DAF: Data Aggregation Frame work

New frame work for supporting data aggregation in NS-2

Karam Madi*   Mouthana Al-Kubeily**

* Engineer in Communications and Electronics Engineering department, Tishreen University.

Email: Madi.karam94@gmail.com

Abstract: Using Wireless Sensors Networks (WSNs) increasingly in the wide life realms, and providing a lot of improvements to people life ways, have made it one of the most issues that scientific research has interested. As a result of its relying on batteries to getting its energy, consumption power reduction have become a major subject to be studied by researches. In networks that serving periodical monitoring applications, it could be very useful to aggregate the sensed data in order to reduce power consumption in the router nodes, and various aggregation protocols have been proposed for different applications such as monitoring and data collection. So that it's very important to make Network Simulator 2 (NS-2), as a simulator specialist for researchers, able to do simulations including data aggregation in WSN. According to that importance, we have introduce in this paper a new frame work called 'DAF' to make data aggregation available to be done by NS-2 users easily, by OTCL commands only.

Keywords: WSN, NS-2, Data aggregation, Frame work, Power Consumption Reduction, periodical sensing.

1. Introduction:

WSNs are networks need low amount of energy to run, due to its work nature which summarizes in tasks like monitoring, data collection and other tasks with similar nature.

On the other hand, WSNs are often used in areas which are hard to reach by humans, so that it is powered by small batteries. As a result, power consumption reduction in WSN has become one of the most important scientific research fields last years.

One of the most efficient way, to preservation nodes' energy, is to use a data aggregation technology, which is considered as one of the fundamental processing procedure for saving the energy[1], where it aims to reduce the transmission times in the router nodes which called in this case 'Aggregator Nodes' as figure (1) shows.

In addition, WSNs uses a lot of routing protocols, but AODV is the most common and a good performer protocol in MANET [2].

Data aggregation technology could be implemented in many ways, for example it could be used to do non-accurate aggregation where apply some functions on many packets received from many sensing nodes to elicit brief data, then transmit it to the sink by one packet, or it could be used to do accurate aggregation where doesn't apply any function on received data, but buffers this data and write it in payload of one packet with source of each data in way that know for sink, and send it to the sink.

This paper will introduce DAF frame work for NS-2, to facilitate users' simulations on it which includes data aggregation.

2. Data Aggregation Technique:

Data aggregation is a technique to aggregate data packets. This technique combines the data packets using an aggregation function (e.g Average, Maximum, Minimum, Count, Median, Rank, Standard Deviation, Variance and Sum) into a single one to transmit or write this data as it is in a new packet with source address for each. It would result in reductions of transmissions and consequently decreasing the communication costs, bandwidth utilization, network congestion, energy consumption and network delay in WSN routing [3].

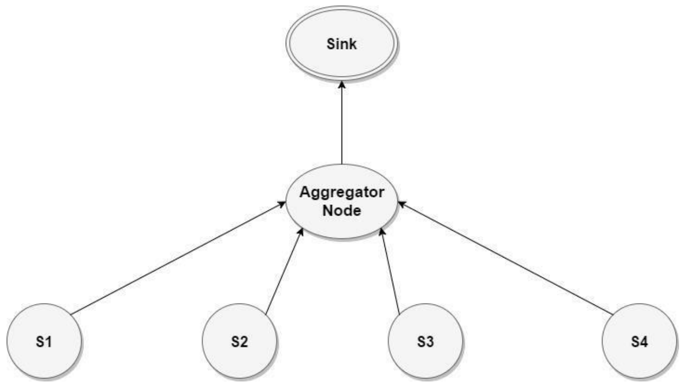

Figure (1) shows data fusion mechanism

3. Network Simulator 2 (NS-2):

System modeling is an act of formulating a simple representation for an actual system [4], and that is what made it the best way for doing our experiments in the context of scientific research without creating real systems for that. One of the most important and popular tools in the field of Network Simulation is Network Simulator (NS-2).

3.1 Basic architecture:

NS-2 is one of the most used simulators in the computer networking field, it is very useful in diagnosing network mechanisms and analyzing its behavior, and it is an open source discrete event network simulator [5].

NS-2 has been written using C++ to defines the internal mechanism (i.e., a backend), and OTCL which sets up

simulation by assembling and configuring the objects as well as scheduling discrete events (i.e., a frontend) [4].
This usage of two languages was to provide fast high performance with ability to making the modifications, on simulated network parameters, easy and fast.
The results of the simulation are written in a trace file after every millisecond intervals containing detailed information about the network attributes (e.g. node id, coordinates, packet size, type etc.)[6]
NS-2 links the both environments using the "TCLCL" class which written in C++, this class consists of main six classes as following: [4]

1. `TclClass` maps class names in the compiled hierarchy to class names in the interpreted hierarchy
2. `InstVar` class binds member variables in both the hierarchies together.
3. `TclCommand` class allows the Tcl interpreter to execute non-OOP C++ statements.
4. `TclObject` class is the base class for all C++ simulation objects in the compiled hierarchy.
5. `Tcl` class provides methods to access the interpreted hierarchy from the compiled hierarchy
6. `EmbeddedTcl` class translates OTcl scripts into C++ codes.

Class hierarchies in both the languages may be standalone or linked together. Figure 3 shows that.

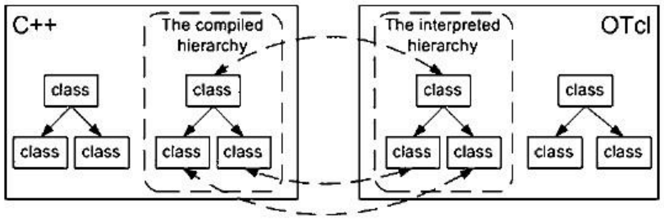

Figure (2), shows that NS-2 includes standalone classes and linked classes.

3.2 Network objects:

NS-2 represents the real network elements in objects using the polymorphism concept in object-oriented programming (OOP), as well as it allows the user to design the commands which is needed by two ways.
1) Non OOP-based way.
2) OOP-based way.

3.2.1 Designing commands in NS-2:

As we presented previously, there are two ways in NS-2 to create commands, OOP-based and non OOP-based commands [4].
In non OOP-based way, a class has to be derived from TCLCommand class (see section 2.1), then name of the created command has to be passed as a parameter to the constructer of the derived class and, the implementation of the command has to be defined in a function called "command" exclusively.
In OOP-based way, NS-2 allows OTcl commands to propagate up the hierarchy owing to its OOP nature [4], two classes have to be linked in each environment (C++ and OTCL).
(Linking mechanism is clearly illustrated in NS manual).

The command in this way has to be in the following syntax [4].

```
$obj <cmd_name> <args>
```

Where:
`$obj` is object of the linked OTCL class.
`<cmd_name>` is the command name.
`<args>` is the argument that passed to the implementation of this command.
In this way, a function called "command" has to be defined in the linked C++ class, and the implementation of the command has to be written in it.

4. Our Proposition:

4.1 Overview:

We introduce DAF frame work as a data aggregation frame work which works relating to AODV protocol, because it needs AODV routing table in some of its tools to perform varietal tasks.
We try in DAF frame work to set up tools that will be useful in research which include data aggregation.
This tools are available and controlled by OTCL commands, in order to help users who don't have required experience to deal with NS-2 backend (C++ environment).

4.2 How DAF works?

DAF tools could be classified based on packet parameters to determine which packet should aggregate and which shouldn't, this parameters are initialized by user. In addition, these tools relies on user OTCL command to determine which node should be aggregator, or relies on hope count in AODV routing table to decide that. All DAF tools allow users to select accuracy type in aggregation whether it's accurate or non-accurate. These two types of accuracy are determined as following:
Accurate aggregation: In this type of aggregation accuracy, node aggregator catches up the packets and save its payload in a buffer specified for packets which sent to the same destination, in addition to save its source address. Then when the number of packets reaches the maximum, or the aggregation duration expired, node aggregator write each value with its source to payload of new packet and sends it to the destination, as illustrated in figure (3.1).
When destination receives this aggregated packet will be able to read the values with its sources.
Note: To reduce simulation complexity in DAF frame work, we made the length of source field constant with 1 byte, and the same length to the value field, but in reality it could be determined in any relevant way.

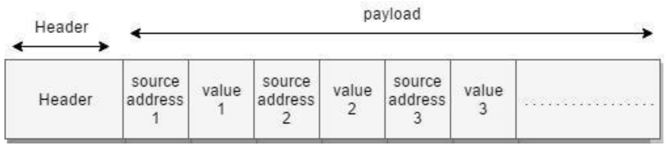

Figure (3.1): shows writing mechanism in the aggregated packet

Non-accurate aggregation: Here, node aggregator catches the packets and saves the value from its payload in a buffer specified for packets which transmitted to the same destination.
Then when the number of packets reaches the maximum, or the aggregation duration expired, node aggregator will apply a function (average, maximum, or minimum) on these values in the buffer and write the result to the payload of a new packet, then sends it to the destination.

4.2.1 Aggregation tools:
All aggregation tools in DAF work according to the following general flowchart.

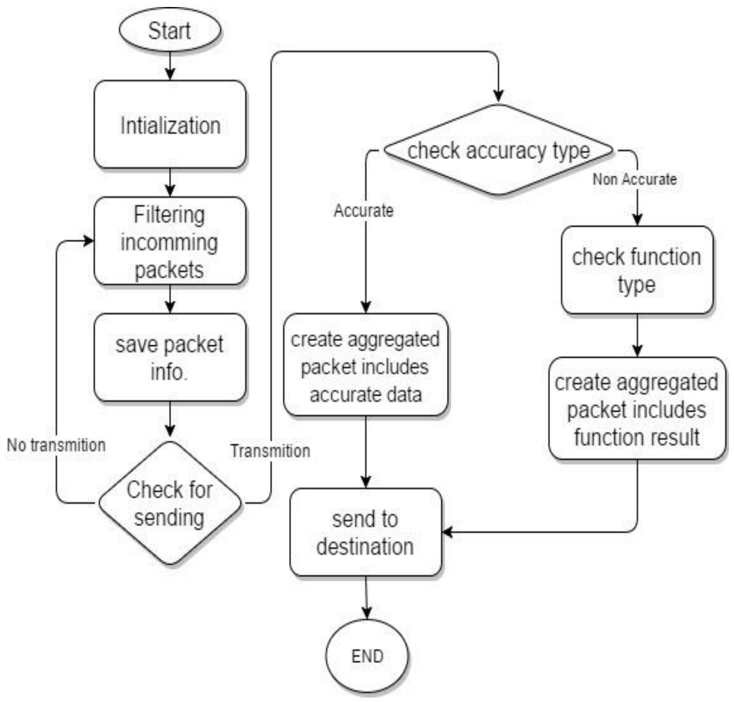

Figure (3): general mechanism for DAF tools working.

These tools differ in ‘Filtering incoming packets’ block which determines criteria that packets should be handled based on. Moreover, these tools might be able to select aggregator nodes automatically based on AODV routing table or do that relying on commands’ user.

3.2.1.1 Based on destination-sources tool (D_S):
This tool makes aggregation relied on the destination address to handle packets. In addition, it allows users to determine the sources that they want to aggregate packet from.
Blocks of general flowchart are defined according to this tool as the following.

- **Initialization** users initialize required parameters which tool will decide the handled packets based on.
- **Filtering incoming packets:** In this stage, aggregator node decides which packets would be aggregated based on the parameters which user initialized in the previous stage.
- **Save packet info:** Aggregator node saves packet source and payload.
- **Check for sending:** In this stage, aggregator node move on to prepare for sending the aggregated packet if aggregation time expired or saved packets number reached the maximum (maximum number of packets is determined by user in the ‘initialization stage’).
- **Check accuracy type:** Aggregator node checks accuracy type for sending. (See paragraph 4.2)
- **Check function type:** Node executes this stage only when accuracy type is non-accurate, where selects function type (AVG, MAX, or MIN) to be applied on payloads of saved packets.
- **Create aggregated packet includes function result:** Create aggregated packet includes value that resulted from applying the previous function.
- **Create aggregated packet includes accurate data:** This stage is executed only if accuracy type is accurate. In this stage aggregator node doesn’t apply any function on the saved values, but write each value with its source in a payload of new aggregated packet.

3.2.1.2 Based on destination tool (D):
This tool works almost in the same mechanism of (Destination-Sources) but it makes aggregation relied on the destination address only to handle packets.
(It differs in ‘Filtering incoming packets’ block).

3.2.1.3 Any_2_any (A):
This tool works almost in the same mechanism of (Destination-Sources) also, but aggregator node in this tool aggregates any data packet received regardless its source and destination address, where creates independent buffer for each destination’s packets and save each packet sent to this destination regardless the source from which coming.
(It differs in ‘Filtering incoming packets’ block only)

3.2.2 Aggregator nodes selection ways:
DAF decides which node should be aggregator node based on one of two ways:
1) Manually, where user selects the aggregator node using OTCL commands. Tools in this case called ‘Manual Tools’.
2) Automatically, where DAF selects the aggregator node according to hops number far from destination, and uses AODV routing table in the node to reach required information. In this way, users should determine the hops number that they want to be far from the destination. Tools in this case called ‘Automatic Tools’.

Tools which use manual way	Tools which use automatic way
D-S-M	D-S-A
D-M	D-A
A-M	A-A

3.3 Results extraction in DAF:

In DAF, users are able to extract results easily by OTCL commands. In this context, users able to display results in three optional forms *Energy Form*, *Aggregation Details form*, *Aggregation delay and received values form.*

- Energy Form: this option makes users able to get measurements about remained energy in any node which user select. Then DAF generate a text file called 'energy_measurements' and write to it pairs of values for time and remained energy of the node at every transmission process in that node. So, user could plot this values by any plotting application. (graphs in this paper have plotted by Gnu Plot v5.0).
- Aggregation Details Form: this option provides detailed information about aggregation process in any node selected by user as well. So that, DAF generates file which reveals the aggregation process (all packets received and all packets sent).
- Aggregation delay: With this option, DAF can generate file to reveal the delay of each aggregation iteration in each node with accumulator counter of aggregation delay over for all simulation time.
- Received Values Form: DAF shows details about aggregated packets which received by destination, and write all these details (including source and payload values) in a file called 'reads_in_destinations'.

Note: Users could make DAF generate one or more of these files.

3.4 Power consumption reduction using DAF:

Aggregation mechanism in DAF aims to reduce transmission times that nodes do, through aggregating many packets which sent to the same destination and sending them collectively in one packet called 'Aggregated Packet'.

We have supposed a WSN scenario and measured power consumption reduction using 'Energy Form' (see section 3.3 to read about 'Energy Form') when DAF was ON.

3.4.1 Scenario Topology:

We supposed the WSN's Topology as figure (7) shown, where all green nodes sends periodical packets to the Sink (node 0) through the router nodes (1, 2, 3 and 4) which work as aggregators according to DAF mechanism.

3.4.2 Scenario Parameters:

We have 12 nodes, 4 routers and one sink, and each of them has 3 Jules energy. Routers are aggregating sent packets, from green nodes to the sink, according to DAF. Simulation time is 500 seconds and all green nodes send CBR packets with interval one second and packet size 20 bytes. UDP protocol is used in the transport layer with packet size 28 bytes. In addition, AODV protocol is used in the Network layer, and IEEE 802.15.4 standard for the first two layers (physical and data link layers).

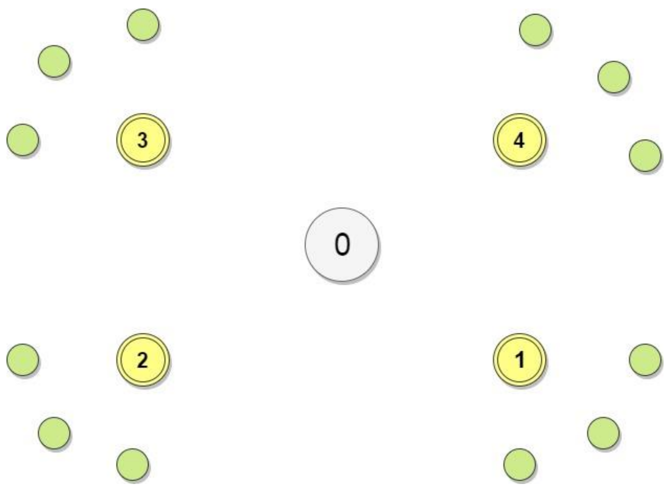

Figure (7): The supposed WSN's scenario topology.

Simulation's results with DAF parameters are shown in table (1).

Node	1	1	1	1	1	1
Tool	-	D-M	D-M	D-M	D-M	D-M
Maximum packets	-	6	9	11	18	25
Delay average (s)	-	1.756	2.716	3.415	5.81	8.186
Life Time (s)	270	388	398	448	458	462

Table (1): Simulation's results and DAF parameters which are used in the scenario.

Graphical view of these results is shown in figure (8).

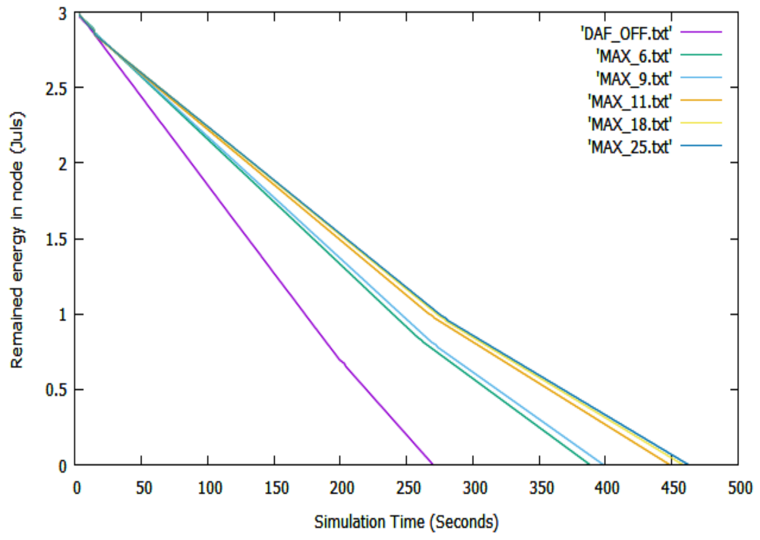

Figure (8): Remained energy in node (1) to simulation time.

3.5 investing traffic to reduce aggregation delay:

Work mechanism in DAF relies on duration of one aggregation process and maximum packets number, when any of them reaches a certain threshold, node aggregator decides to end the ongoing aggregation process and sends

an aggregated packet consists of all saved values from received packets during that aggregation process.
Note: DAF uses the last received packet that ends an aggregation iteration and makes buffer full to send the saved values, where it changes the source of that packet and its 'TTL' field then let it go on instead dropping it. DAF uses this mechanism because that node aggregator is a router node and don't have transport layer agent (UDP, TCP, STCP,. . . . .) to generate new data packet.
This process takes some time, which considered as a delay time resulted from aggregation process because the first packet received during the aggregation process should wait for aggregation time to be expired or number of received packets to reach the threshold.
So that, for aggregator node, it could be said: the more packets received, the less aggregation delay.
In the previous scenario, (see paragraph 3.4.1 and 3.4.2), we repeated the simulation several times, each of them with different CBR interval for green nodes (as it is known the less interval, the more traffic generated) and we noticed that the aggregation delay decreased when the traffic generated increased as figure (9) shows.

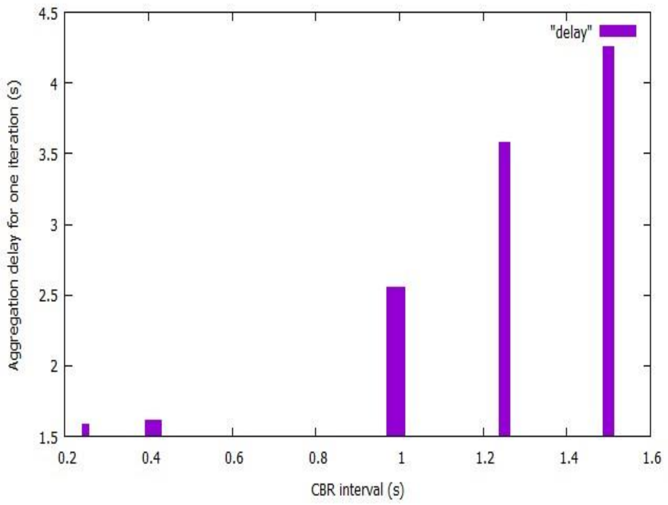

Figure (9): reveal that, when CBR interval got up (traffic got down) the aggregation delay got up.

3.6 Commands for using DAF:

DAF provides many commands to run its tools, and I will illustrate them in this section for Manuals tool and Auto tools as well.
First of all, we are dealing with AODV class that by which push the commands to DAF. So that we have to create objects from class AODV to run the commands as following

```
set obj [new Agent/AODV]     #(command number 1)
```

This way is used for commands that are not related to specific node (as make DAF run for example).
For run DAF should write this command

```
$obj DAF_ON                    #(command number 2)
```

3.6.1 Commands for Manual tools:

These tools are specified for certain node, so that user shouldn't create a new AODV class, because they should use AODV object which is specified for the node that user want to make it aggregator node.

3.6.1.1 Destination_Sources Manual (D_S_M):

This command is used to run destination-sources tool where aggregator nodes are selected manually.

First command:

```
[$node_(X) set ragent_] DAF D_S_M_init max_pkt interval accuracy dest NO_src fun_type #(command number 3)
```

Underlined words are parameters for the tool.
Where:

- $ns: is simulator object (user is free for choice any name he wants).
- $node_(X): is node object (user is free for choice any name he wants)
- [$node_(X) set ragent_]: AODV class object that specified for 'node_(x)'.
- DAF: is key word for DAF commands.
- D_S_M_init: is key word to run and initiate specified node to run Destinatio_Sources tool, and become aggregator node as a result.
- Max_pkt: specifies maximum number of packets in each aggregation iteration that node aggregates them.
- Accuracy: specifies accuracy type (1 for accurate) or (0 for non-accurate).
- Dest: specifies address of destination that node should aggregate packets which sent to it.
- NO_src: number of sources that node should aggregate their packets.
- Fun_type: (in case of accuracy_type is non-ccurate) specifies function type that node should be applied to packets' values (0 for average, 1 for maximum, 2 for minimum).

Second command: #(command number 4)

```
[$node(1) set ragent_] DAF D_S_M_getsrcs src1 src2 src3....
```

Where

- D_S_M_getsrcs: is key word to push sources address that user has specified its number in the D_S_M_init command.

3.6.1.2 Destination Manual (D_M):

This command is used to run destination-sources tool where aggregator nodes are selected manually.

[$node_(X) set ragent_] DAF D_M_init max_pkt interval accuracy dest fun_type #(command number 5)

Underlined words are parameters for the tool.
Where:

- $obj: is AODV class object created previously (see command number 1).
- $node_(X): is node object (user is free for choice any name he wants)
- [$node_(X) set ragent_]: AODV class object that specified for 'node_(x)'.
- DAF: is key word for DAF commands.
- D_M_init: is key word to run and initiate specified node to run Destinatio_Sources tool, and become aggregator node as a result.
- Max_pkt: specifies maximum number of packets in each aggregation iteration that node aggregates them.
- Accuracy: specifies accuracy type (1 for accurate) or (0 for non-accurate).
- Dest: specifies address of destination that node should aggregate packets which sent to it.
- Fun_type: (in case of accuracy_type is non-accurate) specifies function type that node should apply to packets' values (0 for average, 1 for maximum, 2 for minimum).

3.6.1.3 ANY_2_ANY Manual (A_M):
This command is used to run destination-sources tool where aggregator nodes are selected manually.

[$node_(X) set ragent_] DAF A_M_init max_pkt interval accuracy fun_type #(command number 6)

Underlined words are parameters for the tool.
Where:

- $ns: is simulator object (user is free for choice any name he wants).
- $node_(X): is node object (user is free for choice any name he wants)
- [$node_(X) set ragent_]: AODV class object that specified for 'node_(x)'.
- DAF: is key word for DAF commands.
- A_M_init: is key word to run and initiate specified node to run Destinatio_Sources tool, and become aggregator node as a result.
- Max_pkt: specifies maximum number of packets in each aggregation iteration that node aggregates them.
- Accuracy: specifies accuracy type (1 for accurate) or (0 for non-accurate).
- Fun_type: (in case of accuracy_type is non-accurate) specifies function type that node should apply to packets' values (0 for average, 1 for maximum, 2 for minimum).

3.6.2 Commands for automatic tools:
These commands specified generally for all nodes, because aggregator nodes are selected based on AODV routing table (to remain through the path) and on hop counts (to determine how many hops it is far from destination), so that users don't need to use AODV objects of nodes, but need just to create new object from AODV class as I did in (command number 1).
Surely user has to make DAF ON also (command number 2).

3.6.2.1 Destination_Sources Auto (D_S_A):
This command is used to run destination-sources tool where aggregator nodes are selected automatically.

First command:

$obj DAF D_S_A_init max_pkt interval accuracy dest hops NO_src fun_type #(command number 7)

Underlined words are parameters for the tool.
Where:

- $obj: is AODV class object created previously (see command number 1).
- DAF: is key word for DAF commands.
- D_S_A_init: is key word to run and initiate specified node to run Destinatio_Sources tool, and become aggregator node as a result.
- Max_pkt: specifies maximum number of packets in each aggregation iteration that node aggregates them.
- Accuracy: specifies accuracy type (1 for accurate) or (0 for non-accurate).
- Dest: specifies address of destination that node should aggregate packets which sent to it.
- Hops: specifies number of hops that node aggregator should be far from destination.
- NO_src: number of sources that node should aggregate their packets.
- Fun_type: (in case of accuracy_type is non-accurate) specifies function type that node should apply to packets' values (0 for average, 1 for maximum, 2 for minimum).

Second command: #(command number 8)

$obj DAF D_S_A_getsrcs src1 src2 src3....

Where

- D_S_A_getsrcs: is key word to push sources address that user has specified its number in the D_S_A_init command.

3.6.2.2 Destination Auto (D_A):

This command is used to run destination-sources tool where aggregator nodes are selected automatically.

```
$obj DAF D_A_init max_pkt interval accuracy dest
hops fun_type        #(command number 9)
```

Underlined words are parameters for the tool.
Where:

- $obj: is AODV class object created previously (see command number 1).
- DAF: is key word for DAF commands.
- D_A_init: is key word to run and initiate specified node to run Destinatio_Sources tool, and become aggregator node as a result.
- Max_pkt: specifies maximum number of packets in each aggregation iteration that node aggregates them.
- Accuracy: specifies accuracy type (1 for accurate) or (0 for non-accurate).
- Dest: specifies address of destination that node should aggregate packets which sent to it.
- Hops: specifies number of hops that node aggregator should be far from destination.
- Fun_type: (in case of accuracy_type is non-accurate) specifies function type that node should apply to packets' values (0 for average, 1 for maximum, 2 for minimum).

3.6.2.3 ANY_2_ANY Auto (A_A):

This command is used to run destination-sources tool where aggregator nodes are selected automatically.

```
$obj DAF A_A_init max_pkt accuracy hops fun_type
#(command number 6)
```

Underlined words are parameters for the tool.
Where:

- $obj: is AODV class object created previously (see command number 1).
- DAF: is key word for DAF commands.
- A_A_init: is key word to run and initiate specified node to run Destinatio_Sources tool, and become aggregator node as a result.
- Max_pkt: specifies maximum number of packets in each aggregation iteration that node aggregates them.
- Accuracy: specifies accuracy type (1 for accurate) or (0 for non-accurate).
- Hops: specifies number of hops that node aggregator should be far from destination.
- Fun_type: (in case of accuracy_type is non-accurate) specifies function type that node should apply to packets' values (0 for average, 1 for maximum, 2 for minimum).

3.6.3 Command for extract results in DAF:

As I mentioned in section (3.3), DAF could generate results in three forms, and user could determine which these forms he wants by this command:

```
$obj DAF results energy-form aggregation-details reads-in-
dest aggregation-delay node-id
```

Where:

- $obj: is AODV class object created previously (see command number 1).
- DAF: is key word for DAF commands.
- Results: is key word to make user able to set results extraction configuration.
- Energy-form: 1 for ON, 0 for OFF.
- Aggregation-details: 1 for ON, 0 for OFF.
- Reds-in-dest: 1 for ON, 0 for OFF.
- Aggregation delay: 1 for ON, 0 for OFF.
- Node-id: (optional) used just if energy-form is ON, to determine which node will DAF traces its energy values.

4 For getting the source code of DAF:

Please click this link below and follow the set up instructions.
****
link****************************

5. Conclusion:

This paper has introduced a new frame work in NS-2 called 'DAF' to support simulations that including aggregator nodes.
After adding the source code of DAF, users will be able to configure simulations contain aggregator nodes with wide options by OTCL commands only.

6. Future works:

As a future works, may be the ones that will enhance DAF performance to provide more reliability and mode options for users will be the most important.

7. Acknowledgements:
We have to thank Eng. Mouhannad Essa in Tishreen University for the support that he has provided to achieve this frame work.

8. References:

[1] Mousam Dagar and Shilpa Mahajan - Data Aggregation in Wireless Sensor Network: A Survey - International Journal of Information and Computation Technology. ISSN 0974-2239 Volume 3, Number 3 (2013), pp. 167-174.
[2] Bindiya Bhatia, M.K. Soni, and Parul Tomar - Power Optimized Secure AODV Routing Protocol Based on Mobile Agents - International Journal of Applied Engineering

Research ISSN 0973-4562 Volume 12, Number 16 (2017) pp. 5884-5893.

[3] M. Lakshmi, P. Velmani, P. Arockiya Jansi Rani - Study on Data Representation and Aggregation in WSN - International Journal of Computer Applications (0975 – 8887) Volume 165 – No.11, May 2017.

[4] Teerawat Issariyakul, Ekram Hossain - Springer 2012_Introduction to Network Simulator NS2.

[5] International Journal of Advanced Research in Computer Science and Software Engineering, Volume 6, Issue 3, March 2016_Comparative Study of Network Simulator: NS2 and NS3_Pallavi S. Katkar, Dr. Vijay R. Ghorpade

[6] Electrical Engineering and Information Communication Technology (ICEEICT), 2016 3rd International Conference on_A Packet Level Simulation Study of Adhoc networkwith Network Simulator-2 (NS-2)_ Iftekharul Mobin, Sifat Momen, Nabeel Mohammed3.

[8] The VINT Project 2011_The ns Manual (formerly ns Notes and Documentation)_Kevin Fall, Kannan Varadhan
[9] Saeid Pourroostaei Ardakani - Data aggregation routing protocols in wireless   sensor networks: A taxonomy - International Journal of Computer Networks & Communications (IJCNC) Vol.9, No.2, March 2017
[10] Vaibhav Pandey, Amarjeet Kaur and Narottam Chand - A review on data aggregation techniques in wireless sensor network - Journal of Electronic and Electrical Engineering Vol. 1, Issue 2, 2010
[11] NETWORKING 2007. Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet- Raghupathy Sivakumar ،Eylem Ekici ،Jaudelice Cavalcante de Oliveira ،Janise McNair, Ian F. Akyildiz