Graph Database: Neo4J da teoria Ã pratica.

Iuri Andreazza
Unidade AcadÃªmica de GraduaÃ§Ã£o
UNISINOS
SÃ£o Leopoldo, Brazil
iuri.andreazza@gmail.com

Eduardo Pereira da Silva
Unidade AcadÃªmica de GraduaÃ§Ã£o
UNISINOS
SÃ£o Leopoldo, Brazil
eduardobursa@gmail.com

Talita Audibert
Unidade AcadÃªmica de GraduaÃ§Ã£o
UNISINOS
SÃ£o Leopoldo, Brazil
tali.audibert@gmail.com

Paulo GrÃbin
Unidade AcadÃªmica de GraduaÃ§Ã£o
UNISINOS
SÃ£o Leopoldo, Brazil
plgrabin@gmail.com

***Abstract*—**

***Keywords*-Graphs;Databases; Neo4j; No-SQL;**

I. IntroduÃ§Ã£o

What is a Graph Database?

A Graph Database is not a database to store graphics or images as its name may suggest. Itâs a database that uses graph structures such as nodes, properties and edges to represent and store data. In addition, it allows you to represent any kind of data without the limitation of regular databases.

To better understand, letâs talk about each of these structures individually and what they represent:

Node: A node can be used to represent any type of entity that you can think of, be it a business, a blog post, a location, an oil rig, a city, etc. Graph databases donât care what type of data theyâre representing.

Properties: Properties, sometimes called attributes, are named values that relate to nodes. For example, if we take into consideration our City representation of a node, one of the properties would be ânameâ, another would be âpopulationâ, and so on.

Edges: Edges, sometimes called Relationships, connect nodes-to-nodes and organize them into arbitrary structures such as a Map, List or a Tree. Itâs important to note that when a node is the start node of a relationship, the relationship from that nodeâs perspective will be an outgoing relationship. And when a node is at the end of a relationship, the relationship from that nodeâs perspective will be an incoming relationship. Understanding this will make it easier for you to follow the examples.

What can a Graph Database be used for?

There are many scenarios for which one could consider using a Graph Database. Before I list a few examples, as with any architectural and technical decision, you need to analyze all possible solutions. Then, you can select those that are best for you, according to your specifications.

Some of these scenarios include: social networking, fraud detection, people/movie/music recommendation, manufacturing, etc. Using the social networking example, it would be somewhat trivial to do something like âgiven the fact that Bob is my friend, give me all friends that are friendâs of friendâs of friendâs of Bobâ. This is possible because of the path-finding algorithms involved are easy to implement by traversing through the graph. Imagine doing that through a relational database? A nightmare!

Another advantage when using a Graph database is that you can easily and more naturally model a domain using a whiteboard or a piece of paper. Specifically, nouns that are used become nodes, verbs become relationships, and adjectives and adverbs become properties.

Graph Database Example

Letâs examine the following scenario and how we would structure it using a Graph Database: John is friends with Bob, and Bob is friends with Mark. Visually, this is how we could represent the scenario:

You notice that the visual representation is pretty much how we verbally expressed our scenario. This is a very simple example that we can use to compare the implementation of a traditional SQL database structure versus a Graph database structure:

Traditional SQL

In a traditional SQL structure, each row in the users table represents a user, and each row in the friends table represents a relationship between two users. If we decide to add additional properties to the users table at a later time, we would have to alter the base structure of that table. And if we wanted to add new properties to only a subset of users we would still have to alter the entire users table, or create a new table to accommodate the new values just for the subset of users. Not an ideal scenario when dealing with tens of millions of records.

Graph Database

Graph Database Structure

On the other hand, a Graph database has no set structure or schema for the data, much like a NoSQL database. Now, letâs understand what the graph above is representing: each node represents an entity â a User in our case; each node contains property values, in this case itâs the Userâs name; and each line represents a relationship between the nodes. This is as simple as it gets. And to complement the scenario described in the traditional SQL example, if we did decide to add additional properties to a subset of users, we could easily perform this action on a per-node level instead of a table-wide transaction.

What is Neo4j

Neo4j is âThe Worldâs Leading Graph Databaseâ according to Neo Technology, the developer behind the project. Itâs a commercially supported open-source graph database implemented in Java and some of the key characteristics include:

Data representation using an intuitive graph-oriented model. Binding for a number of languages, including Ruby, Python and Closure. Disk-based, native storage manager completely optimized for storing graph structures for maximum performance and scalability. Massive scalability, it can handle graphs of several billion nodes/relationships/properties on a single machine. Easy to use and convenient object-oriented API. Handles large graphs that donât fit in memory with durability and a fully persistent transactional store. Powerful traversal framework for high-speed traversals in the node space. Optimized for highly connected data. Small footprint, only about 750k jar file. It provides a REST interface for languages not supported by its bindings. Itâs capable of traversing depths of over 1000 levels at millisecond speeds. It provides a dual license: open source and commercial. It integrates seamlessly with Lucene, providing full-text search to nodes and relationships, including phrase queries, wildcard queries, proximity queries, ranked searching, sorting, and more.

A. OrganizaÃ§Ã£o

Este artigo estÃ¡ organizado da seguinte maneira, tendo que, na sessÃ£o **??** temos o trecho introdutÃ³rio,

II. Conceito

III. Abordagem PrÃ¡tica

IV. Trabalhos relacionados

V. ConclusÃ£o

At the end of the 20th century and beginning of the 21st century, the evolution of computing and communication technologies made mobile computing and wireless computer networks widespread, which increased their presences in academic, industrial and home environments. This allows anyone to easily access computer networks and databases information. The main issue that stress wireless networks is the increasing presence of notebooks, cell phones and many other mobile devices that enjoy the infrastructure to access the Internet. With the availability of online services and tools for both desktop and mobile devices users, the need for constant information and connectivity becomes essential, as well as the quality of that information [**?**].

Data networks clients are more and more present in the digital world and they demand constant connectivity. With the advent of online and digital tools, the need for quality of connection rises, forcing the current infrastructure to almost achieve its limit. Although networks can support a great amount of users with high level of quality, these models did not predict new incoming users (common day-do-day devices) like refrigerators, cell phones, tablets, printers, information center, cars and houses, for example. The omnipresence of computational devices stresses the network structure, since it foresees that the whole environment will be connected to the devices, thus needing a constant and intense data flow [**?**][**?**].

Most common wireless networks works in a centralized way, *i.e.*, there is a single access point for all clients. In these scenarios, the network becomes exposed to failures and if the access point becomes unavailable or unreachable, all of its clients lose the network connection. An existing pattern that provides a scenario in which even if the access point is down the connection among the clients still exists, is the mesh pattern. This model of network has some important functionalities such as self-repairing, self-configuration and self-layout (among others), giving an advantage over other models [**?**]. However, the mesh model currently does not offer a good support for quality of service, *i.e.*, data transmission inside the mesh is slow and tends to need retransmission, making it impractical to maintain a constant stream of data to the clients in the mesh.

With these new types of users joining the mesh infrastructure out of research environments, this model becomes interesting for a large scale implementation. In this way, it is possible to make an environment in which new devices can join the existing ones. This situation will naturally create a “computational ecosystem” of devices, besides reducing the need for users to manually configure each device.

However, this growth of users inside the mesh creates a totally stochastic environment, which generates a great loss of quality in the routing process algorithms (in the maintenance of the data flow). This problem is a main factor for the resistance against large scale adoption of the mesh model, not allowing users to take advantage of its capabilities.

In the constant search for more flexible mechanisms, capable of adapting to the constant changes in topology (that are natural in a mesh network), we propose the use of Artificial Neural Networks (ANNs) to improve the mesh model packet routing process. ANNs have the ability for non-linear and parallel processing, which gives the capacity to process and classify data in a cognitive way with high

efficiency.

To make them efficient, it is necessary to train the ANNs with a base set of scenarios and its expected results. This learning process is called "supervised training" and has as objective to prepare the internal structures with values (weights) of the ANNs to respond with the expected values whenever they find a known pattern [**?**].

ANNs are built like parallel distributed systems, composed by small simple processing units: the artificial neurons. These neurons calculate mathematical functions, which are usually non-linear functions, and are organized in layers. Artificial neurons are interconnected by a large number of connections, which are usually unidirectional (see Figure **??**). There are still different models that use backpropagation connections, for example [**?**].

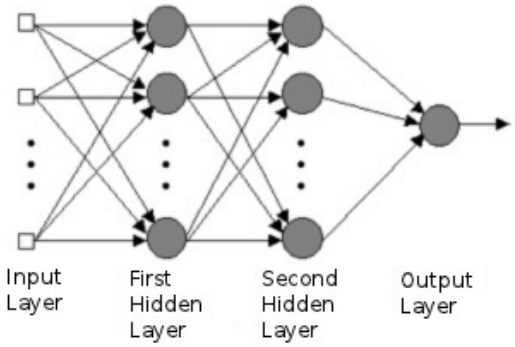

Figure 1. Classic visualization of the structure of a neuron in an Artificial Neural Network [**?**].

VI. Mesh Networks Routing Problems

Mesh network ecosystem brings up a problem related to the dataflow among the different devices and the Internet, impacting directly on connection quality. Considering that a device requires a constant data stream, the network has to be able to maintain the flow of data, always selecting the shortest cost path [**?**]. Given this problem, maintain efficiency in the network becomes a vital issue. In this context, the development of new technologies is expected to allow the mesh model to have the necessary flexibility, in order to hold large scale applications.

The self-organization and self-repairing capabilities are great characteristics of the mesh model. They make the network to be flexible, and almost fail-safe. However, these characteristics impose a great computational cost for the decision process in the retransmission and routing of packets. These problems represent how stochastic is the network topology: with the constant changes in the topology (the entry and exit of client nodes and possible failures in mesh routers) the network unfit for algorithms that converge to static models, that do not adapt to change.

Trivial algorithms are not capable of adjusting themselves in time to support the changes in the network [**?**] [**?**] [**?**]. The authors of [**?**] present the traditional AODV (Ad Hoc On Demand Distance Vector) and its inability to supply a quality routing for the model. As an enhancement, these same authors demonstrate a version of the AODV-GW (which is specialized for mesh networks) as an alternative, and they compare it to other protocols, one of those proposed by themselves: FBR, the Field Beacon Routing.

FBR proposes a proactive alternative to routing, making the client nodes to inspect their environment searching for their neighbours, thus "seeing" the whole ecosystem around and using it in its favour in the decision process of packet routing.

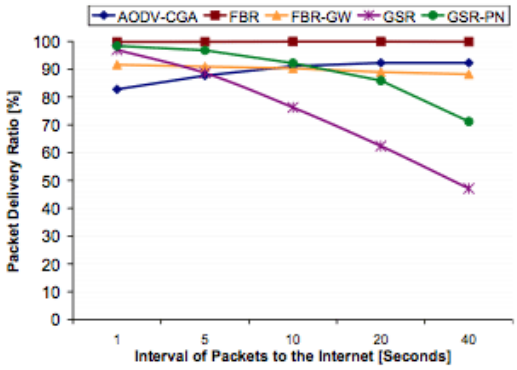

Figure 2. Average packet delivery of packets in a scenario containing pedestrians connected to the ecosystem. [**?**].

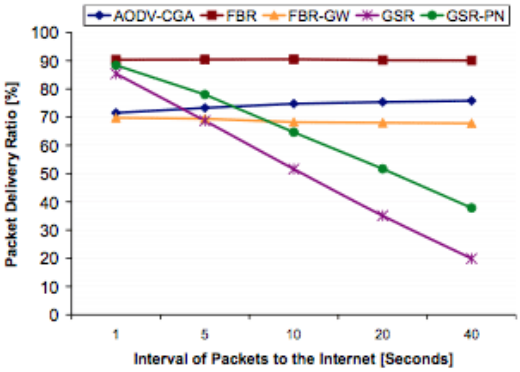

Figure 3. Average packet delivery in a scenario consisting of cars connected to the ecosystem. [**?**].

After analysing the graphics (Figures **??**, **??** and **??**) presented by the author, in [**?**], it can be concluded that, the bigger the mobility of client nodes the higher the interruptions in packet delivery. Thus, package delivery becomes compromised and the throughput drops. Consequently, the end user of the ecosystem will see a significant drop in network performance.

The analysis of Figure **??** shows that scenarios where the client nodes are static, package routing tends to work

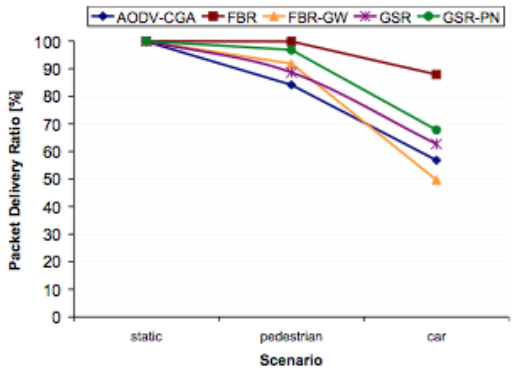

Figure 4. Average packet delivery in a scenario containing static client nodes. [**?**].

correctly, even with classic protocols. However, it is also possible to notice that, in scenarios where client nodes are mobile and have random behaviour, package delivery becomes effectively lower.

VII. Cognitive Routing with Neural Networks

In our work, neural networks are used in the routing protocol of the mesh network. More specifically in the packet retransmission process, like in the works presented by other authors [**?**], which focus their efforts in the search for better paths using traffic analysis and prediction. This work, on the other hand, focuses in the process of decision during retransmission of a packet. The protocol will use the neural network to decide what is the best path for a packet to follow, routing it to less loaded paths, in order to avoid path overloading.

The main idea of cognitive routing is to provide the flexibility that devices participating in the mesh need to use efficiently the available resources. The use of Artificial Neural Networks in the package routing becomes significant in the performance of the mesh because it provides flexibility and agility in the distribution of information through the topology. The employment of traffic engineering in the information flow over the network is extremely important, since the network topology stochasticity imposes great challenges in connection quality. With that in mind, congnitivity appears to allow the network to be flexible enough, foreseeing its future stages, and maintaining quality at an acceptable level [**?**][**?**][**?**].

The Hybrid Wireless Mesh Protocol (HWMP) works proactive and reactive, combining on demand the path discovery and the routing table building. Because of that the strategy, the protocol allows the client nodes to communicate in a peer-to-peer way. This protocol is used for client nodes that are in environments with great and constant changes. The basic routing metric used in this protocol shall be the resource consumption of the transmission channel for a particular link [**?**].

The cognitive routing process in our work is based on the HWMP protocol. This protocol has been adjusted for partially support the use of the ANN during the broadcast of a package: before the next hop of the package, the ANN is performed in order to evaluate and instantiate the target node of the jump. This evaluation determines whether the package will make the jump to the same node or whether the protocol should choose another node to jump.

A. ANN Structure

The ANN used was a Hopfield network with two inputs, one output and three neurons in the hidden layer. During the simulation process, this default setting of the ANN proved to be viable and stable, reaching the threshold that allowed for speed and stability in changing scenarios, *i.e.* mobile client nodes. The ANN uses the sigmoid function to operate using the backpropagation to adjust the network weights. The parameters used to run the network as follows:

- Number of Inputs: two;
- Number of outputs: one;
- Number of Hidden Layers: tree;
- Number of Layers: tree;
- Desired Error: 0.0001;
- Max Epochs: 1000;

The parameters used were stable during the construction of the training sets as well as at the time that the network runs for evaluating the data of the destination nodes. Although the construction naive the results obtained by stable ANN are good and worth of further investigation.

B. Mesh Routing Metrics

The metrics currently used in the assessment of the target mesh are the Delay and the Packet Forwarding PDF (Packet Delivery Fraction). These metrics can determine how much the client node is overloaded, both forwarding packets or other processes taking context and occupying the processing time. With these starting geometries, initial data has been compiled (Table **??**) with the output patterns.

PDF (Packet Delivery Fraction)	Avg. Delay
91.20%	0.015578
85.57%	0.0179017
98.23%	0.00939786
99.55%	0.00592165
91.74%	0.015798
61.54%	0.0734161
65.60%	0.18154
67.48%	0.0329183
79.58%	0.0166734
70.57%	0.0323162

Table I
Sample database collected for the training scenarios of the network.

The threshold for determining whether a node is overloaded was set on seventy percent of PDF, where it can be

seen that there is a low package delivery index, at the same time can be seen that it has high delay deliveries. Such data will inform the client node that it is not able to forward the packets.

C. ANN Training and Execution

Neural network training is done in a disconnected state from the simulation. So the scenarios are being generated prior to the reviews from routing simulation patterns. Such scenarios indicate the cases in which the network should have its positive output indicating that the client node package jump target is within the parameters considered nominal maintenance of data flow.

The implementation of ANN occurs within the process of packet relay, each retransmission the destination of the packet is evaluated by the ANN, which determines (based training scenarios) if the node is overloaded or not.

Such information about the client neighbouring nodes is obtained in the MACAddress that is changed to also transmit their data in PDF and Delay. In order to take the proactive and reactive nature of the network, the neighbours in each update the data is also updated. So the ANN is able to evaluate the destination node and determine whether or not that packet should be used in the forwarded route.

VIII. Simulation Results

To generate and evaluate the results we simulate a mesh network with some characteristics using the NS-3, a simulator based on discrete events for the community of researchers and educators. This simulator aims at providing an environment for the study of large-scale networks (for example, the Internet and huge structures), with controlled environments. NS-3 has the following features for running the simulations [**?**].

- Is written in C++ with an optional interface for Python;
- Provides integration with software that follow their patterns of arrival and departure information, enabling other tools to be used in conjunction with the simulator;
- The research, in general, is validated in two ways: through simulations and real implementations. The NS-3 is an organization that resembles the physical implementations, thus allowing to test more accurately the simulations;
- Contains codes that allow integration with the Linux kernel modules allowing the simulator operates in conjunction with real data. As their model is relatively close to the actual implementation, it supports device drivers as well as IP and Sockets APIs (Linux);
- Eases test simulating in real environments, as well as real time experiments;
- Contains a rich documentation on the operation of their strata and classes, stating what each part of the code available in the simulator does;
- Its new reporting model is compatible with external tools, such as Wireshark.

An example scenario were designed to be configurable, so it can be changed to generate alternative results for comparative basis. The base scenario parameters are:

- *x-size*: Width of the grid, default 6;
- *y-size*: Height of grid, default 6;
- *step*: Space on the grid edge, default 100m;
- *step*: Time from randomization to the start of transmission of packets (0.1 second);
- *time*: Simulation time, default 100 seconds;
- *packet-interval*: Transmission interval of the packets, default 0.001 second;
- *packet-size*: Packet Size, default 1024bytes;
- *interfaces*: Number of transmission interfaces (antennas) that possess a client node, default 1;
- *channels*: Number of transmission channels for a client node, default 1;
- *pcap*: Enable PCAP to generate trace files, default true;
- *stack*: Model of the network stack used, default *ns3::Dot11sStack*;
- *root*: MAC address of the base node of the grid.

The graphs presented serve as a basis for evaluating the use of neural networks in the decision process of the packet routing within the mesh. The graph in Figure **??** shows the average delivery of packets from a data stream, also the average rate of transmission. The relay presented shows which packages have been retransmitted before arriving at the destination.

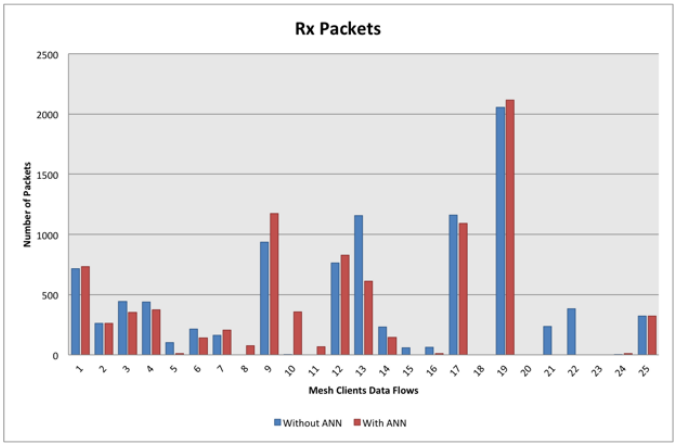

Figure 5. Average delay and packet delivery rate Transmission and Retransmission.

The PDF shown by the ANN varies in different simulations, as seen (Figure **??**) that each data stream in the PDF remains close to the standard network and at times managed to maintain a higher rate of packet delivery than the trivial algorithm.

It is possible to notice that in Figure **??** the ANN represents (in many cases) a much lower rate of packet loss,

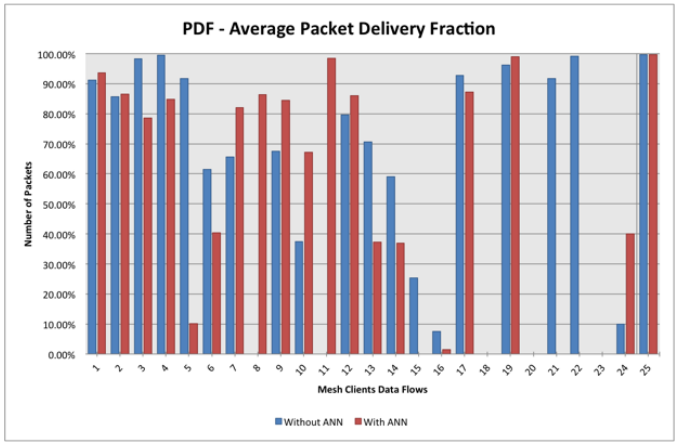

Figure 6. Average packet delivery fraction.

thus demonstrating that the decision to go for alternative paths may be feasible in cases of overload.

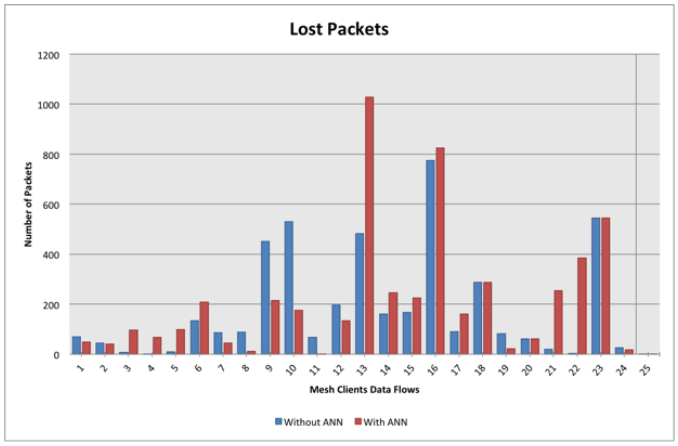

Figure 7. Lost Packets.

Figure **??** compares three simulations using ANN and a simulation of the standard protocol, as well as (Figures **??** **??**) it is possible to note significant changes to the PDF and throughput. Seeing that the rate of delay of the simulations being evaluated as a whole tends to be lower, but because that packages spend more time jumping around so the throughput tends to be lower than the standard without using ANNs.

IX. Related Work

Some authors have worked with models of ANNs to solve problems related to the mesh network and its basic structure. A major problem in this model is the optimal routing of packets, directly influenced by the chaotic structure that form mesh networks. One approach is found solving the optimal path routing in artificial neural networks [**?**].

The authors of [**?**] apply neural networks in order to solve the problem of finding the shortest path between

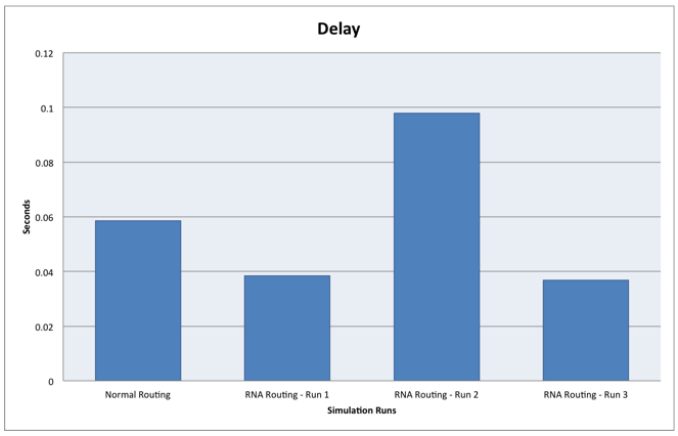

Figure 8. Delivery delay.

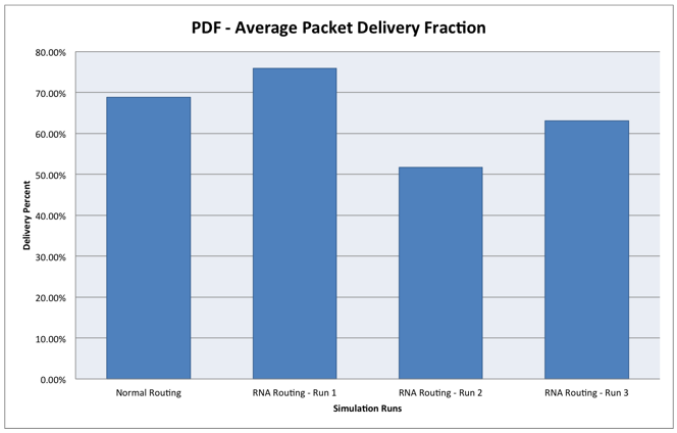

Figure 9. Percentage of packets delivered.

clients, start and end. The same study identified a real application for which ANN would be necessary to route the implementation of the hardware level, because the time required for adjustment of the network weights, and the convergence time in the software level is too expensive.

Another paper that applies ANNs within the model of mesh networks is the work described in [**?**], which points out the challenges in the use of ANNs to achieve high levels of quality in mesh networks. In the paper, the authors present the modified ANNs to support the ability to measure the flow of information within the client-nodes, in order to create a mechanism for routing flexible enough to support the stochasticity of the network topology. This application is based on the ability of ANN from a mathematical analysis of a series to predict whether a future event on the client node will have its ability to relay packets compromised.

The work described in [**?**] is a complete study of the behavior of users of infrastructure mesh networks that is being raised the issue of stochasticity of traffic from the random behavior of users. Given this premise, the paper proposes a fully distributed algorithm for routing, able to be

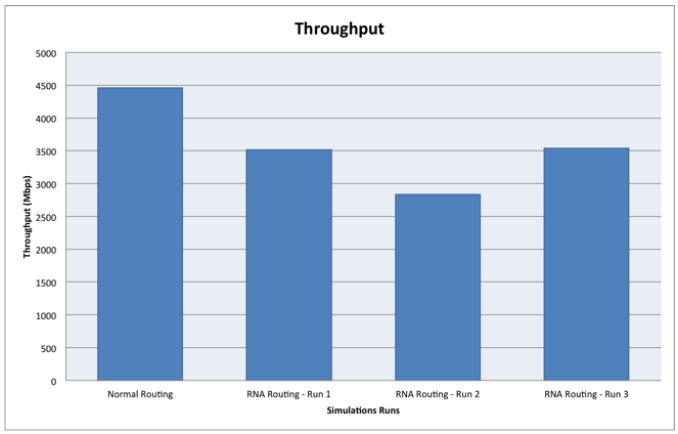

Figure 10. Average Packet lost.

flexible to accommodate changes in network infrastructure.

However, in this paper the idea is to use the ANNs fully operational in routing decision, as in [**?**] and [**?**] to use this cognitive mechanism to decide if the current path where the package is valid and if the next hop will not be compromised. If the path in the near future is not practicable, then forward the packet to another client node being more efficient compared to others available.

X. Conclusion

A point to get is the feasibility to simulate an ANN in each node; now the training is done so disconnected from the simulation process. Thus the training scenarios of the network were created prior to the simulation and only used the process of evaluating the neural network. As future implementation using a dynamically driven recurrent network is used. This model is able to function without the training cases, self-adjusting during the duration of the active node and thereby enabling each node to fit the environment where they are.

The simulations showed consistent results but require further study because in some cases there was a decline in the quality of the network. Such anomalies occur during the simulations because the current algorithm implemented only evaluates the next node that the package will be sent. This ANN does not assess the status of client nodes where being forward. However, in most cases even a client node being evaluated in the path data show promising results, causing the network circumvent stress conditions avoiding paths that become unstable. These good results are promising and require further study, leaving the network with higher connectivity and higher quality.