.

**Route Reversal**

- DSR: Let be $R_1 = (D_1, D_2, ...D_n)$ a Route Request that reaches its adressee. If the link layer is bidirectional, node $d_n$ is able to build directly from it a route to $d_1$, just reversing $(D_n, ...D_2, D_1)$ to store it in cache.
- ADSR. If route $r_1 = (d1, d2, ...d_{n-1}, d_n)$ is reversed, we get $(d_n, ..., d_2, d_1)$ which is no legal ADSR route to $d_1$ as $d_1$ is not a full (collision free) address, so a new Route Request must be initiated.
  Unless we use a workaround: If we *break* the networking stack layers and get from the network layer (i.e. the IP header) the $d_1$ full address, there is no need of a new Route Request.

**Flooding Control**

- DSR: Route Request floods the net. Each Request has a unique identifier. To avoid loops, a node receiving a Route Request checks if it has been processed before by itself, in that case, it discards the Request. To know that the node checks the following:
  1) It keeps a cache of identifiers of recent Route Requests
  2) It checks if its own address is in the addresses where the packet went through.
- ADSR:
  Only the first checking is possible, the second one is illegal as there is no way to now if the address in the path belongs to the node or to a synonym.
  So, under some circustances loops are hypothetically possible, althouch they would be always finite loops.

**Route Simplification**

- DSR: Any route $(D_1, D_2, ..., D_i, ..., D_j, D_{j+1}..., D_n)$ where $D_i = D_j$ , has a loop, so can be simplified as $(D_1, D_2, ..., D_i, D_{j+1}, ..., D_n)$
- ADSR:
  A route
  $r1 = (d_1, d_2, ..., d_i, ..., d_j, d_{j+1}..., d_n)$ where $d_i = d_j$ can't be simplified, as $d_i$ and $d_j$ can be two different synonyms nodes.

## VII. Experimentation

### A. Implementation constraints

To analys ns-2's performance and feasibility we have implemented it over network simulator ns-2, a free [15] widely used tool that supports many network protocols, including DSR.

Next we present some preliminar experimental results. We must notice that the starting point is the DSR implementation, where we make the modifications described in section VI.

That implementation has some optimizations and uses a link layer protocol (IEEE-802.11) that would not be used if the protocol was run over a limited resources architecture as described in section II. We avoid that and trust the well proved effectiveness of the layer-aproach design. From the point of view of performance, layers are not completely isolated in ad-hoc network protocols, as recent studies [16] show. But as performance is not the main purpose yet, we hold following results as useful. We compare a DSR implementation performance with a preliminar ADSR one.

As said in section I-B.2 each node is responsible of making sure that its packets arrive next hop. If the link layer offers that feature, as IEEE-802.11, DSR protocols can trusts the link layer and avoid this task.

In actual ADSR implementation, data is sent by a link layer broadcast. As we shown in section VI, broadcast is not reliable. So in actual implementation, nodes do not perceive packets losts and *Route Error* messages are never generated.

### B. Setup and results

Network setup, workload ans scenarios where ADSR protocol can be used are extremely diverse. It is difficult to settle a *typical* setup or get parameters from the *real world*. Besides, results show great variation with only small changes in any parameter.

So, we replicated faithfully the setup: scenarios, data traffic, connections, and node movements used by Broch *et al* [9] in their comparison of the performance of several ad-hoc network protocols (which is not a trivial task as many things have changed in the simulator since 1998). Our DSR results agree with Broch's, this implies partial validation of our job.

Nodes move according to the *random waypoint* model: Each node remains stationary for *pause time* seconds, then it selects a random destination and moves there with a random speed distributed uniformly between 0 and some maximum value. Recent studies [17] show some imperfections in this model, yet we consider it valid. Every simulation is run 10 times with the same parameters (Scenarios differ as there is a random component in the setup) and the average is computed.

Fig. 6 shows the *Packet delivery ratio*, the ratio between the number of CBR data packets originated by the application layer and the number of packets received.

Obviously, best performance corresponds to DSR, as we are comparing hosts without limitations (where DSR runs) with devices as referd in section II. ADSR can be applied in circumstances where DSR is not suitable, so somehow we could say that the gain is $+\infty$.

At low mobility scenarios, received data packet ratio is high. When mobility rises, the lack of reliability of the current protocol implementation is pronounced.

## VIII. Conclusions

We have seen circumstances where DSR protocol is diffi-coult or imposible to apply. We propose ADSR protocol, based on DSR drastically reduces headers size, allowing not unique identifiers. It produces what we call *collision*, that we have analyced here. If we can guarantee that will be no collisions on each route's last address, only collisions at adjacent nodes may cause problem. Even though collisions prevent many DSR optimizations to be done.

## References


[1] C. E. Perkins, *Ad Hoc Networking*. Addison-Wesley, 2001.

[2] C. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, 1994, pp. 234–244. [Online]. Available: citeseer.nj.nec.com/perkins94highly.html

[3] E. Royer and C. Toh, "A review of current routing protocols for ad-hoc mobile wireless networks," 1999. [Online]. Available: citeseer.nj.nec.com/royer99review.html

[4] S. Murthy and J. J. Garcia-Luna-Aceves, "An efficient routing protocol for wireless networks," *Mobile Networks and Applications*, vol. 1, no. 2, pp. 183–197, 1996. [Online]. Available: citeseer.ist.psu.edu/article/murthy96efficient.html

[5] C. Perkins, "Ad hoc on demand distance vector routing," citeseer.nj.nec.com/article/perkins99ad.html, 1997.

[6] D. Johnson, D. Maltz, and J. Broch, *DSR: The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks*. Addison-Wesley, 2001, ch. 5, pp. 139–172.

[7] M. S. Corson and A. Ephremides, "A distributed routing algorithm for mobile wireless networks," *Wirel. Netw.*, vol. 1, no. 1, pp. 61–81, 1995.

[8] V.D.Park and M. Corson, "A highly adaptative distributed routing algorithm for mobile wireless networks," in *Proceedings of the 3rd Annual ACM/IEEE International Conference on Mobile Computing and Networking* (MobiCom 97). ACM Press, 1997.

[9] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," in *Mobile Computing and Networking*. (ACM MOBICOM'98), 1998, pp. 85–97. [Online]. Available: citeseer.nj.nec.com/broch98performance.html

[10] S. R. Das, C. E. Perkins, and E. E. Royer, "Performance comparison of two on-demand routing protocols for ad hoc networks," in *Proceedings of IEEE INFOCOM - The Conference on Computer Communications*, 2000, pp. 3–12. [Online]. Available: citeseer.nj.nec.com/das00performance.html

[11] C. E. Perkins, E. M. Belding-Royer, and S. R. Das, "IP flooding in ad hoc mobile networks," www.ietf.org/proceedings/01dec/I-D/draft-ietf-manet-bcast-00.txt, 2001, IETF Internet Draft.

[12] K. Fall and K. Varadhan, "The ns manual," http://www.isi.edu/nsnam/ns/doc, uC Berkeley and Xerox PARC.

[13] T. G. Lewis and C. R. Cook, "Hashing for dynamic and static internal tables," *IEEE Computer*, vol. 21, pp. 45–56, 1988.

[14] S.Mullender,G. van Rossum,A.Tanenbaum,R.van Renesse, H.van Staveren, "Amoeba:a distributed operating system for the 1990," *IEEE Computer*, may 1990.

[15] J. M. González-Barahona and C. Daddara, "Free software/open source: Information society opportunities for europe?" http://eu.conecta.it/paper/, April 2000, 001.

[16] C. Barrett, A. Marathe, M. V. Marathe, and M. Drozda, "Characterizing the interaction between routing and mac protocols in ad-hoc networks," in *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*. ACM Press, 2002, pp. 92–103.

[17] J. Yoon, M. Liu, and B. Noble, "Random waypoint considered harmful," 2003. [Online]. Available: citeseer.ist.psu.edu/yoon03random.html

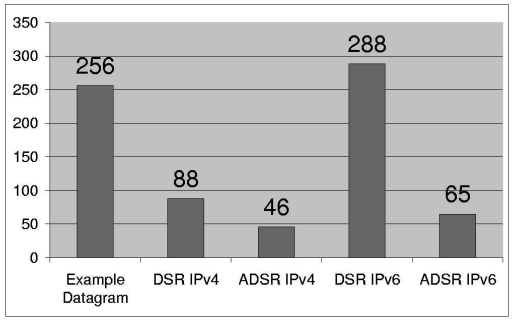


Fig. 1. Datagram size comparison (bytes)

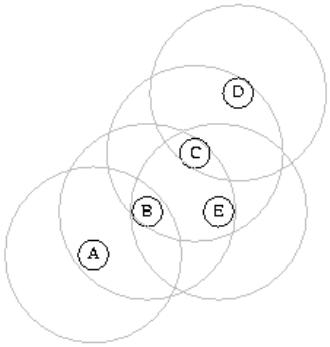


Fig. 2. Ad-hoc Network

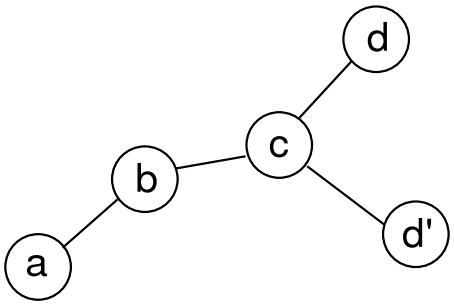


Fig. 3. Adressee Collision

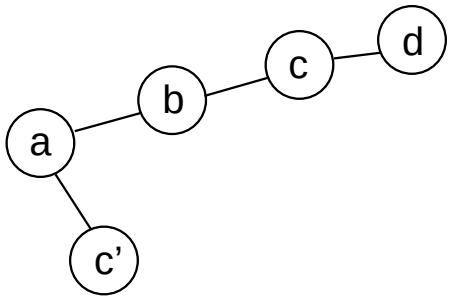


Fig. 4. Distant Collision

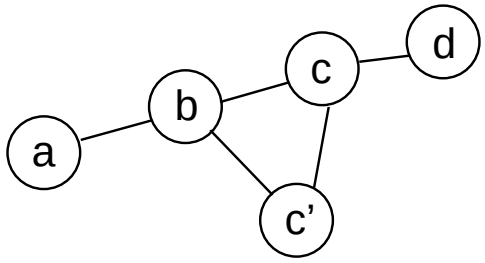


Fig. 5. Adjacent Collision

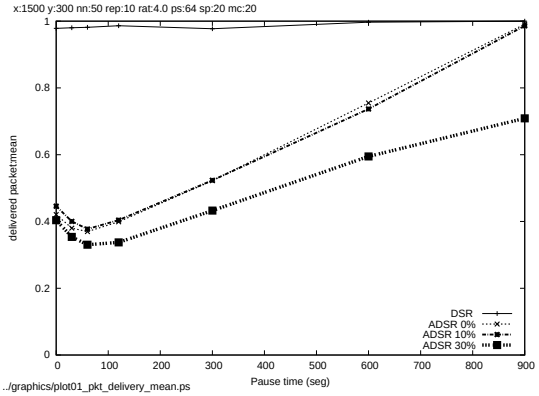


Fig. 6. Packet Delivery