Journal of Practical Information Technology

J. Practical IT, 2(3): 11-16, 2021

Research Paper

Efficient Service Discovery Using NDN In Microservice Architecture

Maryam Eslahi [a] , Dr Reza Azmi [b]

[a] Computer Engineering Department Alzahra University Tehran, Iran
[b] Computer Engineering Department Alzahra University Tehran, Iran

Received: 24 April 2021 Revised: 26 May 2021 Accepted: 27 June 2021

Abstract
The Micro-service architecture (MSA) has become a dominant architectural style choice in the service oriented software industry. This architectural style enables better application performance, flexibility, and ease of deployment. In MSA, services need to know the location of other services to communicate. Addressing this issue needs a process called service discovery. Most of current solutions, discover service base on their IP address. But continuous location changes of services make service discovery a challenge in this regard. In this paper, information centric networking concept is used as an alternative for current IPbased service discovery and powerful forwarding strategy has been used for better performance response.

Keywords: Microservice Architecture; Service Discovery; Named Data Networking; ICN.

How to cite the article:
M. Eslahi, R. Azmi, Efficient Service Discovery Using NDN In Microservice Architecture, J. Practical IT, 2021; 2(3): 11-16,

Introduction

Simplicity in developing, testing and scaling of services has made Monolithic architecture the most commonly used style. But, when applications become more complex and large over the time, building and agile delivery are getting more difficult. To solve these problems, Microservice Architecture (MSA) was introduced. Flexibility, modularity, and scalability of MSA attracted the attention of researchers. MSA is known in [1] as a way software is divided into small running instances in which each instance can dynamically and independently deploy, run, fail, scale or refused.
Every instance runs as a unique process and the locations may change dynamically. Clients send their request to service discovery tools and Service discovery is expected to find the services with the most match with requested one. Current service discovery tools, address this problem by fixing the addresses to the services, but this is in conflict with the nature of moving services in MSA. They store the IP and Port number of the services in a key/value store. Thus, to find the location of a service, service discovery will send a request to key/value store and retrieve the information about the service.

However, central solutions have limitations in scalability, maintainability, and dynamicity since their operation are depend mainly on the cluster designation [2]. Dynamic changes that occur continuously in microservice based application, make it hard to use centeral solutions. Other problems can also be encountered when faced with large scale applications. There are a variety of approaches for ICN architectures [4], [5], [6], [7], [8]. Among these approaches, we consider Named Data Networking (NDN) [8]. In NDN, data is requested by its name and the request is sent to other NDN nodes as an Interest. The Interest is forwarded node by node until it reaches the origin content or a replica of that in an intermediate node. The requested content is returned in response. NDN routers consist of three main modules: *i*) Content
Store (CS): a place in NDN router that memorize data packets which forwarded by the router. ii) Pending Interest Table (PIT): register Interest forwarding information either content name or the reciprocal interfaces. iii) Forwarding Information Base (FIB): point out route information of the router. Although each Interest has a NONCE (unique bit pattern) to prevent looping Interest because NONCE can help to detect which Interest

is duplicated. Comparing the name and NONCE of received Interest with Interests that already exist in the PIT, can determine if the Interest is duplicated or not. IP networks have some typical issues, such as short-term link failure and congestion, and most of that issues occur because IP is not stateful. On the other side, NDN network is stateful and the researches of Yi et al. [9] has shown that NDN can resolve these problems more effectively than IP networks.

Furthermore, in [10] Yi et al. Routing should only provide a reasonable starting point for the forwarding plane, which then should explore different multipath opportunities. In return, adaptive forwarding enables a more scalable routing plane with not much information about the routing.

There some other adaptive forwarding strategies, but none of them can provide all mechanisms foreseen in [9] and [10]. For this reason Posch et al. propose Stochastic Adaptive Forwarding (SAF), a probabilitybased forwarding strategy [11]. Performing Stochastic adaptive forwarding on a per prefix basis, providing effective forwarding with lowest knowledge about routing information, ability to deal with unexpected network topology changes, e.g. link failures and discovering unknown paths to cached replicas are the main objectives that SAF is designed to meet.

I. RELATED WORK

A. Available tools for service discovery

Etcd [12] is a distributed key value store that provides a reliable way to store data across a cluster of machines. etcd gracefully handles leader elections during network partitions and will tolerate machine failure, including the leader. Applications can read and write data into etcd. Etcd is just a key-value store by itself. But some modules can be added to help in registering services in etcd and also some modules to help in reconfiguring the application after any changes accrue.

Consul [13] is a distributed, highly available system. Every node that provides services to Consul runs a *Consul agent*. The agent is used for checking the services and nodes if they are healthy or not. Also, node can communicate with consul server with the help of its agent.

As mentioned, all of above methods use a key-value store. And this key-value store is implemented in a cluster. Beside the benefits that clustering bring with itself, it may cause complexity in the system. Also the number of servers that maintain cluster's activities should be considered. Because without a sufficient number of cluster nodes, cluster operation can be effected and data may be lost. Moreover, when the number of change events increases, advertising these events on every node to synchronize the state of the service, will be increase too. So network overload may be occur.

B. Information-Centric Networking

There are two important aspects in Named Data Networking. The first one is the way contents are being named. Naming Strategy is one of the research subjects today. NDN routing table size may be much bigger than IPs, since the name space is infinite in NDN routing tables. So an efficient name strategy is needed to reduce the size of table to help in searching for requested prefix. The second aspect is Forwarding Strategy. In contrast to the IPs that just have one next hop address per prefix, NDN has multiple face choices for sending an interest with different performance costs. Therefore a forwarding strategy is needed to determine which face is the best next hop to choose.

For the first time, Bau Long and et al. [2] used the concept of Named Data Networking for service discovery. They model the NDN concept in term of MSA since their Information-centric Service Discovery (ISD) mechanism relies on the service names rather than their address to provide service discovery. In ISD, the function of the service is concerned for naming the services, therefore the services with the same functionality, have the same ISD name too. Same ISD name for similar service instances reduce the count of service records in registry compared with central registries that have to store whole services and their backup instances.

In ISD, names are hierarchical and could be similar to URLs. For instance, the ISD name of a service A could be ssu/dcn/μA. Moreover, a service provider can advertise its function that could be satisfied by propagating specific names. Multiple instances for the same name have instance level unique ID. For example, the name ssu/dcn/μA in Fig. 1 is specified by ssu/dcn/μA/_a and ssu/dcn/μA/_b respectively.

BestRoute or ShortestRoute is the Forwarding Strategy that has been used in ISD. In this strategy, Interest is sent to the face with most lowest cost among other faces. Each NDN nodes maintain a FIB that maintains [*name, μCost, NextHop]* structures. The NDN node looks FIB to find longest-prefix match to decide toward which node is better to forward the Interest. When the Interest messages arrive, based on name records in the FIBs, the NDN nodes forward the Interest messages hop-by-hop to the μSLB which provides the required name. The μSLB chooses an appreciate instance and send its location in the form of Response messages back to client applications. When a service instance changes, the central registry

updates the new service records that client application use to communicate with these instances. But in ISD, the new service records are advertised only if the new ISD names are generated. So the number of service records that are needed to propagated over the network could be decreased significantly, compared to centralized solutions. Therefor ISD reduce the network overhead in both communication bandwidth and latency for every service records.
There are some other Forwarding Strategies in NDN, such as:

- Boradcast: Interest are forwarded to all faces supplied by the FIB.
- NCC: faces with lowest delay are chosen to forward Interests.
- iNRR: the strategy tries to send Interests to the nearest face to retrieve data. In this case, nearby caches are preferred than origin content.
- RFA: the forwarding probability of a face is determined by a weight, which is actually a moving average over the reciprocal count of the PIT.

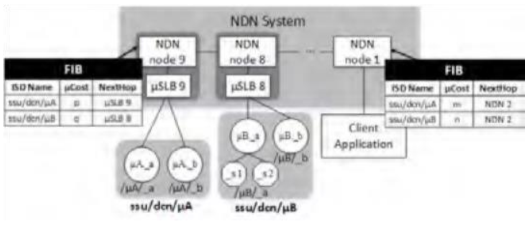

Figure 1: Architecture of ISD

Comparing these strategies with SAF has been shown that in many cases SAF stays in the first level of performance and is the best choice for some architecture with dynamic changes and large scale applications.

II. EFFICIENT SERVICE DISCOVERY USING SAF

In Efficient Service Discovery (ESD) we name a service by its function. When an interest is forwarded to a NDN node, service name which is based on its function, is searched in Content Store of the node. If a service with that prefix exists, a Data packet consists of the address of service, will return back. If there is no service with that prefix, the node will look for the service name in Forwarding Table (FWT) and check for faces with prefixes that match with the service name. If more than one face is found, the node needs a forwarding strategy to determine the forwarding probabilities for each next-hop.
In this section we will describe the design of SAF and how we used this Strategy for service discovery. SAF's object is to maximize the Interest satisfaction ratio and try to achieve to this goal by some updating operations in forwarding probabilities. Finally SAF tries to update this probabilities to achieve an optimal forwarding behavior based on $\mathcal{M}^T$.

A. Definitions and Terminology of SAF

In SAF network model, each node. Also, $\varepsilon \subseteq \nu \times \nu$ for each edge/link $F(\nu, u)$ maintains a physical face ν each node has a virtual face F_{D_ν} which works as overpressure valve. Therefore the set $F_\nu = F'_\nu \cup \{F_{D_\nu}\}$ denotes the entire set of faces known to. In ESD, we concern is determined by a set of content catalogue in. The the name of services as contents. Every node maintains a two-dimensional matrix as Forwarding Table (FWT), where the rows correspond to the set of faces F and the columns correspond to the different contents from the catalogue C. $p(F_i, c_l)$ is denoted as the forwarding probability that an Interest asking for c_l will be forwarded on F_i. As already mentioned, SAF is based on a given measure $\mathcal{M}^T$. This measure is not predefined in SAF, but in this paper, throughput over link failure is considered as $\mathcal{M}^T$. Considering this assumption, $S_{F_i}(A)$ provides a measure for satisfying and $U_{F_i}(A)$ defines a measure for not satisfying Interest. SAF finds the optimal forwarding strategy by maximizing $\mathcal{M}$ over the periods.

B. Use Adaptive Forwarding Strategy of SAF

As mentioned before, SAF is an adaptive forwarding strategy. In fact, the design of FWT and the ability to make changes in the forwarding probabilities is the key of adaptation. Whenever a link failure occurs or a congestion traffic happens, the probabilities in FWT will change and this modification affects the paths chosen by Interests.
The pseudocode of updating operations in SAF is provided in fig 2. In the first step, unsatisfied traffic that has been forwarded on the wrong faces and should be forwarded to other faces during the next period, is determined and denoted as Δ . In this step, the set of physical faces are divided into two disjoint subsets:

```
$\Delta \leftarrow$ determineUnsatisfiedTraffic()
$\Gamma \leftarrow \Delta + p(F_D)$
if $\Gamma > 0$ then
    $\mathcal{F_S}, \mathcal{F_P} \leftarrow$ splitSet ($\mathcal{F_R}$)
    shiftTraffic($\mathcal{F_U}, \mathcal{F_S}, \Gamma$)
    $p(F_D) = \Gamma - \Gamma'$
    if $p(F_D) > 0$ then
        probeOnFaces ($\mathcal{F_P}$)
        if $p(F_D) > (1 - t)$ then
            decreaseReliability(t)
        end if
    end if
else if $I > 0$ then
    increaseReliability(t)
end if
```

Figure 2: pseudocode for the FWT updates in SA

(i) F_R, the set of reliable faces, and (ii) F_u, the set of unreliable faces. The traffic will be shift from the unreliable faces towards the reliable faces. After the amount of Δ has been reached, we also measure the amount of Γ. Γ denotes the sum of unsatisfied traffic Δ and current forwarding probability $P(F_D)$ of the virtual face F_D. It's important to consider the sum of Δ and $P(F_D)$. Because Δ does not consider the discarded traffic on F_D.

The update operations is done if Γ > 0. Because if Γ = 0, the FWT needs no changes, and that's because the is no unsatisfied traffic or dropped traffic. So in the case of Γ > 0, the set of F_R is broken into two subset: i) F_s, faces that have successfully forwarded Interest in current period and ii) F_p, faces that are considered as reliable only because they have not forwarded any Interests in the current period. After that division, the unsatisfied traffic from F_u is forwarded towards F_s. Since the reliability should not decrease below t, the amount of additional traffic that is shifted into F_s should be determined first. This amount of traffic is denoted as Γ′.

In the next step, SAF updates the forwarding probabilities by increasing the forwarding probabilities in F_s. and decreasing the forwarding probabilities in, F_u by Γ′.

Remaining addition traffic that is more that F_s. acceptance, $(\Gamma - \Gamma')$, is forwarded to $P(F_D)$. This traffic is used for probing to identify unknown paths of F_p faces.

In the final step, the amount of $P(F_D)$ is compared with $(1 - t)$. If $P(F_D) > (1 - t)$, the reliability threshold should be decreased, because it cannot be retrained. In contrast, the reliability threshold should be increased, if currently all Interests can be satisfied with a reliability of at least t.

```
$\Delta \leftarrow$ determineUnsatisfiedTraffic()
$\Gamma \leftarrow \Delta + p(F_D)$
if $\Gamma > 0$ then
    $\mathcal{F_S}, \mathcal{F_P} \leftarrow$ splitSet ($\mathcal{F_R}$)
    shiftTraffic($\mathcal{F_U}, \mathcal{F_S}, \Gamma$)
    $p(F_D) = \Gamma - \Gamma'$
    if $p(F_D) > 0$ then
        probeOnFaces ($\mathcal{F_P}$)
        if $p(F_D) > (1 - t)$ then
            decreaseReliability(t)
        end if
    end if
else if $I > 0$ then
    increaseReliability(t)
end if
```

III. EVALUATION

A. Comparing the performance of SAF and other forwarding strategies over link failure

One of the features that makes SAF, a high performance strategy, is its ability to smartly circumvent congested nodes and link failures by taking detours into account. A test has been conducted in [11] to compare the impact of the link failure on different strategies.

Despite the significant impact of link failure in the performance of SAF, it has still maintain itself at a high level of efficiency, relative to rival strategies. Table I depicts the results of experiment in [11] on Data retrieval over link failure in different forwarding strategies in named data networking. Different NDN forwarding strategies have been tested in two phases. (i)without link failure and (ii)with link failure. However, the results show that SAF is more affected by link failure than other strategies, but still has the best performance.

In the next part, we will compare this strategy with the IP-based model. In comparing IP-based service discovery with NDN-based service discovery, we examine data retrieval time on link failures.

B. Comparing the performance of IP-based and NDN-based service discovery over link failure

The design of our NDN network consists of ten Docker 1.13 machines, each one has a Named data Forwarding Daemon (NFD) installed. And every

Docker machine has multiple running services. Service are registered by their name in FIB table of each NDN node. Requests for a service in each machine is forwarded to other machines with the help of NFD. At the end, request is responded by the origin data at destination or a replica of that in the intermediate nodes. The IP of the node which has the requested service, is returned on the way back.

On the other side, the network design of IP-based service discovery consists of ten docker nodes. Docker has some service discovery backends like consul, etcd and zookeeper. In this examination we used consul as service discovery tool. All the nodes are parts of a docker swarm cluster. The first node is swarm manager which consists of consul manager too. Nine other nodes are swarm worker and each of them has a consul agent which is responsible for communication with consul server. Every update in a cluster most be reported to consul server by consul agents. To register a new service, consul agent send the IP and port number of service to the consul agent. And whenever a node needs information about a service on other nodes, sends the request to consul server by the means of consul agents. Fig. 3 shows the design of network, using Consul for service discovery.

To examine the impact of link failure on the performance of these two methods, we test our networks with 1% probability of link failure and 10% probability of kink failure. Fig. 4 and Fig. 5 depict the results of our test.

TABLE I: DATA RETRIEVAL OVER LINK FAILURE IN NDN FORWARDUNG STRATEGIES

strategy	**Data retrieval**	
	Without link failure	***With link failure***
strategy	**Data retrieval**	
	Without link failure	***With link failure***
Broadcast	0.6	0.4
NCC	0.65	0.45
Shortest Route	0.55	0.4
RFA	0.45	0.3
iNRR	0.75	0.57
SAF	0.9	0.65

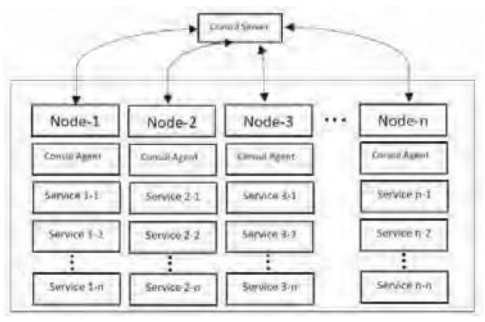

Figure 3: Network design for service discovery using consul

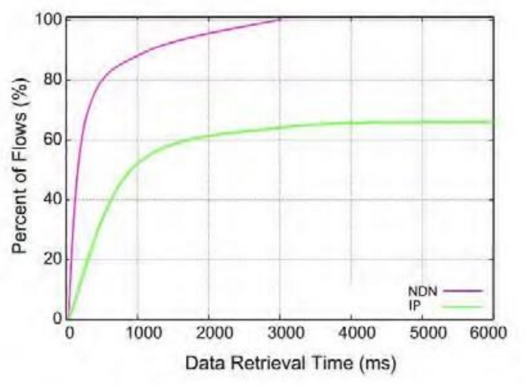

Figure 4: 1% link failure probability

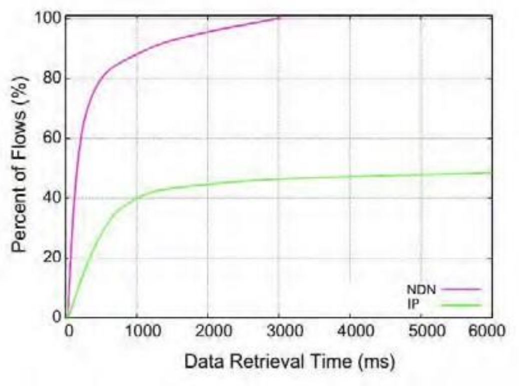

Figure 5: 10% link failure probability

As results show, NDN-based approach is more efficient against link failure, and has better throughput than IP-based approach.

CONCLUSION AND FUTURE WORK

This paper used the concept of NDN for service discovery in microservice architecture, which shows better performance comparing with existing service discovery tools, that are based on IP-based networking. The strategy used in ESB is SAF, which results have proven that is the best strategy in many cases. SAF is an adaptive forwarding strategy. It provides probability-based forwarding on a per-content/per-prefix basis and also can provide effective forwarding even if the routing information is not complete. Link failure and short-term congestion can be handled by this adaptive and stateful strategy. We plan to use more effective NDN forwarding strategies to improve the performance of service discovery in microservice architecture.

REFERENCES

[1] J. Thones, "Microservices", IEEE Software, vol. 32, no. 1, pp. 116, 2015.
[2] Long, Kim Bao, HyunSik Yang, and YoungHan Kim. "ICN-based service discovery mechanism for microservice architecture." Ubiquitous and Future Networks (ICUFN), 2017 Ninth International Conference on. IEEE, 2017.
[3] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos,X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A survey of information-centric networking research," IEEE Communications Surveys & Tutorials , vol. 16, no. 2, 2014.
[4] C. Dannewitz, D. Kutscher, B. Ohlman, S. Farrell, B. Ahlgren, and H. Karl, "Network of Information (NetInf) - An Information-Centric Networking Architecture," Computer Comm., vol. 36, no. 7, 2013.
[5] N. Fotiou, D. Trossen, and G. Polyzos, "Illustrating a Publish-Subscribe Internet Architecture," Telecomm. Systems, vol. 51, no. 4, 2012.
[6] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard, "Networking Named Content," in Proc. of the 5th ACM Int. Conf. on Emerging Networking Experiments & Technologies, 2009.
[7] T. Koponen, M. Chawla, B.-G. Chun, A.
Ermolinskiy, K. H. Kim S. Shenker, and I. Stoica, "A Data-oriented (and Beyond) Network Architecture," SIGCOMM Comp. Comm. Rev., vol. 37, no. 4, 2007.
[8] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named Data Networking," ACM SIGCOMM Comp. Comm. Rev., vol. 44, no. 3, pp. 66–73, 2014.
[9] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A Case for Stateful Forwarding Plane," Computer Communications, vol. 36, no. 7, pp. 779 – 791, 2013.
[10] C. Yi, J. Abraham, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "On the Role of Routing in Named Data Networking," in Proc. of the 1st Int. Conference on Information-Centric Networking. ACM, 2014.
[11] Posch, Daniel, Benjamin Rainer, and Hermann Hellwagner. "Saf: Stochastic adaptive forwarding in named data networking." *IEEE/ACM Transactions on Networking* 25.2 (2017): 1089-1102.
[12] etcd. [Online]. Available:
https://github.com/coreos/etcd.
[13] Consul by Hashicorp. [Online]. Avalable:
https://www.consul.io.