مجلة جامعة دمشق للعلوم الهندسية
المجلد 39- العدد 3 (2023) 41-56

Damascus university Journal for engineering sciences
Vol 39 No.3 (2023) : 41-56

# تصميم آلية استدلال هجينة لتحقيق التوازن الأمثل للحمل في الشبكات المعرفة برمجياً

**نزيه احمد حرفوش[1] د.م محمد مازن محايري[2] د.م رؤوف حمدان[3]**

[1]طالب دكتوراه في هندسة الحواسيب وشبكاتها – كلية الهمك – جامعة دمشق.
[2]أستاذ مساعد في قسم الحواسيب والأتمتة – كلية الهمك – جامعة دمشق.
[3]مدرس في قسم الحواسيب والأتمتة – كلية الهمك – جامعة دمشق.

## الملخص

تعتمد معمارية الشبكات المعرفة برمجيا (Software Defined Networks SDN) على فصل طبقة التحكم (control plane) عن طبقة البيانات (data plane) وتساعد عملية الفصل هذه في جعل الشبكة قابلة للبرمجة بشكل مباشر وقابلة للتوسيع كما وتسمح بإضافة خدمات للشبكة بشكل متكامل مثل موازن الحمل (load balancer) والجدار الناري (firewall) وغيرها الكثير من الخدمات. نقترح في هذه الورقة آلية استدلال هجينة لتحسين أداء المتحكم (controller) في شبكات (SDN) عن طريق استخدام موازن حمل ديناميكي بالاعتماد على كلٍ من خوارزمية البحث بالعمق أولاً (لإيجاد المسارات)، وأمثلة مستعمرة النمل (لاختيار المسار الأمثل للحزم)، وتمَّ تنفيذ وتطبيق موازن الحمل المقترح بطريقة ديناميكية داخل طبقة التحكم وذلك للاستفادة من الرؤية الشاملة لطوبولوجيا الشبكة والتي يوفرها المتحكم في الزمن الحقيقي مما يضمن موازنة الحمل في الزمن الحقيقي وتجنب حالات عنق الزجاجة واستقرار الشبكة في أوقات الذروة في ظروف تغيير الطوبولوجيا أو عند توسيع الشبكة، كما نقترح في سياق العمل آلية لمعالجة مشكلة ضبط بروتوكول حل العنوان (Address Resolution Protocol (ARP)) ضمن متحكم الشبكة لضمان عمل الموازن المقترح بشكل ديناميكي واكتشاف تغيرات الطوبولوجيا والوصلات واتخاذ قرار التوجيه (Routing) الأقل كلفة.

**الكلمات المفتاحية:** الشبكات المعرفة برمجيا، المتحكم، موازن الحمل، بروتوكول حل العنوان، خوارزميات الأمثلة، خوارزمية مستعمرة النمل، خوارزمية البحث بالعمق أولاً.

تاريخ الإيداع:2022/3/21
تاريخ القبول:2022/5/18

# Designing A Hybrid Heuristic Mechanism To Achieve Optimal Load Balancing In Software Defined Networks

**Nazeeh Harfoush[1]**
**Dr. Mohammad Mazen Mahayri[2] Dr. Raouf Hamdan[3]**

[1]PhD student in Department of Computer and Automation Engineering, Faculty of Mechanical and Electrical Engineering, Damascus University.
[2]Professor in Department of Computer and Automation Engineering, Faculty of Mechanical and Electrical Engineering, Damascus University.
[3]Professor in Department of Computer and Automation Engineering, Faculty of Mechanical and Electrical Engineering, Damascus University.

## Abstract

The architecture of Software Defined Networks (SDN) is based on separating the control plane from the data plane, this separation helps making the network to be directly programmable and scalable, and integrating new services such as load balancer, firewall and many other services to the network. In this paper, we propose a hybrid heuristics mechanism to improve performance of SDN controller by using a dynamic load balancer based on both Depth First Search Algorithm (to find paths) and Ant Colony Optimization (to choose the optimal path). The proposed load balancer has designed and run dynamically within the control plane in order to take advantage of comprehensive view of the topology provided by the controller, which ensures load balancing in real time, avoiding bottlenecks and network stability at peak times and under conditions of topology changes or network scaling. In addition, we suggest a new mechanism to solve the issue of handling Address Resolution Protocol (ARP) within the controller to ensure that the proposed load balancer works dynamically, discovers the topology changes, and makes the lowest cost routing decision.

**Keywords:** Software Defined Networks, Controller, Load Balancer, Address Resolution Protocol, Optimization Algorithms, Ant Colony Algorithm, Depth First Search Algorithm.

Received: 21/3/2022
Accepted: 18/5/2022

## 1. المقدمة

تَقسم معمارية (SDN) إلى ثلاث طبقات منفصلة، أولاً: طبقة التطبيقات التي تحتوي على التطبيقات الخارجية كبرامج مراقبة حركة مرور البيانات، ثانياً: طبقة التحكم الممثلة بمتحكم الشبكة (SDN controller) الذي يملك الرؤية الشاملة لطوبولوجيا الشبكة وصانع قرار التوجيه ويَطلق عليه نظام تشغيل الشبكة، ثالثاً: طبقة البيانات التي تقوم بتنفيذ قرار التوجيه من خلال عملية التوجيه أماماً (forwarding) بمعنى آخر هي الطبقة المسؤولة عن توصيل البيانات من المصدر الى الهدف.

يتخذ متحكم الشبكة قرارات التوجيه، بينما تقوم طبقة البيانات بعملية التوجيه أماماً للحزَم وفقًا للتعليمات المقدمة من المتحكم (طبقة التحكم) ومن أهم الميزات لشبكات (SDN) أنَّ طبقة التحكم تتمتع بصفة المركزية أي أنها تمتلك نظرة (رؤية) شاملة لطوبولوجيا الشبكة، بالإضافة الى ذلك يستضيف متحكم (SDN) تطبيقاً شبكياً يوفر الاتصال بين المهام والموارد المطلوبة باستخدام واجهة برمجة التطبيقات الجنوبية (Southbound API)، وتقوم التطبيقات مثل موازن الحمل وجدار الحماية بعرض تجريدي للشبكة وصنع قرار التوجيه المناسب عن طريق جمع المعلومات من المتحكم، بالتالي ليس على مشغل الشبكة إلا وضع سياسات التشغيل ليقوم متحكم الشبكة باتخاذ قرارات التوجيه والتي يتم بموجبها توجيه حركة المرور عبر الشبكة، مما يقلل بدوره من عملية تهيئة (configure) كل جهاز شبكي (مبدلات وموجهات) يدويًا. أما طبقة البيانات فتقوم بالتوجيه الأمامي للحزم (forward) أو إسقاطها (drop) بناءً على القواعد المتوفرة في جداول تدفق (flow tables) تجهيزات طبقة البيانات. يساعد بروتوكول (OpenFlow) المتحكم في الحصول على عرض لهيكلية الشبكة والتواصل مع تجهيزات طبقة البيانات لأغراض مختلفة ويمثل واجهة برمجة التطبيقات الجنوبية، كما وتُستخدم واجهات برمجة التطبيقات الشمالية (Northbound API) لإنشاء الارتباط بين تطبيقات مسؤول الشبكة والمتحكم.

تعدّ هندسة المرور (traffic engineering) في شبكات (SDN) قضية مهمة حيث انه يتم تصميم آلية مرور فعالة من خلال تحليل حركة المرور ضمن الشبكة (network traffic) بحيث يتم تحسين استخدام موارد الشبكة وأيضاً تحقيق معيار جودة خدمة الشبكة (QoS) [1]، لذلك يجب توزيع حركة المرور من المصدر إلى الوجهة بطريقة تتفادى حدوث الازدحام في الشبكة، إلا أنَّ ذلك لا يتم في شبكات (SDN) التي تقوم بنشر العديد من التطبيقات التي تسبب حركة مرور ضخمة في الشبكة مما يجعل الشبكة معقدة ومعرضة لحالات عنق الزجاجة ولتقليل عنق الزجاجة في الشبكة يجب على مسؤول الشبكة مراعاة هندسة المرور في المتحكم. لذلك كان من الضروري تحديد المسار الأمثل (optimal path) من المصدر إلى الوجهة، بحيث يصبح تحديد المسار هو الهدف الأساسي، ويتم تحديد المسار باستخدام التوجيه (routing) وهو عملية تحديد مسار من المصدر إلى الوجهة باستخدام المعلومات الموجودة في جداول التوجيه ويكون هذا إما بشكل ساكن (تحديث المسار يدوياً) أو ديناميكي (تحديث المسار آلياً)[2]. تعتبر الآليات الافتراضية (default mechanism) المُستخدمة لتوجيه البيانات (data routing) في متحكمات (SDNControllers) جيدة بشكل كافٍ لتحقيق المهام المطلوبة منها إلا أنَّها تفتقر في كثير من الأحيان الى تحقيق موازنة الحمل ضمن الشبكة، بالتالي فإنَّ الأداء الفعلي لهذه المتحكمات عند تشغيلها بالنمط الافتراضي (default mode) لا يرقى الى المستوى المطلوب مما يؤثر على الأداء العام للشبكة بشكل سلبي ويؤدي ذلك إلى خروج بعض أجزاء الشبكة عن الخدمة تماما في فترات الذروة.

في الحقيقة فإنَّ مشكلة التوجيه بهدف تحقيق موازنة الحمل والتي تم الإشارة إليها سابقاً هي مشكلة درجة تعقيدها (NP-Hard Problem) ولقد استدعى هذا النوع من المشاكل الكثير من اهتمام الباحثين في الوقت الحاضر بسبب تطبيقاتها ذات الطابع اليومي، إذ لا توجد حتى الآن خوارزمية تقدم الحل الأمثل لهذه المشكلة بسبب تعقيد زمن كثير الحدود وهذا يعني أن زمن الحل ينمو باطراد مع زيادة عدد العقد، وكل الخوارزميات المستخدمة تعطي حلولاً تقريبية لا تتجنب الوقوع في حالة عنق الزجاجة. هناك توجّه لاستخدام خوارزميات مستوحاة من علوم الأحياء وتطبيقها في هندسة المرور لأنواع مختلفة من الشبكات ومنها شبكات (SDN)، يهدف هذا العمل إلى تصميم وتنفيذ آلية مبتكرة لموازن حمل ديناميكي من خلال التهجين بين خوارزمية دمج البحث بالعمق أولاً لاستكشاف كافة المسارات الممكن استخدامها لتوجيه حركة المرور وأمثلَة مستعمرة النمل (Ant Colony Optimization ACO) لاختيار المسار الأمثل صاحب أقل كلفة.

## 2. الأبحاث السابقة

يوجد العديد من الأبحاث في مجال موازنة الحمل في شبكات SDN البعض منها يقترح طرق وآليات ساكنة لموازنة الحمل (static load balancer) حيثُ يعمل موازن الحمل ضمن طوبولوجيا محددة بالتالي هذه الآليات تفقد استقرار الشبكة في حال تم تغيير الطوبولوجيا أو تم توسيع الشبكة وإضافة تجهيزات جديدة، وفشل الآليات المطبقة يؤدي في كثير من الأحيان الى فشل الشبكة وخروجها تماما عن الخدمة. بينما يوجد أبحاث أخرى وضعت آليات ديناميكية (dynamic load balancer) تعتبر جيدة وفعالة مقارنةً مع الطرق الساكنة إلا أن هذه الآليات لم تحقق الهدف المرجو منها وذلك بسبب تنفيذها بطريقة ساكنة مفصولة عن المتحكم، بالإضافة الى كل ما سبق لم تتناول أي من هذه الدراسات عملية ضبط بروتوكول حل العنوان ( Address Resolution Protocol (ARP) نظراً لأهميته في عملية موازنة الحمل عندما يطرأ أي تغيير في طوبولوجيا الشبكة، كما ولجأ العديد من الباحثين الى تطبيق خوارزميات الأمثلة وخوارزميات تعليم الآلة والذكاء الاصطناعي لحل مشكلة عنق الزجاجة التي تصيب الوصلات أثناء توجيه حركة المرور ضمن شبكات (SDN) وذلك لما تقدمه هذه الخوارزميات من نتائج إيجابية في تحسين أداء متحكمات (SDN) بالإضافة الى سهولة تطبيق مثل هذه الخوارزميات بطريقة مرنة.

### 2-1 دراسات ذات صلة

مثلا قام دوبريفيك وزملائه في [3] باقتراح آلية توجيه باستخدام أمثلة مستعمرة النمل وتمحورت كامل الدراسة حول جودة التجربة (QoE) في الشبكات المعرّفة برمجياً حيث كان هدف المؤلفين العثور على أفضل المسارات المتاحة لخدمات الوسائط المتعددة المختلفة. تعتمد جودة تجربة المستخدم (QoE) على جودة الخدمة (QoS) والتي تعتمد بدورها على التأخير (delay) والارتعاش (jitter) وفقدان الحزم ( packet loss) تم تنفيذ واختبار الآلية على طوبولوجيا محددة على تدفقات مختلفة مثل نقل الصوت والفيديو والملفات. كما قام جوان وزميله في [4] باستخدام خوارزمية مستعمرة النمل لإيجاد حل لمشكلة انخفاض معدل النجاح في نقل المعلومات في ظل بيئة حركة المرور الديناميكية، واقترحت هذه الورقة بروتوكول توجيه قائم على خوارزمية مستعمرة النمل في شبكات المركبات المعرَّفة بالبرمجيات (Software Defined Vehicular Networks)، حيث قاموا بإنشاء برنامجاً يحدد بنية شبكة المركبات في ظل بيئة حركة المرور بعد ذلك استخدموا خوارزمية النمل لاستكشاف عُقد الجوار المثلى تدريجياً على المسار لتشكيل ارتباط كامل من العقدة المصدر إلى العقدة الوجهة وفي الوقت نفسه كانت الخوارزمية مسؤولة عن تحديث فرمون المسار، إلى جانب ذلك استخدموا الخوارزمية الشرهة (greedy) لتوصيل حزم البيانات،

وكانت النتائج جيدة من حيث معدل تغيير التوجيه وأداء الاتصال في شبكة المركبات المعرّفة برمجياً. قام ريزا وزملائه في [5] باقتراح تقنية هندسة مرور ذكية لإدارة نظام المراقبة بالفيديو عبر شبكات (SDN)، وكان هدف التقنية تقليل كل من خسارة الحزم والتأخير الكبير في تدفق الفيديو، تم استخدام خوارزمية أمثَلَة مستعمرة النمل لحل المشكلة وتحديد أقصر مسار كقيد للنموذج الرياضي الذي تم استخدامه وأظهرت المقارنات التي ذكروها بين الطريقة المقترحة والطرق السائدة مثل بروتوكول التوجيه (OSPF) فعالية الطريقة المقترحة من حيث تقليل فقدان الحزم والتأخير. كما قام المؤلفون في الورقة [6] بتطبيق آلية التعلم المعزَّز العميق **(Deep Reinforcement Learning Mechanism DROM)** للشبكات المعرفة بالبرمجيات لتحقيق عملية التوجيه الشامل والقابل للتخصيص، وتمَّ في هذه الورقة اقتراح طريقة لتبسيط تشغيل وصيانة الشبكة من خلال تنفيذ (DROM) والتي تؤدي إلى زيادة أداء الشبكة بالمقارنة مع الحلول الأخرى الموجودة حيث يوفر (DROM) تكوينات توجيه أفضل لتحسين أداء الشبكة من خلال تخصيص التوجيه ولكن لم يتم التطرق لموازنة الحمل بطريقة متكيّفة مع تغير طوبولوجيا الشبكة المستخدمة.

### 2-2 خوارزمية البحث بالعمق أولاً ( Depth First Search DFS)

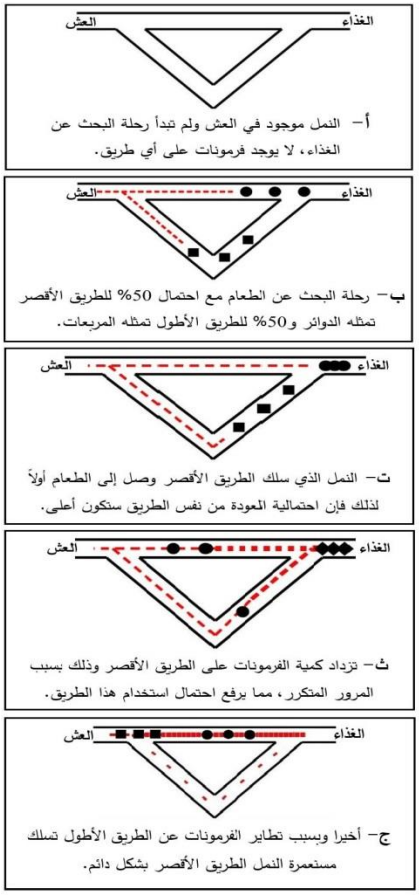
البحث بالعمق أولاً هي خوارزمية يحث في الرسوم البيانية أو في الهيكلية الشجرية حيث يتم تحديد عقدة من عقد الشجرة أو الرسم البياني واتخاذها كجذر ومن ثم يبدأ البحث في العمق ضمن أحد فروع الشجرة ويعرف أيضا بمصطلح طريق عوضا عن فرع وهي تُستخدم في الذكاء الاصطناعي لحل مجموعة متنوعة من المشكلات في مجال التخطيط واتخاذ القرار والأنظمة الخبيرة وما إلى ذلك [7]. ومن الشائع أيضاً أنَّ هذه الخوارزمية تستخدم تقنية التراجع (backtrack) وذلك لحل المشاكل المركبة المختلفة. تُستخدم خوارزمية (DFS) التكرارية لحل مشاكل الأمثلة المتقطعة [8] وتتمثل الميزة الرئيسية لاستراتيجية البحث بالعمق أولاً في أنها تتطلب ذاكرة قليلة جداً. نظراً لأن العديد من المشكلات التي تم حلها بواسطة (DFS) تتطلب عمليات حسابية مكثفة للغاية. في كثير من الأحيان يتم دمج (DFS) مع بعض الاستدلال (الإرشاد) الذي يقدِّر احتمالية وجود الهدف في الأشجار الفرعية المتجذرة للعقدة الداخلية وفي هذه الحالة يتم استدعاء خوارزمية (DFS) بعد أن يتم ترتيب أبناء العقدة من اليسار إلى اليمين عن طريق تقليل قيمة الاستدلال ونقول عن عقدة ما أنّها جيدة إذا احتوت الشجرة الفرعية الموجودة تحتها على العقدة الهدف وسيئة بخلاف ذلك[9]. مع هذا التعريف، يحاول الترتيب الاستدلالي نقل الأبناء الجيدين إلى اليسار والأبناء السيئين إلى اليمين، نظراً لأن (DFS) يبدأ البحث على اليسار وبمساعدة الترتيب الاستدلالي يتم العثور على الهدف بشكل أسرع.

### 2-3 خوارزميات مستوحاة من علم الأحياء (Bio-inspired Algorithms)

الطبيعة هي مصدر إلهام عظيم وهائل لحل المشكلات الصعبة والمعقدة في علوم الحاسب [4] حيث تم استيحاء بعض الخوارزميات من بعض النظريات والآليات البيولوجية، تعد الخوارزميات المستوحاة من علم الأحياء نهجًا ناشئًا يعتمد على مبادئ وإلهام التطور البيولوجي للطبيعة [10] لتطوير تقنيات منافسة جديدة. تتم ملاحظة الأنشطة البيولوجية بعناية ويلاحظ أوجه التشابه مع المشكلات الحسابية المعقدة حيث إنَّ إيجاد التشابه بين النشاط البيولوجي والمشكلات الحسابية يُلهِم الباحثين لبناء نماذج حوسبة عالية الأداء وتطوير خوارزميات ذكية.

### 2-4 أمثَلَة مستعمرة النمل

أمثَلَة مستعمرة النمل هو أسلوب لإيجاد الحل الأمثل تم تقديمه في أوائل التسعينيات، وكان المصدر الملهم لأمثَلَة مستعمرة النمل هو سلوك البحث عن الغذاء لمستعمرات النمل الحقيقية وتمَّ استغلال هذا السلوك وتطبيقه في مستعمرات النمل الاصطناعية للبحث عن حلول تقريبية لمشاكل الاستمثال المتقطعة (مثل مستعمرة النمل)، ومشاكل الاستمثال المستمر (مثل أسراب الطيور)، ولحل المشكلات المهمة في الاتصالات السلكية واللاسلكية مثل التوجيه وموازنة الحمل[11]. تَستَخدِم خوارزمية (ACO) نهجاً احتمالياً لحل مشكلة الهدف تستند هذه الخوارزمية على بعض الاستدلال (الإرشاد heuristic) [12] تم استلهام خوارزمية (ACO) من سلوك النمل حيث أنَّه في البداية تتجول جميع النملات بلا هدف لكنّها تعود إلى العش بعد وضع الفرمون (pheromone) على الطرق أثناء البحث عن الطعام بحيث يمكن للنمل الآخر بعد ذلك اتباع مسارات الفرمون بدلاً من السفر بشكل عشوائي وتم اعتماد هذا النهج لحل بعض مشكلات الشبكات الحاسوبية مثل إيجاد أفضل مسار في الشبكة [13] [14]. تعمل خوارزمية (ACO) في خطوتين في الخطوة الأولى يكتشف النمل الأمامي طرقًا محتملة جديدة بالإضافة إلى جمع معلومات عن المسارات الحالية والتي يشار إليها بالتحديث الأمامي فإذا نجح أحدهم في الوصول إلى العقدة الوجهة تبدأ الخطوة الثانية حيث يتم إرسال بعض النمل مرة أخرى إلى العقدة المصدر على طول المسار الذي تم استكشافه مسبقًا بواسطة النمل الأمامي وأثناء رحلة الرجوع يتم تحديث جداول التوجيه الخاصة بالعقد الموجودة على طول المسار والتي تسمى التحديث العكسي[15]. يعتبر سلوك النمل عشوائي بنسب (stochastic)[16]، حيث أنَّ هذا السلوك يعتمد على الفرمونات، ويبدأ سلوك النمل باكتشاف المساحة التي سيبحث فيها عن الغذاء ويتصرف بشكل متزامن ومستقل، الاعتماد على الفرمونات يحقق جودة عالية لأي حل من خلال التعاون الذي يوفره للمستعمرة. الفرمون هو مادة كيميائية متطايرة تفرزه النملة أثناء رحلة البحث عن الغذاء وتستخدمه باقي النملات فيما بعد للاستدلال على الطريق الأقصر للوصول الى الغذاء وذلك لأن الطريق الأطول سيستغرق وقت أطول وبالتالي احتمال تطاير الفرمون عنه سيكون أعلى بينما الطريق الأقصر ستتركز عليه المادة الكيميائية بالأخص بعد ثاني أو ثالث جولة انتقال للنمل على هذا الطريق بالتالي يعتبر الفرمون هو الذاكرة طويلة الأمد للنمل كما هو مبين في الشكل (1).




**الشكل (1) إعداد تجريبي يوضح قدرة مستعمرات النمل على العثور على أقصر مسار**

يوضح الشكل (1) قدرة مستعمرة النمل على الحصول على أقصر مسار بين العش ومصدر الغذاء، حيث يوجد بين عش

النمل ومصدر الغذاء الوحيد مساران بأطوال مختلفة في الرسومات الخمسة، وتظهر مسارات الفرمون كخطوط حمراء تشير سماكتها إلى تركيز الفرمون ضمن الممرات.

## 3. آلية التحكم وصنع القرار

من الملاحظ أن متحكمات (SDN) لا يتكيف مع طوبولوجيا الشبكة فيما يخص موازنة الحمل وتجنب عنق الزجاجة، وعملية التوجيه الافتراضية (default routing) مشابهة لعملية التوجيه الديناميكي للشبكات التقليدية سواء على مستوى بروتوكولات التوجيه الداخلية (Interior protocols) أو الخارجية (Exterior protocols) وفي كثير من الأحيان تتغلب البروتكولات الديناميكية الحديثة في الشبكات التقليدية على أداء التوجيه الافتراضي غير المتكيف في متحكمات (SDN)، حيث يكون اهتمام آلية التوجيه الافتراضي في المتحكم هي إيجاد أحد الطرق الموصلة للهدف بشكل عشوائي، فلو كانت الطوبولوجيا تفرض طريق واحد بين المصدر والهدف سيكون هذا الحل مثالياً من ناحية اختيار الطريق بينما في حال وجود طريق آخر فستكون حالة من ضربة الحظ إصابة الطريق الأفضل ضمن الظروف اللحظية للشبكة، مثلاً من الممكن أن يكون الطريق (أ) مثالياً لنوع معين من الحزم في اللحظة (t=1) ولكن في اللحظة (t=5) من الممكن أن يكون الطريق (ب) هو المثالي لهذا النوع من الحزم الموجهة ولكن بالنسبة للأداء الافتراضي للمتحكم لا يزال الطريق (أ) مثالي لطالما يؤدي الغرض المطلوب ألا وهو إيصال الحزم من المصدر إلى الوجهة، لذلك وجب في البداية تشخيص هذا الأمر على أنه مشكلة بحثية في عالم الشبكات المعرفة برمجياً، وتفتح للباحثين الباب لإيجاد حلول دائمة قادرة على التكيُّف مع الحالة اللحظيَّة للشبكة. في هذا العمل تمَّ تصميم وتنفيذ آلية استدلال (إرشاد) متكيفة مع تغييرات طوبولوجيا الشبكة ومتطلبات التوجيه الخاصة بنوعية الحزم وذلك من خلال التهجين بين خوارزمية (DFS) وخوارزمية أمثلة مستعمرة النمل وتم اعتمادها لعدة أسباب نذكر منها: أولاً: إمكانية التحكم في نقل المسار من طريق الى آخر بسهولة عند كل حساب. ثانياً: توفر الخوارزمية معلومات كافية لاختيار الطريق بشكل متكيف في كل لحظة عن طريق المعلومات التي توفرها الفرمونات. ثالثاً: تحقق الخوارزمية أرضية مناسبة لتوزيع الحمل بشكل كفؤ، حيث أن خريطة الفرمونات توفر معلومات عن الحالة اللحظية للمسارات المتوفرة كما ويمكن إضافة محددات (parameters) إضافية لعملية الحساب لتحقيق موازنة الحمل المطلوبة.

### 3-1 رسالة وصول حزمة (Packet IN Message)

عند وصول حزمة جديدة الى أحد مبدلات شبكة (SDN) (التي تعمل على بروتوكول Open Flow 1.3) سيبحث المبدل عن مطابقة لترويسة الحزمة مع احدى قواعد جداول التدفق الخاصة بهِ (flow tables)، في حال تمت المطابقة سيقوم المبدل بتنفيذ التعليمة (instruction) (في الإصدار Open Flow 1.1 كانت تسمى فعل (action)) الذي تنص عليه القاعدة المُطابقة، أما في حال عدم وجود أي قاعدة مطابقة سيسأل المبدل السؤال التالي: ماذا أفعل بهذه الحزمة؟ في الحقيقة فإنَّ المبدل يوجه هذا السؤال إلى متحكم الشبكة وذلك بإرسال رسالة وصول حزمة (Packet IN) الى المتحكم وتحتوي هذه الرسالة على الحزمة بالإضافة الى معلومات المبدل نفسه، يتلقى المتحكم الرسالة ويقوم بفك ترميزها وإرسالها الى مصنع القرار وعند كل محطة في هذا المصنع يقوم مُنصِت (listener) من المنصتين المسؤولين عن صناعة القرارات الفرعية (بعض هؤلاء المنصتين يمثِّل الآلية المقترحة) بإجراء الحسابات اللازمة ومن ثم اتخاذ قرار فرعي مناسب.

### 3-2 مصنع القرار (Decision Factory)

يمكن التعبير عن مصنع القرار بشكل مجازي على أنه ممر لصناعة القرار تمر من خلاله رسائل وصول الحزمة ويحتوي مجموعة من المُنصتات المسجلة على المتحكم مرتبة

على طول الممر حسب الأولوية من الأهم إلى الأقل، وللمُنصتات ثلاثة أنواع من حيث الصلاحيات والمهام (مراقب (Observer)، ناصح (Advisor)، مدير (Director)) يمكن لكل من الناصح والمدير قراءة و تعديل رسالة وصول الحزمة ومن ثمَّ إضافة التغييرات المطلوبة قبل تسليمها الى المُنصِت التالي وفي حال نتج عن هذه التغييرات قرار يقوم المدير بإرسال مُدخَل جديد (new entry) ضمن رسالة تعديل التدفق (flow-mod) (واحدة من رسائل بروتوكول OpenFlow 1.3 لتعديل جداول التدفق) إلى المبدلات التي ستمر من خلالها الحزمة ليتم إضافة هذا المُدخَل إلى جداول تدفق المبدلات والتعامل فيما بعد مع الحزمة، بينما تقتصر وظيفة المراقب على المراقبة (ليس له أي صلاحية تعديل). صناعة القرار عملية تجميعية تمر بعدة مراحل، تقع مسؤولية صناعة القرارات الفرعية وإجراء الحسابات الخاصة بكل مرحلة على عاتق مجموعة من الإجرائيات، تقوم هذه الإجرائيات بإجراء العمليات الحسابية المناسبة واتخاذ القرار الفرعي ليتم في نهاية المطاف الخروج من مصنع القرار بقرار مناسب للحزمة الواصلة الى المتحكم، مصنع القرار الافتراضي ( default decision maker) الموجود ضمن متحكم شبكات (SDN) يؤدي المهام المطلوبة منه على أكمل وجه ولكن بطريقة غير متكيفة مع التغيرات اللحظية، بمعنى آخر لا يقدم أية ضمانات لموازنة الحمل و تجنب حالات عنق الزجاجة، وأغلب الأبحاث التي تمت فيما سبق وتم استعراضها في قسم المؤلفات السابقة كانت لا تتناول مصنع القرار ولا تتدخل بصناعة القرار بشكل مباشر، وإنَّما اقتصرت على تصميم وتشغيل تطبيقات خارجية تعمل على طبقة التطبيقات وتتواصل مع المتحكم عن طريق الواجهة الشمالية مما يقلل من قدرة هذه التطبيقات على التكيف مع التغييرات اللحظية لحالة الشبكة والوصلات وذلك لأنها تعتمد في عملها على مصنع القرار الافتراضي الذي يستخدمه المتحكم والذي يوفر كمية كبيرة من المعلومات ولكن يعاني من سوء إدارة وتصرف بهذه المعلومات. تقوم آلية الاستدلال الهجينة المنفذة في هذا البحث بالتدخل بشكل مباشر في صناعة القرار وتُعتَبر جزء لا يتجزأ من مصنع القرار بالنسبة للمتحكم بحيث تمَّ إنشاء وإضافة عدة عمليات (processes) فرعية تتدخل في صناعة قرار التوجيه المناسب الذي يحقق موازنة الحمل على مستوى الوصلات والشبكة بشكل عام لتحسين الأداء.

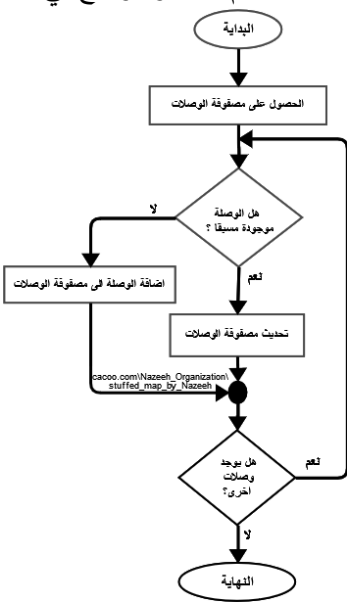
## 4. الآلية المقترحة

تعمل الآلية المقترحة بشكل متكامل ضمن طبقة التحكم وهي عبارة عن مجموعة من الإجرائيات والعمليات المسؤولة عن جمع المعلومات المتوفرة لدى المتحكم وضبط رسائل بروتوكول (ARP) وتحديد وحفظ جميع المسارات (paths) الموجودة بين المصدر والوجهة ضمن قاعدة بيانات المتحكم ويتم بعد ذلك اختيار المسار المناسب للحفاظ على حمل متوازن ضمن الشبكة وإطلاق رسائل (flow-mod) لإعلام المبدلات بالمسارات. سيتم شرح خطوات عمل الآلية المقترحة في القسم التالي مع عرض مخططات التدفق (flowchart) لكل مرحلة، وعند وقوع حدث (catch event) سيتحقق المتحكم من نوع الحدث كما هو موضح في الشكل (2).


**الشكل (2) مخطط تدفق لمُنصتات مصنع القرار**

يوضح الشكل (2) مخطط التدفق الخاص بالمنصتات الثلاث الخاصين بآلية موازنة الحمل المقترحة (ملاحظة: جميع العمليات المشار عليها باللون الغامق سيتم عرض المخططات الخاصة بها).

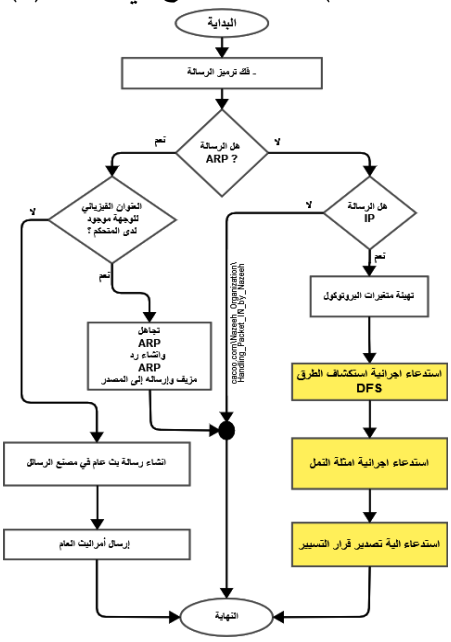
في حال كان الحدث المُستَلَم من قبل المُنصِت هو تغيير في الطوبولوجيا يتم حذف المعلومات الموجودة في جدول المعطيات (Data) الخاص بتخزين المبدلات (يتم التمييز بين كافة المبدلات عن طريق الرقم المميز DPid) وجدول الوصلات (Links) الخاص بتخزين كافة وصلات الطوبولوجيا وجدول العناوين الفيزيائية (MAC) الخاص بتخزين العناوين الفيزيائية بين طرفي الوصلات ومن ثمَّ إطلاق إجرائية (stuffed_map) لاستكشاف الطوبولوجيا الجديدة وتحديث قاعدة بيانات المتحكم كما هو موضح في الشكل (3).




**الشكل (3) مخطط التدفق لإجرائية استكشاف الطوبولوجيا الجديدة**

يوضح الشكل (3) خطوات عمل إجرائية استكشاف التغييرات في طوبولوجيا الشبكة حيث يتم جمع كافة المعلومات عن وصلات الشبكة ونلاحظ استدعاء إجرائية إضافة وصلة (links.Add) ضمن المخطط لإضافة الوصلات الجديدة الى جدول الوصلات والجدير بالذكر هنا أنّه يتم التمييز بين الوصلات من خلال ما يلي: لكل وصلة بداية ونهاية، يتم تمييز بداية الوصلة برقم التعريف الخاص بالمبدل المصدر (DPid_Src)، أمّا نهاية الوصلة فيتم تمييزها بزوج مكون من رقم المنفذ المصدر (Src_port) ورقم المنفذ الوجهة (Dst_port) بالإضافة إلى رقم التعريف الخاص بالمبدل الوجهة (DPid)، بالإضافة الى ذلك يتم تخزين طول هذه الوصلة (length) وفي نهاية عمل إجرائية استكشاف التغييرات في طوبولوجيا الشبكة يكون المتحكم قد حصلَ على رؤية شاملة لمواقع كافة المبدلات مع كامل الوصلات (links) ضمن الشبكة.

أما في حال كان الحدث هو رسالة وصول حزمة سيقوم المتحكم بالتحقق من نوع الرسالة ومن ثمَّ التحقق من المعلومات المتعلقة بموقع كل من المصدر (يرمز له Src) والوجهة (يرمز له Dst) كما هو موضح في الشكل (4).

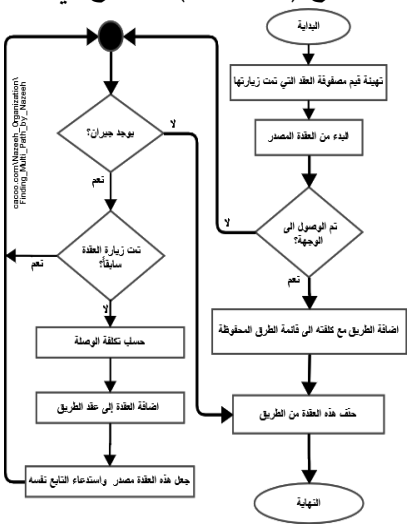



**الشكل (4) مخطط تدفق إجرائية معالجة رسالة وصول حزمة**

يوضح الشكل (4) مخطط التدفق لإجرائية معالجة رسالة وصول الحزمة بالإضافة الى الإجرائية الخاصة بضبط عمل بروتوكول (ARP)، حيث أنّه في حال عدم توفر معلومات

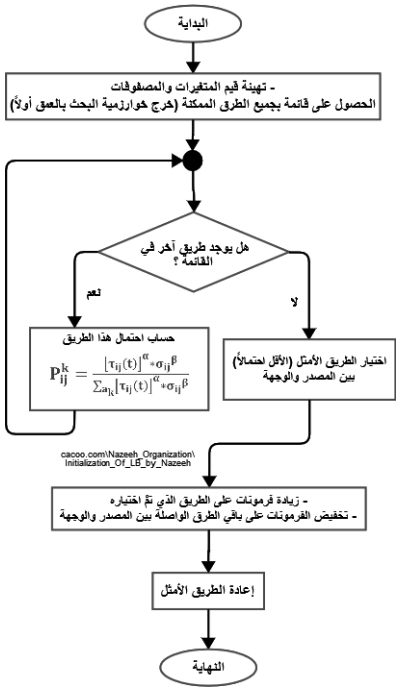
عن الوجهة في قاعدة بيانات المتحكم ستقوم هذه الإجرائية بالاستعلام عن الوجهة برسالة بث عام، تم إضافة هذه الإجرائية الى مصنع القرار لتتواصل بالشكل المطلوب والمناسب مع المُنصتات الأخرى في مصنع القرار.

كما هو موضح في الشكل (4) عند معالجة رسالة وصول الحزمة فإنه من الضروري حساب وتخزين كافة المسارات التي تربط المصدر بالوجهة ويتم ذلك باستدعاء الإجرائية (Compute_Path()) والتي تعتمد على خوارزمية البحث بالعمق أولاً ومبدأ التراجع (backtrack) الموضح في الشكل (5).




**الشكل (5) مخطط تدفق اجرائية حساب كافة المسارات بين المصدر والوجهة**

بعد حساب كافة المسارات بين المصدر والوجهة لابد من المفاضلة بين هذه المسارات و تحديد المسار المناسب للحفاظ على موازنة الحمل ضمن الشبكة تتم هذه العملية باستدعاء الإجرائية (Select_Path()) التي تعتمد على أمثلة مستعمرة النمل لتحديد المسار الأمثل كما هو موضح في الشكل (6).




**الشكل (6) مخطط تدفق اختيار المسار الأمثل بتطبيق خوارزمية (ACO)**

باعتبار كل حزمة ضمن الشبكة عبارة عن نملة من مستعمرة النمل ويمكن أن تبدأ بالحركة من أي عقدة في الشبكة بشكل عشوائي، ونفترض أنَّ (m) هو العدد الكلي للنمل و(n) هو العدد الكلي للمبدلات، وباعتبار $(b_i(t))$ هو عدد النمل الموجود في المبدل $(\mathcal{S}_i)$ في اللحظة (t)، ينتج لدينا التصميم الموضح في المعادلة1.

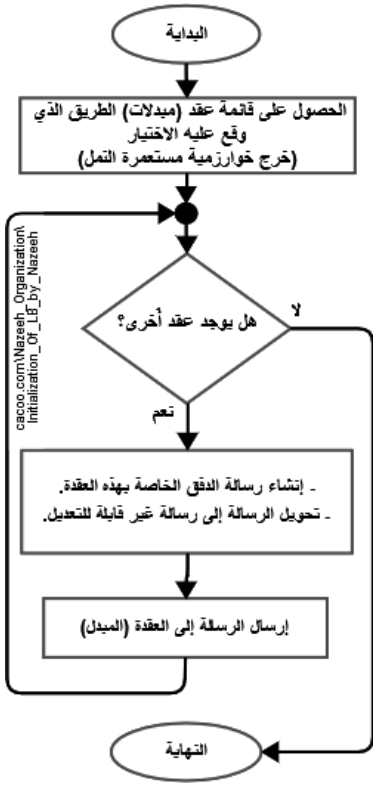
المعادلة (1): $\mathbf{m = \sum_{i=1}^{n} b_i(t)}$

توضِّح المعادلة (1) عدد الحزم الموجودة في المبدل $(\mathcal{S}_i)$. ونعبر عن كل نملة (حزمة) بالاسم $(A_k)$ بحيث (k=1,2,…,m) يتم تسجيل كافة العقد التي زارتها الحزمة ضمن جدول يسمّى جدول المحرمات، ويتم حساب احتمال الانتقال من مبدل الى آخر من خلال حساب الفرمون المتبقي على كل مسار، نلاحظ أنَّ $(P_{ij}^k)$ هو احتمال زيارة النملة $(A_k)$ للمبدل $(\mathcal{S}_j)$ انطلاقا من المبدل $(\mathcal{S}_i)$. بالتالي ينتج لدينا

التصميم التالي الذي يمثل احتمال اختيار المسار الأمثل كما هو موضح في المعادلة (2).

المعادلة (2): [17] $P_{ij}^{k} = \frac{[\tau_{ij}(t)]^{\alpha} * \sigma_{ij}{}^{\beta}}{\sum_{a_k}[\tau_{ij}(t)]^{\alpha} * \sigma_{ij}{}^{\beta}}$ توضح المعادلة (2) احتمال اختيار المسار حيث $(a_k)$ تمثل المبدلات المتاحة عندما تكون $(A_k)$ في المبدل $(\mathcal{S}_i)$، و $(\tau_{ij}(t))$ تمثل الفرمون الكلي الموجود على المسار من المبدل $(\mathcal{S}_i)$ إلى المبدل $(\mathcal{S}_j)$، تمثل بيتا (β) طول المسار بينما ألفا (α) تمثل وزن الفرمون المتبقي. وتمثل الدالة $(\sigma_{ij}{}^{\beta})$ دالة الاستدلال الذاتي (heuristic function).

نسعى في هذا البحث الى إيجاد طريقة مناسبة وديناميكية لموازنة الحمل فيمكن الاستفادة من المعلومات المقدمة من قبل خوارزمية النمل لتبيان وضع الوصلات في الشبكة عند كل لحظة حساب، أي لحظة ورود رسالة الى المتحكم من قبل مبدل (packet in). تتم الاستفادة من مصفوفة الفرمونات لمعرفة وضع الوصلات (links) وبتطبيق المعادلة (2) نحصل على احتمال كل طريق من هذه الطرق، للوصول الى أفضل نتيجة يتم اختيار الطريق ذو الحمل الأقل، من الواضح أن القضية في هذه الحالة هي قضية تعظيم أو تصغر ( Max or Min) في حال ذهبنا باتجاه التعظيم ينتج لدينا اكثر الطرق تفضيلا من قبل الرسائل (وهذا يحتمل نسبة من العشوائية) و في حال ذهبنا باتجاه التصغير نقدم عرضا مسرحياً مبتكراً في توزيع الحمل، ويعود الفضل في ذلك الى اختيار الطرق الأقل ازدحاماً ليس من ناحية عدد الرسائل المتواجه عليه حاليا ولكن الأقل تفضيلا من قبل باقي الرسائل فنحقق بذلك مساواه اجتماعية بين الوصلات على الشبكة و تعم السعادة أرجاء المملكة (الطوبولوجيا). بعد اختيار المسار الأمثل واتخاذ قرار التوجيه، تقوم الآلية بتحديث خريطة الفرمونات لتناسب التغيير الحاصل، حيث يتم رفع مستوى الفرمونات الموجودة على وصلات هذا المسار وخفض مستوى الفرمونات على باقي وصلات الشبكة مع احترام الفاصل الزمني لكل قيد موجود على المبدل، ويتبع ذلك لنوع الرسالة وحجمها ومدة الإرسال، ومن ثمَّ تقوم الآلية بتصدير قرار التوجيه على شكل (flow-mod) وإرساله الى كل المبدلات الموجودة على طول هذا المسار ويتم تصدير الرسالة من مصنع القرار بصيغة (immutable message) أي أنّها غير قابلة للتعديل نهائياً كما هو موضح في الشكل (7).




**الشكل (7) مخطط التدفق لتصدير قرار التوجيه إلى كافة مبدلات المسار**

## 5. التصميم

تم إنشاء تصميمين مختلفين للشبكة من حيث الطوبولوجيا، وتم اعتماد التصاميم المعتمدة في مراكز البيانات[18]، وذلك من أجل إجراء التجارب وتسجيل نتائج التجارب على كل طوبولوجيا قبل إجراء أي تعديلات على عمل المتحكم (المتحكم يعمل بالنمط الافتراضي) وتمَّ إعادة نفس السيناريو

على كل طوبولوجيا بعد تطبيق الآلية المقترحة من أجل تسجيل ومقارنة النتائج، تمَّ إنشاء الطوبولوجيا المخصصة لكل تصميم باستخدام الأداة (Miniedit) المرفقة مع (Mininet) وهي أداة مكتوبة باستخدام واجهة برمجة تطبيقات بايثون (Python API) [19] [20]، وتساعد هذه الأداة على إنشاء طوبولوجيا معقدة بشكل بسيط ومرئي.

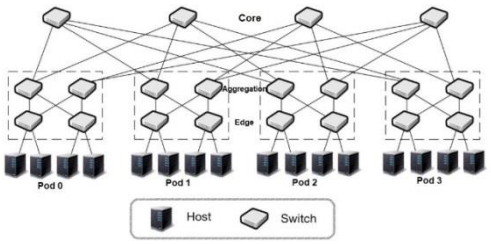
تم استخدام جهاز حاسب واحد لتشغيل التصميمين المقترحين وتنفيذ التجارب، الحاسب المُستَخدم يعمل بوحدة معالجة (Intel Core i5-4210U CPU @ 1.70GHz 2.40 GHz) وذواكر وصول عشوائي (GB 16.0).

ملاحظة: تمَّ تحويل نمط جميع المبدلات عن طريق المتحكم من (hybrid-mode) إلى (pure Open Flow mode).

### 5-1 الطوبولوجيا Fat-Tree:

في هذا التصميم تم استخدام طوبولوجيا (fat-tree)، تُعتبر هذه الطوبولوجيا مناسبة للاستخدام في الحوسبة السحابية ومراكز البيانات بسبب قابليتها للتوسع بشكل سهل وذلك بتكرار طوبولوجيا الشجرة من ثلاث طبقات وضمها الى الطوبولوجيا المستخدمة دون الحاجة الى أعادة عنونة الأجهزة بشكل كامل بالإضافة الى ذلك تتميز هذه الطوبولوجيا بإمكانية تحقيق التصيف الكامل لعرض الحزمة على كامل العقد الموجودة ضمن المجموعة الواحدة (مبدلات التوزيع والحافة المتصلة مع بعضها بشكل مباشر) وتعتبر طوبولوجيا الشجرة الثخينة واحدة من أهم الطوبولوجيات المستخدمة في مراكز بيانات شركتي (Google, Facebook) [21][22]، الشكل (8) يوضِّح التصميم الأول للشبكة.

يتم التحكم بكامل الطوبولوجيا المقترحة عن طريق المتحكم C0 وهو من نوع (HPE VAN).




**الشكل(8) الطوبولوجيا fat-tree المستخدمة في التصميم الأول [22]**

يوضح الشكل (8) طوبولوجيا شبكة (fat-tree) المكونة من ثلاث طبقات النواة والتجميع والحافة (core, aggregation and edge) تمَّ إنشاء هذه الطوبولوجيا باعتبار (K=4) حيث (K) تمثل عدد منافذ المبدل الواحد، ينتج لدينا عدد مبدلات طبقة (core) بتطبيق المعادلة 3 وعدد مبدلات كلاً من طبقة (aggregation) وطبقة (edge) بتطبيق المعادلة (4).

المعادلة (3): $N_{ofCore} = K^2/4$ [22]

المعادلة (4): $N = K^2/2$ [22]

بذلك يكون العدد الكلي للمبدلات في الطوبولوجيا موضحاً في المعادلة (5)، بينما العدد الكلي للمضيفين موضحاً في المعادلة (6).

المعادلة (5): $S = 5K^2/4$

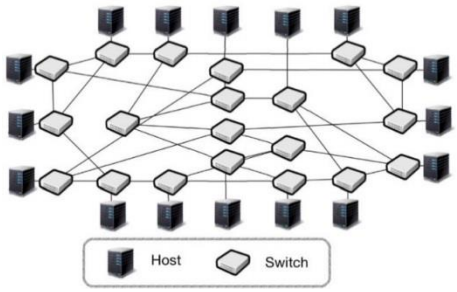
المعادلة (6): $H = K^3/4$

### 5-2 الطوبولوجيا Jellyfish:

في هذا التصميم تم استخدام طوبولوجيا قنديل البحر (Jellyfish) وهي عبارة عن شبكة ربط عالي السعة، وتعتمد على رسم مخطط (graph) عشوائي، ويتم تشبيه هذه الطوبولوجيا بقنديل البحر الذي ينتج نفسه بشكل طبيعي من أجل التوسع التدريجي، يُعتبر تصميم طوبولوجيا (Jellyfish) تصميماً غير بنيوي (unstructured) ويجلب تحديات جديدة في عمليات التوجيه والتوزيع المادي للأسلاك، يوجد العديد من التقييمات التي تشير أنه يمكن نشر قنديل البحر في مراكز البيانات الحالية بكفاءة ومرونة وبالأخص في حالة اقترانها

بالتحكم الأمثل في التوجيه وإدارة الاختناقات[23][24]. تم بناء طوبولوجيا قنديل البحر باعتبار (K=4) حيث أن (K) تُمثِل عدد منافذ المبدل، يتم حساب عدد المبدلات (S) والمضيفين (H) باستخدام المعادلتين (5) و(6) على التوالي.

تم بناء الطوبولوجيا (jellyfish) باستخدام الأداة (Miniedit) ويتم التحكم بكامل الطوبولوجيا عن طريق متحكم (c0) كما هو موضح في الشكل (9).

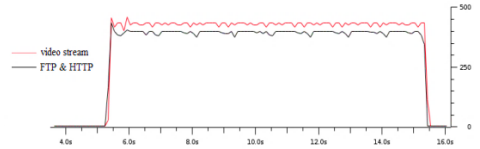



**الشكل (9) الطوبولوجيا jellyfish المستخدمة في التصميم الثاني[22]**

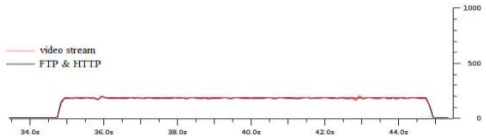
## 6. التنفيذ والنتائج

تمَّ استخدام الأداة (iperf) وهي أداة مكتوبة بلغة بايثون تقوم بإرسال حزم (TCP) و(UDP) بين الأجهزة، تمَّ تشغيل تعليمات (iperf) على جميع الأجهزة الموصولة في كل تصميم من التصميمين وذلك من اجل زيادة حركة المرور ضمن الشبكة ولضمان توليد حمل كافٍ لإيصال الوصلات لحالة الاختناق، حيث تقوم الأجهزة بسحب ملفات مختلفة وطلب صفحات ويب ونقل تدفق فيديو من بعضها مسببةً بذلك زيادة الحمل على الشبكة من خلال زيادة حركة مرور الحزم على الوصلات التي يتم اختيارها تلقائيا من قِبَل المتحكم.

تم تكرار التجربة ذاتها وبذات الظروف على كل تصميم على حدا دون التدخل في عملية التوجيه التي يستخدمها المتحكم بشكل افتراضي، وتسجيل النتائج حيث الشكل (10) والشكل (11) يوضحان عدد الحزم التي تم نقلها خلال زمن تنفيذ التجارب على كل من التصميم الأول والتصميم الثاني على التوالي حيث يمثل المخطط باللون الأسود عدد الحزم المنقولة باستخدام بروتوكولي (FTP, HTTP) بينما النقاط باللون الأحمر تمثل عدد الحزم المنقولة لتدفق الفيديو التي تجري بين الأجهزة.

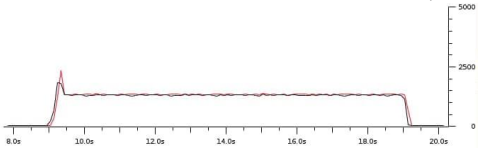



**الشكل (10) تدفق الحزم في طوبولوجيا fat-tree قبل تطبيق الآلية**

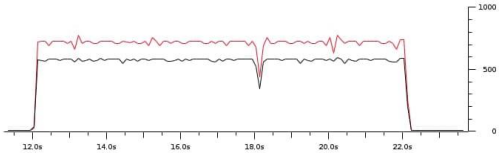



**الشكل (11) تدفق الحزم في طوبولوجيا jellyfish قبل تطبيق الآلية**

بعد تسجيل النتائج السابقة تمَّ تكرار التجربة ولكن بتطبيق آلية الاستدلال الهجينة المقترحة وذلك عن طريق إيقاف برامج التوجيه الافتراضية في المتحكم وإضافة مجموعة من ( OSGI Bundles) مكتوبة بلغة جافا عوضاً عنها، وتم تسجيل النتائج الجديدة لمعدل نقل الحزم على نفس التصميمين السابقين كما هو موضّح في الشكل (12) والشكل (13).



**الشكل (12) تدفق الحزم في طوبولوجيا fat-tree بعد تطبيق الآلية**



**الشكل (13) تدفق الحزم في طوبولوجيا jellyfish بعد تطبيق الآلية**

## 7. التحليل ومقارنة النتائج

عند مراقبة مدخلات جداول الدفق (flow tables) في مبدلات الشبكة في طوبولوجيا fat-tree ومقارنة المخططات الظاهرة في الشكل (10) والشكل (12) نلاحظ بأن السلوك الذي يتبعه المتحكم في توجيه التدفقات يسبب تأخير في وصول الحزم نتيجة الضغط الحاصل على بعض الوصلات

التي تم اختيارها بشكل افتراضي وهذه هي الحالة الافتراضية في شبكات SDN التي يتحكم في توجيه الحزم فيها المتحكم (HPE Van)، في التجربة الأولى قبل تعديل سلوك التوجيه نلاحظ أنّ عدد الحزم الواصلة تقريبا بمعدل 400 حزمة في الثانية الواحدة بالنسبة لحزم بروتوكولي (FTP, HTTP) وفي أفضل الحالات إلى 450 حزمة بالنسبة لحزم تدفق الفيديو.

أمّا في التصميم الثاني الذي يستخدم طوبولوجيا jellyfish عند مراقبة مدخلات جداول الدفق (flow tables) في مبدلات الشبكة ومقارنة المخططات الظاهرة في الشكل (11) والشكل (13) نلاحظ أنَّ عدد الحزم الواصلة تقريبا بمعدل 200 حزمة في الثانية الواحدة بالنسبة لحزم بروتوكولي (FTP, HTTP) ومثلها بالنسبة لحزم تدفق الفيديو.

يقوم المتحكم في الحالة الافتراضية باختيار أول طريق يتم اكتشافه من قبل تطبيقين موجودين ضمن نظام تشغيل المتحكم وهما (Path Diagnostics, Path Daemon) حيث يتم قراءة ترويسة الحزمة واستخراج مصدر ووجهة الحزمة ومن ثم اكتشاف الطريق الواصل بينهما وذلك بعد وصول حدث (packet-in) من أول مبدل لا يجد أي حقل يطابق حقول ترويسة الحزمة مع مدخلات جدول التدفق الخاص بهذا المبدل ومن ثم إرسال حدث (packet-out) إلى جميع المبدلات المتواجدة على المسار المكتشف بتوقيت كافي لتمرير الحزمة على الطريق المطلوب عن طريق بروتوكول (Open Flow)، وتقوم المبدلات باستلام حدث (packet-out) وتحديث جداول التدفق لديها ومن ثم تمرير الحزمة إلى الهدف وعند انتهاء الوقت الممنوح من قبل المتحكم لهذه التدفقات (Flows) يقوم المبدل بحذفها من جداوله.

عندما استخدم المتحكم بالحالة الافتراضية هذه الإجرائية تجاهل استخدام الوصلات الأخف حمولة مسببا بذلك ضغط على الوصلات الموجودة على الطرق التي حددت من قبله وبالتالي نشأت هذه المشكلة.

أما بعد تطبيق آلية الاستدلال الهجينة المقترحة لموازنة الحمل عن طريق اختيار المسارات الأمثل نلاحظ أن زمن وصول الحزم بشكل كامل قد انخفض وذلك بسبب زيادة عدد الحزم الواصلة في الثانية الواحدة الذي ارتفع ليصبح تقريباً بمعدل 1500 حزمة بالنسبة لحزم بروتوكولي (FTP, HTTP) ومثلها في تدفق الفيديو للتصميم الأول (fat-tree)، أمّا بالنسبة للتصميم الثاني (jellyfish) فإنَّ عدد الحزم الواصلة في الثانية الواحدة ارتفع إلى 600 حزمة تقريباً بالنسبة لحزم بروتوكولي (FTP, HTTP) بينما في تدفق الفيديو ارتفع عدد الحزم الواصلة إلى 800 حزمة في الثانية الواحدة، وبمقارنة هذه النتائج نلاحظ التحسن الواضح في تقليل التأخير و زيادة الإنتاجية (throughput) من خلال توزيع الأحمال بشكل عادل و استغلال الوصلات الأقل حمولة كما يمكن ملاحظة الفرق الحاصل في من تغيير الطوبولوجيا سواء كان المتحكم يعمل بالطريقة الافتراضية أو بعد تطبيق الآلية المقترحة حيث تثبت التجربة أن أداء المتحكم يتغير بتغير طوبولوجيا الشبكة وفي كلا التصميمين زاد عدد الحزم المنقولة في الثانية الواحدة إلى الضعف تقريباً.

## 8. الخاتمة

الفكرة الرئيسية من هذا العمل هي تطبيق آلية للتدخل في التوجيه وتغيير نمط التوجيه الافتراضي لمتحكم الشبكات المعرفة برمجياً (HPE VAN SDN Controller) عن طريق قراءة معاملات الشبكة والتدخل في صناعة قرارات التحكم والتوجيه في الزمن الحقيقي عن طريق طبقة التحكم وذلك بغرض تخفيف الحمل وإيجاد المسار الأمثل ضمن المسارات المتاحة بين جميع التجهيزات الشبكية التي تعمل في مركز البيانات، بعد تطبيق الآلية المقترحة نلاحظ التحسن الملحوظ في معدل نقل البيانات.

# 9. References

[1] Z. Shu *et al.*, "Traffic engineering in software-defined networking: Measurement and management," in IEEE Access, vol. 4, pp. 3246-3256, 2016.

[2] Shreya, T., Mulla, M. M., Shinde, S., & Narayan, D. G. (2020, July). Ant colony Optimization-based dynamic routing in Software defined networks. In *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)* (pp. 1-7). IEEE.

[3] Dobrijevic, O., Santl, M., & Matijasevic, M. (2015). Ant colony optimization for QoE-centric flow routing in software-defined networks. 2015 11th International Conference on Network and Service Management (CNSM). https://doi.org/10.1109/cnsm.2015.7367371

[4] Kong, D., & Zhang, G. (2020). Ant Colony Algorithm Based Routing Protocol in Software Defined Vehicular Networks. Proceedings of the 2020 the 4th International Conference on Innovation in Artificial Intelligence. https://doi.org/10.1145/3390557.3394324

[5] Mohammadi, R., Javidan, R., & Keshtgari, M. (2018). An intelligent traffic engineering method for video surveillance systems over software defined networks using ant colony optimisation. International Journal of Bio-Inspired Computation, 12(3), 173. https://doi.org/10.1504/ijbic.2018.094625

[6] Yu, C., Lan, J., Guo, Z., & Hu, Y. (2018). DROM: Optimizing the Routing in Software-Defined Networks With Deep Reinforcement Learning. IEEE Access, 6, 64533–64539. https://doi.org/10.1109/access.2018.2877686

[7] Rao, V. N., & Kumar, V. (1987). Parallel depth first search. Part I. Implementation. International Journal of Parallel Programming, 16(6), 479–499. https://doi.org/10.1007/bf01389000

[8] Korf, R. E. (1988). Optimal Path-Finding Algorithms. Search in Artificial Intelligence,223–267. https://doi.org/10.1007/978-1-4613-8788-6_7

[9] Meseguer, P. (1997, August). Interleaved depth-first search. In IJCAI (Vol. 97, pp. 1382-1387).

[10] Kar, A. K. (2016a). Bio inspired computing – A review of algorithms and scope of applications. Expert Systems with Applications,59,20–32. https://doi.org/10.1016/j.eswa.2016.04.018

[11] Ying-Tung Hsiao, Cheng-Long Chuang, & Cheng-Chih Chien. (2004). Computer network load-balancing and routing by ant colony optimization. Proceedings. 2004 12th IEEE International Conference on Networks (ICON 2004) (IEEE Cat. No.04EX955). https://doi.org/10.1109/icon.2004.1409157

[12] Xue, H., Kim, K., & Youn, H. (2019). Dynamic Load Balancing of Software-Defined Networking Based on Genetic-Ant Colony Optimization. Sensors, 19(2), 311. https://doi.org/10.3390/s19020311

[13] Hailong Zhang, & Xiao Guo. (2014). SDN-based load balancing strategy for server cluster. 2014 IEEE 3rd International Conference on Cloud Computing and Intelligence Systems. https://doi.org/10.1109/ccis.2014.7175817

[14] Blum, C. (2005). Ant colony optimization: Introduction and recent trends. Physics of Life Reviews, 2(4), 353–373. https://doi.org/10.1016/j.plrev.2005.10.001

[15] Janacik, P., Orfanus, D., & Wilke, A. (2013). A Survey of Ant Colony Optimization-Based Approaches to Routing in Computer Networks. 2013 4th International Conference on Intelligent Systems, Modelling and Simulation. https://doi.org/10.1109/isms.2013.20

[16] Meuleau, N., & Dorigo, M. (2002). Ant Colony Optimization and Stochastic Gradient Descent. Artificial Life, 8(2), 103–121. https://doi.org/10.1162/106454602320184202

[17] Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. IEEE

Computational Intelligence Magazine, 1(4), 28–39. https://doi.org/10.1109/mci.2006.329691

[18] Lebiednik, B., Mangal, A., & Tiwari, N. (2016). A survey and evaluation of data center network topologies. arXiv preprint arXiv:1605.01701.

[19] Kaur, K., Singh, J., & Ghumman, N. S. (2014, February). Mininet as software defined networking testing platform. In International Conference on Communication, Computing & Systems (ICCCS) (pp. 139-42).

[20] De Oliveira, R. L. S., Schweitzer, C. M., Shinoda, A. A., & Prete, L. R. (2014, June). Using mininet for emulation and prototyping software-defined networks. In 2014 IEEE Colombian Conference on Communications and Computing (COLCOM) (pp. 1-6). IEEE.

[21] Andreyev, A. (2018, June 27). Introducing data center fabric, the next-generation Facebook data center network. Engineering at Meta. Retrieved May 19, 2021, from https://engineering.fb.com/2014/11/14/production-engineering/introducing-data-center-fabric-the-next-generation-facebook-data-center-network/

[22] Al-Fares, M., Loukissas, A., & Vahdat, A. (2008). A scalable, commodity data center network architecture. ACM SIGCOMM computer communication review, 38(4), 63-74.

[23] Singla, A., Hong, C. Y., Popa, L., & Godfrey, P. B. (2012). Jellyfish: Networking data centers randomly. In 9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12) (pp. 225-238).

[24] Shafiee, M., & Ghaderi, J. (2017). A simple congestion-aware algorithm for load balancing in datacenter networks. IEEE/ACM Transactions on Networking, 25(6), 3670-3682.