Distributed Load Balancing for the Resilient Publish/Subscribe Overlay in SeDAX

Michael Hoefling, *Graduate Student Member, IEEE,* Cynthia G. Mills, *Member, IEEE,*
and Michael Menth, *Senior Member, IEEE*

***Abstract*—SeDAX is a publish/subscribe information-centric networking architecture where publishers send messages to the appropriate message broker over a Delaunay-triangulated overlay network. Resilient data forwarding and data redundancy enable a high level of reliability. Overlay nodes and topics are addressed via geo-coordinates. A topic is stored on primary and secondary nodes, those nodes closest and second-closest to the topic's coordinate, respectively. The overlay automatically reroutes a topic's messages to its secondary node should its primary node fail. In the original proposal, SeDAX determines the coordinate of a topic by hashing its name. This kind of topic allocation is static, which can lead to unintended load imbalances.**

We propose a topic delegation mechanism to make the assignment of topics to nodes dynamic. Our proposed mechanism is the only existing method to improve the flexibility and resource management of the SeDAX architecture so far. We define three resilience levels that allow information on the SeDAX overlay to survive 0, 1, or 2 node failures, imposing different loads on SeDAX nodes. For this elaborated SeDAX approach, we suggest a distributed resource management system that detects traffic imbalances among SeDAX nodes and re-assigns topics to other coordinates for load balancing purposes. We evaluate the load imbalance for the different resilience levels, for different topic characteristics, and in particular for topics with storage requirements growing over time. The proposed algorithm leads to well balanced load on SeDAX nodes while keeping load redistribution at a reasonable level.

***Index Terms*—SeDAX, smart grid communication, publish/subscribe, information-centric networking, load balancing, optimization, performance evaluation.**

I. Introduction

THE SEcure Data-centric Application eXtension (SeDAX) architecture [1] is a scalable, resilient, and secure data delivery and sharing platform for smart grids. SeDAX applies the *publish/subscribe* (pub/sub) and *information-centric networking* (ICN) paradigm to the electric utility network of sensors and controls for electricity generators, consumers, and brokers. In the following introduction we present the SeDAX architecture and point out the intrinsic potential for load imbalance in terms of stored topic data on SeDAX nodes which is due to SeDAX's static mapping of topics to geo-coordinates using their names. In the remainder of this work, we describe architectural extensions to pure SeDAX allowing for dynamic mapping of topics to coordinates that facilitate load balancing. We further describe a centralized and various distributed load balancing algorithms for this novel architecture and evaluate them. This work extends prior work of the authors [2] by proposing mechanisms for continuous load balancing over time while keeping the overhead at a reasonable level and showing the effectiveness of that approach by simulation results. This work addresses resource management on the SeDAX layer only and does not consider operation of the lower layers, i.e., latency issues that may result from lower layer bottlenecks are not addressed in this work and are out of scope.

A. The SeDAX Architecture

SeDAX's pub/sub communication paradigm decouples information contributors from information consumers by organizing information into *topics*. In this context, a topic is an abstract representation of a unidirectional information channel, and is addressed using its unique name and probably attributes, e.g., data type, location, and time. Publishers of a topic send messages to brokers that forward them to subscribers. This requires that publishers and subscribers have registered with the broker for that topic. The broker stores published topic data and keeps it available for some time. Additionally, the broker has to store metadata for each topic, e.g., the list of subscribers to a topic. Topic data and metadata create load on the server in terms of storage requirements.

Formally, SeDAX stores a set of topics $\mathcal{T}$ on a set $\mathcal{V}$ of brokers which are called SeDAX nodes. An overlay network steers messages addressed to a certain topic to the right SeDAX node. Thus, publishers and subscribers do not need to know the addresses of the corresponding SeDAX node to send registration and data messages, they just need to have access to the overlay network. As a result, SeDAX does not require a mapping system, that may be compromised or fail, to resolve topics to SeDAX nodes.

Figure 1 illustrates the overlay network which is organized as follows: SeDAX nodes $v \in \mathcal{V}$ are equipped with geo-coordinates denoted as $C(v)$. Nodes are connected to selected geographic neighbors via TCP transport connections to form a Delaunay triangulated (DT) overlay network. The DT overlay

Manuscript received March 4, 2016; revised July 15, 2016 and November 24, 2016; accepted December 18, 2016. The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7-ICT-2011-8 under grant agreement n° 318708 (C-DAX). The authors alone are responsible for the content of this paper.

M. Hoefling and M. Menth are with the University of Tuebingen, Department of Computer Science, Chair of Communication Networks, Germany (e-mail: {hoefling,menth}@uni-tuebingen.de). M. Hoefling is the formal Corresponding Author for this submission.

C. G. Mills is with the SySS GmbH, Tuebingen, Germany, and was with the University of Tuebingen, Department of Computer Science, Chair of Communication Networks, Germany while the work was done.

Digital Object Identifier 10.1109/TNSM.2016.2647678

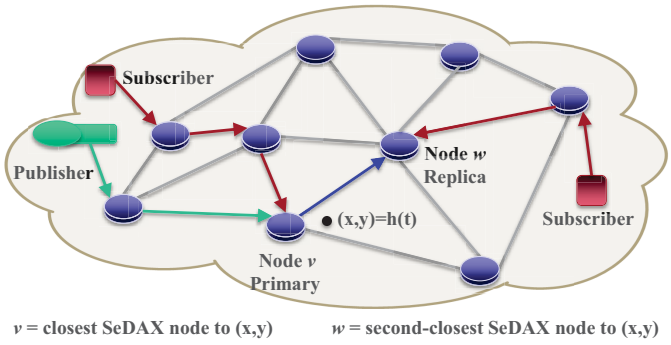


Fig. 1. Topic-group communication in SeDAX uses geographic forwarding.

network enables SeDAX nodes to forward a message addressed to a certain coordinate to the closest SeDAX node. All coordinates for which a node v is closest form its Voronoi cell $Voronoi(v)$. The SeDAX authors [1] have shown that this kind of overlay forwarding creates only little path stretch compared to the shortest path in the overlay. Furthermore, the DT overlay is self-healing: if a node fails, the DT property is restored after some local and self-organized reconfiguration.

A geographic hashing function (GHF) derives a Euclidean coordinate $(x, y) = h(t)$ from the name of a topic t. A topic is stored on the SeDAX node closest to that coordinate, i.e., on the node with the least Euclidean distance $d(C(v), h(t)), v \in \mathcal{V}$. The GHF and the DT overlay enable other SeDAX nodes to forward messages destined to a topic to the SeDAX node responsible for that topic.

SeDAX can be made resilient against node failures. The data and information of a topic t are stored on the SeDAX nodes that are closest (primary) and second-closest (secondary) to the topic's coordinate $h(t)$. This is simple, as they are neighboring nodes. The failure of a node is detected via broken TCP connections. This triggers self-healing of the DT overlay, i.e., the failed node is excluded from the network of forwarding node. Details are described in [1]. Messages for topic t are then automatically forwarded to the respective alternate node, which starts delivering messages to subscribers. This resilience concept may be extended to protect against consecutive failures by ensuring that topic data and metadata are always kept on the closest and second-closest working SeDAX node. Thus, the self-healing property of the DT overlay combined with the backup concept constitutes a simple and effective resilience concept in SeDAX that can survive even multiple consecutive failures.

B. Problem Statement

SeDAX statically assigns topics to coordinates using the hash value $h(t)$. If a SeDAX node is accidentally primary or backup node for too many or too large topics, it may become overloaded in terms of storage capacity. The original SeDAX architecture does not provide any features to take away topic responsibility from such a node.

C. Contributions of this Paper

(1) We propose adding *topic delegation* to SeDAX which allows dynamic assignment of topics $t \in \mathcal{T}$ to configurable coordinates $C(t)$ instead of to a fixed hash value $h(t)$.

(2) We introduce three *resilience levels* that allow information on the SeDAX overlay to survive 0, 1, or 2 node failures, imposing different loads on SeDAX nodes.

(3) We define *load metrics* for SeDAX nodes and coordinates for the introduced resilience levels.

(4) We provide distributed *coordinate selection algorithms* that make use of these definitions to retrieve appropriate delegation coordinates.

(5) We present distributed *load balancing algorithms* that combine these coordinate selection algorithms and the topic delegation mechanism.

(6) We show by *simulations* that static topic assignment can lead to significant load imbalance among SeDAX nodes and analyze the causes for that imbalance.

(7) We demonstrate that the proposed load balancing schemes can almost equalize the load over the SeDAX nodes for different resilience levels when topic loads remain static.

(8) Finally, we show the influence of topic growth on balanced SeDAX networks, and demonstrate that our algorithms can still equalize the load over the SeDAX nodes for different resilience levels.

D. Organization of this Paper

The paper is structured as follows. In Section II we review related work. Section III suggests the topic delegation mechanism as an extension to SeDAX. Section IV defines the loads of SeDAX nodes and coordinates for different levels of resilience. In Section V, various distributed coordinate selection algorithms are presented which are an important building block for distributed load balancing as described in Section VI. Section VII and Section VIII quantify and analyze the load imbalance in the original and improved SeDAX architecture, and show that the proposed load balancing algorithms almost equalize the load over all SeDAX nodes. Finally, Section IX concludes this work.

II. Related Work

SeDAX [1] builds upon prior work in the area of pub/sub [3] and ICN [4]. In recent work [5], we investigated the storage requirements of SeDAX necessary to survive the failure of multiple SeDAX nodes without storage shortages. This led to high storage requirements on SeDAX nodes that could be reduced by assignment of optimized coordinates to SeDAX nodes, which is generally difficult to implement.

SeDAX uses the DT overlay and GHF to locate its pub/sub-based message brokers. Most existing ICN architectures such as PSIRP/PURSUIT [6], 4WARD/SAIL [7], NDN/CCNx [8], [9], DONA [10], and CAN [11] are based on distributed hash tables (DHTs) and pub/sub. They differ in the way topic names are resolved, data is forwarded, and whether the organization of data distribution is hierarchical [12] or flat as in SeDAX. Quality of Service (QoS) constraints for replication in more complex topologies with hierarchical data stores are discussed

in [13], [14]. LIPSIN [15] uses bloom filters to quickly resolve names and find topic stores.

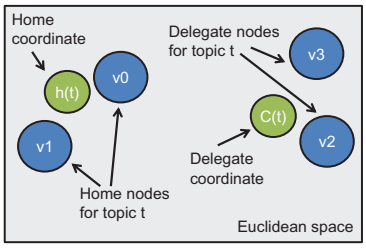
Chord [16] allocates coordinates on a ring to predecessor and successor nodes. Others like CAN [11] allocate rectangular areas to a primary node, further subdividing or combining rectangles as nodes join or exit the network. Greedy routing schemes like SeDAX organize the space into Voronoi cells so that the closest node to a coordinate is the home node for that coordinate, thus avoiding the need to maintain routing tables.

ICN systems can be viewed as structured P2P systems [17]. In a structured system like SeDAX, some nodes may provide more centralized services such as directory services (e.g., maintaining a lookup table of underloaded nodes) or security services (authoritatively authenticating a node, publisher, or subscriber). Most load balancing approaches in P2P systems focus on unstructured P2P systems [17] where nodes with different capacities frequently join and leave the network. SeDAX nodes are both more structured and less ephemeral whereas SeDAX publishers and subscribers can readily be mobile without requiring updates to the node routing overlay.

Load balancing schemes differ as well. Felber et al. [18] give an excellent overview on load balancing mechanisms for peer-to-peer systems based on DHTs. Their taxonomy divides load balancing into three different categories: object placement, routing, and underlay. Our approach falls into the object placement category, i.e., load balancing is achieved by placing objects (topics in SeDAX) or nodes on the overlay so that the load among nodes is equalized. We briefly summarize solutions that have been surveyed in [18] and that come close to the proposed load balancing approach in the following.

Kenthapadi et al. [19] propose a mechanism for load balanced overlay node addition by placing new overlay nodes between most loaded nodes. Stoica et al. [16] propose virtual servers for load balancing. In a nutshell, a physical node may host several virtual servers that are each responsible for a certain identifier (coordinate in SeDAX). Physical nodes can exchange virtual servers to achieve better load balance, i.e., virtual servers facilitate fair (virtual) overlay node placement. This scheme may be very difficult to implement for SeDAX because of SeDAX' resilience scheme, i.e., primary and backup virtual server have to be adjacent nodes on the overlay but should be hosted on two different physical nodes. Rao et al. [20] propose three methods for physical nodes to exchange load information about their virtual servers; their methods are comparable to our distributed coordinate selection algorithms in Section V. Godfrey et al. [21] extend the methods of [20] by periodic load balancing and emergency load balancing. Byers et al. [22] propose the use of multiple hashing functions to find storage nodes on the overlay, and the use of redirection pointers at destination nodes resembles our topic delegation mechanism in Section III.

Most load balancing approaches, including those described in this paper, benefit from the "power of 2 choices" described by Mitzenmacher [23], [24] in ball-bin load balancing. As Bridgewater et al. summarize in Balanced Overlay Networks (BON) [25], "The important result from ball-bin systems is that if one probes the population of more than one bin prior to assigning a ball, the population of the most full bin will be reduced exponentially in N." Even and Medina [26] further discuss lower bounds for ball-bin load balancing.



Delegate list of nodes v0, v1, v2 and v3

Topic	Coordinate	Timestamp	…
t	h(t)	0	…
t	C(t)	2	…
…	…	…	…

Fig. 2. Topic delegation in SeDAX. The home nodes ($v0$ and $v1$) are responsible for all messages of topic t from time zero until the next entry on the delegate list, time 2. All messages on or following time 2 are the responsibility of the delegate nodes ($v2$ and $v3$). The delegate list is synchronized among home and delegate nodes.

In BON, nodes change the number of immediate incoming neighbors in response to the node's availability. Thus, the overlay network can be viewed as a directed graph that is dynamically reconfigured to reflect the current system load. BON uses random walks through the directed graph to select the least loaded node on the path. BON's target application is job allocation in grid computing. In this environment, jobs enter and leave the network frequently whereas SeDAX's storage requirements tend to be of longer if not permanent duration. Our random query approach might use such a random walk to include the least loaded node on the path to the queried location, effectively increasing the scope of queries.

III. Topic Delegation

If a SeDAX node is overloaded, diverting load to other nodes may be helpful. However, due to static assignment of topic coordinates $C(t)$ to coordinates $h(t)$, the original SeDAX architecture cannot support load shifting by design. We propose topic delegation for SeDAX which uses $h(t)$ as the default coordinate of a topic, but allows for a reassignment of $C(t)$ to any other coordinate. Topic delegation adds flexibility to SeDAX without sacrificing its benefits, e.g., resilient overlay forwarding, decentralized control, and the ability to cope without a mapping system. In the following, we explain the principle and operation of topic delegation in SeDAX.

A. Topic Delegation Principle

The node closest to a topic's default coordinate $h(t)$ is the topic's *home node*. By default, the topic coordinate $C(t)$ equals the topic's hash value $h(t)$ and is called *home coordinate* of topic t. When the topic coordinate $C(t)$ is set to a value other than $h(t)$, the coordinate $C(t)$ is called *delegate coordinate* of topic t and the node closest to that coordinate is called the *delegate node* for topic t. Delegate nodes are responsible for the topic, i.e., they store published topic data and metadata.

Home nodes track where all topic data is stored via a *delegate list*. A delegate list holds the active topic coordinates $C(t)$ for topics $t \in \mathcal{T}$ and a timestamp of the first message stored at the respective coordinate; this list is shared and synchronized among home and delegate nodes.

Figure 2 shows a delegate list shared among topic t's home and delegate nodes. The current coordinate of a topic is the most recent coordinate on the delegate list. The home nodes ($v0$ and $v1$) are responsible for all messages of topic t from time zero until the next entry on the list, time 2. All messages on or following time 2 are the responsibility of the delegate nodes ($v2$ and $v3$). If the delegate nodes at coordinate $C(t)$ have retired themselves from service for topic t, the home nodes resume responsibility by default and enter the home coordinate $h(t)$ as most recent topic coordinate on the list.

Once home and delegate nodes agree to participate in a forwarding relationship, the home nodes add to their list of delegates an entry containing the delegate coordinate $C(t)$ and the *start time*. The start time is the timestamp of the first data packet stored at $C(t)$. If the start time is not known, e.g., no data has been stored at $C(t)$ yet, the start time is the time at which forwarding to $C(t)$ began. When a topic moves from one delegate node to another, registrations are transferred to the new delegate node. It is up to the implementer whether the old or new delegate node informs the clients about that event. The first entry in the delegate list always contains the topic coordinates $h(t)$ and start time zero to ensure that the home nodes remain responsible for all requests prior to any other delegation.

Delegate nodes that wish to retire, e.g., because they have become overloaded themselves, simply inform the home nodes that they are retiring. Retiring delegate nodes should exit gracefully to minimize both the potential for data loss and unnecessary network traffic, normally by waiting until all of their data has expired, alternatively by gradually shifting their load to their predecessor and/or successor. When retiring delegate nodes no longer contain data for a topic, they notify the home nodes and the delegation is deleted from the delegate list. All registered clients are informed of the new delegate coordinate by the delegates' predecessor and/or successor; any future topic data is handled by the new delegate.

This simple yet robust arrangement elegantly enables the continuous service of requests while avoiding the excessive data transfers, forwarding loops, data loss, and service delays that commonly accompany such transitions. This paper investigates the effectiveness of the resource management but does not provide experimental results on signaling delay.

B. Topic Delegation Operation

When a SeDAX client (publisher or subscriber) joins a topic, the client first sends a join message over the overlay to the topic's home coordinate $h(t)$ so that the message reaches the home node. The topic's home node checks its delegate list for that topic. If there is no entry, the home node itself is the message broker for that topic; no modification to the existing SeDAX architecture is needed. If the delegate list holds an entry for that topic, the home node forwards the join message to the delegate coordinate $C(t)$; this can be achieved by encapsulation to the delegate coordinate or rewrite of the destination coordinate. Upon receipt of the join message, the delegate node registers the client for the requested topic and informs the client to use the new topic coordinate $C(t)$ instead of $h(t)$ in all subsequent messages. In particular, publishers will address all data messages to $C(t)$ instead of $h(t)$. Should the topic be moved for some reason to another node, all registered clients are informed of the new delegate coordinate.

C. Robustness Considerations

Each topic data store has a secondary node ($v1$ and $v3$ in Figure 2) to which it replicates the topic data and control structures. Should a home or delegate node fail, the secondary seamlessly takes over, since it is now the node nearest to the coordinates in question. It can now start replicating to a new secondary because it has already been pre-populated with appropriate topic data and metadata. A returning primary must check with its secondary before resuming operations. The secondary then becomes a delegate for topic data that was stored during the primary's absence. It is not necessary to copy the interim data back to the primary, unless other factors, such as load balancing, make the shift of data desirable. The adjacent delegate shifting mechanism can then facilitate an orderly, efficient, and gradual shift of topic data under home's direction, even after cascading node failures.

IV. Load Definitions for SeDAX Nodes and Coordinates

We now propose load metrics for different resilience levels. We first introduce auxiliary functions that facilitate later definitions and consider three different resilience levelsfor the operation of SeDAX. We define load metrics for SeDAX nodes and coordinates, based on which we determine a SeDAX node's best coordinate. These concepts are used by the coordinate selection and load balancing algorithms in Section V and Section VI.

A. Auxiliary Functions

The following auxiliary functions facilitate the formulation of subsequent definitions and formulae.

- $\mathcal{V}$: set of SeDAX nodes.
- $\mathcal{T}$: set of topics.
- $C(v), v \in \mathcal{V}$: coordinate of node v.
- $C(t), t \in \mathcal{T}$: (delegate) coordinate of topic t.
- $N_j(c)$: node whose coordinate is j-closest to coordinate c among all other SeDAX nodes, e.g., $N_1(c)$ is the closest node, $N_2(c)$ is the second-closest, etc.
- $\mathcal{T}_j(v) = \{t : t \in \mathcal{T}, N_j(C(t)) = v\}$; set of topics for which v is the j-closest node.
- $L_T(t), t \in \mathcal{T}$: load of topic t.

Since topic data may expire, SeDAX nodes require only sufficient capacity to store current, i.e., non-expired, topic data, and are not intended for archival purposes. Therefore, limited storage is sufficient for the data of a topic $t \in \mathcal{T}$ which is given by the topic load $L_T(t)$.

B. Considered Resilience Levels

We consider three different resilience levels for SeDAX operation.

1) No resilience. Topic data and topic information are stored only on SeDAX node $N_1(C(t))$. If the node fails, the topic information is lost.
2) Resilience against one node failure. Topic data and information are stored redundantly on two SeDAX nodes $N_1(C(t))$ and $N_2(C(t))$. If $N_1(C(t))$ fails, messages are automatically rerouted to $N_2(C(t))$ so that they can be forwarded to the registered subscribers. If both $N_1(C(t))$ and $N_2(C(t))$ fail, the topic information is lost and publishers cannot longer reach a broker.
3) Resilience against two node failures. Topic information is stored redundantly on two SeDAX nodes $N_1(C(t))$ and $N_2(C(t))$ like above. If $N_1(C(t))$ fails, messages are automatically rerouted to $N_2(C(t))$ so that they can be forwarded to the registered subscribers. In addition, if $N_1(C(t))$ or $N_2(C(t))$ fails, topic data and information are copied to SeDAX node $N_3(C(t))$. Should the remaining node $N_1(C(t))$ or $N_2(C(t))$ also fail, then $N_3(C(t))$ takes over.

More than two successive node failures are repetitions of the two node failure scenario.

C. Load Definitions

We provide definitions for a topic's *load on a SeDAX node* and the *load on a coordinate* for different resilience levels. While the node loads serve to quantify load imbalance among nodes, the coordinate loads are used to find appropriate coordinates for load balancing.

1) Topic Load $L_T(t)$: Each topic $t \in \mathcal{T}$ induces a certain load $L_T(t)$ on the node on which it is stored. As an alternative to storage capacity, load may be measured in terms of required processing power or I/O capacity if these quantities are the limiting system resource. To facilitate further considerations and calculations, we assume the topic load to be an additive metric.

2) Node Load $L^i_N(v)$: The node load $L^i_N(v)$ is the maximum load on a node $v \in \mathcal{V}$ induced by topics in any failure scenario considered by resilience level i. It is the minimum capacity for v to guarantee operation on resilience level i without capacity shortage.

a) Resilience Level 1: A SeDAX node v is responsible only for topics $t \in \mathcal{T}$ for which it is the closest node. The maximum load induced by topics on this node is

$$L^1_N(v) = \sum_{t \in \mathcal{T}_1(v)} L_T(t). \tag{1}$$

b) Resilience Level 2: A SeDAX node v is responsible for topics for which it is the closest or second-closest node. The maximum load induced by topics on this node is

$$L^2_N(v) = \sum_{t \in (\mathcal{T}_1(v) \cup \mathcal{T}_2(v))} L_T(t). \tag{2}$$

c) Resilience Level 3: As above, a SeDAX node v is responsible for topics for which it is the closest or second-closest node; the resulting base load is $L^2_N(v)$. With resilience level 3, node v becomes responsible for additional topics for which it is third-closest if their closest or second-closest node fails. The imposed load depends on the failure of a specific primary or secondary node $x \in \mathcal{V}$. Therefore, we determine the maximum load over all relevant single node failures. The failure of a specific node $x \in \mathcal{V}$ is relevant only if it is closest or second-closest for a topic $u \in \mathcal{T}$, i.e., $N_1(C(u)) = x$ or $N_2(C(u)) = x$, for which the considered node v is third-closest, i.e., $\{u \in \mathcal{T}_3(v)\}$. Thus, the maximum additional load imposed on node v in case of a node failure is:

$$L^3_{aN}(v) = \max_{w \in \left\{ \substack{x: x \in \mathcal{V}, u \in \mathcal{T}_3(v), \\ N_1(C(u))=x \vee \\ N_2(C(u))=x} \right\}} \sum_{t \in \left\{ \substack{s: s \in \mathcal{T}_3(v), \\ N_1(C(s))=w \vee \\ N_2(C(s))=w} \right\}} L_T(t) \tag{3}$$

and the node load for resilience level 3 is

$$L^3_N(v) = L^2_N(v) + L^3_{aN}(v). \tag{4}$$

3) Minimum and Maximum Coordinate Load ($L^i_{min}(c)$ and $L^i_{max}(c)$): We define the minimum (maximum) load of a coordinate c as the minimum (maximum) of all node loads that are affected by topics assigned to coordinate c.

a) Resilience Level 1: A topic assigned to coordinate c is stored only on the closest node $N_1(c)$ so that $N_1(c)$ stores only information of topics for which it is closest node. The (minimum and maximum) coordinate load is

$$L^1_{min}(c) = L^1_{max}(c) = L^1_N(N_1(c)). \tag{5}$$

b) Resilience Level 2: A topic assigned to coordinate c is stored on the closest node $N_1(c)$ and on the second-closest node $N_2(c)$. These nodes store the information of topics for which they are closest or second-closest. The coordinate loads are

$$L^2_{min}(c) = \min\left(L^2_N(N_1(c)), L^2_N(N_2(c))\right) \text{ and} \tag{6}$$

$$L^2_{max}(c) = \max\left(L^2_N(N_1(c)), L^2_N(N_2(c))\right). \tag{7}$$

c) Resilience Level 3: Like above, a topic assigned to coordinate c is stored on the closest node $N_1(c)$ and on the second-closest node $N_2(c)$. The maximum load of those nodes is $L^3_N(N_1(c))$ and $L^3_N(N_2(c))$. Moreover, the topic may be stored on the third-closest node $N_3(c)$ if $N_1(c)$ or $N_2(c)$ fails. That node $N_3(c)$ carries the load $L^2_N(N_3(c))$ from topics for which it is closest or second-closest node. If $N_1(c)$ or $N_2(c)$ fails, node $N_3(c)$ carries in addition the load from all topics that have $N_1(c)$ or $N_2(c)$ as closest or second-closest node, and $N_3(c)$ as third-closest node. Thus, the failure-set-specific additional node load $L^3_{faN}(N_3(c), c)$ of $N_3(c)$ for coordinate c is

$$L^3_{faN}(N_3(c), c) = \max_{w \in \{N_1(c), N_2(c)\}} \sum_{t \in \left\{ \substack{s: s \in \mathcal{T}_3(N_3(c)), \\ N_1(C(s))=w \vee \\ N2(C(s))=w} \right\}} L_T(t). \tag{8}$$

Hence, the coordinate loads are

$$\begin{aligned} L^3_{min}(c) = {} & \min(L^3_N(N_1(c)), L^3_N(N_2(c)), \\ & L^2_N(N_3(c)) + L^3_{faN}(N_3(c), c)) \text{ and} \end{aligned} \tag{9}$$

$$\begin{aligned} L^3_{max}(c) = {} & \max(L^3_N(N_1(c)), L^3_N(N_2(c)), \\ & L^2_N(N_3(c)) + L^3_{faN}(N_3(c), c)). \end{aligned} \tag{10}$$

D. Definition of Best Coordinates

We define C^* as a set of coordinates. If a new topic should be assigned to a coordinate from that set, the coordinate should be carefully selected such that it minimizes the maximum load of all nodes, maximizes the minimum load of all nodes, and minimizes the required backup capacity. This translates to the following three criteria based on the metrics L^i_{max}, L^i_{min}, and coordinate-specific spare capacity:

1) Select a coordinate with a small maximum coordinate load L^i_{max}.
2) Select a coordinate with a small minimum coordinate load L^i_{min}.
3) Only for resilience level 3: select a coordinate with a large coordinate-specific spare capacity on the coordinate's third-closest node $N_3(c)$. It is the spare capacity on $N_3(c)$ if either the closest node $N_1(c)$ or the second-closest node $N_2(c)$ fails. That capacity is calculated as $L^3_N(N_3(c)) - (L^2_N(N_3(c)) + L^3_{faN}(N_3(c), c))$.

We define that a coordinate c_0 is better than a coordinate c_1 if it is better in the first criterion (small $L^i_{max}(c)$). Or if it is equal in the first criterion but better in the second one (small $L^i_{min}(c)$). Or if it is equal in the first two criteria and better in the third one (coordinate-specific spare capacity). A coordinate of a coordinate set C^* is best if there is no better coordinate in that set. Thus, several best coordinates may exist. These criteria combine the best heuristics in our experiments.

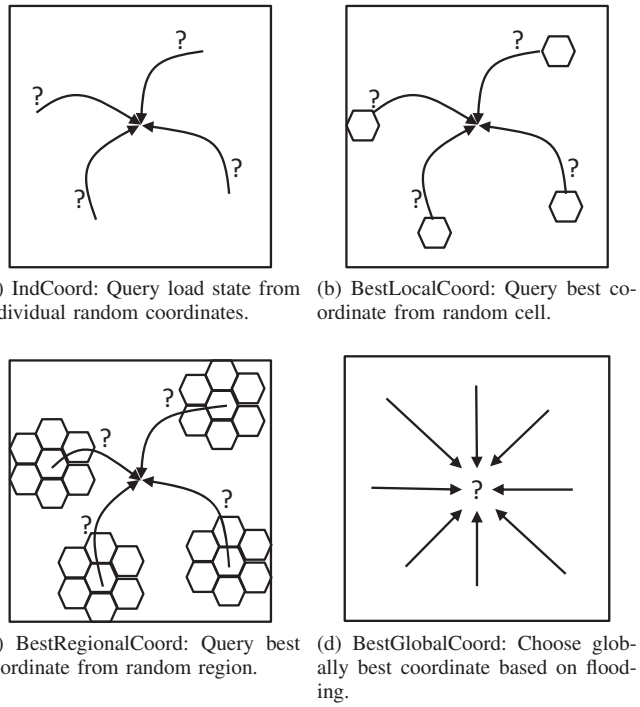
For resilience level 1, all coordinates of a Voronoi cell $Voronoi(v)$ of a node $v \in \mathcal{V}$ are equally good. This is different for resilience level 2 and 3. Here, a mathematical analysis yields the area of best coordinates. Alternatively, a best (or at least a good) coordinate may be found empirically by selecting the best coordinate of a set of random coordinates within a node's Voronoi cell. This is much simpler, but may not find the absolute best coordinate.

V. Distributed Coordinate Selection Algorithms

We present four different algorithms for distributed coordinate selection in SeDAX that support resilience levels 1, 2, and 3. If a node v wants to delegate a topic with a coordinate $C(t) \in Voronoi(v)$ within its own Voronoi cell to another coordinate, we call it a delegating node. This delegating node needs to find a better coordinate according to the definitions in Section IV-D. Load metrics in this section should be computed excluding the topic to be delegated.

A. Querying for Individual Coordinates (IndCoord)

A delegating node may send a query to a random coordinate c that is forwarded to its closest node $N_1(c)$ over the DT overlay. This node locally computes the metrics L^i_{max}, L^i_{min}, and coordinate-specific spare capacity as proposed in Section IV-D and returns them to the delegating node. The delegating node may issue $n_{queries}$ such queries so that it eventually knows the relevant loads of $n_{queries}$ other coordinates and the load $L_T(t)$ of the topic to be delegated. On this basis the delegating node can choose the best coordinate and assign the topic. This method is illustrated in Figure 3a.



(a) IndCoord: Query load state from individual random coordinates.

(b) BestLocalCoord: Query best coordinate from random cell.

(c) BestRegionalCoord: Query best coordinate from random region.

(d) BestGlobalCoord: Choose globally best coordinate based on flooding.

Fig. 3. Algorithms for finding a delegation coordinate.

B. Querying Locally Best Coordinates (BestLocalCoord)

This differs from IndCoord in that the node $N_1(c)$ determines a locally best coordinate within its Voronoi cell $Voronoi(v)$ according to Section IV-D. It returns that coordinate including the relevant metrics to the delegating node. Thus, the delegating node receives $n_{queries}$ locally best coordinates and also computes its own locally best coordinate. The topic is assigned to the best coordinate among them. This method is illustrated in Figure 3b. It causes more computational overhead than IndCoord, but it is likely to find better coordinates.

C. Querying Regionally Best Coordinates (BestRegionalCoord)

Here, the node receiving the query returns a regionally best coordinate selected from the coordinates of its own cell and of those cells within n_{hops} hops. Thus, the delegating node receives $n_{queries}$ regionally best coordinates and also computes its own regionally best coordinate. The topic is assigned to the best coordinate among those. This method is illustrated in Figure 3c. It causes more computational overhead and involves more communication than the methods presented above, but it is more likely to find better coordinates.

D. Determining Globally Best Coordinates Based on Flooding (BestGlobalCoord)

The delegating node floods a request to all other nodes (or at least one node in each region) for their best coordinates. The responses allow the delegating node to determine a globally best coordinate to which the topic is assigned. This method is illustrated in Figure 3d. It may require more computation

Algorithm 1 Load-balanced topic addition.

Require: balanced SeDAX network, $t \notin \mathcal{T}$
1: $C(t) \leftarrow h(t)$ {Geographical hash of topic t.}
2: $C^* \leftarrow$ Distributed coordinate selection
3: $c_{cand}^{best} \leftarrow$ Tiebreaker(C^*) {Best coordinate from C^*.}
4: **if** c_{cand}^{best} is better than $C(t)$ **then**
5: $\quad C(t) \leftarrow c_{cand}^{best}$ {Delegate t to $c_{cand}^{best} \in C^*$.}
6: **end if**
7: $\mathcal{T} \leftarrow \mathcal{T} \cup t$ {Add topic to SeDAX network.}
8: **return** balanced SeDAX network.

Algorithm 2 Continuous load balancing every $\Delta\tau$.

Require: node $v \in \mathcal{V}$, resilience level i
1: $t_{min}^v \leftarrow \underset{t \in \bigcup_{j=1}^{i} \mathcal{T}_j(v)}{\arg\min} L_T(t)$ {Smallest topic of v.}
2: $C^* \leftarrow$ Distributed coordinate selection
3: $c_{cand}^{best} \leftarrow$ Tiebreaker(C^*) {Best coordinate from C^*.}
4: **if** c_{cand}^{best} is better than $C(t_{min}^v)$ **then**
5: $\quad C(t_{min}^v) \leftarrow c_{cand}^{best}$ {Delegate t_{min}^v to $c_{cand}^{best} \in C^*$.}
6: **end if**
7: **return** more balanced SeDAX network.

and communication than the methods presented above, but it is able to find a network-wide best delegation coordinate given the current network state.

E. Implementation Aspects

In our simulation implementation, we use a heuristic coordinate generation approach to produce best coordinates for querying nodes. We generate 200 random coordinates per SeDAX node which must lie in the Voronoi cell of the respective node and which serve as best coordinate candidates. We calculate the load metrics for each coordinate according to the desired resilience level and cache them. When a node is queried by another node, it returns its best coordinate according to the preferred selection mechanism. Cache refresh is necessary when topics are added to or removed from the system, and when topics are delegated from one coordinate to another. Appropriate data structures allow for a significant reduction of recalculations for the latter because only affected nodes at the delegation source and destination have to refresh the load metrics of their best coordinate candidates.

VI. Distributed Load Balancing

We distinguish two types of load balancing for SeDAX: load-balanced topic addition and continuous load balancing. In the following, we show the basic steps for each type and briefly discuss the differences. A combined version of the two approaches is briefly presented at the end of this section.

A. Load-balanced Topic Addition

Load balancing by *load-balanced topic addition* means that best coordinates for new topics are determined before the actual topic addition to the overlay. Topic coordinates are chosen so that new topics have minimal negative impact on the load imbalance on the overlay.

Algorithm 1 shows the simplified steps that are necessary when a topic t is added to a balanced overlay. Regardless of whether a topic will be delegated or not, its home coordinate $h(t)$ and original coordinate $C(t)$ is calculated via geographic hashing. A set C^* of best delegation coordinates is constructed using one of the distributed coordinate selection algorithms described in Section V. Applying a tie-breaker of Section IV-D on this set yields the best coordinate c_{cand}^{best}. Topic t is delegated to c_{cand}^{best} if c_{cand}^{best} is better than $C(t)$. Otherwise, topic t is stored at $C(t)$. Eventually, topic t is added to the set $\mathcal{T}$ of SeDAX topics.

When topic loads remain static, any topic addition to a balanced overlay leads to a still balanced overlay with only minimal signaling effort because at most the newly added topic is delegated. When topic loads change, the optimized yet static topic assignment may lead to load imbalance on the overlay. We will investigate the effects of changing topic loads on an initially balanced SeDAX network in Section VIII.

B. Continuous Load Balancing

In contrast, *continuous load balancing* means that balancing decisions are made during system runtime. Each SeDAX node has a dedicated load balancer process that is triggered from time to time. The load balancer tries to shift topic load away from its node and then waits for another $\Delta\tau$. This process runs on each SeDAX node and does not require additional synchronization for triggering. This fully distributed approach allows the overlay to react on topic load changes.

Algorithm 2 shows the simplified steps when load balancing is triggered after $\Delta\tau$ for node v and resilience level i. The rationale is discharging nodes by delegating their smallest topics to other less-loaded nodes. We select the smallest topic t_{min}^v of all topics for which v is responsible for according to the desired resilience level; this may include topics for which node v is closest, second-closest or third-closest depending on the resilience level. The best coordinate c_{cand}^{best} for a potential topic delegation is determined analogously to Algorithm 1. If delegating t_{min}^v to c_{cand}^{best} decreases the node load imbalance compared to the original coordinate $C(t_{min}^v)$, t_{min}^v is delegated to c_{cand}^{best}. Otherwise, t_{min}^v remains at $C(t_{min}^v)$. Finally, the load balancer waits for the next load balancing trigger event after $\Delta\tau$.

C. Combined Approach

Load-balanced topic addition and continuous load balancing can be integrated into a *combined approach* to take advantage of the benefits of both approaches. In such a combined approach, load-balanced topic addition minimizes the impact of new topics to node load imbalance, and continuous load balancing adapts the topic coordinates to changing topic loads. We will investigate the impact of $\Delta\tau$ on load balancing quality and signaling effort when applying the combined approach to a SeDAX network in Section VIII.

VII. Evaluation for Static Topic Sizes

This section investigates potential load imbalance in SeDAX overlays for static topic sizes by simulation experiments.

TABLE I
MEAN VALUE $\bar{x}$, 5% AND 95% QUANTILES OF NODE LOAD $L_N^i(v)$ FOR $n_{nodes} = 100$ WITHOUT TOPIC DELEGATION.

	$n_{node}^{topics} = 1000$			$n_{node}^{topics} = 100$						$n_{node}^{topics} = 10$					
	homogeneous topic loads			homogeneous topic loads			heterogeneous topic loads			homogeneous topic loads			heterogeneous topic loads		
	$q_{5\%}$	$\bar{x}$	$q_{95\%}$	$q_{5\%}$	$\bar{x}$	$q_{95\%}$	$q_{5\%}$	$\bar{x}$	$q_{95\%}$	$q_{5\%}$	$\bar{x}$	$q_{95\%}$	$q_{5\%}$	$\bar{x}$	$q_{95\%}$
L_N^1	29.5%	100.0%	194.7%	24.8%	100.0%	192.5%	24.7%	100.0%	197.3%	12.0%	100.0%	204.0%	7.1%	100.0%	239.7%
L_N^2	85.0%	200.0%	332.9%	80.2%	200.0%	335.3%	78.6%	200.0%	345.7%	60.8%	200.0%	349.6%	37.3%	200.0%	391.7%
L_N^3	122.6%	257.4%	411.6%	117.3%	258.0%	410.6%	113.5%	259.2%	419.2%	102.0%	262.9%	433.5%	78.2%	271.3%	488.8%

First, the simulation setup is described. The complementary cumulative distribution functions (CCDFs) of node loads illustrate that the existing SeDAX can lead to significant load imbalance for which we analyze the causes. We show that load-balanced topic addition based on global information can equalize the load among all nodes and highlight the importance of respecting the resilience level for load balancing. As global information may be difficult to obtain, we show that simpler coordinate selection approaches can also lead to good load balancing results.

A. Experiment Setup and Methodology

We use a square plane as coordinate space on which n_{nodes} nodes are positioned randomly. Each node is assigned n_{node}^{topics} topics on average. We generate $n_{topics} = n_{node}^{topics} \cdot n_{nodes}$ topics, and each t of these topics comes with a random coordinate $h(t)$. These topics are iteratively added to SeDAX. When load-balanced topic addition is enabled, a load balancer may reassign each topic to a different coordinate $C(t)$ depending on the current load situation in the overlay; otherwise the original random topic coordinates remain.

Typical parameters are not known and relevant parameter ranges will depend on the application scenario. Therefore, we consider a wide range of parameters so that relevant parameter ranges are covered.

We study three choices for static topic loads.

- Homogeneous topic load: each topic has the same load $L_T = 1$.
- Heterogeneous topic load: 80% of the topics have load $L_T = \frac{1}{4}$, 20% of the topics have load $L_T = 4$. This distribution yields also an average load $E[L_T] = 1$ and its coefficient of variation is 1.5.
- Exponentially distributed topic sizes: $L_T(i) = e^{\lambda \cdot \frac{i}{(n-1)}}$ with $0 \le i < n$. Their mean and coefficient of variation is $E[L_T] = 3.9247$ and 0.6472 for $n = 100$ different topics and $\lambda = 2.3026$. We use that model in Section VII-D. Parameter λ is chosen that the topic sizes equal those of the topic growth model in Section VIII.

After the successive generation of topics, assignment to coordinates, and load balancing, node loads are calculated for all nodes $v \in \mathcal{V}$ and the CCDF of these loads is determined. We perform each experiment 100 times, average the CCDFs from single simulation runs, and show 95% confidence intervals where appropriate. We use the same seeds for all corresponding experiments to make the simulation results comparable with each other. The quantiles in the following evaluations are derived from the averaged CCDFs.

TABLE II
CORRELATION COEFFICIENTS BETWEEN VORONOI CELL SIZE $A(v)$ AND NODE LOAD $L_N^i(v)$ FOR $n_{nodes} = 100$.

$corr$	$n_{node}^{topics} = 1000$	$n_{node}^{topics} = 100$		$n_{node}^{topics} = 10$	
	homogeneous	homo-geneous	hetero-geneous	homo-geneous	hetero-geneous
L_N^1	0.9984	0.9844	0.9522	0.8680	0.6996
L_N^2	0.6830	0.6740	0.6528	0.6026	0.4914
L_N^3	0.6028	0.5954	0.5770	0.5336	0.4365

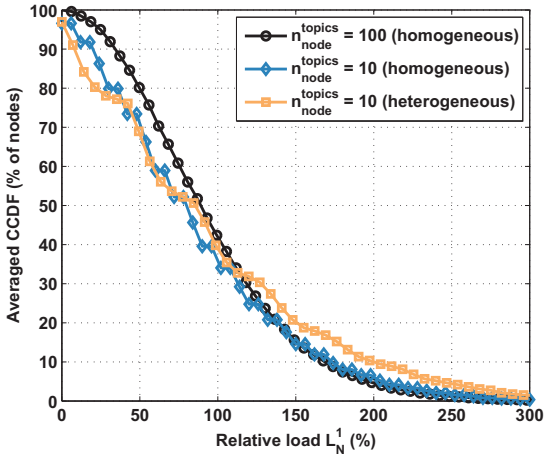


Fig. 4. CCDFs of the node loads L_N^1 for resilience level 1 demonstrate a significant load imbalance. The experiments were conducted with $n_{nodes} = 100$ using homogeneous and heterogeneous topic loads. The CCDFs are averaged over 100 simulation runs.

B. Load Distribution without Topic Delegation

We first investigate the load distribution in SeDAX without load balancing. We simulate $n_{nodes} = 100$ nodes in the plane with $n_{node}^{topics} \in \{1000, 100, 10\}$ homogeneous-load topics per node on average and $n_{node}^{topics} \in \{100, 10\}$ heterogeneous-load topics per node on average. Figure 4 shows the CCDF of the node loads $L_N^1(v)$ for resilience level 1 for $n_{node}^{topics} \in \{100, 10\}$. The curve for $n_{node}^{topics} = 1000$ is omitted in the figure as it visually coincides with the curve for $n_{node}^{topics} = 100$. Node loads are relative, i.e., 100% relative load corresponds to a node load of n_{node}^{topics}. The lines are interpreted as follows: for a node load x on the x-axis, the y-axis gives the percentage of nodes whose node load X is greater than x. In the best case, i.e., perfect load balancing, equal load on any node would result in a vertical line at 100% node load. In the worst case, all load is handled by a single node although many other nodes

are available, e.g., this would result in a horizontal line starting at 1% on the y-axis with an abrupt drop at 10000% relative load $L_N^1(v)$ for $n_{nodes} = 100$. However, this worst case is unlikely and the simulation yields the CCDF of the load per node: a continuous decrease over a load range between 0% and 250% is observed for $n_{node}^{topics} = 100$. The curves for $n_{node}^{topics} = 10$ homogeneous-load topics have a slightly greater load imbalance which increases for heterogeneous-load topics.

Figure 5a shows in addition to the distribution of node load L_N^1 the distribution of node loads L_N^2 and L_N^3, i.e., the loads for resilience levels 2 and 3. The loads are significantly larger than the load of resilience level 1. While the L_N^1 loads have a mean of 100%, the L_N^2 loads have a mean of 200% because each topic has to be stored twice, and they range between 0% and 450% per node. The L_N^3 loads have a mean of about 260% and range between 0% and 550%. The mean of the L_N^3 load is less than 300% because topics can share the normally unused backup capacity of SeDAX nodes if they have different primary and secondary nodes. As exact values for load imbalance are hard to determine from the figures, Table I shows the 5% and 95% quantiles of the loads. We observe that the relative load imbalance increases with fewer topics per node and with increasing variance of topic loads. Furthermore, these values increase with increasing resilience level. The 95% quantiles may be useful for capacity provisioning. They can easily amount to 200% – 250% of the respective mean values. This is highly inefficient but necessary in the absence of load balancing capabilities because pure SeDAX cannot resolve such bottlenecks.

A good part of the strong load imbalance is caused by the strong imbalance of the Voronoi cell sizes. The average Voronoi cell size is $\frac{A_{square}}{n_{nodes}}$, where A_{square} is the area of the coordinate space in our experiment. If we take this as 100%, the 5% and 95% quantile of the cell sizes is 29.0% and 191.4%. This is very close to the quantiles of the load distribution with $n_{node}^{topics} = 1000$ homogeneous-load topics. Table II shows the correlation coefficients between the Voronoi cell size and the load of SeDAX nodes for different topic loads and resilience levels. We observe high correlations for all cases. The correlation is largest for resilience level 1 and 1000 homogeneous-load topics per node, and decreases for fewer topics per node, heterogeneous topic loads, and higher resilience levels. Thus, the observed load imbalance is largely due to different cell sizes. *

C. Load Distribution with Topic Delegation

We now examine the impact of load-balanced topic addition and the various coordinate selection algorithms presented in Section V on the load balancing outcome.

1) Load-Balanced Topic Addition Using Global Knowledge: We first investigate load-balanced topic addition using coordinate selection based on global knowledge as proposed in Section V-D. We add topics one after another to SeDAX and perform a load balancing decision for each new topic, i.e., whether it should be assigned to its default coordinate $C(t) = h(t)$ or to another recommended coordinate $C(t)$.

To validate the correctness of the load balancing results for load balancing goal L_N^3, we check that the following equation is met after the assignment of a topic with load $L_T^{assigned}$:

$$\min_{c \in \mathcal{C}}(L^3_{max,old}(c)) \leq \max(\max_{v \in \mathcal{V}}(L^3_{N,new}(v)) - L_T^{assigned}, L^3_{N,new}(N_3(c_{assigned}))) \quad (11)$$

The subscripts "old" and "new" in the equation refer to the respective metric before and after topic addition, and $L_T^{assigned}$ and $c_{assigned}$ refer to the load and the coordinate of the last assigned topic.

In the following, we perform load-balanced topic addition with various objectives, namely to equalize the L_N^1, L_N^2, or L_N^3 load.

a) Equalizing L_N^1 Node Load: Figure 5b illustrates the CCDF of the node loads L_N^1, L_N^2, and L_N^3 when topics are load-balanced for L_N^1. The L_N^1 load is well balanced over all nodes and the maximum L_N^1 load is near 100%. However, the L_N^2 load ranges between 100% and 500%. Thus, this simple load balancing approach does not lead to equalized data volumes on SeDAX nodes when SeDAX is operated under failure-free conditions in a resilient mode. For resilience level 3, the node load also ranges between 100% and 500%.

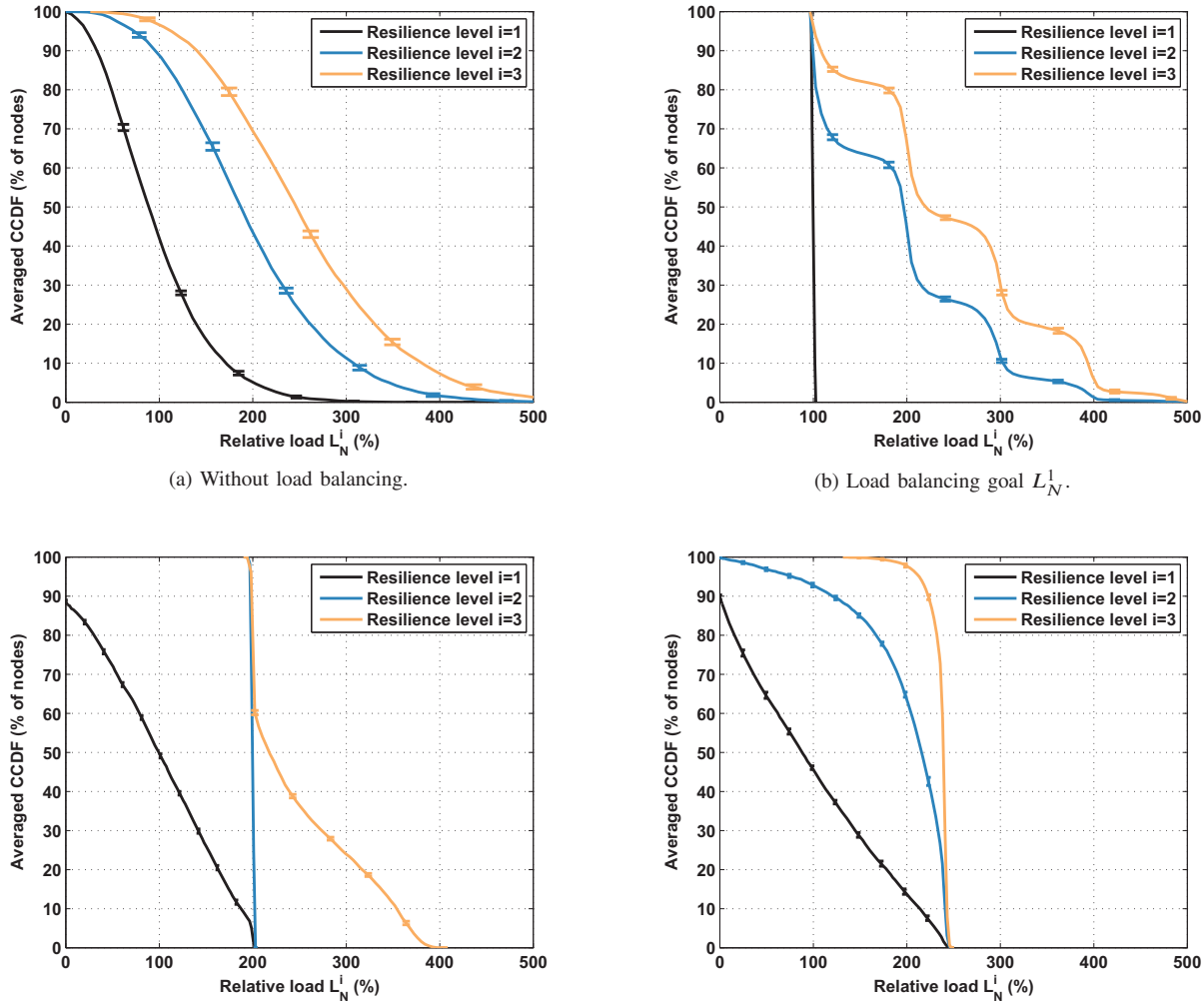
b) Equalizing L_N^2 Node Load: Figure 5c shows the respective results when L_N^2 is used as load balancing goal. The L_N^1 load is almost equally distributed between 0% and 200% which is far from being equally balanced. However, the L_N^2 load is well equalized among all nodes, which is the balancing goal. That means, the data volumes on SeDAX nodes are about the same on all nodes when SeDAX is operated under failure-free conditions in a resilient mode. The CCDF of the L_N^3 load shows the distribution of the maximum node load during single node failures. During single node failures, heavy load spikes in terms of additional load from other topics can occur on nodes with values ranging from 200% to 400%.

c) Equalizing L_N^3 Node Load: Figure 5d presents the load distribution for load balancing objective L_N^3. The L_N^1 load is approximately uniformly distributed between 0% and 240% whereas the L_N^2 load is not. The maximum load node L_N^3 is about 240%; this means that no SeDAX node carries much more than 240% even during single node failures. This is a desirable feature even though the distribution of the actual load under failure-free operation is far from being equalized.

These investigations demonstrate that the load balancing objective for SeDAX needs to be carefully chosen. The simple L_N^1 load balancing goal cannot equalize the load of resilient SeDAX under failure-free conditions. The more complex L_N^2 load balancing goal achieves that objective, but cannot avoid load spikes during single node failures. Only the more complex L_N^3 load balancing goal is able to minimize load spikes during single node failures.

2) Load-Balanced Topic Addition Using Limited Knowledge: Load balancing with coordinate selection based on global knowledge requires the calculation of the best coordinates of all SeDAX nodes and their communication to the load balancing node. That can be expensive in networks with

*We also conducted experiments with more and fewer nodes, but the results are so similar that we omit them here.



(a) Without load balancing.

(b) Load balancing goal L_N^1.

(c) Load balancing goal L_N^2.

(d) Load balancing goal L_N^3.

Fig. 5. CCDF of node loads L_N^1, L_N^2, and L_N^3 without load balancing and for load balancing goals L_N^1, L_N^2, and L_N^3. The experiments were conducted with $n_{nodes} = 100$, $n_{node}^{topics} = 100$ heterogeneous-load topics per node. The CCDFs are averaged over 100 simulation runs with 95% confidence intervals.

many nodes and topics, so it is important to explore coordinate selection approaches that require less effort.

In the following, we examine the impact of the various coordinate selection algorithms presented in Section V on the load balancing outcome. We focus on balancing of the L_N^3 load with $n_{nodes} = 100$ nodes and $n_{node}^{topics} = 100$ heterogeneous-load topics. All investigated approximation algorithms are based on the principle of random queries. In all experiments, we use $n_{queries} = \{1, 10, 100\}$ queries per topic delegation decision, and perform load-balanced topic addition.

Table III shows the mean L_N^3 load, the 5% and the 95% quantiles of the averaged CCDFs of the experiments including the values without load balancing from Table I for comparison. The simulation results show that *all* selection algorithms can limit the 95% load quantile to about 240% while the 95% load quantile without load balancing is 419%, i.e., they reduce the 95% quantile of the load by as much as $\frac{419.2\%-237.9\%}{419.2\%} \approx 43\%$. However, IndCoord and BestLocalCoord require at least $n_{queries} = 10$ to achieve good results but that is feasible. Hence, distributed load balancing yields similar results as load balancing with global knowledge (BestGlobalCoord), but is more scalable. Nevertheless, all presented approaches are

TABLE III
IMPACT OF COORDINATE SELECTION ALGORITHMS AND $n_{queries}$ ON MEAN VALUE $\bar{x}$, 5% AND 95% QUANTILES OF NODE LOAD L_N^3.

	IndCoord			BestLocalCoord		
$n_{queries}$	$q_{5\%}$	$\bar{x}$	$q_{95\%}$	$q_{5\%}$	$\bar{x}$	$q_{95\%}$
–	113.5%	259.2%	419.2%	113.5%	259.2%	419.2%
1	176.8%	254.0%	282.3%	175.9%	245.3%	312.7%
10	225.5%	245.8%	251.1%	230.6%	235.6%	237.9%
100	234.2%	237.7%	239.1%	223.8%	235.4%	238.4%
	BestRegionalCoord			**BestGlobalCoord**		
$n_{queries}$	$q_{5\%}$	$\bar{x}$	$q_{95\%}$	$q_{5\%}$	$\bar{x}$	$q_{95\%}$
–	113.5%	259.2%	419.2%	113.5%	259.2%	419.2%
1	226.8%	235.8%	239.4%	213.5%	236.2%	243.3%
10	226.2%	235.1%	238.4%	213.5%	236.2%	243.3%
100	214.6%	236.1%	242.0%	213.5%	236.2%	243.3%

TABLE IV
DISTRIBUTION OF NODE LOAD L_N^3 FOR EXPONENTIALLY DISTRIBUTED TOPIC SIZES AND VARYING TOPIC ADDITION ORDER.

Topic addition order	$q_{5\%}$	$\bar{x}$	$q_{95\%}$
Ascending size	207.7%	248.6%	264.6%
Descending size	205.7%	248.6%	257.1%
Random size	207.0%	248.3%	261.3%

heuristics. The general load balancing problem maps to the NP-hard 0/1 knapsack problem and all proposed algorithms greedily assign topics to coordinates, one after another. Therefore, none of the results is likely to be fully optimal.

The fact that coordinate selection algorithms with limited knowledge can outperform the coordinate selection algorithm with global knowledge seems surprising. By incrementally equalizing existing load before adding large topics, BestGlobalCoord can cause load spikes on a few nodes. In contrast, coordinate selection algorithms with limited knowledge equalize the load for only a limited set of coordinates, leading to a globally imperfect balance with larger load differences between coordinates. This leaves room for larger topics to be more evenly distributed, since when a large topic is assigned, the probability of a coordinate having significantly less load is larger than for BestGlobalCoord. Although this helps explain the observed phenomena, it also hints that future research can further improve coordinate selection algorithms, particularly for the investigation of load balancing in larger networks.

D. Impact of Topic Addition Order on Load Distribution

Finally, we investigate the impact of the topic addition order on the load balancing result. We simulate $n_{nodes} = 10$ and $n_{node}^{topics} = 10$ topics per node on average. We use exponentially distributed topic sizes and perform load-balanced topic addition with BestGlobalCoord as coordinate selection algorithm to balance the node loads in each experiment run. We add the topics in *ascending topic size order*, *descending topic size order*, and *random order*. We perform each experiment 100 times and use the 5% and 95% quantiles of the averaged CCDFs of the experiments to calculate the load imbalance.

Table IV shows the mean L_N^3 load, the 5% and 95% quantiles of the averaged CCDFs of the experiments. We observe that the topic addition order has some effect on the load balancing quality. Adding topics in descending topic size order leads to the best load balancing results because this addition order always leaves room for smaller topics to fill holes in the overlay. Conversely, adding topics in ascending order makes it more challenging for the last few topics to be placed on an already well-balanced overlay without causing some load imbalance. Random topic addition leads to an imbalance between both topic size orders.

VIII. EVALUATION FOR DYNAMIC TOPIC SIZES

In this section, we assume that topic sizes grow over time, some grow slowly and some grow fast. We first present a model for this growth. We use it to illustrate the impact of heterogeneously growing topic sizes on load distribution in SeDAX without any load balancing. Then, we show how heterogeneous topic growth impacts load distribution after load balancing. Finally, we assume that topics are initially added to the system in a load-balanced way and then investigate the impact of continuous load balancing. The latter algorithm has only a single parameter and we illustrate its impact.

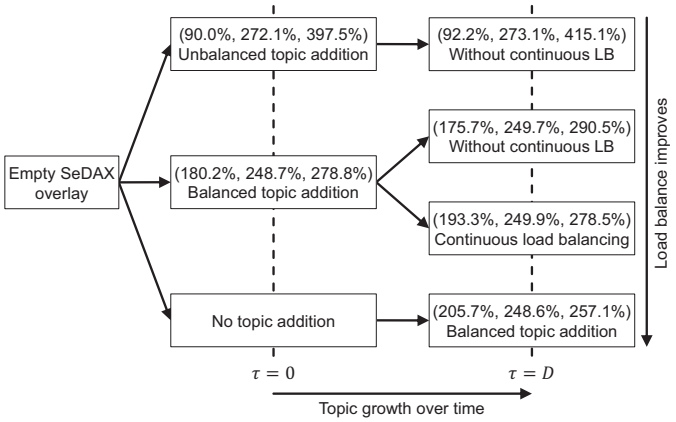


Fig. 6. Distribution of node load L_N^3 for dynamic topic sizes before topic growth (left-hand) and after topic growth (right-hand) under different load balancing configurations. The (x, y, z) values correspond to the 5% quantile, the mean, and the 95% quantile of the averaged CCDFs of the node load L_N^3.

A. Model for Topic Growth

We assume n topics t whose sizes $L_T(t, \tau)$ grow exponentially over time $\tau \in [0, D]$ within an experimentation interval of duration D. Initially, all topics t have equal size $L_T(t, 0) = 1$. They grow with different rates so that the smallest topic is still of size 1 at the end of the experimentation interval and the largest topic is L_T^{max} large. Thus, the largest growth rate is

$$\lambda_{max} = \frac{ln\left(L_T^{max}\right)}{D} \tag{12}$$

while the growth rate of the other topics is linearly spaced within $[0, \lambda_{max}]$. This yields exponentially distributed topic sizes as already used in Section VII-D.

We simulate $n_{nodes} = 10$ nodes and $n_{node}^{topics} = 10$ topics per node on average. We quantify the load imbalance by the mean node load, and the 5% and 95% quantiles of the averaged CCDFs of the node load. We normalize node loads after topic growth for better comparison. That means, for all following experiments 100% normalized load corresponds to the sum of all topic loads $L_T(t, D)$ at the end of the experiment divided by n_{nodes}.

B. Impact of Topic Growth without Load Balancing

We first conduct reference measurements with an unbalanced system. We add all topics to the overlay without balanced topic addition, and then let all topics grow according to the topic growth model. The top row in Figure 6 shows the mean L_N^3 load, the 5% and 95% quantiles of the averaged CCDFs of the experiments before (left-hand) and after (right-hand) topic growth. We observe only minor changes in the mean load from 272.1% to 273.1% but more significant

changes in the load imbalance. The 95% quantile increases from 397.5% to 415.1%. This is because topic growth leads to a heterogeneous topic size distribution and increases the variance of the node loads. This observation is consistent with our initial investigation from Section VII-B and the illustration in Figure 4. We use the values from the top row in Figure 6 as reference for comparison in the remaining part of the performance evaluation.

C. Impact of Topic Growth on Load-Balanced Topic Addition

We now investigate the impact of topic growth on load distribution in a balanced system. In contrast to the previous experiment, we now use load-balanced topic addition with BestGlobalCoord as coordinate selection algorithm to initially balance the node loads in each experiment run. When all topics have been added to the system, we let all topics grow according to the topic growth model without any further load balancing. The middle row in Figure 6 shows the mean L_N^3 load, the 5% and 95% quantiles of the averaged CCDFs of the experiments before (left-hand) and after (right-hand, upper box) topic growth. The mean load changes from 248.7% to 249.7% and the 95% quantile increases from 278.8% to 290.5%. That means, we observe a similar trend of load imbalance change after topic growth like in an unbalanced system. For completeness, we included the results for load-balanced topic addition after topic growth from Section VII-D in the bottom row of Figure 6.

D. Benefits of Continuous Load Balancing

As final experiment, we perform continuous load balancing on initially balanced SeDAX overlays, i.e., effectively a combined approach, showing the influence and tradeoffs of the control parameter $\Delta\tau$ on the load balancing quality. A load balancer is triggered every $\Delta\tau$ for a randomly selected node which may reassign its smallest topic t to a different coordinate $C(t)$ based on the current load situation in the overlay.

1) Impact of Continuous Load Balancing on Load Distribution: The middle row in Figure 6 shows the mean L_N^3 load, the 5% and 95% quantiles of the averaged CCDFs of the experiments before (left-hand) and after (right-hand, lower box) topic growth for $\Delta\tau = 0.01$. The mean load changes from 248.7% to 249.9% and the 95% quantile decreases from 278.8% to 278.5%. This is a significant improvement compared to the previous experiments, and demonstrates that our algorithm keeps the load imbalance constant over time for heterogeneously growing topics.

We now investigate the impact of different $\Delta\tau$ on load balancing quality and the necessary communication effort.

2) Impact of $\Delta\tau$ on Load Balancing Quality: Table V shows the mean L_N^3 load, the 5% and 95% quantiles of the averaged CCDFs of the experiments for $\Delta\tau = \{0.1, 0.01, 0.001\}$. For easier comparison, we include the results from the experiments without continuous load balancing in the first row of the table. We observe that the load imbalance improves for smaller values of $\Delta\tau$. Compared to our result from Section VIII-C, the 95% quantile improves by $\frac{290.5\%-276.0\%}{276.0\%} \approx 5\%$ but falls behind the heuristically achievable load balancing results for exponentially distributed topic sizes in Table IV by $\frac{276.0\%-257.1\%}{276.0\%} \approx 7\%$. Nevertheless, the results for continuous load balancing are a good indicator that the proposed mechanisms can well handle dynamic topic load changes.

TABLE V
IMPACT OF CONTROL PARAMETER $\Delta\tau$ ON MEAN VALUE $\bar{x}$, 5% AND 95% QUANTILES OF NODE LOAD L_N^3.

$\Delta\tau$	$q_{5\%}$	$\bar{x}$	$q_{95\%}$
–	175.7%	249.7%	290.5%
0.1	188.6%	249.2%	282.8%
0.01	193.3%	249.9%	278.5%
0.001	199.0%	249.8%	276.0%

3) Impact of $\Delta\tau$ on Moved Load Rate: Finally, we investigate the impact of $\Delta\tau$ on the *moved load rate* R_{ML}. This allows quantifying the tradeoff between improving load balancing and minimizing the moved load. We first give the necessary definitions to quantify R_{ML}.

Let $\delta(\tau) = [\delta_1(\tau), \delta_2(\tau), \ldots, \delta_n(\tau)]$ be a vector of size $||T||$. Each $\delta_t(\tau)$ equals 1 if topic t was delegated at time τ; otherwise $\delta_t(\tau)$ equals 0. The *volume of moved load* $V_{ML}(\tau)$ at time τ is defined as

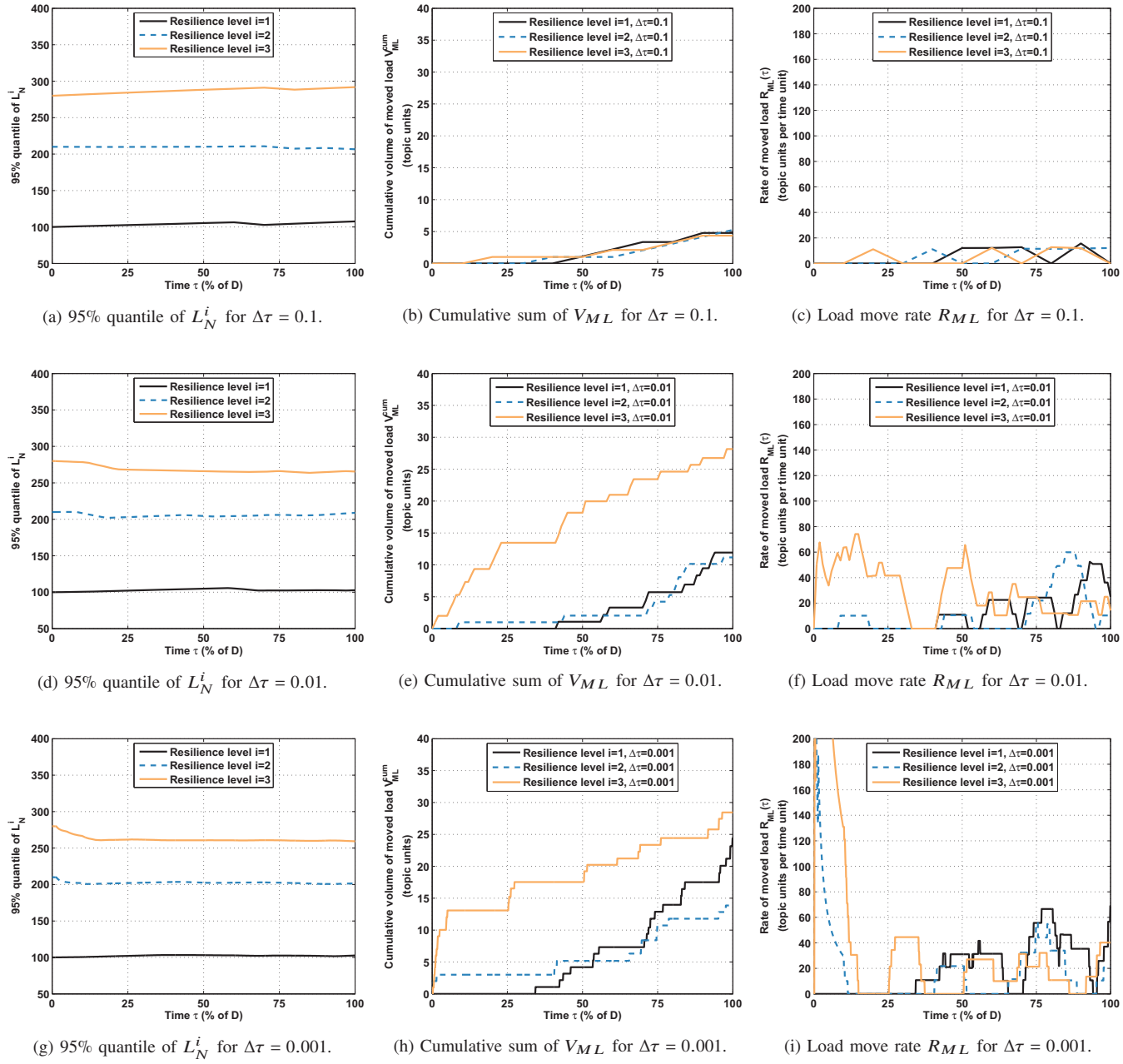
$$V_{ML}(\tau) = \sum_{t \in \mathcal{T}} \delta_t(\tau) \cdot L_T(t, \tau). \tag{13}$$

We calculate the rate of moved load $R_{ML}(\tau)$ over a time window of size $W = \frac{1}{10} \cdot D$ by

$$R_{ML}(\tau) = \frac{1}{min(\tau, W)} \cdot \sum_{\tau'=max(0,\tau-W)+1}^{\tau} V_{ML}(\tau'). \tag{14}$$

Figure 7 shows the 95% quantile of node load L_N^i, the cumulative volume of moved load V_{ML}^{cum} and the moved load rate R_{ML} over the experimentation period D for load balancing goals L_N^1, L_N^2 and L_N^3, and varying control parameter $\Delta\tau$. The cumulative moved load volume V_{ML}^{cum} is given in topic units, the rate of moved load R_{ML} is given in topic units per time unit, and the 95% quantile is given as normalized node load. In this context, one topic unit corresponds to the load of a topic t with $L_T(t) = 1$. The plotted values represent the results from one arbitrarily selected simulation run of the 100 distinct simulation runs.

The time-dependent evolution of the 95% quantile of L_N^i in Figure 7a, Figure 7d and Figure 7g shows that continuous load balancing can keep the system balanced over experimentation period D. Figure 7b, Figure 7e and Figure 7h illustrate the influence of $\Delta\tau$ on the cumulative volume of moved topic load over time. We observe a significant increase in the amount of moved load between $\Delta\tau = 0.1$ and $\Delta\tau = 0.01$, and a saturation with regard to the absolute amount of moved load when comparing $\Delta\tau = 0.01$ and $\Delta\tau = 0.001$. Figure 7f and Figure 7i show the rates of moved load R_{ML} for $\Delta\tau = 0.01$ and $\Delta\tau = 0.001$. We observe that the fast saturation of the cumulative volume of moved load for $\Delta\tau = 0.001$ comes at the price of a large and erratic moved load rate R_{ML} at the beginning of the experiment. In our experiments, $\Delta\tau = 0.1$ achieves worse load balancing results than $\Delta\tau = 0.01$. In



(a) 95% quantile of L_N^i for $\Delta\tau = 0.1$. (b) Cumulative sum of V_{ML} for $\Delta\tau = 0.1$. (c) Load move rate R_{ML} for $\Delta\tau = 0.1$.

(d) 95% quantile of L_N^i for $\Delta\tau = 0.01$. (e) Cumulative sum of V_{ML} for $\Delta\tau = 0.01$. (f) Load move rate R_{ML} for $\Delta\tau = 0.01$.

(g) 95% quantile of L_N^i for $\Delta\tau = 0.001$. (h) Cumulative sum of V_{ML} for $\Delta\tau = 0.001$. (i) Load move rate R_{ML} for $\Delta\tau = 0.001$.

Fig. 7. 95% quantile of L_N^i, cumulative volume of moved load V_{ML}^{cum} and rate of moved load R_{ML} over experimentation period D for load balancing goals L_N^1, L_N^2 and L_N^3 and varying control parameter $\Delta\tau$ of a selected simulation run of 100 simulation runs. The experiments were conducted with $n_{nodes} = 10$, $n_{node}^{topics} = 10$ growing topics per node with $L_T^{max} = 10$. R_{ML} and 95% quantiles are smoothed using a moving average with window size $W = \frac{1}{10} \cdot D$.

contrast, $\Delta\tau = 0.001$ leads to equally good load balancing results as $\Delta\tau = 0.01$ at the price of very high communication overhead. $\Delta\tau = 0.01$ provides a good tradeoff between reasonable communication overhead and still very good load balancing results. We conclude that re-assignment during operation is challenging and causes significant communication overhead in the form of large, probably erratic load move rates. Therefore, the $\Delta\tau$ should be set to a reasonable value. This value depends on the number of topics, number of nodes, and their placement.

IX. Conclusion

SeDAX statically assigns topics to coordinates. We showed that this can lead to severe load imbalance on SeDAX nodes. Therefore, we proposed a modification allowing dynamic reassignment of topics to coordinates while retaining the benefits of SeDAX, i.e., resilient overlay forwarding, decentralized control, and the ability to cope without a mapping system.

We defined metrics for load on SeDAX nodes for three different resilience levels. We observed strong load imbalance in existing SeDAX which is due to topological structures, i.e.,

varying Voronoi cell sizes, and does not vanish with scaling to larger number of topics or nodes.

We developed load balancing algorithms and demonstrated that they work well for all considered resilience levels, i.e., they significantly reduce the 95% quantile of the load on all nodes. For resilient SeDAX that survives at least two node failures, relative reduction is 43% and also the amount of shared backup capacity is clearly reduced. As load balancing using global knowledge may raise scalability concerns due to required signaling, we also proposed coordinate selection algorithms that work with only limited knowledge.

We showed that a balanced SeDAX system may run out of balance if topic sizes change over time. Therefore, we presented a distributed algorithm for continuous load balancing offering a single parameter to trade off load balancing quality against load balancing effort in terms of moved load rates. In our evaluations, it keeps a balanced system well balanced when topic sizes grow exponentially over time with different rates. Thus, the distributed and resilient SeDAX pub/sub system can be managed in a distributed way both at its initialization and during operation.

ACKNOWLEDGMENT

The authors thank M. Thottan, Y.-J. Kim, H. Kim, M. Mampaey, W. Braun, and F. Heimgaertner for valuable input and stimulating discussions.

REFERENCES

[1] Y.-J. Kim, J. Lee, G. Atkinson, H. Kim, and M. Thottan, "SeDAX: A Scalable, Resilient, and Secure Platform for Smart Grid Communications," *IEEE JSAC*, vol. 30, no. 6, 2012.

[2] M. Hoefling, C. G. Mills, and M. Menth, "Distributed Load Balancing for Resilient Information-Centric SeDAX Networks," in *IEEE Network Operations and Management Symposium (NOMS)*, Krakow, Poland, May 2014.

[3] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The Many Faces of Publish/Subscribe," *ACM Computing Surveys*, vol. 35, no. 2, 2003.

[4] A. Ghodsi, T. Koponen, B. Raghavan, S. Shenker, A. Singla, and J. Wilcox, "Information-Centric Networking: Seeing the Forest for the Trees," in *ACM HotNets*, Nov. 2011.

[5] M. Hoefling, C. G. Mills, and M. Menth, "Analyzing Storage Requirements of the Resilient Information-Centric SeDAX Architecture," in *IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Vancouver, Canada, Oct. 2013.

[6] D. Trossen and G. Parisis, "Designing and Realizing an Information-Centric Internet," *IEEE Mobile Communications*, vol. 50, no. 7, 2012.

[7] R. Alimi, L. Chen, D. Kutscher, H. H. Liu, A. Rahman, H. Song, Y. R. Yang, D. Zhang, and N. Zong, "An Open Content Delivery Infrastructure using Data Lockers," in *ACM SIGCOMM Workshop on Information-Centric Networking (ICN)*, 2012.

[8] C. Yi, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "Adaptive Forwarding in Named Data Networking," *ACM SIGCOMM CCR*, vol. 42, no. 3, 2012.

[9] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking Named Content," in *ACM CoNEXT*, 2009.

[10] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinsky, K. H. Kim, S. Shenker, and I. Stoica, "A Data-Oriented (and Beyond) Network Architecture," in *ACM SIGCOMM*, Aug. 2007.

[11] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," *ACM SIGCOMM CCR*, vol. 31, no. 4, 2001.

[12] A. Ghose, J. Grossklags, and J. Chuang, "Resilient Data-Centric Storage in Wireless Ad-Hoc Sensor Networks," in *International Conference on Mobile Data Management (MDM)*, 2003.

[13] M. Shorfuzzaman, P. Graham, and R. Eskicioglu, "Distributed Placement of Replicas in Hierarchical Data Grids with User and System QoS Constraints," in *IEEE 3PGCIC*, 2011.

[14] X. Tang and J. Xu, "QoS-Aware Replica Placement for Content Distribution," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 10, 2005.

[15] P. Jokela, A. Zahemszky, C. Esteve Rothenberg, S. Arianfar, and P. Nikander, "LIPSIN: Line Speed Publish/Subscribe Inter-Networking," *ACM SIGCOMM CCR*, vol. 39, no. 4, 2009.

[16] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," *ACM SIGCOMM CCR*, vol. 31, no. 4, 2001.

[17] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A Survey and Comparison of Peer-to-Peer Overlay Network Schemes," *IEEE Communications Surveys & Tutorials*, vol. 7, no. 2, 2005.

[18] P. Felber, P. Kropf, E. Schiller, and S. Serbu, "Survey on Load Balancing in Peer-to-Peer Distributed Hash Tables," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 473 – 492, 2014.

[19] K. Kenthapadi and G. S. Manku, "Decentralized Algorithms Using both Local and Random Probes for P2P Load Balancing," in *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, Jul. 2005.

[20] A. Rao, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, "Load Balancing in Structured P2P Systems," in *International Workshop on Peer-to-Peer Systems (IPTPS)*, Feb. 2003.

[21] B. Godfrey, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, "Load Balancing in Dynamic Structured P2P Systems," in *IEEE Infocom*, Mar. 2004.

[22] J. Byers, J. Considine, and M. Mitzenmacher, "Simple Load Balancing for Distributed Hash Tables," in *International Workshop on Peer-to-Peer Systems (IPTPS)*, Feb. 2003.

[23] M. D. Mitzenmacher, "The Power of Two Choices in Randomized Load Balancing," Ph.D. dissertation, Univ. of California at Berkeley, 1996.

[24] M. Mitzenmacher, A. W. Richa, and R. Sitaraman, "The Power of Two Random Choices: A Survey of Techniques and Results," in *Handbook of Randomized Computing*. Kluwer, 2000, pp. 255 – 312.

[25] J. S. A. Bridgewater, P. O. Boykin, and V. P. Roychowdhury, "Balanced Overlay Networks (BON): Decentralized Load Balancing via Self-Organized Random Networks," *CoRR*, vol. cs.DC/0411046, 2004.

[26] G. Even and M. Medina, "Parallel Randomized Load Balancing: A Lower Bound for a More General Model," in *SOFSEM 2010: Theory and Practice of Computer Science*. Springer, 2010, vol. 5901.

Michael Hoefling (GS'10) is a researcher and PhD student at the Chair of Communication Networks at the University of Tuebingen/Germany. He holds a B.Sc. and M.Sc. in Computer Science from the University of Umeaa/Sweden, and a German diploma in Computer Science (Dipl.-Inform.) from the University of Wuerzburg/Germany. His current research interests include future Internet routing, smart grids, information-centric networking, data center networks, and grid and cloud computing.

Cynthia G. Mills (M'13) received the B.A. degree in German from Tufts University, and the Dipl.-Inform. degree (German Diploma) in computer science from the University of Tuebingen, Germany. She has been active in the internetworking arena since 1978. In addition to conducting Internet research in traffic measurement and electronic commerce, she has shipped several network and Internet security products. She is currently an IT Security Consultant with the SySS GmbH, Tuebingen.

Michael Menth (M'05–SM'09) is a Professor with the Department of Computer Science, University of Tuebingen, Germany, and the Chairholder of Communication Networks. He has published over 150 papers in the field of computer networking. His special interests are performance analysis and optimization of communication networks, resilience and routing issues, resource and congestion management, software-defined networking and Internet protocols, industrial networking, and Internet of Things.