Resource and Virtual Function Status Monitoring in Network Function Virtualization Environment

To cite this article: MS Hai et al 2018 J. Phys.: Conf. Ser. 1087 052037

View the article online for updates and enhancements.
Resource and Virtual Function Status Monitoring in Network Function Virtualization Environment

MS Hai, P Yi, YM Jiang
National Digital Switching System Engineering & Technological Research Center, Zhengzhou, China
33xywz@163.com

Abstract: In Network Function Virtualization, in order to improve the utilization of the underlying resources and effectively deploy the service chain dynamically, we will use the management and orchestration to monitor the network resources and virtual function status in real time, but real-time monitoring will result in a certain communication overhead. In this paper, a resource state agent monitoring strategy is designed. The improved label propagation algorithm divides network to subnet and selects the agent monitoring node, which realizes the real-time monitoring of resource status and minimizes the monitoring information communication overhead. Simulation results show that the monitoring strategy proposed in this paper reduces the overhead of monitoring information communication about 15%.

1. Introduction
In traditional networks, operators use a large number of dedicated hardware devices to provide network services for users, which need huge investment costs and maintenance costs, and the traditional network configuration is complex. In order to solve these problems, the industry proposed the concept of Network Function Virtualization (NFV) [1]. In a network function virtualization environment, if there is a service request in the network, the management and orchestration deploys the service chain[2] for the request, as service requests in the network increase with time, the service traffic of each service chain will also fluctuate constantly, and the status of virtual functions will also change continuously[3]. In order to complete service chain deployment according to user service requests in a timely manner and provide users with high-performance services, the management and orchestration requires real-time monitoring of the underlying physical resources[4], virtual resources, and network function status.

Gardikis[5] proposed a monitoring architecture to monitor the status of physical resources, virtual resources, and network functions in the network through an integration framework. However, there is a large time delay in the monitoring information of the integrated environment. Pfitscher[6] proposed a strategy for each node to monitor the state of its neighboring nodes. It can monitor resource status information in real time, but it would generate a large amount of communication overhead. Cao[7] identified performance bottlenecks by monitoring hardware resource utilization and other specific indicators, but only hardware resources are monitored and network function status monitoring is not considered; Naik[8] designed the agent monitoring strategy, but its agent monitoring node needs to be deployed in advance, and cannot be dynamically adjusted according to the underlying network changes after deployment.

In the network function virtualization environment, the management needs to monitor the resources...
and statuses in the network in real time, and the real-time monitoring will consume a large amount of link bandwidth resources. In order to improve the real-time monitoring and reduce the communication overhead, this paper proposes an agent node monitoring strategy.

2. System model and problem description
The underlying physical network is constructed as an undirected graph \( G = (V; E) \), where \( V = \{v_1, v_2, \ldots, v_n\} \) is the set of physical nodes in the network topology, \( n \) is the number of physical nodes, and \( E \) represents the set of links in the network. Assume that the underlying physical network is divided into \( K \) sub-networks, denoted by \( S = (S_1, S_2, \ldots, S_K) \), and there are no duplicate nodes between different sub-networks. We select one agent monitoring node in each subnet, and make other nodes in the subnet send the resource information and virtual function status to the agent monitoring node. The proxy node aggregates the data and reports it to the management and orchestration.

Considering that the actual network topology has community attributes[9], the network can be divided into sub-networks based on the correlation between the nodes. Therefore, the network nodes in the same sub-network are closely connected, and the correlation is large. The transmission of monitoring information can reduce the delay and link bandwidth. The label propagation algorithm can divide the network into multiple sub-networks according to the network community structure. However, the sub-networks will have large gaps, the difference in sub-network size will cause load unbalanced. Because the influence and importance of each node in the network are different, we can divide the subnet according to the importance of the node and the load of the monitoring data in the subnet. First, we divide network according to the degree of importance of the nodes[10] and the load of the sub-networks so that the sub-networks are commensurate in size. Second, the sub-network agent monitoring nodes are searched for by minimizing the communication overhead.

2.1. Dividing Subnet
The label propagation algorithm updates the label information between the target node and the neighbor nodes according to the relationship between the nodes in the network. After repeated iterations, the label area is stable, and finally the sub-network is divided according to the node labels. The improved label propagation algorithm proposed in this paper constrains the subnet allocation according to the importance of the nodes and the subnet load to minimize the network communication overhead.

Define the node density function \( \rho \) to represent the number of neighbor nodes. Define node connectivity \( \text{Connection} \), if node \( v_a \) and node \( v_b \) belong to the same subnet then \( \text{Connection} \) equals 1, otherwise 0. Define the sub-network load-balancing index \( \text{LoadBalance} \), which is the maximum difference in the number of nodes between different subnets. As shown in the formula (1), \( |S_i| \) represents the number of network nodes in sub-network \( S_i \).

\[
\text{LoadBalance} = \max(|S_i| - |S_j|) \quad \forall i, j \in k
\]  

First, we initialize the network and set different initial label propagation numbers \( C^0(V) = (C^0(v_1), C^0(v_2), \ldots C^0(v_n)) \) for each node in the network, then according to the node density function, sort the nodes in descending order, update the node labels and control the label propagation capability according to the importance of the nodes. Meanwhile, We use load constraints between subnets to control subnet size. Let \( C^m(V) = (C^m(v_1), C^m(v_2), \ldots C^m(v_n)) \) be the label of each node in the network after \( m \) times of label propagation and update.

2.2. Set up the proxy monitoring node
In the process of monitoring information, the communication overhead mainly includes bandwidth and delay. Assuming that each node has the same monitoring information bandwidth, the communication overhead is mainly the link transmission delay, and the communication overhead of transmission
monitoring information between nodes is the shortest delay between nodes, as shown in equation (2).

\[
Overhead(v_a, v_b) = \min \text{PathDelay}(v_a, v_b)
\]  

(2)

Assuming that node \( v_i \) is the agent node in the subnet \( S \), the communication overhead for transmitting monitoring information is shown in equation (3).

\[
Overhead_i = \sum_{j=1}^{n} Overhead(v_i, v_j) + Overhead_{Orchestration}
\]  

(3)

For large-scale NFV networks, the resource monitoring optimization objective function is shown in Equation (4).

\[
\min \sum_{i=1}^{k} Overhead_i
\]  

(4)

The resource monitoring in the NFV network also needs to satisfy the following constraints:

\[
\text{LoadBalance} \leq \text{StandardLoad}
\]  

(5)

\[
\sum \text{Connection}(v_a, v_{\text{else}}) = 1 \quad \forall v_a \in V
\]  

(6)

Equation (5) is a constraint condition for load balancing across subnets, and StandardLoad is the upper limit of load balancing set by the network operator. Equation (6) ensures that non-agent nodes in the network can transmit monitoring information to the agent node and can only transmit to one agent node.

3. Algorithm solving

The label propagated algorithm proposed in [11] can reasonably divide the network into multiple subnets in a linear time with less computation and storage resources according to the community structure characteristics existing within the network. However, the difference in the size of the subnets divided by the LPA is large, resulting in unbalanced load. For the resource monitoring large-scale NFV networks and reduce communication overhead, an improved LPA Subnet Agent Monitor (ILSAM) is designed in this paper. It is mainly divided into two steps: the first step is to use an improved label propagation algorithm to divide large-scale network into subnets showed in table 1.

| Table 1. Subnet dividing algorithm |
|-----------------------------------|
| **Algorithm Function**            |
| **Divide-Subnets**                |
| Set standard of load balance      |
| StandardLoad                      |
| \( C^0(V) = (C^0(v_1), C^0(v_2), \cdots C^0(v_n)) \) |
| for each \( v_j \) do             |
| calculate \( \rho(v_j) \)         |
| end for;                          |
| \( V' = \text{Sort}(\rho(v_j)) \) |
| Set \( t=0 \);                    |
| while not stop                    |
| \( t++ \);                        |
| for each \( v_m \) in \( V' \) do |
| find \( v_m \) adjacent nodes     |
| for each adjacent node \( v_a \) in \( V' \) |
| if \( \text{LoadBalance} \geq \text{StandardLoad} \) |
| break;                            |
| end if;                           |
| end for;                          |
| update \( C(v_m) \);              |

3
In the sub-network division step, nodes are sorted in descending order according to the importance of each node in the network. In the process of updating labels, the labels are updated in order according to their importance (line 9). If the number of multiple labels in the neighboring nodes is the same, the labels are selected according to the importance of the nodes (line 11), and the size of each subnet network is controlled (line 12). If the node label is no longer updated, the dividing algorithm is completed (line 19).

The second step is to deploy the agent monitoring node in each sub-network to minimize the communication overhead of resource monitoring in the network, the detailed algorithm is shown in table 2.

| Table 2. Agent node deployment algorithm |
|------------------------------------------|
| Algorithm Function Deploy Agent Node     |
| Create N node                           |
| Input $S = (S_1, S_2, \ldots, S_k)$     |
| 1  for each $S_i$ do                    |
| 2    for each $v_j$ do                  |
| 3     $\sum Overhead(v_j, v_{v_{\text{new}}}) + Overhead_{\text{local}} \forall v_{v_{\text{new}}} \in S_i$ |
| 4     end for;                          |
| 5     find $v_{\text{agent}}$ make Overhead minimize; |
| 6     end for;                          |
| 7     return $v_{\text{agent}}$         |

4. Experiments and results

4.1. Simulation environment
In order to verify the performance of the strategy, this article uses the GT-ITM tool to generate the underlying network topology and simulates it using MATLAB software. The simulation computer is Windows 7 operating system configured with 3.4GHz Intel Core i7-4790 processor, 8GB of memory, and 1 Gbps network interface. In the experiment, it is assumed that the amount of monitoring information transmitted by each underlying node is the same, and the nodes do not affect each other. The difference in the number of nodes in each sub-network does not exceed 3, and the delay of the underlying link network is uniformly distributed in the range of [1, 30].

4.2. Simulation results and performance analysis
In order to verify the performance of the ILSAM algorithm, we compare it with the distributed resource monitoring algorithms DRM, NFVPerf and LSAM.

In the ILSAM algorithm, the proxy node is set according to the network community structure and the importance of the node. For different network sizes and topologies, the algorithm automatically adjusts the number and location of the proxy nodes. The DRM and NFVPerf algorithm's proxy nodes are deployed in advance and cannot adjust the number of agent nodes according to the dynamic
changes of the underlying network. Figure 1 shows the change of the proxy node of ILSAM algorithm under different network scales.

![Figure 1. Number of monitoring agents under different network sizes](image1.jpg)

Then, the number of monitoring agents under different algorithms is simulated. Figure 2 shows the variance maps of the number of the underlying nodes monitored by each agent node in different algorithms. From the simulation results, we can see that the NFVperf algorithm has the smallest difference in the number of nodes in each subnet, followed by the ILSAM algorithm, and the difference in the number of subnet nodes in the LSAM is the largest. Since the NFVperf algorithm sets the number of agent nodes in advance for the static underlying network, the number of agent node monitoring is relatively balanced. LSAM sets up the agent monitoring node according to the network community structure completely. Therefore, the difference between the number of sub-network nodes is large. The ILSAM algorithm balances the monitoring of the agent node overhead and updates the label according to the importance of the node, so its subnet size more balanced.

![Figure 2. The variance of the monitoring number of agent nodes under different algorithms](image2.jpg)

Figure 3 shows the communication overhead of monitoring information. We can see that the communication overhead of the ILSAM algorithm is the smallest, followed by the LSAM and NFVperf algorithms. NFVperf does not dynamically adjust the proxy node mechanism, resulting in a large communication overhead; LSAM considering only the community structure of the nodes, the subnet size varies greatly, leading to a large communication overhead. ILSAM improves the LSAM, combines the importance of the nodes, and restricts the size of the subnet so that communication overhead is minimized. ILSAM communication overhead is reduced by approximately 15% compared to sub-optimal strategies.
5. Conclusion and future work

This paper aims at the problem of real-time monitoring of resource status in network function Virtualization environment, designs the agent monitoring strategy. Monitoring strategy proposed in this paper reduces the communication overhead of monitoring information on the basis of real-time monitoring, and experimental results also show that the performance is better than other strategies. However, when considering the communication overhead, this paper assumes that the monitoring information of each node is the same and does not consider the differences between nodes. The next major work will further consider the impact of node information on the deployment of agent monitoring nodes.

References

[1] Gray K, Nadeau T D. Network Function Virtualization[M]. 2016.
[2] Liu J, Lu W, Zhou F, et al. On Dynamic Service Function Chain Deployment and Readjustment[J]. IEEE Transactions on Network & Service Management, 2017, PP(99):1-1.
[3] Racheg W, Ghrada N, Zhani M F. Profit-driven resource provisioning in NFV-based environments[C]// IEEE International Conference on Communications. IEEE, 2017.
[4] Xu Z, Liang W, Galis A, et al. Throughput maximization and resource optimization in NFV-enabled networks[C]// IEEE International Conference on Communications. IEEE, 2017:1-7.
[5] Gardikis G, Koutras I, Mavroudis G, et al. An integrating framework for efficient NFV monitoring[C]// Netsoft Conference and Workshops. IEEE, 2016.
[6] Pfitscher R J, Scheid E J, Santos R L D, et al. DREAM - a distributed result-aware monitor for Network Functions Virtualization[C]// Computers and Communication. IEEE, 2016:663-668.
[7] Cao L, Sharma P, Fahmy S, et al. NFV-VITAL: A framework for characterizing the performance of virtual network functions[C]// Network Function Virtualization and Software Defined Network. IEEE, 2016:93-99.
[8] Naik P, Shaw D K, Vutukuru M. NFVPerf: Online performance monitoring and bottleneck detection for NFV[C]// Network Function Virtualization and Software Defined Networks. IEEE, 2017:154-160.
[9] Jiguang, WANG, Yuqing, et al. REMARKS ON NETWORK COMMUNITY PROPERTIES[J]. Journal of Systems Science & Complexity, 2008, 21(4):637-644.
[10] Shive M S, Anderson J M. Biodegradation and biocompatibility of PLA and PLGA microspheres.[J]. Advanced Drug Delivery Reviews, 2012, 64(1):72-82.
[11] Raghavan U N, Albert R, Kumara S. Near linear time algorithm to detect community structures in large-scale networks[J]. Physical Review E Statistical Nonlinear & Soft Matter Physics, 2007, 76(3 Pt 2):036106.