Timers in State Vector Sync

Varun Patil
varunpatil@cs.ucla.edu
UCLA
Los Angeles, USA

Seiji Otsu
seijiotsu@ucla.edu
UCLA
Los Angeles, USA

Lixia Zhang
lixia@cs.ucla.edu
UCLA
Los Angeles, USA

ABSTRACT

State Vector Sync (SVS) is a Distributed Dataset Synchronization (Sync) protocol designed to support distributed applications running over NDN. The design of SVS has two unique features that set it apart from all the previous Sync protocol designs. First, SVS encodes the raw information of data namespace to be synchronized in its Sync Interest packets. Second, and related, it uses Sync Interests as notifications which do not solicit data replies. To reveal insights of how its unique design features enable SVS to outperform its counterparts, in this poster we describe the operation of two types of timers used in SVS and their effectiveness in minimizing protocol overhead while keeping synchronization delay low.

CCS CONCEPTS

• **Networks** → **Network protocol design**; **Transport protocols**; *Network performance evaluation.*

KEYWORDS

Named Data Networking, Distributed Dataset Synchronization, NDN Transport, State Vector Sync, Timers

ACM Reference Format:
Varun Patil, Seiji Otsu, and Lixia Zhang. 2023. Timers in State Vector Sync. In *ACM ICN 2023 Posters and Demos (ICN '23), October 9–10, 2023, Reykjavik, Iceland*. ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3623565.3623754

1 OVERVIEW

Distributed Dataset Synchronization (Sync) [3] provides the transport layer functionality in the Named Data Networking (NDN) [1] architecture. The State Vector Sync (SVS) [5] protocol is one of the newer Sync designs, providing improved Sync performance compared to previous protocols [6]. Quick re-synchronization within a Sync group and effective suppression of redundant Sync Interests are two essential factors that lead to this improved performance; both rely on the setting of Sync Interest timers. In this poster, we briefly explain the working of SVS timers, and demonstrate the effects of network conditions through simulations. We hope these insights would facilitate improvements to Sync, and better equip application developers to choose default values.

ICN '23, October 9–10, 2023, Reykjavik, Iceland

ACM ISBN 979-8-4007-0403-1/23/10.
https://doi.org/10.1145/3623565.3623754

2 LOSS RECOVERY AND TIMERS

SVS provides namespace synchronization among the participants of a Sync group. Each producer in the group uses a sequence number to advertise the number of data objects produced, which can then be used by consumers to fetch the data if they desire. SVS synchronizes the list of sequence numbers at all producers using a single type of message: a Sync Interest carrying the entire *raw state*[1] that is multicast to the entire group. On receiving the Sync Interest, group members can compare the received state to the locally known state and thus learn about new data production.

To provide reliable synchronization with loss recovery, SVS retransmits Sync Interests by a combination of event-driven notifications and soft-state timers.

(1) **Triggered by new data production** Each data producer sends a SVS Sync Interest upon data production.
(2) **Triggered by obsolete Sync Interest** If a member M of the Sync group receives a Sync Interest I_R and notices that I_R's state vector contains a lower sequence number for one or more producers, M will plan to send a Sync Interest I_M with its own state, thus updating I_R's sender.
(3) **Triggered by soft-state refresh** If no Sync Interest is received by a group member M for a defined time period, M will send a Sync Interest carrying its own state, ensuring the eventual consistency of the Sync group.

However, care must be taken to avoid simultaneous retransmission of Sync Interests by multiple group members, since one outdated Sync Interest may trigger a flood of responses to correct it, and the number of periodic updates would increase linearly with the group size. SVS mitigates this potential overhead by applying a suppression mechanism to all outgoing Sync Interests. Before sending any outgoing Sync Interest, a node M sets a timer to a randomly chosen value between zero and a configured upper bound.

(1) While the suppression timer is active, M merges all incoming Sync Interests during this time by picking the highest sequence number for each producer from all the received state vectors to create a state vector V_S. If V_S contains updated information than the local state vector V_M, then V_M is updated.
(2) When the suppression timer expires, M compares V_S with its local state vector V_M. If the two state vectors are identical, the outgoing Sync Interests is discarded (*suppressed*). Otherwise if the two vectors differ and V_M contains more advanced state (i.e. the sequence number for one or multiple producers is higher), M sends a Sync Interest carrying V_M.

The SVS implementation uses a single timer for both the suppression and soft-state refresh timers, because only one of the two is active at any given moment. This suppression mechanism of SVS ensures that an outdated Sync Interest triggers a single or a small

[1]The state of the group is encoded as a list of tuples of [producer, seq#].

number of responses, and ensures that only one periodic Sync Interest is sent every refresh period. This enables the lower overhead of SVS while maintaining resiliency to losses.

3 EVALUATION

We use ndnSIM [4] to recreate the GÉANT network topology [2] with 45 nodes. We run SVS at each node and create a single Sync group. Each link has a 10ms latency, and we vary packet drop rates from 0 to 50%. To test effects of connectivity, we randomly remove edges from the GÉANT topology and create five variations. The *node degree* is the number of neighbors for a node, with the avg. varying from 1.9 to 3.2. Correspondingly, the avg. hop count varies from 6.3 to 3.1, and the longest path delay ranges between 70ms to 150ms.

In each run, each node randomly chooses a time in the [0s, 45s] range for the first publication, and publishes new data every 45s thereafter. This yields an overall publication rate of one per second. We set the suppression timer to range between [0, 200ms], and test periodic timer values of 250ms, 1s and 4s, both timers having a 25% random jitter. We define the 90%tile of the time taken for a node to learn about data production as Sync latency, and the total number of packets sent as the overhead metric.

From Fig. 1, we can infer that SVS increasingly benefits from richer connectivity in the presence of losses. This is expected, since a node can receive Sync Interest from any of its multiple links. With sparse connectivity and high losses, the probability of any Sync Interest reaching a given node goes down, resulting in some nodes falling behind in the dataset state updates. Under these conditions, periodic Sync Interests are likely to help correct outdated state, and thus a shorter period can significantly reduce latency.

Fig. 2 highlights the corresponding overhead trade-off. As the period gets shorter, more Sync Interests are produced thus increasing overhead. Further, with a very short period that is comparable to the network diameter, suppression becomes ineffective leading to a great increase in overhead.

By observing the results, we can draw the following conclusions:

(1) SVS benefits significantly from richer connectivity, improving metrics of latency and overhead especially in lossy conditions.
(2) SVS successfully mitigates overhead to a large degree using the Sync Interest suppression mechanism, despite Sync Interests not being aggregatable in the network[2].
(3) The ideal timer values have a positive correlation to the network diameter. Values that are too short make suppression ineffective, greatly increasing overhead.
(4) A shorter periodic timer can improve Sync latency in the presence of losses, at the cost of increased overhead.

4 FUTURE WORK

The optimal choices for the values of SVS timers depend on the application, topology and network conditions, which may change dynamically over time. We recognize self-adjusting timer values as a future research direction for SVS. Such automated adjustments may use heuristics such as the maximum latency within the group, the data production rate, and the observed packet loss rate.

[2]Each Sync Interest in SVS is signed with the sender's private key, making it unique.

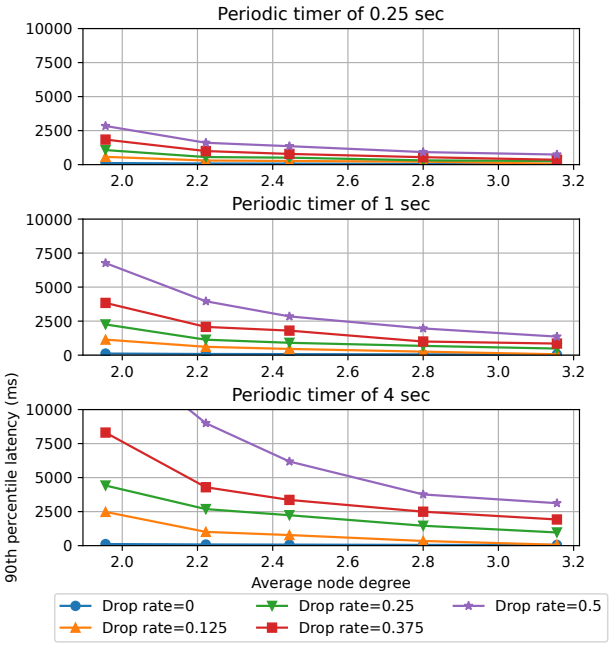

Figure 1: Comparison of Sync latency.

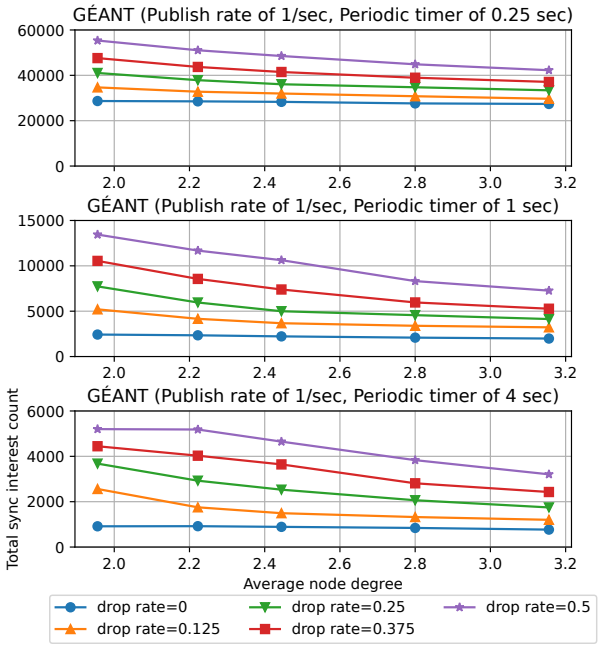

Figure 2: Total packet count

REFERENCES

[1] Alexander Afanasyev, Tamer Refaei, Lan Wang, and Lixia Zhang. 2018. A Brief Introduction to Named Data Networking. In *Proc. of MILCOM*.

[2] GÉANT project. 2018. GÉANT topology map. https://network.geant.org/ accessed: 2023-06-15.

[3] Tianxiang Li, Wentao Shang, Alex Afanasyev, Lan Wang, and Lixia Zhang. 2018. A Brief Introduction to NDN Dataset Synchronization (NDN Sync). In *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*. IEEE, 612–618.

[4] Spyridon Mastorakis, Alexander Afanasyev, and Lixia Zhang. 2017. On the Evolution of NdnSIM: An Open-Source Simulator for NDN Experimentation. *SIGCOMM Comput. Commun. Rev.* 47, 3 (sep 2017), 19–33. https://doi.org/10.1145/3138808.3138812

[5] Philipp Moll, Varun Patil, Nishant Sabharwal, and Lixia Zhang. 2021. *A Brief Introduction to State Vector Sync.* Technical Report NDN-0073. NDN.

[6] Philipp Moll, Varun Patil, Lan Wang, and Lixia Zhang. 2022. SoK: The Evolution of Distributed Dataset Synchronization Solutions in NDN. In *Proceedings of the 9th ACM Conference on Information-Centric Networking* (Osaka, Japan) *(ICN '22)*. Association for Computing Machinery, New York, NY, USA, 33–44. https://doi.org/10.1145/3517212.3558092