The SHR Family of Protocols

Mark Lisee

February 8, 2007

1 Packet Fields

The SHR family of protocols consists of SHR-M, SHR-P, SHR and SHR-R. These fields are used in the packets transmitted by the SHR family of protocols. Not all of the fields are used in all packet types. In addition, some of the packets types are not used by some of the protocols. The packet formats are shown in the sections that describe the processing of those packets.

The field descriptions are

`SrcID`	The ID of the originating node. This field's value is fixed for the life of the packet.
`DestID`	The ID of the target node. This field's value is fixed for the life of the packet.
`SeqNum`	A number assigned by the originating node. This should increase for each packet sent by the originating node. This value is fixed for the life of the packet.
`ActHC`	The number of hops that the packet has traveled from the originating node. This field's value is updated each time the packet is broadcast.
`ExpHC`	The number of additional hops that should be required to reach the target node. This field's value is updated each time the packet is broadcast.
`MaxHop`	The upper limit on the number of times a packet may be forwarded before it must be dropped. This field's value is fixed for the life of the packet.
`RBC`	A Boolean value that indicates if this packet is the second broadcast by a node. This field's value is updated each time the packet is broadcast.
`Data`	The information that is traveling from the originating node to the target node. This field's value is fixed for the life of the packet.

The pair $\langle$`SrcID`, `DestID`$\rangle$ is called a flow ID and represents traffic originating at `SrcID` and terminating at `DestID`. If there is two way traffic between a pair of nodes, there will be two flow IDs.

The triple ⟨`SrcID`, `DestID`, `SeqNum`⟩ is called a packet ID. It represents all instances of a specific packet in the flow ID.

2 Node Information

These data structures are maintained at each node.

2.1 Cost Vector

This associates a cost with a destination node. The cost may be a function of several characteristics, such as the number of hops to the destination, the quality of the link, the amount of energy available at this node, etc.

2.2 APID

The APID is a list of all the packet IDs that are active at this node. (APID is an acronym for Active Packet IDs.) The APID is a two level data structure. The top level consists of a list of flow IDs that have been seen by the node. Associated with each flow ID is a queue of sequence numbers ordered from the oldest to the newest. Each sequence number is mapped to a state and some state specific information. There are two states, `New` and `Ignore`, that are common to all automata and are defined below. States that are specific to an automaton are defined in the section that describes that automaton. The state specific information is listed in the sections that describe the automata.

`New` This node has not yet seen the packet ID.

`Ignore` This node will no longer react to the packet ID.

3 Meta Processing

When a node receives a packet, it performs the steps

Update *ls*: If the packet is not an ACK packet, the source's entry in the cost vector, *ls*, is set to min(*ls*, `ActHC`).

Update *ld*: If the packet is a DATA packet, then the destination's entry in the cost vector, *ld*, is set to min(*ld*, `ExpHC` + 1).

Get packet state from APID: Locate the flow ID associated with the packet ID. If the sequence number is found, the associated state is used. If the sequence number is not found and the sequence number is greater than the oldest sequence number in the queue, then the sequence number is added to the APID with the state `New`. Otherwise, the state `Ignore` is associated with this packet ID.

Apply automaton processing: If the packet is a DREQ packet see section 5.1. If the packet is a DREP packet see section 5.2. For other packet types, see section 5.3, 5.4, 5.5 or 5.6 for the protocols SHR-M, SHR-P, SHR or SHR-R respectively.

Update the APID: The processing of the packet may change the state of the APID. Because of this, the list of sequence numbers for this flow ID is examined. If there are more than five sequence numbers, the oldest one is removed from the APID. Repeat the step of removing the oldest remaining sequence number as long as both of these conditions apply

- The state is `Ignore`, and
- The list has more than one sequence number.

4 Source Node Processing

This section applies to all protocols. When a node needs to send a packet to the node identified by `DestID`, it first determines if `DestID` has an entry in the cost vector. If there is an entry, then the node

1. Creates the packet DATA⟨`MyID`, `DestID`, `SeqNum`, `0`, `ExpHC`, `Data`⟩. The value for `ExpHC` comes from the cost vector.
2. Transmits the DATA packet.
3. Enters the packet ID ⟨`MyID`, `DestID`, `SeqNum`⟩ into the APID with the state `Ignore` (SHR-M) or `Owner` (SHR-P, SHR or SHR-R).
4. Increments `SeqNum`.

If the cost vector does not contain an entry for the destination, the node

1. Puts `Data` on a deferral list.
2. Creates the packet DREQ⟨`MyID`, `DestID`, `SeqNum`, `0`⟩.
3. Transmits the DREQ packet.
4. Enters the packet ID ⟨`MyID`, `DestID`, `SeqNum`⟩ into the APID with the state `Ignore`.
5. Increments `SeqNum`.

5 Received Packet Processing

This section applies to all packets that are received, even packets that originated at this node. The DREQ and DREP processing is identical for all of the protocols. The protocols differ in their processing of DATA packets and, where applicable, ACK packets.

These notations are used in the description of the state transitions.

ls	The distance from the source's entry in the cost vector.
ld	The distance from the destination's entry in the cost vector.
timer($delay$)	A timer should be started with the given delay.

5.1 DREQ Packets

The processing of DREQ packets is identical for all protocols in the SHR family. Their format is

DREQ ⟨`SrcID`, `DestID`, `SeqNum`, `ActHC`⟩

DREQ packets are flooded from the `SrcID` to the `DestID` using the automaton shown in Figure 1 and the actions listed in Table 2. Table 1 lists each state and the information that must be retained when the automaton enters that state.

An intermediate node forwards a DREQ packet if the DREQ packet is new or it represents a better distance to the source. When the DREQ packet is forwarded, it is after a delay of $\mathrm{U}(0, (log_{10}(ActHC)+1)\lambda)$. The delay is indicated by $logBackoff$ in table that describes the automaton. The delay reduces the number of DREQ packets that should be sent by increasing the probability that a node will receive the optimal DREQ packet before forwarding one of its own. The average delay increases with the distance to the source to compensate for the fact that packets may arrive via disjoint paths. The destination sends only one DREP packet and that is after a period of 10λ in which no DREQ packets have been received. This allows the DREQ broadcast storm to subside and increases the probability that all of the nodes in the network will know their distance to the source. This last point is important for the forwarding of DREP packets.

The DREQ automaton defines these states in addition to `New` and `Ignore`.

`Delay` This state is not used by the destination node. The intermediate node has received a DREQ packet and will forward the packet when the timer expires.

`Listen` There are two sets of actions based on whether the node is the destination or an intermediate.

Destination The node is waiting for the 10λ timer to expire before sending the DREP packet.

Intermediate This node has forwarded the packet. It starts a timer and listens for DREQ packets. If the node receives a packet that improves the distance to the source, then the node moves to the `Delay` state. If a better packet is not received before the timer expires, the node moves to the `Ignore` state.

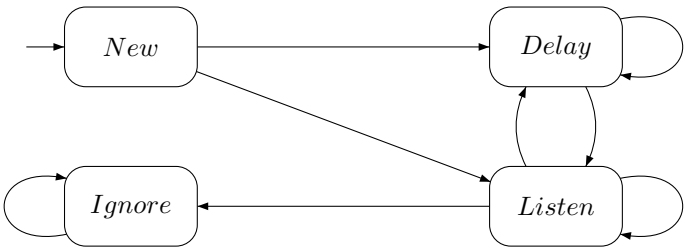

Figure 1: Automaton for DREQ and DREP Packets

Table 1: DREQ Automaton State Information

State	**Data Field**	**State Field**
`New`	None	
`Delay`	`s#`	`sn`
`Listen`	`s#` or `new s#`	`sn`
`Ignore`	None	

When the destination node transitions from the state `New` to `Listen`, it generates a new sequence number that will be used when the DREP packet is transmitted. When an intermediate node transitions from the state `New` to `Listen`, it saves the sequence number from the incoming DREQ packet.

Table 2: DREQ Automaton Events

State	Event	Action	New State
N	DREQ⟨S, MyID, s#,*⟩	Start timer(10λ)	L
	Any other packet	Start timer($logBackoff$)	D
D	Any packet	No action (The APID processing has already updated ls.)	D
	Timer expired	Send DREQ⟨S, D, s#, $ls+1$⟩; Start timer(10λ)	L
L	DREQ⟨S, MyID, s#, *⟩	Cancel timer; Start timer(10λ)	L
	DREQ⟨S, D, s#, $<ls$⟩	Cancel timer; Start timer($logBackoff$)	D
	Any other packet	No action	L
	Timer expired, D = MyID	Send DREP⟨MyID, S, sn, 1, ls⟩	I
	Timer expired, D ≠ MyID	No action	I
I	Any packet	No action	I

5.2 DREP Packets

The processing of DREP packets is identical for all protocols in the SHR family. Their format is

DREP ⟨`SrcID`, `DestID`, `SeqNum`, `ActHC`, `ExpHC`⟩

DREP packets are forwarded from the source to the destination using the same method as for the DREQ packets with the exception that a node will forward a DREP only if the node knows its distance to the destination. The automaton is shown in Figure 2 and the actions are listed in Table 4. Table 3 lists each state and the information that must be retained when the automaton enters that state.

The DREP automaton defines these states in addition to `New` and `Ignore`.

`Delay` The node has received a DREP packet and is waiting for a timer to expire. When the timer expires, the node will perform an action based on if the node is the destination or intermediate node. The destination node will send any pending DATA packets to the source. An intermediate node will forward the DREP packet.

`Listen` This state is not used by the destination node. The intermediate node has forwarded a DREP packet and listening for additional DREP packets. If a better packet is received, the node moves to the `Delay` state. If a better packet is not received before the timer expires, the node moves to the `Ignore` state.

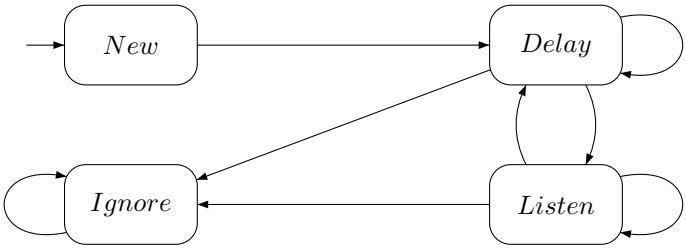

Figure 2: Automaton for DREP Packets

Table 3: DREP Automaton State Information

State	Data Field	State Field
`New`	None	
`Delay`	None	
`Listen`	None	
`Ignore`	None	

Table 4: DREP Automaton Events

State	Event	Action	New State
N	DREP⟨S, MyID, s#,*,*⟩	Start timer(1.5λ)	D
	DREP⟨S, D, s#,*,*⟩	Start timer($logBackoff$)	D
D	Any packet	No action (The APID processing has already updated ls.)	D
	Timer expired, D = MyID	Send any DATA packets destined for S	I
	Timer expired, D ≠ MyID	Send DREP⟨S, D, s#, ls+1, ld⟩; Start timer(10λ)	L
L	DREP⟨S, D, s#, $<ls$,*⟩	Cancel timer; Start timer($logBackoff$)	D
	Any other packet	No action	L
	Timer expired	No action	I
I	Any packet	No action	I

5.3 The SHR-M Protocol

The SHR-M (for Minimal) protocol is the simplest member of the SHR family of protocols. It uses only a single packet type, DATA, whose format is

DATA ⟨`SrcID`, `DestID`, `SeqNum`, `ActHC`, `ExpHC`, `Data`⟩

Ideally, the SHR-M protocol will use only one broadcast for each hop traversed by the packet. Only nodes that are closer to the destination may forward the DATA packet. The nodes also ignore the second and all subsequent receptions of the same DATA packet.

The automaton for the SHR-M DATA packet processing is shown in Figure 3. Table 5 lists each state and the information that must be retained when the automaton enters that state. Table 6 lists the events and their associated actions.

The SHR-M DATA automaton defines one state in addition to `New` and `Ignore`.

`Possible` This node is participating in the election to forward the DATA packet.

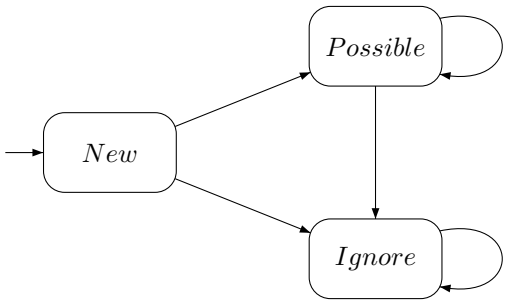

Figure 3: Automaton for SBCR DATA Packets

Table 5: SHR-M DATA Automaton State Information

State	Data Field	State Field
`New`	None	
`Possible`	`ActHC`	`sAHC`
	`ExpHC`	`sEHC`
	`Data`	`sData`
`Ignore`	None	

Table 6: SHR-M DATA Automaton Events

State	Event	Action	New State
N	DATA⟨S, MyID, s#,*,*,*⟩	Forward data to the application	I
	DATA⟨S, D, s#,*, $>ld$,*⟩	Start timer(U(0,λ))	P
	Any other packet	No action	I
P	DATA⟨S, D, s#,*, <sEHC,*⟩	Cancel timer	I
	Any other packet	No action	P
	Timer expired	Send DATA⟨S, D, s#, sAHC+1, ld, sData⟩	I
I	Any packet	No action	I

5.4 The SHR-P Protocol

The SHR-P (for Passive) protocol is the next member of the SHR family of protocols. It uses only a single packet type, DATA, whose format is

DATA ⟨`SrcID`, `DestID`, `SeqNum`, `ActHC`, `ExpHC`, `Data`⟩

The SHR-P protocol adds some robustness to SHR-M by allowing a node to broadcast a DATA packet twice. If neither of the packets is forwarded, then the node increases by two its distance to the destination. When the destination receives the DATA packet, it broadcasts an empty DATA packet to indicate that the data has arrived.

The automaton for the SHR-P DATA packet processing is shown in Figure 4. Table 7 lists each state and the information that must be retained when the automaton enters that state. Table 8 lists the events and their associated actions.

The SHR-P DATA automaton defines these states in addition to `New` and `Ignore`.

`Possible` This node is participating in the election to forward the DATA packet.

`Owner` This node has forwarded the packet and is waiting to see if the packet is forwarded.

`Resend` This node has forwarded the packet twice and is waiting for the timer to expire.

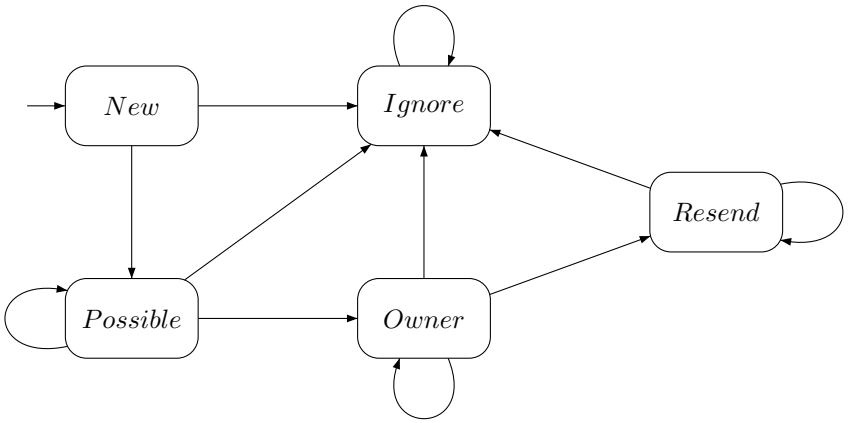

Figure 4: Automaton for SHR-P DATA Packets

Table 7: SHR-P DATA Automaton State Information

State	Data Field	State Field
New	None	
Possible	ActHC	sAHC
	ExpHC	sEHC
	Data	sData
Owner	ActHC	sAHC
	Data	sData
Resend	None	
Ignore	None	

Table 8: SHR-P DATA Automaton Events

State	Event	Action	New State
N	DATA⟨S, myID, s#,*,*,*⟩	Send DATA⟨S, myID, s#, AHC+1, 0, null⟩; Forward data to the application	I
	DATA⟨S, D, s#,*, $>ld$,*⟩	Start timer(U(0,λ))	P
	Any other packet	No action	I
P	DATA⟨S, D, s#,*, <sEHC,*⟩	Cancel timer	I
	Any other packet	No action	P
	Timer expired	Send DATA⟨S, D, s#, sAHC+1, ld, sData⟩; Start timer(U(1.25λ,1.75λ))	O
O	DATA⟨S, D, s#,*, $<ld$,*⟩	Cancel timer	I
	Any other packet	No action	O
	Timer expired	Send DATA⟨S, D, s#, sAHC+1, ld, sData⟩; Start timer(U(1.25λ,1.75λ))	R
R	DATA⟨S, D, s#,*, $<ld$,*⟩	Cancel timer	I
	Any other packet	No action	R
	Timer expired	ld += 2	I
I	Any packet	No action	I

5.5 The SHR Protocol

The SHR protocol is the base version of the SHR family of protocols. It uses both DATA and ACK packet, whose formats are

DATA ⟨`SrcID`, `DestID`, `SeqNum`, `ActHC`, `ExpHC`, `MaxHop`, `Data`⟩
ACK ⟨`SrcID`, `DestID`, `SeqNum`⟩

As with SHR-P, if a node does not detect that its packet was not forwarded after the second broadcast, the node increases by two its distance to the destination. Unlike SHR-P, the node will perform a third broadcast with the new distance. The protocol uses ACK packets in two cases. The first case is by the destination node to indicate that the data has arrived. The second is when a node detects that two downstream nodes have forwarded the DATA packet. This condition happens only when all the neighbors do not form a clique. Neighbors that hear the ACK and either did not forward the DATA packet or were the second node to forward the DATA packet will ignore the next $IgnoreCount$ packet IDs. (Currently, $IgnoreCount$ is set to 9.)

The automaton for the SHR DATA packet processing is shown in Figure 5. Table 9 lists each state and the information that must be retained when the automaton enters that state. Table 10 lists the events and their associated actions.

The SHR DATA automaton defines these states in addition to `New` and `Ignore`.

`Possible` This node is participating in the election to forward the DATA packet.

`Owner` This node has forwarded the packet and is waiting to see if the packet is forwarded.

`Father` This node has forwarded the packet and detected that exactly one downstream node has forwarded the packet.

`Resend` This node has forwarded the packet twice and is waiting for the timer to expire.

`Waiting` This node was in the Possible state and detected another node forwarding the packet. It is waiting to see if an ACK is broadcast.

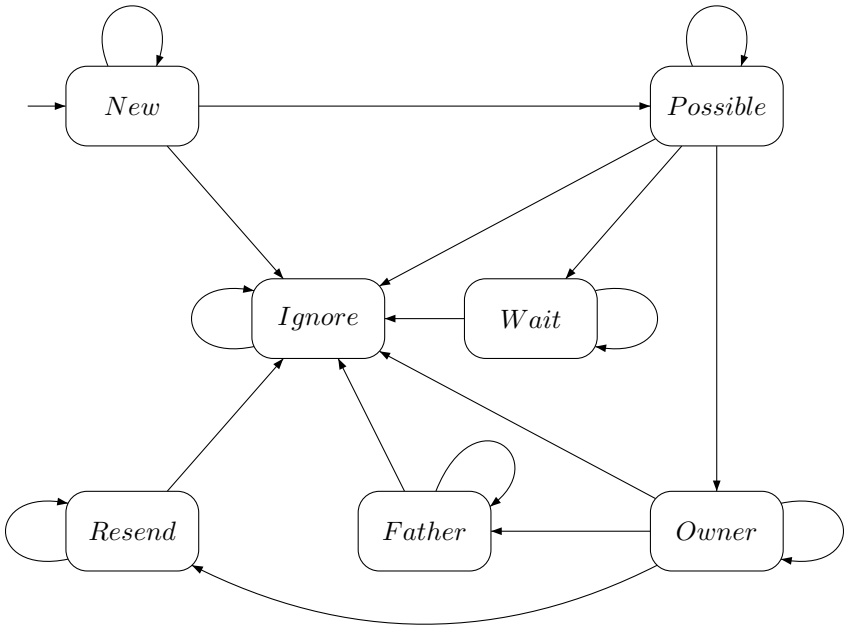

Figure 5: Automaton for SHR DATA Packets

Table 9: SHR DATA Automaton State Information

State	Data Field	State Field
New	None	
Possible	ActHC	sAHC
	ExpHC	sEHC
	MaxHop	sMH
	Data	sData
Owner	ActHC	sAHC
	MaxHop	sMH
	Data	sData
Father	None	
Resend	ActHC	sAHC
	MaxHop	sMH
	Data	sData
Wait	ExpHC	sEHC
Ignore	None	

Table 10: SHR DATA Automaton Events

State	Event	Action	New State
N	DATA⟨S, myID, s#,*,*,*,*⟩	Send ACK⟨S, myID, s#⟩; Forward data to the application	I
	DATA⟨S, D, s#,*, $>ld$,*⟩	If ignore > 0 then ignore–	I
		If ignore $= 0$ then Start timer(U(0,λ))	P
	ACK⟨S, D, s#⟩	No action	I
	Any other packet	No action	N
P	DATA⟨ S, D, s#,*,<sEHC,*,*⟩	Cancel timer, start timer(λ/4)	W
	ACK⟨S, D, s#⟩	ignore = $IgnoreCount$; cancel timer	I
	Any other packet	No action	P
	Timer expired	Send DATA⟨S, D, s#, sAHC+1, ld, sMH, sData⟩; Start timer(U(1.25λ,1.75λ))	O
O	DATA⟨S, D, s#,*, $<ld$,*,*⟩	No action	F
	ACK⟨S, D, s#⟩	Cancel timer	I
	Any other packet	No action	O
	Timer expired	Send DATA⟨S, D, s#, sAHC+1, ld, sMH, sData⟩; Start timer(U(1.25λ,1.75λ))	R
F	DATA⟨S, D, s#,*, $<ld$,*,*⟩	Send ACK⟨S, myID, s#⟩; cancel timer	I
	ACK⟨S, D, s#⟩	Cancel timer	I
	Any other packet	No action	F
	Timer expired	No action	I
R	DATA⟨S, D, s#,*, $<ld$,*,*⟩	Cancel timer	I
	ACK⟨S, D, s#⟩	Cancel timer	I
	Any other packet	No action	R
	Timer expired	ld += 2; if ($ld + sAHC < sMH$) then Send DATA⟨S, D, s#, sAHC+1, ld, sMH, sData⟩	I
W	DATA⟨S, D, s#,*,<sEHC,*,*⟩	Cancel timer	I
	ACK⟨S, D, s#⟩	ignore=$IgnoreCount$; cancel timer	I
	Any other packet	No action	W
	Timer expired	No action	I
I	DATA⟨S, myID, s#,*,*,*,*⟩	Send ACK⟨S, myID, s#⟩	I
	Any other packet	No action	I

5.6 The SHR-R Protocol

The SHR-R (for Reactive) protocol is the most sophisticated member of the SHR family of protocols. It uses both DATA and ACK packet, whose formats are

DATA ⟨`SrcID`, `DestID`, `SeqNum`, `ActHC`, `ExpHC`, `MaxHop`, `RBC`, `Data`⟩
ACK ⟨`SrcID`, `DestID`, `SeqNum`, `ActHC`⟩

MWL: This section will not be completed until the SHR-R protocol is finalized.

The automaton for the SHR-R DATA packet processing is shown in Figure 6. Table 11 lists each state and the information that must be retained when the automaton enters that state. Table 12 lists the events and their associated actions.

The SHR-R DATA automaton defines these states in addition to `New` and `Ignore`.

`Possible` This node is participating in the election to forward the DATA packet.

`Owner` This node has forwarded the packet and is waiting for some event.

`Resend` This node has forwarded the packet twice and is waiting for some event.

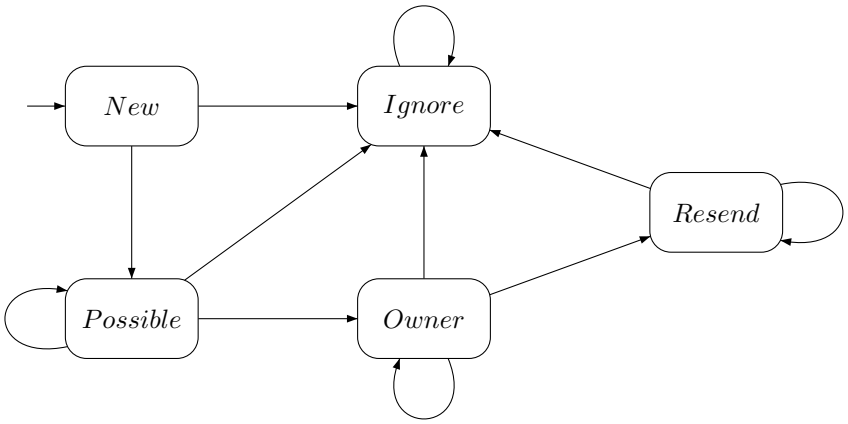

Figure 6: Automaton for SHR-R DATA Packets

Table 11: SHR-R DATA Automaton State Information

State	Data Field	State Field
New	None	
Possible	ActHC	sAHC
	ExpHC	sEHC
	Data	sData
Owner	ActHC	sAHC
	Data	sData
Resend	None	
Ignore	None	

Table 12: SHR-R DATA Automaton Events

State	Event	Action	New State
N	DATA⟨S, D, s#,*, $>ld$,*⟩	Start timer(U(0,λ))	P
	DATA⟨S, myID, s#,*,*,*⟩	Send DATA⟨S, myID, s#, AHC+1, 0, null⟩; Forward data to the application	I
	Any other packet	No action	I
P	DATA⟨S, D, s#,*, <sEHC,*⟩	Cancel timer	I
	Timer expired	Send DATA⟨S, D, s#, sAHC+1, ld, sData⟩; Start timer(U(1.25λ,1.75λ))	O
	Any other packet	No action	P
O	DATA⟨S, D, s#,*, $<ld$,*⟩	Cancel timer	I
	Timer expired	Send DATA⟨S, D, s#, sAHC+1, ld, sData⟩; Start timer(U(1.25λ,1.75λ))	R
	Any other packet	No action	O
R	DATA⟨S, D, s#,*, $<ld$,*⟩	Cancel timer	I
	Timer expired	ld += 2	I
	Any other packet	No action	R
I	Any DATA	No action	I