Implementing Resource-aware Multicast Forwarding in Software Defined Networks

Poderys, Justas; Sunny, Anjusha; Soler, José

Published in:
Proceedings of the 6th World Conference on Information Systems and Technologies (WorldCist'18)

Link to article, DOI:
10.1007/978-3-319-77712-2_29

Publication date:
2018

Document Version
Peer reviewed version

Citation (APA):
Poderys, J., Sunny, A., & Soler, J. (2018). Implementing Resource-aware Multicast Forwarding in Software Defined Networks. In Á. R. (Ed.), *Proceedings of the 6th World Conference on Information Systems and Technologies (WorldCist'18)* (pp. 299-308). Springer. Advances in Intelligent Systems and Computing https://doi.org/10.1007/978-3-319-77712-2_29

Implementing Resource-aware Multicast Forwarding in Software Defined Networks

Justas Poderys[1], Anjusha Sunny[2], and Jose Soler[1]

Department of Photonics Engineering,
Technical University of Denmark,
Kongens Lyngby, Denmark
[1]{juspo, joss}@fotonik.dtu.dk,
[2]s150896@student.dtu.dk

Abstract. Using multicast data transmissions, data can be efficiently distributed to a high number of network users. However, in order to efficiently stream multimedia using multicast communication, multicast routing protocols must have knowledge of all network links and their available bandwidth. In Software Defined Networks (SDN), all this information is available in a centralized entity - SDN network. This work proposes to utilize the SDN paradigm to perform network-resources aware multicast data routing in the SDN controller. In a prototype implementation, multicast data is routed using a modified Edmonds-Karp algorithm, by taking into account network topology and links load information. This paper presents the algorithm, implementation details, and an analysis of the testing results.

Keywords: multicasting, SDN, Edmonds-Karp Algorithm

1 Introduction

Streaming television delivery is a core part of a triple-play offering (data, telephony and television) by telecommunication operators. Streaming services require low delay data transmission with specific bandwidth requirements known in advance [4]. Since data-delivery service guarantees are highly desirable for these applications, streaming multimedia data paths should be engineered by taking into account network bandwidth, jitter, packet loss, and delay [7].

The use of unicast data delivery for streaming services consumes a large amount of network bandwidth resources. This can be mitigated with the introduction of multicast streaming. Multicast communication allows to deliver information to multiple receivers while consuming less network bandwidth than unicast communication. In multicast communication, the network infrastructure is used to duplicate data whenever required. Thus, it is possible to send data to all the interested receivers in a single transmission.

The emerging Software Defined Networking (SDN) paradigm promises to provide better network resource management, traffic control, and application classification in order to design an efficient Quality-of-Service (QoS)-aware data routing mechanism [3]. SDN uses a centralized control plane which is separated from the data plane. The SDN controller's global visibility of the network contributes to a better information for the routing algorithms, which aids to build optimal data routing trees [5].

The work presented here describes an implementation of network-resources aware multicast data routing in a SDN network. In the prototype implementation, the SDN controller computes the best paths for multicast data flows that meet the QoS requirements of the traffic transmitting applications. This is done by using the information available to the SDN controller about the different network parameters such as network's topology, links capacity and load.

2 Related Work

Several different algorithms have been proposed to find an optimal routing for the multicast packets in SDN. A summary of the previous works focusing on an efficient routing of multicast data in SDN is listed below.

A proposal for a reliable multicast tree, Recover-aware Steiner Tree (RST) has been described in [11] for reliable multicast routing in SDN. The RST minimizes the tree and recovery costs. Here, the tree cost refers to the total cost of sending data along all edges of the tree. Finding the RST is NP-Hard. To solve the RST, a k-approximation algorithm, recover average edge reduction algorithm could be deployed in the SDN controller to minimize the tree and recovery costs [11].

A routing model for multicast data in SDN networks with segment routing has been proposed in [12] which ensures that the data is routed along feasible paths with service guarantees. The paper focuses on building a bandwidth-efficient multicast routing tree for the requests of the multicast group which minimizes the likelihood of rejecting the traffic requirements and increases the network throughput. It also proposes an algorithm which considers the residual bandwidth, node loading, link criticality, and scalability.

An SDN based multicast algorithm has been proposed in [5], that enables multicast in data center switches. The paper proposes the AvRA routing algorithm [5], which attempts to minimize the size of the routing tree of the multicast group for data center topologies. It tries to find the shortest path to the existing tree node rather than finding the shortest path from the multicast server to the receiver.

Research on implementation of load-balancing and multicast routing algorithms based on the extended Dijkstras algorithm for SDN is presented in [6]. In the proposal, clients send data to the Virtual IP address which is an IP address that does not correspond to an actual physical network interface. The requests are forwarded to one of several servers. Here the load balancing algorithm routes the request to the nearest server whose link-load is below the predefined threshold value. The multicast routing algorithm is based on the multicast tree construction algorithm which uses the extended Dijkstras algorithm for a multicast data routing.

3 Background

3.1 Multicast data delivery

Multicast is a group-based data distribution method. In IP networks, data transmitted to the multicast group address is delivered to all registered members of the group. Multicast groups are uniquely identified using class-D IP addresses [2]. In IP networks, multicast data delivery is implemented using two sets of protocols described in the following.

The first set of protocols is used by hosts to manage the membership of the groups. Internet Group Management Protocol [2] (IGMP) is used to manage the membership of the multicast groups in IPv4 networks and the Multicast Listener Discovery [13] (MLD) protocol is used to manage the membership of the multicast groups in IPv6 networks. When a host wants to join, query or leave a multicast group, it sends an IGMP or MLD message which is processed by the first-hop router on the sending host's network.

The second set of protocols is used to forward multicast data between the routers. Protocols in this group can implement full routing protocol functionality, like DVMRP, or MP-BGP [10]. Alternatively, multicast routing protocols can utilize information maintained by interior gateway protocol for multicast data delivery, like the family of Protocol Independent Multicast protocols [3, 10].

3.2 SDN and OpenFlow

Software Defined Networking (SDN) is a network paradigm allowing dynamic network management with by decoupling data and control planes. An SDN network consists of data-plane devices (switches), and a central network entity implementing the control-function - an SDN controller.

SDN data-plane devices (SDN switches) operate by matching every received packet to a set of rules maintained in the SDN switch, by the SDN controller. If the received packet does not match any of the rules, the packet is forwarded to the SDN controller for processing. Upon processing the packet, the controller can choose to install new rules in the SDN switches that will be used for the subsequent packets. Typically, SDN switches communicate with the controller using the OpenFlow protocol.

4 Solution Design and Implementation

Centralization of network functions in the SDN controller allows to significantly reduce the complexity of multicast data delivery. By performing multicast data routing in the SDN controller, the whole set of distributed protocols can be eliminated from the network. For example, consider the network shown in Fig. 1.A, consisting of 4 routers and 2 hosts. In this network, all routers run instances of a multicast routing protocol. Furthermore, routers connecting user hosts to the network must run protocols used to join and leave multicast groups.

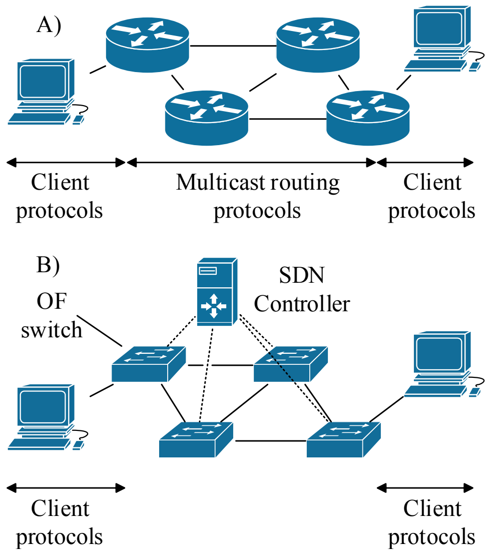

Fig. 1. Multicast implementation comparison between conventional and SDN-based networks.

The same network using OF switches and a centralized controller is shown in Fig. 1.B. In this network, OF switches forward all group joining or leaving messages received from the user hosts to the SDN controller for processing. The SDN controller processes these membership messages by updating the membership lists of the groups. It additionally creates and installs flow rules used to forward data to the multicast group members, as required.

In addition to reducing the protocol overhead in the network, using a centralized entity to manage the network allows to implement network resources-aware multicast routing. As the SDN controller has full and up-to-date information about the network topology and link speeds, this information can be used to improve routing decisions. Specifically, multicast groups in the proposed SDN network can be assigned with a reserved bandwidth value. These values later can be used to route multicast data flows in an optimal way. The exact method used to route multicast data flows with respect to available bandwidth is described in the next section.

Centralizing multicast data routing in a single network entity (SDN controller) requires special consideration regarding the controller's availability. In case of a controller failure, the described system would still be able to route multicast data. However, no new members would be able to join, and data-paths would not be reconfigured after network topology changes.

To increase the SDN controller's availability, multiple controllers can be run in parallel, utilizing different active/standby strategies[8]. The ONOS controller used in this work supports running in a cluster mode[1]. When multiple ONOS controllers run in a cluster mode, they maintain a single synchronized view of the network. Running multiple controllers in a cluster allows increasing the availability and OpenFlow messages processing capacity, compared to running a single controller. Furthermore, as all controllers work using a shared network view, failure of a single controller in a cluster does not affect the remaining controllers.

4.1 Dimensioning Multicast flows

Dimensioning of traffic flows refer to the process of installing flows in a way that bandwidth constraints (if any) are met. ONOS SDN controller is using Dijkstras Shortest Path First (SPF) algorithm to dimension flows multiple-destination flows. By definition, the SPF protocol finds paths having the lowest number of intermediate hops (OpenFlow (OF) switches). Such approach is not always suitable when dimensioning multicast data flows that have specific bandwidth requirements. In order to dimension flows with respect to the required and available bandwidth a different approach is needed.

One possible way to perform resources-aware dimensioning is to use a maximum flow algorithm. Given the topology of a network and its link capacities, a maximum flow algorithm returns a path from source to destination that has the maximum available capacity. Among the best known maximum flow algorithms are Ford-Fulkerson algorithm, Edmonds-Karp algorithm, King, Rao, Tarjan's (KRT) algorithm and other. This work uses a modified Edmonds-Karp (EK) algorithm to find the best path for multicast flows following the protocol described in [9]. The choice of the algorithm was motivated primarily by the faster algorithm run-time and the ready availability of software libraries implementing it.

The full algorithm to build the multicast data distribution tree is shown in listing 1 below, and works as follows. Given the source node and a list of receiver nodes, run the EK algorithm using the network's topology to find a maximum-flow from the source to each destination. Use the available bandwidth of each link in the network as a corresponding edge weight. In case EK algorithm produces more than one path between the source and destination nodes having identical maximum-flow, use a path with a lower hops count. To produce the final multicast tree, superimpose the paths produced by the EK algorithm. In the current implementation, it is done by appending a list with edges from all maximum-flow paths and removing duplicates.

In order for the algorithm to run successfully, it must have information about network topology and available link resources. ONOS controller tracks this information by using the *Topology Store* and *Resources Service*. Furthermore, the resources service was extended to track the allocated resources in addition to the overall capacity of the link. The execution of the algorithm was triggered by changes in topology or group membership events. The topology event is triggered when the SDN controller detects that new devices or links are added to the network. Similarly, a group change event is triggered when the IGMP message informing about change in group membership is processed by the SDN controller.

```
1  Function buildMcastTree(source, destinations[])
       Data: The source and a list of destination nodes/vertices
       Result: mcastTreeEdges - a list of multicast tree edges
2      foreach dest in destinations do
3          candidates ← EdmondKarps(source, dest, NetworkTopo, EdgeWeights);
4          if num(candidates) == 1 then
5              mcastTreeEdges←candidates;
6          else
7              mcastTreeEdges←min(candidates.hop_count);
8      end
9      RemoveDuplicates(mcastTreeEdges)
10 end
```

Algorithm 1: Resource-aware multicast tree building algorithm.

5 Performance Evaluation

In order to test the functioning and performance of the proposed system, a virtual testing environment was used having the following configuration. The network was implemented using Mininet network emulator. OF switches were implemented using Open vSwitch (v. 2.0.2) software. Open Network Operating System (ONOS) (v. 1.5) was used as a SDN controller. Multicast data delivery and reception were performed by streaming a video recording using VideoLAN (VLC) software.

5.1 Testing Strategy

The network topology shown in Fig. 2 was used for testing. The test network consisted of six OF switches (SW1-6), two streaming servers (SRV1-2), and six hosts used to receive the multicast video streams (H1-6). The topology and bandwidth of the links between the OF switches was chosen specifically to illustrate various features of the algorithm. The configuration of the multicast groups used in testing is shown in Table 1.

Table 1. Multicast group parameters used in testing.

Gr.	Group IP	Source	Clients	Bandwidth constrain
1	239.5.2.1	SRV1	H1, H2	2.5 Mbps
2	239.5.2.2	SRV2	H3, H4	2 Mbps
3	239.5.2.3	SRV1	H5, H6	2.5 Mbps

It is important to note, that tests described here were intended to show the correct functioning of the algorithm and hence used only a small number of nodes. Any wide-scale deployment would require further load testing with a higher number of devices. The performance of any wide-scale deployment would be affected by three factors: the rate of users arrival and departures, the frequency of network topology changes, and the number of controllers processing above-mentioned events. Users arrival and departure requires only a minor change in the forwarding table of the switch users are connected to. A special case, when all users connected to the same switch leaves, requires an extra flows pruning step in the upstream switch. The performance impact of network topology change and how topology change events can be processed in parallel are left for future work. However, such network topology change events should not occur frequently.

Implementation testing followed three different scenarios. In the first scenario, flows were created using the ONOS built-in point-to-multipoint intent framework utilizing the SPF algorithm. The second scenario repeated the test using the proposed multicast-routing algorithm. Finally, the third scenario utilized the proposed multicast routing algorithm in the same network. However, in the third scenario, two paths with equal number of hops were available to deliver multicast data to all receivers.

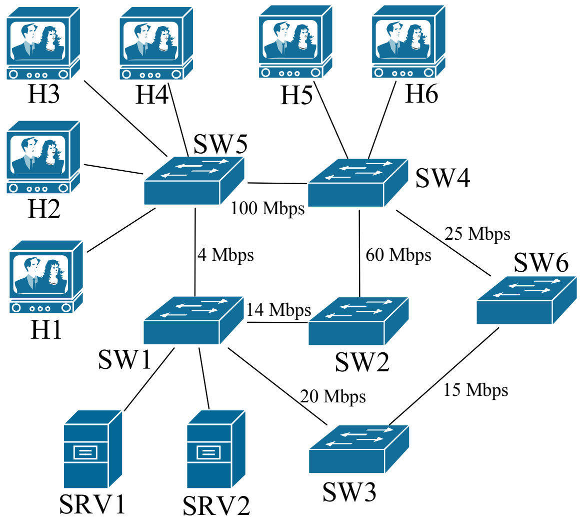

Fig. 2. Test network topology. Multimedia streaming was performed from servers SRV1 and SRV2 to clients H1-H6. Numbers next to the links indicate the link's bandwidth.

5.2 Flow Selection Results

Scenario 1. The first test used ONOS built-in path creation method using the SPF algorithm. After both clients (H1 and H2) joined the multicast group, the resulting data path for the group was: `SRV1-SW1-SW5-{H1, H2}`. Subsequently, the remaining two clients (H3 and H4) joined the second multicast group. The resulting data path was: `SRV2-SW1-SW5-{H3, H4}`. As expected, the SPF algorithm ignored the link capacities constrain and routed both pats via the smallest number of intermediate switch hops.

Scenario 2. The second test used a modified EK algorithm to set-up the multicast data paths. After members of the first group joined the multicast group, the resulting data path was: `SRV1-SW1-SW5-{H1, H2}`. Subsequently, clients H3 and H4 joined the second group. The path created for the multicast data was: `SRV2-SW1-SW2-SW4-SW5-{H3, H4}`. Here, it can be seen that the link `SW1-SW5` was not used by the second data path. This was expected, as the available bandwidth after provisioning the first group was 1.5 Mbps. Furthermore, the algorithm did not select the `SW1-SW3-SW6-SW4` path, even though it had higher available bandwidth (15 Mbps). This shows, that the algorithm ranks all the feasible paths and selects the one with the lowest number of intermediate hops.

Scenario 3. The third test used the modified EK algorithm as in the second test. In this test, clients H5 and H6 joined the third multicast group. Two paths, each having 3 hops, were available for the controller: `SW1-SW5-SW4` with 4 Mbps available, and `SW1-SW2-SW4` with 14 Mbps available. As the second path had a higher available bandwidth, it was the chosen path by the controller.

5.3 Flow Setup Time

The replacement of the path setup algorithm can increase the complexity and the run-time of the algorithm. Hence, the average flow setup time using the SPF and modified EK algorithms was measured using the following procedure. The ONOS controller creates flows in a reactive way: when a packet arrives at an OF switch and there is no flow matching that the packet, the packet is sent to the controller. After inspecting the packet, the controller setups the required path and resends the packet.

The flow setup time was measured by observing the time difference between the time when the first packet in a flow was sent and received by the destination host. As all virtual hosts performing the test were running on the same physical server, the timestamps could be compared directly. Table 2 shows the average flow setup times in the test network observed over 10 tests.

Table 2. Flow setup times

Num.	Algorithm	Flow Setup Time, ms
1	Shortest Path First	16.0
2	Modified Edmonds-Karp	28.9

As expected, due to the higher algorithm complexity, the flow setup time using the modified EK algorithm is longer than using the SPF algorithm.

6 Conclusions

Implementing multicast data communication with resource-aware data routing is not a trivial task. This paper presents a prototype implementation of resources-aware data routing in SDN networks by using a modified Edmonds-Karp algorithm. By implementing multicast routing algorithm in the SDN controller, routing decisions can take into account up-to-date information about network topology and links load. This allows reducing the protocol overhead in the network and route data using the optimal data-path. The implemented algorithm has worse run-time complexity compared to the default routing algorithm (Shortest Path first). However, a longer run-time allows the algorithm to perform resources-aware routing - a key requirement for high quality streaming multimedia delivery.

References

1. Introducing ONOS - a SDN network operating system for service providers (2014). URL http://onosproject.org/wp-content/uploads/2014/11/Whitepaper-ONOS-final.pdf. Accessed: 2017-09-01
2. Deering, S.: Host extensions for IP multicasting. RFC 1112 (Internet Standard) (1989). DOI 10.17487/RFC1112. URL https://www.rfc-editor.org/rfc/rfc1112.txt. Updated by RFC 2236
3. Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z., Zheng, L.: Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised). RFC 7761 (Internet Standard) (2016). DOI 10.17487/RFC7761. URL https://www.rfc-editor.org/rfc/rfc7761.txt
4. Huang, L., Zhi, X., Gao, Q., Kausar, S., Zheng, S.: Design and implementation of multicast routing system over SDN and sFlow. In: Communication Software and Networks (ICCSN), 2016 8th IEEE International Conference on, pp. 524–529. IEEE (2016)
5. Iyer, A., Kumar, P., Mann, V.: Avalanche: Data center multicast using software defined networking. In: Communication Systems and Networks (COMSNETS), 2014 Sixth International Conference on, pp. 1–8. IEEE (2014)
6. Jiang, J.R., Yahya, W., Ananta, M.T.: Load balancing and multicasting using the extended dijkstra's algorithm in software defined networking. In: ICS, pp. 2123–2132 (2014)
7. Kenyon, T.: Data networks: routing, security, and performance optimization. Digital Press (2002)
8. Lee, B., Park, S.H., Shin, J., Yang, S.: Iris: the openflow-based recursive sdn controller. In: Advanced Communication Technology (ICACT), 2014 16th International Conference on, pp. 1227–1231. IEEE (2014)
9. Mallick, K.K., Khan, A.R., Ahmed, M.M., Arefin, M.S., Uddin, M.S.: Modified EDMONDS-KARP algorithm to solve maximum flow problems. Open Journal of Applied Science **6**, 131–140 (2016)
10. Savola, P.: Overview of the Internet Multicast Routing Architecture. RFC 5110 (Informational) (2008). DOI 10.17487/RFC5110. URL https://www.rfc-editor.org/rfc/rfc5110.txt

11. Shen, S.H., Huang, L.H., Yang, D.N., Chen, W.T.: Reliable multicast routing for software-defined networks. In: Computer Communications (INFOCOM), 2015 IEEE Conference on, pp. 181–189. IEEE (2015)
12. Sheu, J.P., Chen, Y.C.: A scalable and bandwidth-efficient multicast algorithm based on segment routing in software-defined networking. In: Communications (ICC), 2017 IEEE International Conference on, pp. 1–6. IEEE (2017)
13. Vida, R., Costa, L.: Multicast Listener Discovery Version 2 (MLDv2) for IPv6. RFC 3810 (Proposed Standard) (2004). DOI 10.17487/RFC3810. URL https://www.rfc-editor.org/rfc/rfc3810.txt. Updated by RFC 4604