Identifying Nearest Fog Nodes With Network Coordinate Systems

Simon Huber, Tobias Pfandzelter, David Bermbach
Technische Universität Berlin & Einstein Center Digital Future
Mobile Cloud Computing Research Group
{shr,tp,db}@mcc.tu-berlin.de

***Abstract*—Identifying the closest fog node is crucial for mobile clients to benefit from fog computing. In this paper, we analyze the performance of the Meridian and Vivaldi network coordinate systems for this task. To that end, we simulate a dense fog environment with mobile clients. We find that while network coordinate systems really find fog nodes in close network proximity, a purely latency-oriented identification approach ignores the larger problem of balancing load across fog nodes.**

I. Introduction

Fog computing bridges the gap between central cloud data centers and mobile clients with intermediary and edge nodes that provide computing services in close proximity [1]. To benefit from this, clients must be able to identify their nearest fog node, which is non-trivial in the loosely coupled fog [2]. Network-only techniques are insufficient as the node must also be available and have sufficient capacity. Similarly, approaches based on geographic location of nodes and clients ignore network characteristics [3]. Fog node selection is a continuous process as clients are usually mobile, e.g., connected vehicles or IoT devices, changing their network and physical position [4]. The naive approach of probing each node in the network will theoretically lead to a perfect result but is not scalable due to the large communication overhead.

There exist some approaches for identifying a nearest node for peer-to-peer (P2P) systems, yet it is unclear if those can directly be applied to fog computing. In this paper, we aim to close this gap by applying the *Meridian* [5] and *Vivaldi* [6] *network coordinate systems* to fog computing. We analyze the performance of both systems in a simulation of mobile clients moving through a distributed fog network.[1]

II. Simulation

We simulate a fog network with mobile clients in a $2.25\,\mathrm{km}^2$ area in Berlin, Germany for ten minutes using *SimPy* [8].[2] Clients in our simulation follow predefined movement patterns obtained from the *Open Berlin Scenario* [9]. Clients periodically send tasks to the fog platform, measuring response latency. There are 29 fog nodes in this area, based on cell tower locations [10]. To abstract from heterogeneous hardware resources, each node has a number of "slots", i.e., concurrent client tasks it can fulfill. We assign the number of client slots randomly, with a total of 318 available slots. To observe the impact of client load on NCS behavior, we vary the ratio between total available slots and clients between 0.1 and 1.0.

Communication latency between parties is based on transmission, propagation, processing, and queuing delays, using values given by Burk and Lemberg [11]. We compare the Meridian and Vivaldi NCS with a baseline and a random selection approach. The baseline is the theoretical optimal node selection based on the omniscient simulation environment.

III. Results

The total number of messages per client (Fig. 1a) increases with client ratio, likely caused by nodes not being able to accept client requests. We see that Meridian has a higher number of messages with low client ratios but is more scalable than Vivaldi and our baseline at higher ratios.

Vivaldi and the baseline show higher numbers of lost client messages, which occurs when the fog node is overloaded and cannot process more tasks (Fig. 1b). As a result of unprocessed tasks, we observe higher reconnections for clients (Fig. 1c).

The rate of optimal node selection is 100% for the baseline but less than 10% for all other techniques, including the random approach. Nevertheless, Meridian and Vivaldi show a low RMSE selection error in terms of additional network delay over the optimal node selection. While they mostly cannot find the *optimal* node, both NCS find *good* nodes.

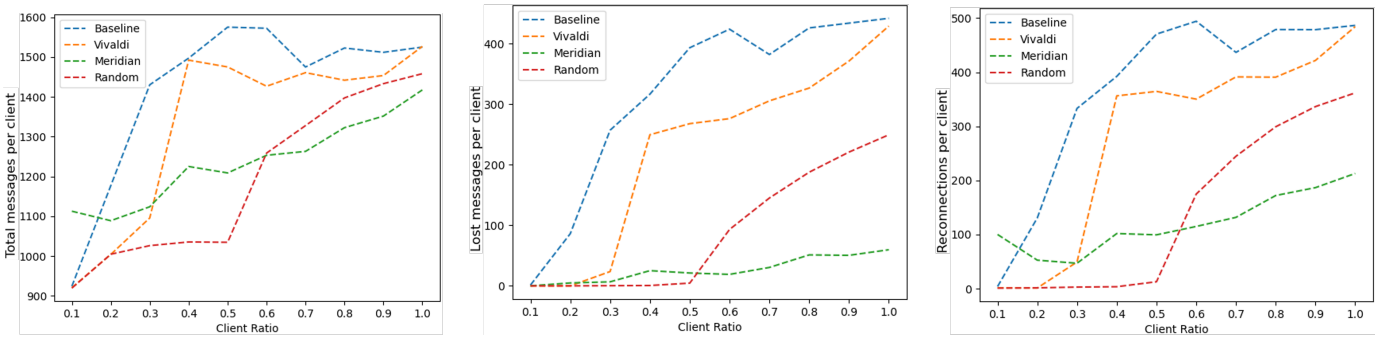
The mean communication latency from clients to identified fog nodes (Fig. 2a) increases with client ratio because of increased load on the fog platform and decreased bandwidth available per client. The baseline and Vivaldi show higher mean latency per client compared to Meridian and the random approach, which we explain with the uniqueness of node selection (Fig. 2c). Because the baseline and Vivaldi do not distribute the load of the clients over the entire network, the bandwidth of these nodes decreases. This translates into a higher transmission and queuing delay. By not accounting for fog nodes with less capacity the random approach overloads these nodes, leading to increased latency. Only the Meridian system is aware of the resources of the fog nodes as fully loaded nodes do not take part in the selection process.

We show the error between achieved RTT and optimal RTT in Fig. 2b. Surprisingly, we observe the highest connection error for Vivaldi, especially in the higher client ratios.

Supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 415899119.

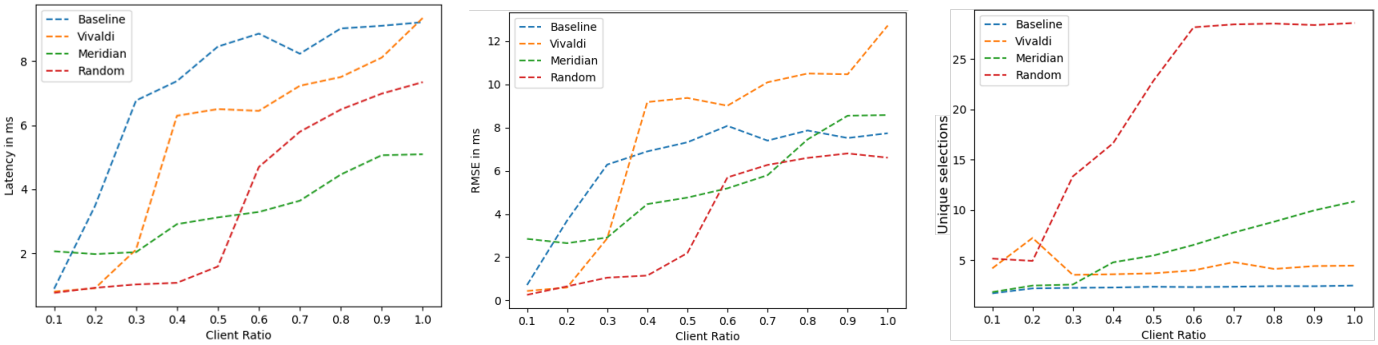
[1] An extended version of this paper is available as technical report [7].

[2] https://github.com/pfandzelter/nearest-fog-nodes



(a) Total Messages (b) Lost Messages (c) Reconnection Requests

Fig. 1: Number of messages handled per client for each of the tested algorithms. The total number of messages (Fig. 1a) grows with higher client ratios, with baseline requiring the most messages. With increasing client ratios, the number of *lost* messages per client (Fig. 1b) and number of reconnections per client (Fig. 1c) also grows for all tested algorithms, with the Meridian approach requiring the least messages at higher ratios.



(a) Mean Latency (b) Selection Error (c) Unique Selections

Fig. 2: Actual client access latency also depends on available slots and available bandwidth, and surprisingly the mean latency results (Fig. 2a) show the random approach outperforming others with low client ratios. At higher client ratios, Meridian is best. Both approaches are good at load balancing (higher number of unique node selections, Fig. 2c), reducing additional delay caused by reconnection and limited bandwidth.

IV. Conclusion

The results of our simulation show that NCS are powerful techniques to find nearest nodes in a distributed system that consider real network distance instead of, e.g., geographical location. Surprisingly, however, we find that identifying the closest node is often not of upmost concern to clients in a fog network as dense as that of our simulation: In most cases, the additional latency of suboptimal nodes was negligible. Instead, we find the major driver for the efficiency of a node selection algorithm to be how well it balances load across fog nodes. For future research we thus suggest focussing on distributed load balancing within bounds of service-level objectives.

References

[1] F. Bonomi *et al.*, "Fog computing and its role in the internet of things," in *Proc. MCC '12*, 2012, pp. 13–16.

[2] T. Pfandzelter *et al.*, "Managing data replication and distribution in the fog with fred," *Software: Practice and Experience*, 2023.

[3] J. Hasenburg and D. Bermbach, "DisGB: Using geo-context information for efficient routing in geo-distributed pub/sub systems," in *Proc. UCC '20*, Dec. 2020, pp. 67–78.

[4] T. Pfandzelter and D. Bermbach, "Towards predictive replica placement for distributed data stores in fog environments," in *Proc. IC2E '21*, Oct. 2021, pp. 280–281.

[5] B. Wong, A. Slivkins, and E. G. Sirer, "Meridian: a lightweight network location service without virtual coordinates," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4, pp. 85–96, 2005.

[6] F. Dabek *et al.*, "Vivaldi: a decentralized network coordinate system," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 15–26, 2004.

[7] S. Huber, T. Pfandzelter, and D. Bermbach, "On the applicability of network coordinate systems for fog computing," TU Berlin & ECDF, Mobile Cloud Computing Research Group, Tech. Rep. MCC.2023.2, Jul. 2023.

[8] O. Lünsdorf, S. Scherfke, and P. Grayson, "Simpy," https://gitlab.com/team-simpy/simpy, SimPy Development Team, Oct. 2021, accessed: June 17, 2023.

[9] D. Ziemke, I. Kaddoura, and K. Nagel, "The matsim open berlin scenario: A multimodal agent-based transport simulation scenario based on synthetic demand modeling and open data," in *Proc. ABMTRANS 2019*, Apr. 2019, pp. 870–877.

[10] "Elektromagnetische felder (emf)," https://www.bundesnetzagentur.de/DE/Vportal/TK/Funktechnik/EMF/start.html, Bundesnetzagentur, Jun. 2023, accessed: June 17, 2023.

[11] A. Burk and D. Lemberg, "5g overview and predictive analysis for latency optimized telecommunication networks," https://www.hs-rm.de/fileadmin/persons/khofmann/Gastvortraege/Vortragsfolien/20181026-Burk-Lemberg-vodafone-5G.pdf, Vodafone Germany, Oct. 2018, accessed: June 17, 2023.