Mitigating Intermittent Links With Name-Based Networking

Behrooz Farkiani
b.farkiani@wustl.edu
Washington University in St. Louis
St. Louis, MO, USA

John DeHart
jdd@wustl.edu
Washington University in St. Louis
St. Louis, MO, USA

Jyoti Parwatikar
jp@wustl.edu
Washington University in St. Louis
St. Louis, MO, USA

Patrick Crowley
pcrowley@wustl.edu
Washington University in St. Louis
St. Louis, MO, USA

ABSTRACT

We present a name-based sidecar-assisted networking approach that enables off-the-shelf TCP/IP and HTTP protocols to effectively handle network disruptions. We compare the performance of our approach with the Name Data Networking by deploying and testing them in a hardware networking testbed and show, on average, in 85% of experiments, the sidecar approach manages to transfer data.

CCS CONCEPTS

• **Networks** → **Network design principles**; **Application layer protocols**; **Routing protocols**; **Transport protocols**; **Intermediate nodes**; **Middle boxes / network appliances**.

KEYWORDS

Sidecar Architecture, Envoy, HTTP/1.1, TCP, NDN

ACM Reference Format:
Behrooz Farkiani, John DeHart, Jyoti Parwatikar, and Patrick Crowley. 2023. Mitigating Intermittent Links With Name-Based Networking. In *ACM ICN 2023 Posters and Demos (ACM ICN '23), October 9–10, 2023, Reykjavik, Iceland.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3623565.3623756

1 INTRODUCTION

This work answers the following question: "In the absence of an end-to-end network path, is it possible to effectively use off-the-shelf TCP/IP and HTTP/1.1 protocols and tools without utilizing other special protocols or changing application logic?" We answer "Yes" to this question and introduce our sidecar-assisted networking approach, in which we utilize a set of proxies to route application data in the presence of network disconnections. In this way, the application logic is not changed since applications are not aware of proxies, and only the traffic endpoint addresses change. The significance and importance of the question and our solution lay in the facts that 1) network application developers could be assured that their applications work in stable and unstable network conditions with the same application logic, and 2) our solution is easily deployable in current systems without the need for utilizing special networking protocols.

We use Envoy proxy to implement our sidecar-assisted approach [8] [4]. In our approach, we build an overlay network of Envoy HTTP proxies and route the application traffic through them. Although recent research has been focused on the load balancing features of the Envoy proxy [3] [12], to the best of our knowledge, this is the first study that investigates the usage of Envoy proxy as a general-purpose overlay network. Since we configure Envoy HTTP proxies to route traffic based on the requested paths (names), the sidecar-assisted approach has a name-based architecture.

We compare the performance of our approach with Named Data Networking (NDN) [13]. In NDN, a name is associated with each piece of data. Data is retrieved by sending *interests*, and content is returned following the path of the *interests*. The NDN Forwarding Daemon (NFD) receives, aggregates, and forwards *interests*. In the context of lossy networks, research on NDN has mostly been focused on congestion control techniques or the use of synchronization protocols [1] [5].

The key contribution of this work is as follows: 1) For the first time, by using a name-based overlay network, we enable traditional internet protocols to operate efficiently in highly unstable network conditions. 2) We show our approach is easily implementable since it doesn't require modification in application logic or undelaying protocols.

2 TEST ENVIRONMENT

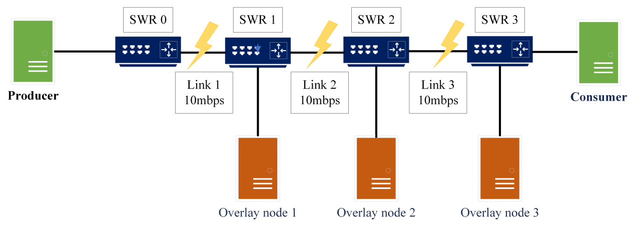
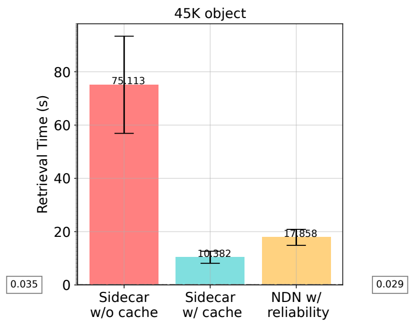


Figure 1: Testbed Topology

We implemented the sidecar and NDN approaches in the network shown in Figure 1. The components were executed inside separate Ubuntu 20.04.6 physical machines connected via 1Gb/s Ethernet. We restricted the bandwidth of Links 1-3 between software routers (SWRs) to 10Mb/s using the TC utility [11]. Links 1-3 were initially

Architecture		Success Ratio (%)	
		45K	500K
Direct HTTP/TCP		0	0
Sidecar	No Cache	95	80
	Cache	100	90
NDN	No Reliability	0	0
	Reliability	100	100



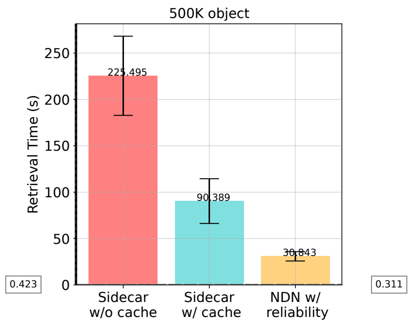


Figure 2: Three intermittent links results.

set to a 100% loss ratio. Every second, we sequentially reset one link to 0% loss, then revert it back to 100% before moving to the next link.

For the sidecar architecture, we utilized the Bitnami Envoy container (v1.26.2) [2] on overlay nodes 1-3. The consumer used Curl (v7.68.0), and the producer used the Nginx container (v1.23.3) [7] for content delivery. In direct connection, Curl HTTP GET request passed through the SWRs, and in the sidecar-assisted approach, requests went to the Envoy HTTP listener on node 3, then to the producer via overlay nodes. The return path was the same as the forward path. Curl was configured with a "connect-timeout" of "720s" and a "max-time" parameter of 900s. For Envoy proxies, we used HTTP connection manager, HTTP router, and HTTP cache filters. In addition, we set "0" for both "timeout" and "idle_timeout," and set "retry_on" for "5xx, reset, refused-stream, connect-failure, gateway-error" errors, "num_retries" to "4" and "per_try_timeout" to "180s" as the Route [9] component configurations. For Nginx, we set "public, max-age=900" for "cache-control," "sendfile on," "sendfile_max_chunk 1k," "if_modified_since off," and "keepalive_timeout 0" as additional configuration parameters.

We used NDN version 22.02 [10]. An NFD executed on the producer, consumer, overlays, and overlay nodes acted as routing nodes. The request and response path were the same as the sidecar approach. We used ndnputchunks with a chunk size of 1000 bytes for the producer and ndncatchunks for the consumer. NFD caches were valid for 15 minutes, and we used NDNLP [6] over UDP as the NFD link protocol to provide reliability.

To measure the performance of approaches, we use the success ratio, defined as the percentage of time the data object is transferred completely, and retrieval time, defined as the duration between the moment the consumer requests a data object and when the consumer receives it. In each experiment, the consumer waited for a random duration between 5 to 15 seconds and then requested an object of approximate size 45 KB or 500 KB from the producer. Then, it waited for completion or failure and requested the content again. We repeated this process 20 times and averaged the results. We used the direct HTTP connection to the producer and NDN with no reliability protocol, both with no intermittent links, as baseline measurements, and their values are shown on the outside left and the right side of each graph, respectively. Also, the standard error of the mean is shown as error bars.

3 RESULTS

Results are shown in Figure 2. The direct HTTP/TCP failed in all tests, and the average success ratio of sidecar-assisted no-cache and cache-enabled scenarios are 97.5% and 85% for 45K and 500K data objects, respectively. The average retrieval time is 41s and 153s for 45K and 500K data objects, respectively. Therefore, although the direct HTTP/TCP connection fails to transfer data objects, sidecar-assisted HTTP/TCP still can transfer the data objects with at least 85% success ratio. The reasons for the failure of direct TCP connection are TCP timeouts and exponential backoff. When the link is up, only 0.4s are needed to retrieve the 500K object. If TCP begins exponential backoff when the link is down, many backoff iterations can occur until retransmission happens with the link up. This can cause long delays and eventually trigger Curl failures. Since we also utilized TCP protocol in the sidecar approach, the situation might still happen, which led to the 15% failure and long 153s retrieval time. However, since Envoy proxies retry in case of TCP failure, we still observe an 85% success ratio.

NDN with no reliability feature failed in all tests. On the other hand, NDN with reliability could transfer 45K and 500K objects in 17s and 30s, respectively. NDN also shows a success ratio of 100%. However, NDN requires special protocols and tools to achieve this success ratio, while the sidecar-assisted approach uses the widely deployed TCP/IP protocols and tools, and the only change in application is a change in the endpoint from the producer to the overlay node 3.

4 CONCLUSION AND FUTURE WORK

We showed that although a direct HTTP/TCP fails when we have three intermittent links, the sidecar architecture enables HTTP/TCP to perform with a success ratio of 85% without changing the application logic or using special protocols. While NDN achieves a 100% success ratio, it also requires special protocols and tools.

The main reason behind the 15% failures of the sidecar architecture is TCP congestion control and its exponential backoff mechanism. One possible improvement could be using HTTP/3, which utilizes QUIC/UDP as the transport protocol. Therefore, our next set of experiments will focus on the effect of utilizing the sidecar-assisted approach with HTTP/3 to deal with highly unstable network conditions.

ACKNOWLEDGMENTS

This work was supported by NSF CNS Award 2213672.

REFERENCES

[1] Hila Ben Abraham, Jyoti Parwatikar, John DeHart, Adam Drescher, and Patrick Crowley. 2018. Decoupling information and connectivity via information-centric transport. In *Proceedings of the 5th ACM Conference on Information-Centric Networking (ICN ’18)*. Association for Computing Machinery, New York, NY, USA, 54–66. https://doi.org/10.1145/3267955.3267963

[2] Bitnami. 2023. *bitnami/envoy Tags | Docker Hub*. https://hub.docker.com/r/bitnami/envoy/tags Accessed: 2023-06-14.

[3] Boutheina Dab, Ilhem Fajjari, Mathieu Rohon, Cyril Auboin, and Arnaud Diquelou. 2020. Cloud-native Service Function Chaining for 5G based on Network Service Mesh. In *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*. 1–7. https://doi.org/10.1109/ICC40277.2020.9149045 ISSN: 1938-1883.

[4] Behrooz Farkiani and Raj Jain. 2022. *Service Mesh: Architectures, Applications, and Implementations*. Technical Report. Washington University in St. Louis. https://www.cse.wustl.edu/~jain/cse574-22/ftp/svc_mesh/

[5] Yi Hu, Constantin Serban, Lan Wang, Alex Afanasyev, and Lixia Zhang. 2021. PLI-Sync: Prefetch Loss-Insensitive Sync for NDN Group Streaming. In *ICC 2021 - IEEE International Conference on Communications*. 1–6. https://doi.org/10.1109/ICC42927.2021.9500452 ISSN: 1938-1883.

[6] NDN. 2023. *NDNLPv2 - NFD - NDN project issue tracking system*. https://redmine.named-data.net/projects/nfd/wiki/NDNLPv2 Accessed: 2023-06-14.

[7] Nginx. 2023. *nginx Tags | Docker Hub*. https://hub.docker.com/_/nginx/tags Accessed: 2023-06-14.

[8] Envoy Proxy. 2023. *Envoy Proxy - Home*. https://www.envoyproxy.io/ Accessed: 2023-06-14.

[9] Envoy Proxy. 2023. *HTTP route components (proto)*. https://www.envoyproxy.io/docs/envoy/latest/api-v3/config/route/v3/route_components.proto Accessed: 2023-06-14.

[10] NDN Tools. 2023. *Release ndn-tools-22.02: Version 22.02 named-data/ndn-tools*. https://github.com/named-data/ndn-tools/releases/tag/ndn-tools-22.02 Accessed: 2023-06-14.

[11] Ubuntu. 2023. *Ubuntu Manpage: tc - show / manipulate traffic control settings*. https://manpages.ubuntu.com/manpages/focal/man8/tc.8.html Accessed: 2023-06-14.

[12] Xiaojing XIE and Shyam S. Govardhan. 2020. A Service Mesh-Based Load Balancing and Task Scheduling System for Deep Learning Applications. In *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*. 843–849. https://doi.org/10.1109/CCGrid49817.2020.00009

[13] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, kc claffy, Patrick Crowley, Christos Papadopoulos, Lan Wang, and Beichuan Zhang. 2014. Named data networking. *ACM SIGCOMM Computer Communication Review* 44, 3 (2014), 66–73. https://doi.org/10.1145/2656877.2656887