"Mircea cel Batran" Naval Academy Scientific Bulletin, Volume XVIII – 2015 – Issue 1
Published by "Mircea cel Batran" Naval Academy Press, Constanta, Romania // The journal is indexed in: PROQUEST SciTech Journals, PROQUEST Engineering Journals, PROQUEST Illustrata: Technology, PROQUEST Technology Journals, PROQUEST Military Collection PROQUEST Advanced Technologies & Aerospace

EXPLORING THE POSSIBILITIES OF A SELF-REGULATING SDN CONTROLLER

Rares MANIU[1]
Laurentiu Alexandru DUMITRU[2]
[1] Eng., Ph.D. (c), Military Technical Academy, 39-49 George Cosbuc Bvd., Bucharest, Romania, rares.maniu@yahoo.com
[2] Eng., Ph.D. (c), Military Technical Academy, 39-49 George Cosbuc Bvd., Bucharest, Romania dlaur@nipne.ro

Abstract: *As the number of networked devices increases, traditional routing algorithms tend to be non-optimal for mesh topologies. With the power of controlling the data flows in Software Defined Networks, a controller can implement a dynamic communication path for each flow. If, in the same context, the controller would also implement a history evaluation algorithm combined with a genetic search method, it could achieve a dynamic resource allocation that tends to an optimal solution. This paper proposes the implementation of such a system on top of an Open Flow controller.*

Introduction

In the context of increasing data speeds being delivered to low-powered processing devices and the emerging of the highly discussed Internet of Things, computer networks will also expand, and evolve, as the need for more bandwidth, higher speeds and lower latencies become more obvious. These networks will contain more nodes in which the delimitation between client node, edge router or core router will not be so clear as is it now. Classical computer networks use routers with algorithms that have only one major dynamic factor: the distance. Typically, the bandwidth usage, link quality, and other subjective factors do not play a role into the routing process. Furthermore, the Internet, in its current form, runs manly with IP. Thus, if someone wants to run a particular protocol, it must encapsulate it over IP. Route optimizations are point-to-point virtual circuits are now in the responsibility of the network administrators. Since traffic patterns can change over time and networks are being reconfigured automatically (e.g. cloud networks), the help of an automated solution that can optimally configure a network and, at the same time, provide on-demand private links can be seen in the form of Software Defined Networks (SDN) governed by an intelligent, self-aware controller.

Software defined networks allow decoupling the control plane from the date plane. Network switches can be reconfigured in real-time to change their CAMs rules according to the indication of a controller to which the switch is connected. There is, at least one, SDN controller that has a global view of the network and controls the participating switches. Since all the switches are configurable, there is no need of distance-based, hop-to-hop routing algorithms. A flow can be imposed by the controller, from entry to exit, according to a set of rules. This permits multi-link routing based on different criteria. Network switches that participate in the routing process must support the OpenFlow protocol. All of them connect to the SDN controller. It is the controller's responsibility to configure the flow tables on each switch.

The proposed solution will provide optimal multi-link routing in a SDN environment based on the SAEN [1] architecture that uses multiple classifiers. These classifiers are particular to each type of traffic and are a core element for providing an optimal real-time traffic distribution. The search for solutions is done using genetic algorithms, statistic prediction and real-time analysis. By using these methods, the network will provide natural routing paths that approach satisfy both lower-cost routing and human-like, heuristic decisions. A routing method that keeps track of history and it's able to self-adjust in order to respond to the environment's needs can be extremely useful when the node numbers increase and manual configuration time also increases.

Related work

Software defined networks are relatively new and have not yet received widespread implementations. However, several major vendors have already started to commercialize OpenFlow-capable switches that can participate in SDNs. OpenFlow [2] was originally proposed as an alternative for the development of experimental protocols on university campuses, where it is possible to test new algorithms without disrupt or interfere with the normal operation of traffic of other users. Vendor alternatives exists, but, these come at a cost and are closed-source. OpenFlow is an open standard that can be implemented on many hardware modules without any major difficulties. Its major advantage is that is leverages existing functions such add/delete entries in the RIB and uses standard packet actions such as accept or drop. Although still evolving, there are a few SDN controllers that have proven to be reliable and can provide both performance and configurability. Khondoker, Rahamatullah, et al. have compared there controllers in their study [4]. An overview of the SDN evolution can be found in [3]. The paper shows the evolution of this technology, why it is so important and where it may be applicable.

The intention of this study is to explore the possibilities of having a self-regulating SDN controller, using genetic algorithms and history extrapolation for dynamically reconfiguration of the network. Such an approach, not in a SDN context, is proposed by the Self Adaptive Evolutionary Network (SAEN) architecture [1]. The paper discusses the already mentioned limitations of classical routing algorithms and proposes history recording for prediction purposes combines with evolutionary algorithms that compute the routing paths, instead of shortest path methods.

CODA [6] proposes a hop by hop back pressure-based, with periodical sampling of the link and buffer loads, method for congestion avoidance and control in sensor networks. Genetic algorithms are also used by Arnab Raha et. al [5], but, as with traditional algorithms, multi-link is not a conditional factor in the route-decision process.

SDN Architecture

The main idea of the Software Defined Networking is that is possible to apply a high level of abstractization in network control. It allows administrators to manage services, independent from the systems that make decisions about routing, from systems that forward traffic to destination. It means that in SDN exist two planes: control plane and data plane.

The SDN architecture is:

Directly programmable: Network control is directly programmable because it is decoupled from forwarding functions.

Agile: Abstracting control from forwarding lets administrators dynamically adjust network-wide traffic flow to meet changing needs.

Centrally managed. Network intelligence is (logically) centralized in software-based SDN controllers that maintain a global view of the network, which appears to applications and policy engines as a single, logical switch.

Programmatically configured. SDN lets network managers configure, manage, secure, and optimize network resources very quickly via dynamic, automated SDN programs, which they can write themselves because the programs do not depend on proprietary software.

Open standards-based and vendor-neutral. When implemented through open standards, SDN simplifies network design and operation because instructions are provided by SDN controllers instead of multiple, vendor-specific devices and protocols.

At the heart of the SDN lies the SDN controller that communicates with the switches using the OpenFlow protocol. The protocol itself stated from the production version 1.0 and has now reached version 1.4. Each version has added new functionality and improved the existing one. The latest version provides support for applying rule bundles in an atomic-like operation. OpenFlow messages regard the control plane of the switch, which, in turn influences the data path. A switch can connect to multiple SDN controllers in order to achieve fault tolerance or, higher throughput. When there is not default action and the incoming packet does not match any rules in the forwarding base, the controller is responsible for generating an action for that packet.

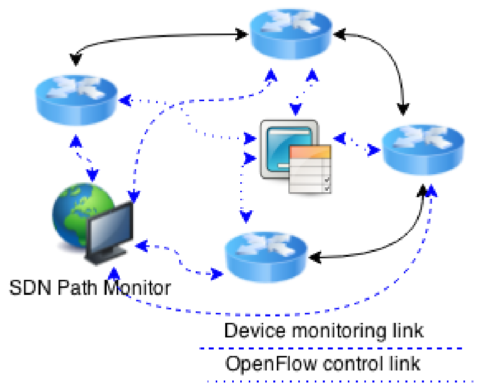

Figure 1

OpenFlow supports two methods of flow insertion: proactive and reactive. Reactive flow insertion occurs when a packet reaches an OpenFlow switch without a matching flow. The packet is sent to the controller, which evaluates it, adds the appropriate flows, and lets the switch continue its forwarding. Alternatively, flows can be inserted proactively by the controller in switches before packets arrive. In the case of proactive flow insertion, the arriving packet will never be sent to the controller for evaluation since it matches the proactively-inserted flow.As normal FIB entries, these contain selectors, counters and actions. Selectors and actions decide what to do with the packet while counters can be used for statistical information. The counters are particularly important due to the fact that SAEN is history-based. As the Route Monitor in SAEN can be compared to a SDN controller, blending both of them will create the desired solution – a network node capable of having a top level view of the network and, using this information, capable of reconfiguring the devices in order to implement a optimal paths based on the multi criteria evaluation that is has computed. This node, the SPM (SDN Path Monitor) will implement the history-based route search algorithm described in SAEN.
As discussed in [4], different controllers offer different programming approaches for interacting with the network. The SPM can be implemented as a standalone server or, it can blend in the existing controller, given that this provides the necessary API functions. In the first case, the SPM will implement a minimal set of functions from the OpenFlow protocol that will allow it to query network devices for their counters and events. The full list of SDN-capable devices would be retrieved from the SDN controller, along with their interconnections. This information is critical for the evolutionary search. If the SDN controller can receive all the information requested by the SPM, then the SPM could be implemented as a module. The other approach would be to fully integrate the OpenFlow protocol into the existing SAEN RouteMonitor. However this will create a redundant element that doubles the SDN controller. No particular advantage can come from this approach.
When implementing the SPM as a distinct software component, it is essential to easily communicate with the controller. Five major controllers were compared.
The NOX/POX pair are one of the oldest. POX (python-based) is a continuation of NOX (C-based). In terms of speed, NOX performs very good.
Ryu is a component-based controller supported by NTT. It has a particular feature that enables it to use many programming languages for constructing new model from existing components or from new ones.
A controller based on Ruby which is centered around building easy code is Trema.
OpenDaylight is one major initiative for powering SDN networks. The goal of this project is to provide a robust code platform that covers all the major components of any SDN architecture. It is open-sourced and java based.
The FloodLight controller, a project which is derived from Beacon, is self-contained and can run out-of-the-box on any operating system. It is also java based but also exposes a REST API that makes it extremely useful for the testing of the proposed SPM daemon. The SPM makes use of the Static Flow Pusher API. The Static Flow Pusher is a Floodlight module, exposed via a REST API, that allows a user to manually insert flows into an OpenFlow network.

An evolutionary approach

Even if SDN is a new concept in network management, and is developed for large and dynamic networks it is not enough when a network presents an exponential growth, following an anarchic scalability.
The idea of "Internet of Things" that offer connectivity between devices, systems, services, covering a large number of protocols, applications and domains of activity and the concept of machine-to-machine communication increase the dynamicity. And the multitude of devices with communication and computing facilities sustain the development of high networks. In this context, Software Defined Networking will need to work for all users, not just for very specific IT systems like clouds or data-centers.
Software Defined Networking made the transition between a global-fixed network path and and a dynamically modifiable one. SDN means a combination between a virtualization level and a network operation system. The operator works with an abstract view of network. The link between this view and the configuration of network devices is being generated by software. The virtualization level translates the abstract view in a global view of network. The network operation system translates the global view in a configuration for network elements.
The global network view provides a real-time state of the entire infrastructure. With this centralized control, SDN is capable to build a global view of topology for all network elements connected and this produces a simplified network management. The central database with information about the structure of network is important in the implementation of optimized routing engine. Actually, the routing calculating process that is based on Dijkstra's algorithm, works well for classical network management, but due to the fact that it runs in time of $O(V^2)$, where V is the number of nodes, it is time consuming for big networks and can't offer solutions in timely manner for big, evolving networks.
Because of the high-dynamicity of modern networks, a SDN will need to present very quickly a solution for new configuration, when changes occur. It must generate proper routes and, based on this, the configuration for network elements, not necessary the best configuration, but one that can respect all demands. That means it is a multi-criteria optimization. Genetic algorithms represent one of techniques well adequate to solve such types of problems, based on natural selection. It can offer solutions for high-complexity problems.
The main feature of a genetic algorithm is that it can perform a global search in solution space, maintaining a number of possible solutions from generation to generation. Because of the interdependence between objectives, the genetic algorithm will generate a set of solutions, called Pareto optimal solution.

Unlike other optimization techniques, genetic algorithms can analyze in parallel a high number of solutions, they don't need apriori knowledge, just only objective function value, operates

with an encoded set of parameters and uses stochastic transition rules. As advantages, a genetic algorithm is easy to implement and reconfigure, the solution is closed to the global optimum, offers good results for large seeking space, with a big number of variables. Can be used for non-linear, multi-objective optimization, generating several solutions that meet the requested criteria. Require just objective function, with little knowledge about the problem and is very robust due the large number of solutions processed in parallel at each stage.
As disadvantages, the process is a high resource consumer due to the many iterations needed and to the parameters that are dependent on the implementation of genetic operators, number of generation, type of selection and other criteria.
A genetic algorithm can be described as:

```
procedure GENETIC-ALGORITHM

Generate initial population P0;
Evaluate population P0;

Generation counter g=0;
While fitness function not satisfied repeat
        Select chromosomes from Pg to copy into Pg+1;
        Crossover chromosomes from Pg and put into Pg+1;
        Mutate chromosomes from Pg and put into Pg+1;
        Evaluate some elements of Pg and put into Pg+1;
        Generation counter g = g+1;
End while
End procedure
```

The genetic algorithm starts with initial population generation, each element in the population is called as chromosome. Each chromosome is a solution by itself. It is generated randomly or heuristic, from all possible routes in the network, starting from initial note. The initial population must be distributed in all search space. A small population produces a local optimum and a large population needs more computational resources. Each chromosome is evaluated for fitness. This value indicates the quality of this chromosome in population and is generated based on fitness function. New offspring are generated from the chromosomes; using operators like selection, crossover or mutation add the chromosomes with the best values for fitness function are moved to the next generation. The process is repeated until the chromosomes have the best solution for problem.

Implementation

A genetic algorithm was implemented to test the possibility to generate a set of routing solution in a timely manner for a network with a big number of nodes and links.
The chromosome defines the route and its genes represent nodes. A chromosome encodes the problem by presenting the node IDs between source to destination. First gene is the source and last is the destination. The chromosome length is the number of nodes in network. Because the length of route is variable, the remaining locus in chromosome is completed with value 0.
The initial population affects the performance of genetic algorithms. Its size and structure in important because it must be uniform distributed in the entire search space to obtain a global minimum not a local minimum of fitness function. The number of elements in population size is important because the computational time of algorithm is proportionally with this number. The initial population can be generated heuristic or randomly. Here, we choose to generate it using random initialization to provide a adequate distribution of chromosome in search space.
Fitness function measures the quality of a chromosome in population. It is an important element of a genetic algorithm because it introduces a criterion for selection between chromosomes. In this algorithm, the fitness function is a sum of link costs in a route, being like this:

$$fitness_i = \sum_{j=0}^{l_i-1} C_{g_i(j),g_i(j+1)}$$

$Fitness_i$ is the fitness value for chromosome nr. I, l_i is the length of the chromosome (for this implementation is the number of nodes in network) and C is the link cost between 2 adjacent nodes in route.
All the values about existence a link between nodes and the values for cost of links are provided from central database of the SDN infrastructure, the SPM.
The transition between populations is made using the selection procedure. It improves the average quality of population and ensures the promotion of well-adjusted individuals to the next generation. In genetic search wo types of selections exist: proportional and ordinal based. Proportional selection means that the chromosome is selected based on its fitness relative to the fitness of other chromosomes in population. Ordinal-based selection means that the chromosome is selected based on its rank (direct proportionally with fitness value) within population. In this implementation, two chromosomes are evaluated according fitness value and the one that is fitter is selected.
Crossover operators produce the exchange information between two chromosomes, being the most important operator in a genetic algorithm. It exists two types of crossover: with one-cut-point and multi-cut point. For one-cut point, from every chromosome is chosen a locus. They are cut in two parts and every new chromosome will contain one part from every parent. For crossover in multi-cut points, more than one cutting point are chosen randomly and the resulting chromosomes will contain information from each parent. For this application, the crossover in one-cut point is used. After the crossover, a function will eliminate loops from routes and check if the new chromosomes are valid solutions. Just valid-routes are selected after crossover.
In a genetic algorithm, mutation ensures that all of the solution space will be taken into consideration to be searched and a global solution will be generated by algorithm, not a local solution. This operator generates random changes in population and will replace the lost genes in selection process. The mutation ratio is important. A low value for mutation value will increase the probability to convergence in a local solution. A high value for mutation probability will increase the time to convergence and, because the difference between parents generation and offspring generation will be too high, the algorithm will not be able to learn from search history. Mutation operator in this implementation will randomly replace (according to the value from mutation probability) a number of chromosomes from offspring population.
The termination criterion is in fact the convergence of algorithm. Here, a minimum value for fitness value of next generation (the sum of fitness function from all chromosomes in next generation), combined with stall generation and with small changes in population fitness means that the algorithm found a set of solutions.

Experimental results

The implemented genetic algorithm use a one-point crossover operator, the selection function based on fitness value and a percent of 5% as mutation probability. The parent generation, offspring generation, and next generation have the same number of chromosomes (50 chromosomes per generation). The termination criterion is a combination between minimum sum of fitness functions for all chromosomes in next generation, the small changes in population fitness and the stall generation. The initial generation was randomly generated. The network is randomly generated; it is a medium-connected network and has 2000 nodes and 15000 links. The cost per link is randomly generated, being between 1 and 10.
After 10 runs of the algorithm, using 10 different network configurations and 10 different initial generations, these results about convergence, fitness and solutions for routing problem were obtained:

Running number	Number of iterations until convergence	Fitness for solution
1	15	5
2	13	6
3	13	4
4	11	3
5	6	1
6	16	6
7	11	5
8	16	3
9	18	3
10	13	5

The running of a genetic algorithm on a network with high number of nodes and links, as it can be seen in experimental results, will generate in a relative low number of iterations a set of solutions for routing problem. The result is not necessary the minimum cost route, but a route with a good cost, generated in timely manner. And, because of construction, these kinds of algorithms are easy to fit on dynamic network.

As future works, the algorithm can be implemented using dedicated hardware devices, it can be implemented on parallel machines or using a combination between genetic algorithm and classic minimum route-search algorithms.

The generation of initial population can be modified, injecting a number of some well-fitted chromosomes generated in anterior runs of algorithms (the main idea is that, the evolution of a big network is relatively constant and don’t present high discontinuity points).

Conclusions

The self-adaptive property of a network will be a mandatory requirement for the future. With the opened technical possibilities that SDNs are offering and the dynamical reconfiguration strategies controlled by a intelligent single point orchestrator, networks will provide optimal services with minimal intervention. As Software Defined Networks are evolving, their controllers need to become more capable and autonomous. By implementing solutions such as the Self Adaptive Evolutionary Network architecture into the SDN controllers, the network can truly become more effective. On a wider scale, such optimizations have larger impacts such as lower energy requirement, fail-save paths and automatic virtual networks.

Bibliography

[1] Maniu, Rares, L.A. Dumitru. "Self-adaptive networks with history extrapolation, evolutionary selection and realtime response." Optimization of Electrical and Electronic Equipment (OPTIM), 2014 International Conference on. IEEE, 2014.
[2] N. McKeown, T. Anderson, H. Balakrishnan et al., “OpenFlow: enabling innovation in campus networks,” ACM SIGCOMM Computer Communication Review, vol. 38, pp. 69–74, 2008.
[3] Valdivieso Caraguay, Ángel Leonardo, et al. "SDN: Evolution and Opportunities in the Development IoT Applications." *International Journal of Distributed Sensor Networks* 2014 (2014).
[4] Khondoker, Rahamatullah, et al. "Feature-based comparison and selection of Software Defined Networking (SDN) controllers." Computer Applications and Information Systems (WCCAIS), 2014 World Congress on. IEEE, 2014.
[5] Arnab Raha, Mrinal Kanti Naskar, Et. Al. “A Genetic Algorithm Inspired Load Balancing Protocol for Congestion Control in Wireless Sensor Networks using Trust Based Routing Framework (GACCTR)”, I. J. Computer Network and Information Security, 2013, 9, 9-20
[6] C.Y.Wan, S.B.Eisenman and A.T. Campbell, “CODA: Congestion Detection and Avoidance in Sensor Networks”, SenSys’ 03, Los Angeles, USA, pp. 266-279, ACM, Nov 2003.
[7] Open Networking Foundation SDN Architecture Overview
[8] Fernandez, Marcial P. "Comparing openflow controller paradigms scalability: Reactive and proactive." Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on. IEEE, 2013.