ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 12, December 2013

Adhoc On-Demand Distance Vector Protocol For Energy Efficiency

Abhilash C S[1], Abhishek Varshney[2], Dilip S[3] , Navik Yogesh Laljibhai[4] , Pradyot H Adavi[5]

M.Tech student, Department of Networking and Communication, I.I.I.T.B., Bangalore, India [1,3,5]

M.Tech student, Department of Computer Science, I.I.I.T.B., Bangalore, India [2]

M.Tech student, Department of Embedded Systems, I.I.I.T.B., Bangalore, India [4]

Abstract: The use of computer networks is drastically growing and the need for enhancing the existing network protocols and enforcing communication security thus is increasing. Tools like network simulators are used by researchers in order to test new scenarios and protocols in a controlled and reproducible environment. They allow the user to represent various topologies, simulate network traffic using different protocols, visualize the network and measure the performances. Although they are very useful, most of the widely used network simulators do not scale. Trying to simulate medium to large networks will result in a long simulation time unsuitable for investigating protocols. Some of the current network simulators implement various methods for accelerating the simulations by means of using parallel computation or changing the data representation of the nodes. However, most of the optimization techniques require additional hardware resources as a computational grid or cluster and deep knowledge of the structure of the simulator. A wide assortment of network simulators are available and most of them share the same issues when it comes to computation time and scalability. Simulating a network of 100 nodes on a normal CPU using standard simulator takes a long time. In this project we try to alleviate these problems by building a simulator on a Graphical Processing Unit [GPU] based on parallel execution on using the CUDA programming model developed by nVIDIA.

Keywords: AODV, NS2, GPU, CUDA, Energy, Runtime

I. INTRODUCTION

A Wireless sensor network consisting of mobile nodes must be energy efficient in order to be useful in areas where power is scarce. Power failure of a mobile node affects overall network efficiency to forward packets. Most of the energy gets consumed when an event generated at any node tries to reach a base station using a protocol. Simulate a network of about 100 nodes and measure energy consumption on the average for a random event to reach the base station under a given protocol. Any standard protocol such as DSR or AODV may be used for implementation.

In this project we focus the possibilities for parallel implementations of wireless sensor network of 100 mobile nodes using *AODV* protocol [1]. Specifically we investigate and implemented its simulation on *GPU* in order to utilize its resources and obtain faster simulations. We propose a possible architecture for achieving this goal.

This paper is organized as the follows: Section II provides a brief survey on similar systems; Section III proposes the project architecture; Section IV describes the implementation; Section V shows the observation and results of the experiment; Section VI concludes this paper.

II. BRIEF SURVEY ON SIMILAR SYSTEMS

There are two types of approaches for developing a parallel network simulator. One can create the parallel simulator from scratch, where all the simulation software is custom designed for a particular parallel simulation engine. For this approach a significant amount of time and effort are necessary to create a usable system. This is so, because new models must be developed, and therefore validated for accuracy. The second approach for developing parallel/distributed simulation involves interconnecting with existing simulators. These federated simulations may include multiple copies of the same simulator (modeling different portions of the network), or entirely different simulators. Few parallel implementations of this approach are presented in the following:

A. Global Mobile Information System Simulator (GloMoSim): It is a scalable simulation library designed at UCLA Computing Laboratory to support studies of large-scale network models, using parallel and/or distributed execution on a diverse set of parallel computers [2]. GloMoSim beside sequential adopts parallel simulation model using libraries and layered API. The libraries are developed using Parsec [3], which is a parallel C based programming language which uses message based approach.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 12, December 2013

B. Scalable Simulation Framework (SSFNet): It claims that is a standard for parallel discrete event network simulation [3, 4]. SSFNET's commercial Java implementation is becoming popular in the research community, but SSFNet for C++ (DaSSF) does not seem to receive nearly as much attention, probably due to the lack of network protocol models. It is a high performance network simulator designed to transparently utilize parallel processor resources, and therefore scales to a very large collection of simulated entities and problem sizes.

C. Georgia Tech Network Simulator (GTNetS): It is a network simulation environment which uses C++ as a programming language [5]. GTNetS is designed for studying the behavior of moderate to large scale networks. The simulation environment is structured as an actual network with distinct separation of protocol stack layers.

D. OMNeT++: It is a network simulation library and framework, primary used for simulation of communication networks, but because of its flexible architecture can be used to simulate complex IT systems too. OMNeT++ offers an Eclipse based IDE and the programming language used is C++ [6, 7].

In our approach we have developed an application which runs AODV protocol on GPU. Our application takes much lesser time for many events triggering at the same time at the network of mobile nodes when compared with any application running on CPU. As events triggering at the same time are served in parallel using GPU in our application while on CPU it is being served sequentially.

III. PROJECT ARCHITECTURE : AODV PROTOCOL ON GPU

In our architecture each node performs its operations independently without getting affected by operations of any other node in the network. Every node in the network is mapped to a single thread working independently from other threads in GPU. We have implemented Single Instruction Multiple Data (SIMD) concept to achieve parallelism [8].

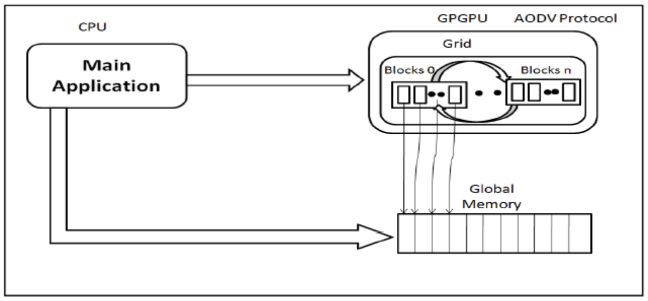

Fig.1- Project architecture

In the above Fig 1., the main application creates an array of structures on the global memory where each element of the structure defines the properties of the node. The number of threads created is equal to the number of nodes specified in the network. Every element of the structure array is mapped to its corresponding thread. Every thread can access any of the structure elements to perform its operation. Every node i.e. a thread executes the following set of functions:

- Sending Hello Packets.
- Serving its Events-Queue for communication.
- Serving its Request-Queue.
- Refreshing its
  - Neighbor's list.
  - Routing table.
  - Broadcast-id table.

IV. IMPLEMENTATION

System setup for the project was done on an Ubuntu 11.04 system containing an NVIDIA GeForce GTX 480 graphics card.

1. CUDA development environment was set up on the host OS with the following software versions installed:
   - CUDA Toolkit for Ubuntu Linux 10.10.
   - Developer Drivers for Linux version 270.41.19
2. NS-2.35 was compiled and installed from its source codes using NVCC compiler.
3. Eclipse SDK 3.5.2 using Java JDK 1.6 for ubuntu 64-bit.

A. MANET with AODV protocol on CPU

Lowering energy consumption is a key goal in many multi-hop wireless networking environments, especially when the individual nodes of the network are battery powered. This requirement has become increasingly important for new generations of mobile computing devices (such as PDAs, laptops, and cellular phones) because the energy density achievable in batteries has grown only at a linear rate, while processing power and storage capacity have both grown exponentially [9]. As a consequence of these technological trends, many wireless-enabled devices are now primarily energy-constrained; while they possess the ability to run many sophisticated multimedia networked applications, their operational lifetime between recharges is often very small (sometimes less than *1* hr). In addition, the energy consumed in communication by the radio interfaces is often higher than, or at least comparable to, the computational energy consumed by the processor. Various energy-aware routing protocols have thus been proposed to lower the communication energy overhead in such multi-hop wireless networks. In contrast to conventional wired routing protocols that try to utilize the minimum-hop route (one that minimizes the number of unique links), these protocols typically aim to utilize the most energy-efficient route [9].

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 12, December 2013

These protocols exploit the fact that the transmission power needed on a wireless link is a nonlinear function of the link distance, and assume that the individual nodes can adapt their transmission power levels. As a consequence of this, it turns out that choosing a route with a large number of short-distance hops often turns out to consume significantly less energy than an alternative one with a few long-distance hops. For wireless links, a signal transmitted with power Pt over a link with distance D is attenuated and is received with power [9].

$$Pr = C * K(D) >= 2$$

Where C is a constant, K(D) depends on the propagation medium, antenna characteristics, and channel parameters, such as the radio frequency. Since most wireless receivers are able to correctly decode the received signal as long as its power is above a certain fixed threshold, energy-efficient algorithms typically set the transmission power to be proportional to DKðDÞ. If the link cost in a routing algorithm is then assigned proportional to this transmission power, a minimum-cost path will then correspond to a route that consumes the lowest cumulative energy for a single-packet transmission. To prove the above mentioned concept i.e. choosing a route with a large number of short-distance hops often turns out to consume significantly less energy than an alternative one with a few long-distance hops [9].

B. **AODV Protocol for GPU**

To simulate the actual network scenario we prepared various models as follows:

1. Power and coverage radius modeling:

We have used two-ray ground model to define the coverage radius for the nodes.

• Parameters that has been fixed
– Transmission power (Pt): 165E-3 W
– Transmitter Gain (Gt): 1
– Receiver Gain (Gr): 1
– Transmitter antenna height (ht): 1.5 m
– Receiver antenna height (hr): 1.5 m
– Wavelength (l): 0.328 m
– Transmission frequency: 914 MHz
– Transmission speed: 1 Mbps
– Threshold reception power (Prt): 6.3E-9 W
• Based on these parameters we calculate the coverage radius for the node from the following equation [10]

$$D = [(Pt * Gr * Gt * l2) / (4 * pi)*2 *Prt)]^{-2}$$

* It turns out to be 133.58m

2. Delay modeling:

Overall delay for a packet to reach from one node to another consists of transmission delay at the node and the propagation delay in the medium [10].

•***Transmissiondelay*=(*packetsizeinbits*)/(*transmissionspeed*)**
*****It turns out to be 0.000576sec

•***Propagationdelay=(maxdistancebetweentwonodes)/(wavepropagationspeed)***
***** It turns out to be 0.0000006679sec

• ***Overalldelay* = *Transmissiondelay* + *Propagationdelay***.
*****It turns out to be 0.0005766679

3. Energy modeling:

During transmission and reception of the packet, a node consumes finite amount of energy. The energy modeling is done in the following manner [10].

• Parameters fixed for energy modeling
– Transmission or receive energy per bit for a node Ebit: 50nJ
– Average control packet size: 72 bytes.
– Initial Energy of node: 1000 J
• *TransmissionEnergyperpacketEt* : 72 * 8 * (50*nJ*).

4. Finding neighbors:

• Each node finds its coverage radius (D) based on the above equation and decides it neighbor.
• At every HELLO_INTERVAL (1000ms) nodes update their neighbor list.

V. Observation and Results

A. **Remaining energy in MANET using AODV protocol on CPU.**

Figure below shows the simulation for a network of 100 mobile nodes. The event shown is a scenario where a source - node 0 and destination (sink) - node 1 are communicating using AODV protocol.

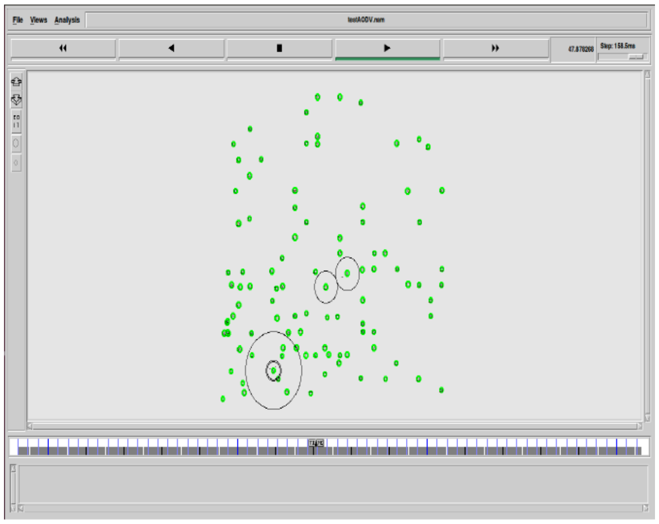

Fig.2 - Node communication in NS2

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 12, December 2013

The simulation has been done on NS2 and the graph is as shown in the *Fig 2*. The graph plotted is Number of intermediate nodes between source and sink on the x-axis and the Remaining energy of the node in the y-axis. It is observed that as the number of intermediate nodes increases between source and sink, the Energy remaining of the node increases following the equation [9],

$$E = 7.578 \ln(n) + 21.17 \text{—— } \textit{equation.1}$$

Where, E = Energy Remaining of the node after transmission in Joule. n = Number of intermediate nodes between source and sink. The transmitter and all the other nodes can be adjusted.

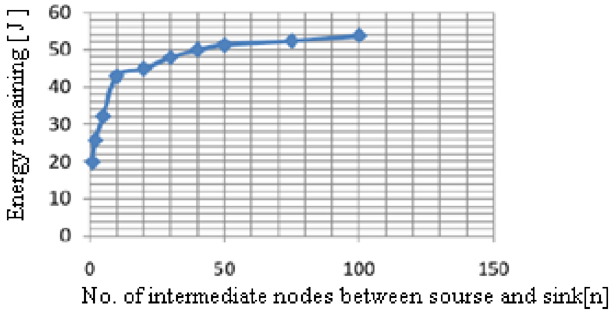

Fig.3 - Average energy remaining in the source node[J]

Conclusion from the Fig 3. is that, consider a scenario wherein the transmitter(T) sends signal with transmission power 'Pt' to a Receiver(R) at the distance of 'D' m, if there were no intermediate nodes between T and R then the T should transmit with its full power Pt so that the signal reaches R. When an intermediate node (N1) is placed between T and R. T should transmit with *Pt/2* power to N1 and then N1 transmits the signal to R with *Pt/2* power. Thus the Average power/Energy of the T node is reduced. The same explanation holds when the Number of intermediate nodes is increased. It follows a particular curve given by *equation 1* [9].

Another Graph was plotted between Number of events on the x-axis and Energy Remaining of the node on the y-axis as in *Fig 4*. It is observed that the Energy of the node decreases as the number of events increases in the network. Conclusion from the above graph Consider a scenario wherein there is only event in the network (say TCP connection), then the Remaining energy of the node is say 'Er'. When the number of events is increased the 'Er' is decreased as each event consumes power [9].

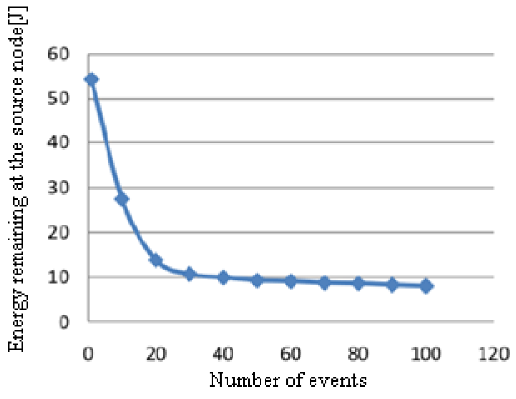

Fig.4 - Energy remaining at the node[J]

B. AODV Protocol on GPU –

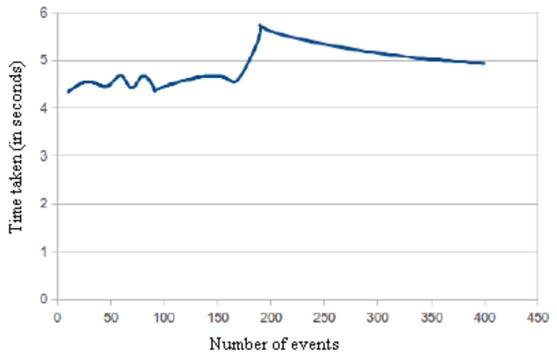

Fig. 5. Runtime on GPU

The Fig 5. shows the Statistical Analysis of Simulation of AODV protocol running on GPU for 20 seconds for multiple events. And the graph shows the independence of the simulation time taken with respect to the number of events.

From the above graph we can conclude that there is not much deviation in the simulation time as we increase the number of events. This is mainly because all the events are served in parallel and all the threads representing the nodes in the network are running in parallel and independent of each other.

We would also like to mention that a one-to-one comparison with NS2 running on CPU/GPU cannot be made because NS2 makes use of all the higher layer protocols which are absent in our simulator running on GPU. Hence the time taken would not be will not be accurate although they would provide an approximate estimate.

VI. CONCLUSION

We have successfully developed a Simulator which runs AODV protocol on GPU. The proposed simulator takes very less time for simulating a network of mobile nodes with many events triggering at the same time as compared to simulating on CPU. Each of the multiple events triggered at the same time are assigned a thread and run in parallel in GPU as opposed to CPU where the multiple events occurring at the same time are served sequentially.

From our experiment it is evident that there is a significant reduction in time it takes to perform simulation of 100 nodes. Hence other networking protocols can be implemented on GPU and a robust simulator like NS2 can be built. Also the input for defining the topology of the network can done using the TCL language and then hook up this with the CUDA files where the logic of the protocols are present. On the similar lines, Standard network protocols such as NS2 can be made to work on GPU for Scalability and Robustness.

ACKNOWLEDGMENT

We are extremely grateful to those who have helped and supported us during the project. Our deepest thanks to our advisors Prof. Shrisha Rao and Prof Poonacha P G. for their continuous encouragement and suggestions throughout the course of this work. It was our pleasure to work under their guidance regarding project by explaining many scenarios where the project can be used.

REFERENCES

[1] Perkins, C.; Belding- Royer, E.; Das, S. (July 2003) "Adhoc On-Demand Distance Vector (AODV) Routing".[RFC 3561]
[2] Zeng, X., Bagrodia, R., Gerla, M., "*GloMoSim: A Library forParallel Simulation of Large-Scale Wireless Networks*", in Proc.12th Workshop on Parallel and Distributed Simulation, Banff, Alta.Canada, 1998, p. 154-161.
[3] Parallel Simulation Environment for Complex Systems (PARSEC), retrieved June 2010 from http://pcl.cs.ucla.edu/projects/parsec/.
[4] Cowie, J.H., Nicol D.M., and Ogielski A.T., "Modeling the GlobalInternet*", Computing in Science and Engineering*, 1999.
[5] Riley, G.F., "The Georgia Tech Network Simulator", in Proc. of the Workshop on Models, Methods, and Tools for Reproducible Network Research (MoMe Tools), 2003.
[6] Varga, A., "The OMNeT++ discrete event simulation system", Proc. of the European Simulation Multiconference (ESM '2001), Prague, Czech Republic, 2001.
[7] Sekercioglu, Y. A., Varga, A., Egan, G. K., "Parallel Simulation Made Easy With Omnet++", in Proc. of the European Simulation Symposium (ESS2003), Oct. 2003, Delft, The Netherlands.
[8] Buck, Ian, "*GPU Computing: Programming a Massively Parallel Processor*" in Code Generation and Optimization, 2007. CGO '07. International Symposium, 11-14 March 2007.
[9] S. Banerjee and A. Misra, "*Minimum energy paths for reliable communication in multihop wireless networks*", Proc Mobihoc Conf , June 2002.
[10] Wendi Beth Heinzelman,"*Application-Specific Protocol Architectures for Wireless Networks*",in his thesis document in Massachusetts Institute of Technology,June 2000