Increasing Resilience of SD-WAN by Distributing the Control Plane [Extended Version]

Friedrich Altheide, Simon Buttgereit, and Michael Rossberg, *Member, IEEE*

***Abstract*—Modern WAN interconnects utilize SD-WAN to automatically respond to network changes and improve link utilization, latency, and availability. Therefore, they incorporate controllers with a centralized view, which collect network state from managed gateways, calculate suitable forwarding actions, and distribute them accordingly. However, this limits the robustness and availability of the network control plane, especially in the event of node or partial network outages. In this paper, we propose a distributed and highly robust SD-WAN control plane without any central or regional controller. Our solution can handle arbitrary device failures as well as network partitioning. The distributed forwarding decisions are based on user-defined, dynamically evaluated path cost functions, and consider not only path quality but also quality fluctuations. The evaluation shows that our approach can handle several thousand SD-WAN gateways and hundreds of network policies in terms of computation. Further, the communication overhead introduced due to its distributed architecture is discussed and shown to be negligible compared to a central approach. This paper is an extended version of our work published in 2023. It introduces novel insights, including an in-depth analysis of the information transmitted between sites, a new strategy for policy deployment, a discussion as well as a detailed analysis of approaches that reduce communication bandwidth, and the introduction of a method for grouping multiple flows without the need for explicit coordination.**

***Index Terms*—SD-WAN, SDN, robustness, distributed systems.**

I. Introduction

TO ENABLE per-application-based traffic engineering, modern networks are continuously monitored to automatically react to network changes e.g., by adapting the routes of flows inside the controlled network. This improves the per-flow throughput, end-to-end latency, and ultimately, experienced quality. The presumably most common technique to achieve such an automatic, fine-grained traffic control is Software Defined Networking (SDN). When used between remote locations connected over one or more Wide Area Network (WAN), it is referred to as Software Defined WAN (SD-WAN).

Manuscript received 15 December 2023; revised 23 February 2024; accepted 16 March 2024. Date of publication 19 April 2024; date of current version 12 July 2024. The associate editor coordinating the review of this article and approving it for publication was X. Fu. *(Corresponding author: Simon Buttgereit.)*

Friedrich Altheide is with the CTO Office, Secunet Security Networks AG, 45138 Essen, Germany (e-mail: friedrich.altheide@secunet.com).

Simon Buttgereit and Michael Rossberg are with the Telematics and Computer Networks Group, Technische Universität Ilmenau, 98693 Ilmenau, Germany (e-mail: simon.buttgereit@tu-ilmenau.de; michael.rossberg@tu-ilmenau.de).

Digital Object Identifier 10.1109/TNSM.2024.3386962

In general, SD-WAN comprises of centralized SD-WAN controllers, programmed with abstract policies, which decide where to route traffic. Each policy defines how different traffic should be handled between the SD-WAN gateways under the management of the controller (see Fig. 1), and can be updated if requirements change. Based on these global policies, SD-WAN controllers select a connection suitable for a specific traffic flow. In some cases, it also initiates the establishment of connections between the gateways. In this context, a *connection* typically refers to an end-to-end association between two gateways, which can be established through a variety of means such as a VPN tunnel or an MPLS label path, combined with a unique connection ID. To make qualified decisions where to route flows based on the current network state, the controller regularly acquires network topology information from the managed SD-WAN gateways. This information is obtained either by querying the gateways or by proactively receiving relevant data, such as the currently established tunnels and their properties, e.g., latency, error rate, or current usage, but also device-specific information like the current utilization of the gateway or its WAN uplinks. Once a decision is made, corresponding forwarding rules are sent to the SD-WAN gateways, which realize the forwarding decisions.

The centralized view enables the programming of complex network behaviors with relatively simple and comprehensive network programs, which are executed on the controller. Today's widely deployed SD-WAN solutions offer comparatively simple, automatic, flexible, and fine-grained traffic control to optimize routing, both per network flow, but also on a more global scale. But to retain reactivity to link failures or overload situations, a permanent connection to (some kind of) centralized controller is required. Although it is common to deploy multiple controllers to enhance availability within an SD-WAN infrastructure, they can still pose a neuralgic point, particularly in scenarios involving partial network partitioning. Firstly, the deployed SDN controllers may not be able to exchange state appropriately due to high latency, packet errors, or reachability/connectivity issues. Secondly, WAN partitioning may result in SDN devices losing connectivity to all controllers, making them incapable of responding to new events such as topology modifications. Consequently, the robustness and availability of the network control plane are decreased significantly.

Based on the foundational SDN requirements outlined by Stallings [2] and the fundamental observations about Internet routing by Paxson [3], we state that a solution for an automatic,

fine granular SD-WAN traffic control should support the implementation of global network policies on a global and local scale and facilitate:

(F1) decisions based on the current Quality of Service (QoS) properties of the used WAN connections,
(F2) flexible packet matching,
(F3) symmetric traffic flows, e.g., to ease network troubleshooting [3], or to cope with Network Address Translation (NAT) and stateful firewalling,
(F4) load balancing traffic across multiple gateways of a single site (gateway cluster),
(F5) high availability support, e.g., clusters with geo-redundancy, as well as
(F6) simple firewalling (blocking of flows).

Furthermore, a solution should be:

(N1) highly available,
(N2) robust against network failures and partitioning as well as arbitrary node failures,
(N3) reactive towards network changes, yet provide stable end-to-end connections,
(N4) be able to handle thousands of SD-WAN gateways,
(N5) offer a high flexibility for an administrator, yet still be comprehensible and
(N6) should be independent of any kind of exposed infrastructure.

Unlike existing solutions, this paper presents an SD-WAN architecture that satisfies the stated requirements by completely distributing the control plane onto all SD-WAN gateways.

By dividing global network policies into gateway-local policies and selecting WAN connections based on user-defined cost function values, as well as estimations of value fluctuations, we achieve highly robust and available traffic engineering that remains responsive even in the event of device and network failures.

This paper is an extended version of our work published in [1] and is organized as follows: The subsequent section presents the current state of the art. Compared to [1] it is extended by specifying the information exchanged between the different SDN components. Sections III and IV describe new methods of consequently distributing the SD-WAN control plane and how to program such a distributed system. Section III is extended by describing a fundamental policy deployment strategy, while Section IV additionally discusses approaches that reduce the communication bandwidth. Furthermore, it also introduces methods for grouping multiple flows without the need for explicit coordination. Following, in Section V the proposed architecture is evaluated, extending the original paper with a more detailed analysis of the required bandwidth. The paper closes with a conclusion and a description of possible future work.

II. Related Work & Background

One solution to overcome the described issues resulting from a centralized architecture could be the use of well-established distributed routing protocols. The Border Gateway Protocol (BGP) is the major routing protocol for large networks. It offers exceptional scalability, availability, and robustness by avoiding the need for a central instance. Further, BGP supports flexible parameterization to differentiate paths through a network. Yet, aside from destination prefixes, it is not capable of routing fine granular traffic flows over different paths without sacrificing scalability. Additionally, BGP does not natively support the representation of bandwidths or latency in its parameters, making it unable to automatically optimize these network characteristics. To achieve more flexibility, distributed routing protocols can be extended by a central coordinator. *Fibbing* [4] is such a hybrid approach, allowing a central entity to influence unmodified Open Shortest Path First (OSPF) routers by introducing fake links and nodes [4]. While this approach is very promising in regards to its robustness, the steering granularity still cannot match SDN.

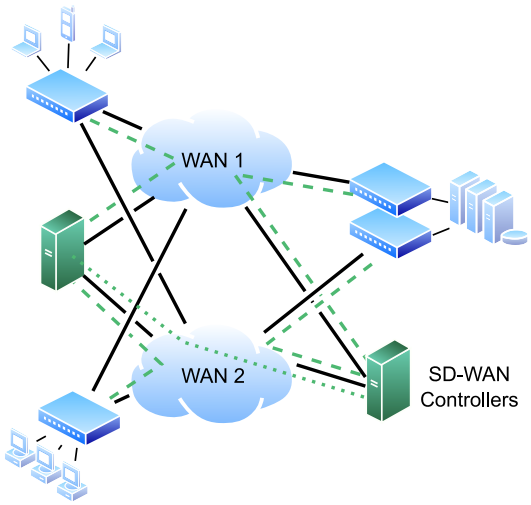

Fig. 1. A centralized SD-WAN infrastructure decides how to route traffic between the different single-gateway and cluster sites. Any connectivity issue between an SD-WAN gateway and the controller or the different controllers results in reactivity issues of the control plane. Thus, the robustness of the control plane is quite limited.

To support a fine-grained traffic control while overcoming the described issues of SDN, a common approach is to deploy distributed controllers. The simplest design is the usage of a clustered controller infrastructure with common state being synchronized between the elements. Every SDN device is assigned to one controller [5]. This assignment can either be static or dynamic to support automatic load-balancing between the cluster elements [6], [7]. If synchronization between the controllers is done with some kind of consent protocol such as Paxos [8] the number of cluster nodes and latency among them, as well as the number of consensuses required [9] become a limiting factor. To limit this problem, some SDN controllers use eventual consistency [10], yet the amount of data to synchronize becomes enormous for a large number of SDN controllers. While horizontal clustering improves the availability of the cluster itself, the reactivity of the controlled network remains limited when gateways cannot reach the controllers due to network issues. Hence, clusters are typically distributed among different regions to compensate regional

TABLE I
RELEVANT INFORMATION REQUIRED FOR A DECIDING ENTITY

Data	Category				Interval		
	per connection	per uplink	per policy	per device	regularly	on change	once
Latency	✗				✗		
Rx error rate	✗	✗			✗		
Tx bandwidth	✗	✗			✗		
Rx bandwidth	✗	✗			✗		
CPU load				✗	✗		
Temperature				✗	✗		
Flow rule information			✗		✗		
Connection → Uplink	✗						✗
Link speed		✗				✗	
Booked bandwidth		✗					✗
Reachability of site-local network				✗		✗	✗

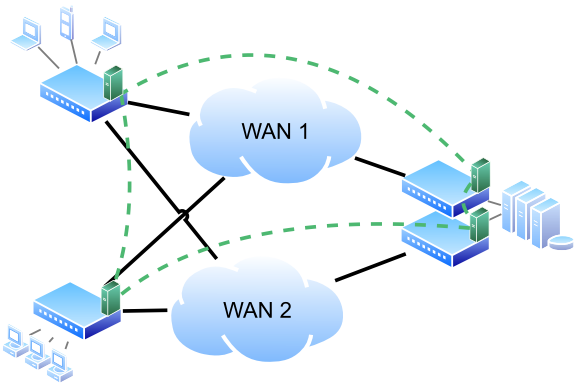

Fig. 2. SD-WAN controller elements (green) are placed on each gateway and coordinate their decisions on whether and how to route/distribute network traffic between their sites.

network outages [11], [12]. While this improves the availability, it also introduces additional latency to the synchronization messages, thus limits the scalability.

This limitation can be lifted by using a hierarchical controller design [13], [14], [15], which splits the network in several independent network regions. Each region is managed by its own independent regional controller. To steer traffic between regions, a global controller is introduced. Hence, failures of a regional controller do not affect other regions. Although this design seems promising and offers good scalability, it suffers from availability issues as the routing between regions cannot adapt to network changes, e.g., high link utilization and changed error rates, once the global controller becomes unreachable. Consequently, connectivity problems between controllers can result in inter-regional network partitions. Additionally, depending on the sizes of the regions, outages of regional controllers may still have a substantial impact.

Background on network statistics: As outlined before, deciding instances (the SD-WAN controllers) always have to acquire the current network state to make qualified decisions. In a scenario with a single controller, all gateways must send information about their established connections and gateway-specific information to the controller. Table I lists and categorizes these based on how often they must be transmitted and if there are collected per connection, uplink, policy, or per device. Note that, depending on the concrete scenario the required information may vary. Some information only needs to be sent once, e.g., during the establishment of a connection, while other information requires transmission upon change or at regular intervals. Some data describes characteristics of the gateway itself, yet the majority of the information concerns the status of uplinks or the established connections. It is worth noting that in scenarios with regional controllers, the decision-making process would only require data about the connections within their specific region. Yet, inter-regional information is usually relayed by the regional controllers. Subsequently, the regional controllers receive all data and selectively send information pertaining the inter-region connections to the central controllers. Additionally, it has become common practice to periodically retrieve information about the currently deployed flow rules for reasons of robustness. The controller subsequently sends flow rules, connection establishment requests (e.g., in form of a tunnel), and, depending on the used southbound protocol, statistic requests to the gateways.

III. DISTRIBUTING THE SD-WAN CONTROL PLANE

In this work, we propose to fully distribute the control plane and thus the decision process into the distributed SD-WAN gateways (see Fig. 2). These distributed entities must then coordinate how to route flows while achieving scalability and minimizing the control overhead. The entire concept is designed to function without the need for any kind of external instance. It ensures that any pair of sites that is able to communicate, e.g., by establishing a connection between its SD-WAN gateways, directly coordinates their respective intra-site traffic. Hence, a failure of one site or its gateways only affects connections with or within that site, while other sites are unaffected and can still realize their network policies. But, to achieve high availability (A) and partition tolerance (P), it may be necessary to sacrifice strong consistency (C) as explained by the CAP theorem [16]. Consequently, protocols such as Paxos [8], are not suitable under all conditions. Instead, we employ the method of eventual consistency [17], which ensures that the entire network converges to a consensus on how to handle network traffic. This means that, after a network failure, any discrepancies or conflicts that arose will be resolved over time, and the network will be configured with a homogeneous view.

A. Distributed Global Network Policies

A global SD-WAN policy typically comprises a *traffic description* and an *objective*.

1) The *traffic description* defines the flow that is described by the policy. It is typically based on the packet headers and can contain multiple conjunctive or disjunctive matching criteria, e.g., HTTPS traffic (usually identified by TCP port 443 or 8443) tagged with VLAN ID 20.
2) The *objective* describes how to handle the traffic flow defined by the *traffic description*. In the context of SD-WAN the *objective* determines whether matched

packets should be dropped (firewalling) or the desired quality of the used connection.

Implementing such a global policy in a distributed manner by coordinating decisions among a potentially large number of gateways can be complex and may not scale well if done improperly. In existing SD-WAN solutions, this problem is typically avoided as central SD-WAN controllers have global knowledge of network topology and state. While a centralized instance may be necessary for some complex SD-WAN policies to coordinate all decisions, it is not required for the majority. This is because SD-WAN policies usually involve paths between at most two sites, which means that coordination on routing decisions can be limited to the two sites alone. Global policies that relate solely to traffic within a single site can be managed within that specific site but fall outside the scope of this work. Note that we assume that the IP ranges of the site-local networks of different SD-WAN gateways are disjoint unless they connect the same site in form of a cluster (more details follow).

To describe policies that are limited to one pair of sites we introduce the term *inter-site policy*. A global network policy is transformed into multiple inter-site policies, each describing a specific flow between exactly two sites or more precisely between two subnets (see Fig. 3). Limiting each inter-site policy to exactly one subnet at each site simplifies selecting a connection for an inter-site policy, as different connections to the same remote site may announce different local subnets (clustering). Without this limitation, some policies may not be feasible (details in Section IV-C). Consequently, a global policy encompassing multiple subnets of a pair of sites is split into multiple independent *inter-site policies* each with one source and one destination subnet. This approach enables local decision-making by SD-WAN gateways (details follow in Section III-B).

The transformation of a global policy into multiple inter-site policies can be executed by a centralized management/monitoring component, a local administration tool, or even the gateway itself. Implementing this transformation in the gateway allows for the distribution of global policies between gateways without involving a third party. An administrator could create a global policy, transfer it to a single gateway, and as long as the network is unpartitioned, the policy may propagate to all gateways. While this approach can distribute policies, its design necessitates additional memory on the gateways to store all network policies. Further, there has to be a method of handling conflicts between different policy revisions. The following rules implement a relatively basic approach (more sophisticated approaches can be considered):

- R1 Every policy can be identified by a deterministic ID, e.g., the hash of the traffic description and policy.
- R2 The current policy configuration of a gateway is called *policy set* and represented by the IDs of all configured policies along with a set counter.
- R3 If a gateway receives a policy set, it checks if the received is newer than the currently configured one, e.g., by comparing a set counter. If it is, it checks for unknown policy IDs within the new set, requests the appropriate policies from the remote gateway, switches to the new policy set, and further distributes the new policy set. Otherwise, it keeps the old configuration.

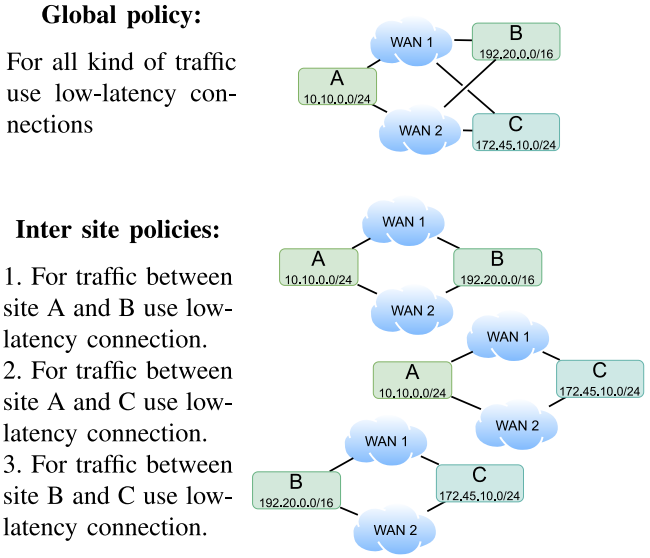

Fig. 3. Global policy is transformed into three inter-site policies between each pair of sites/networks.

For this simple approach, the used counters must be synchronized out-of-band, else (different) administrators may configure and deploy different policy sets with the same counter. This synchronization mechanism could be implemented by some kind of deployment and monitoring platform, which does not affect the availability and robustness of the decision algorithm itself yet provides an overview over the currently active policies.

B. Realizing Inter-Site Policies

We aim to support two types of SD-WAN policies: Firewalling (drop) policies and routing policies. Firewalling policies do not require advanced synchronization because each gateway involved can independently check packets to determine whether to forward or drop them. Routing policies, in contrast, require coordination between the participating instances. To preempt potential instabilities that may arise from diverse underlay conditions, or overlay NAT and statefull firewalling, it is advantageous for flows between two sites to employ symmetric paths in both directions. To achieve this symmetry, protocols that allow each site to make independent decisions should be avoided. Instead, both sites must coordinate which WAN connection to choose. As the available connections between two sites may have a high latency, e.g., due to the physical distance, classical consensus protocols may limit the responsiveness. Instead, we propose to deterministically select one site as the leading site. The leading site is then responsible of choosing a single connection for each inter-site policy between itself and the remote site. The latter must follow these decisions. To select the leading site for each pair of sites, we compare the hashes of the concatenated site IDs and prefer the smaller one:

$$hash(ID_{\text{site}_1}|ID_{\text{site}_2}) \overset{?}{<} hash(ID_{\text{site}_2}|ID_{\text{site}_1})$$

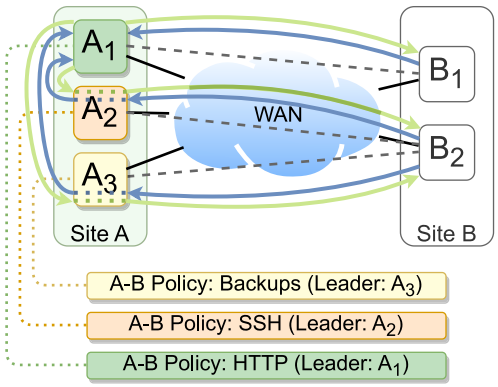

Fig. 4. Two multi-gateway sites *A* and *B* are configured with three inter-site policies with site *A* being the leading site. To achieve a good scalability, the role of the inter-site policy leader for the three different policies can be distributed across gateway A_1, A_2, and A_3. The two sites are connected by three connections (- - -). For the HTTP policy, which is led by A_1, the regularly transferred and relayed statistic data (←) as well as the distributed decisions (→) are depicted.Note that this process happens for all policies, yet is not displayed for the sake of simplicity.

This comparison prevents the site with the smallest *ID* from always taking the role of the leading site. Instead, the number of leader roles per site is evenly distributed in larger networks (details follow in evaluation).

For reasons of high availability, one or both sites may consist of multiple SD-WAN gateways, called a *cluster* of gateways. To allow a good scalability, availability, and robustness, we propose to further distribute the decision-making of the individual inter-site policies within a cluster. We therefore introduce the role of the *inter-site policy leader*. Gateways assigned to this role make all decisions for a concrete inter-site policy. Other gateways within the same site, as well as those in the remote site, then follow the decisions made by the leader for that particular inter-site policy. Since a site typically will be configured with multiple inter-site policies, the different inter-site policy leaders can be distributed across all SD-WAN gateways of the cluster (see Fig. 4).

The decision-making process is illustrated for three sites as an example in Fig. 5. It starts with each gateway notifying the inter-site policy leader of all connections that are appropriate for the given policy by transmitting the current connection and device statistics (more details follow in Section IV). This action is triggered by a local and policy-specific *statistic timer*. To deal with outliers, statistics are fetched more frequently than specified by the *statistic timer* from the gateway's data plane and smoothened, e.g., with a moving average. Additionally, the fluctuation for each statistical value is determined by computing the moving average of the difference between the smoothened and current values. This corresponds to an iteratively approximated mean absolute deviation, i.e., the first absolute central moment. The smoothened statistics and fluctuations are sent over the available connections (see Fig. 4) and relayed within the leading site, either over the site's local network or additional intra-site connections. This process ensures that the leading gateway has information about all available connections as well as remote site gateways.

When a *decision timer* triggers, the leading gateway selects a distinct connection to the remote site. It then notifies the remote gateway by sending an activation message containing the policy ID over the selected connection. If the decision alters a previous one that involved another gateway, the former decision is revoked by sending a deactivation message containing the policy ID through the previously chosen connection. If the previous connection no longer exists, this step can be skipped. In cases where the decision remains the same, a small message is dispatched to keep the soft-state alive. Upon receiving a decision, a gateway generates and installs new forwarding rules into its data plane.

All these steps are repeated periodically and can be divided into rounds/periods. However, in the event of a connection failure, this information must be processed immediately. The leading gateway calculates the new decisions and disseminates it without waiting for a timer. It should be noted that each gateway can take on the leader role for multiple inter-site policies simultaneously, e.g., site C in Fig. 5 for *A*-*C* and *B*-*C*.

To limit the impact of gateway or network failures, the inter-site policy leader role is assigned to the gateway that the policy's network flow is currently sent over. This approach enhances both robustness and availability compared to randomly distributing inter-site policy leader roles. After startup, each gateway first selects itself as the leader for all inter-site policies where it is part of the leading site. If a gateway discovers another gateway within the same site that it is currently unaware of, it establishes a bidirectional channel. Both gateways then exchange information about the inter-site policies they are currently leading and the quality of their selected connections. In cases where both gateways are equally suited for a particular policy, the gateway with the higher *ID* relinquishes its role, resulting in the establishment of a single leader. In all other cases the gateway with the better connection wins and stays the policy leader. After both gateways exchanged their leading roles, exactly one leader exists for every inter-site policy. If the leader selects to route the flow via another gateway, it informs all gateways and hands over the policy leader role.

To address situations where multiple gateways claim the role of the inter-site policy, each leader distributes its current decision and its quality to all other cluster members after each round. Hence, by applying the rules stated above each cluster member is aware of the current leader for each policy. In order to sustain responsiveness in the event of a leading gateway failure, each decision has a lifetime $t_{exp} = t_{timer} + c$ with $c > 0$. Each gateway in the leading site periodically checks for expired decisions. For each expired decision, the gateway announces itself as the new leader to all other gateways within the site. The presented rules as well as the regular exchange of the current decisions ensure that over time only one inter-site policy leader will remain, even if multiple gateways choose to promote themselves as the leader simultaneously.

Due to connection failures, a clustered leading site may split into multiple partitions that are unable to communicate with each other. We assume that each of the site's subnets is connected to at most one partition, since otherwise a connection between the partitions would exists. Since inter-site

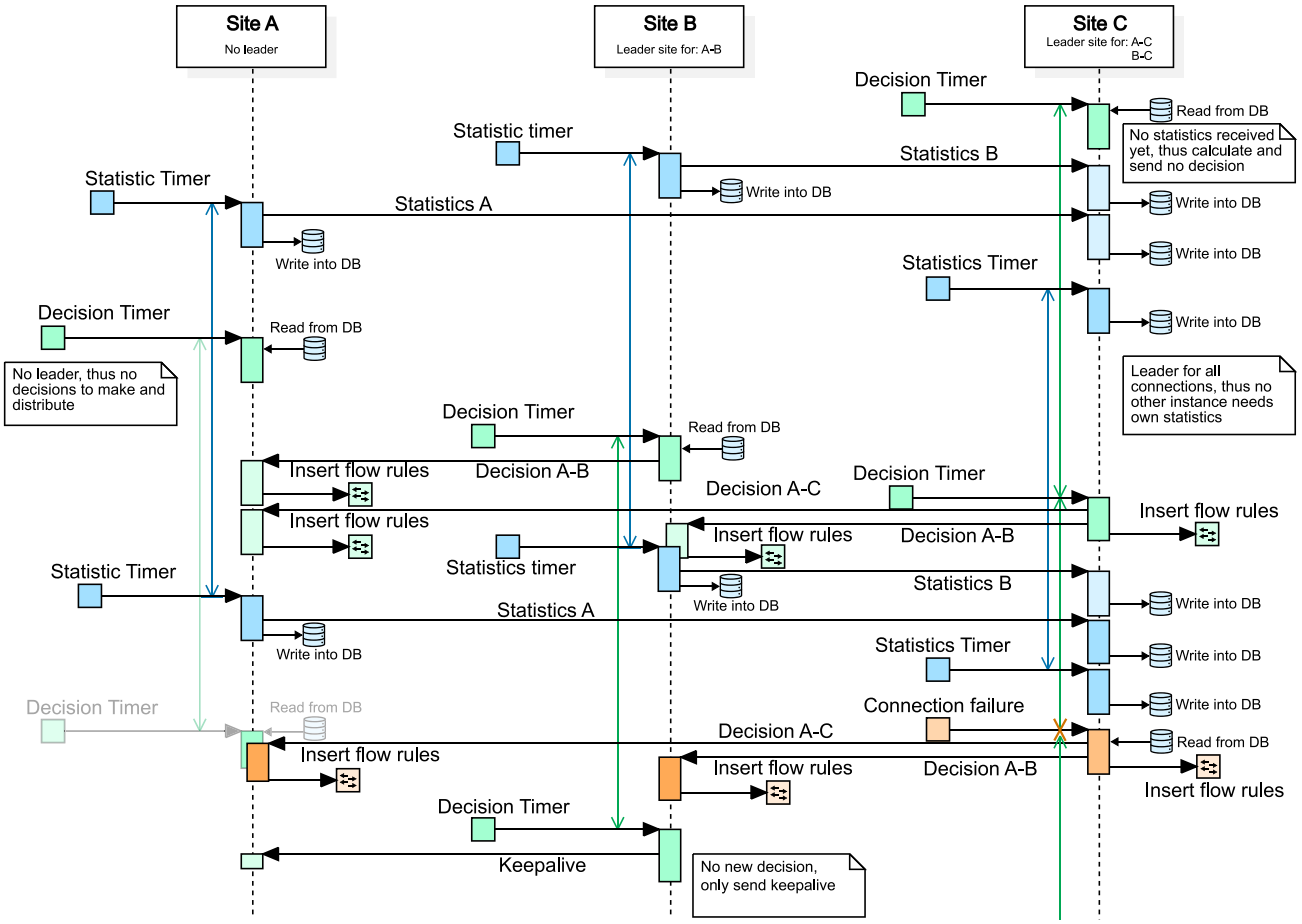

Fig. 5. Time sequence of decision making between three sites. Site *A* with no leader role only sends statistics and statistic fluctuation, receives decisions, and writes appropriate flow rules into the data plane. If a site has one (*B*) or multiple leading roles (*C*), it receives and stores statistics along with their fluctuations. Based on these, a decision is calculated and then transmitted to the appropriate remote site. After a decision was received by a site, flow rules are written into the data plane of the site's gateways. Both steps of sending statistics and calculating decisions are triggered by timers. In case of a connection failure, a new decision is calculated immediately, and the timer is restarted. For sake of readability, the statistic timers as well as the decision timers of one site are executed at the same time. It should be noted that in reality all timers are not synchronized and operate independently. The process of sending/receiving statistics and calculating/receiving a decision for a specific inter-site policy can be divided into rounds. Each round lasts for one timer interval and starts when the statistic timer is triggered.

policies exclusively contain a single subnet per site, as defined in Section III-A, their functionality remains intact. Even if two gateways should temporarily elect themselves as leaders after connections are regained, the protocol ensures that eventually only one leader remains.

IV. Programming a Distributed Control Plane

By utilizing the described architecture, decisions can be made in a fully distributed manner, eliminating the requirement for a central component or global/regional knowledge. Thus, telling the control plane on how to react to changing network characteristics is not realizable by one simple network program deployed on a central component. However, for reasons of usability, a simple but comprehensive method of programming the distributed system is required. We therefore propose to let the distributed gateways weight every connection based on locally available data such as connection statistics, gateway utilization and/or the current time. The calculation of a specific connection's weight is accomplished through the use of customizable metric functions within the distributed SD-WAN gateways. These functions, as demonstrated in 1, determine a weight > 0 for each connection. The idea is to let the inter-site policy leader compare different connections using their calculated weights with lower cost considered as the better connection. As a connection involves two endpoints, either both endpoints weight their statistics independently and send the weight to their policy leaders, or the connection statistics are transferred to the leading site, where they are weighted and then sent to the leaders. In both cases the leader can then sum both weights to a connection weight, which is then used for comparing connections. If the goal is to balance the execution time between both sites, it could be beneficial to evaluate the statistics independently on each endpoint. Yet, [1] revealed that while such a design is technically feasible, it significantly reduces scalability and works only for very small numbers of policies. This is because, instead of transmitting statistics once, the solution requires the transmission of all metric weights of each connection, leading to a linear bandwidth increase when new policies are installed. Thus, all weights of an inter-site

```
fn costs(con_stats, util, timestamp) -> f64 {
  if(timestamp between 8am and 10am) {
    // classify connections in 10ms steps
    return con_stats.latency().as_ms().round() / 10
  }
  if(timestamp between 10pm and 4am) {
    // select no connection and block traffic
    return f64::MAX;
  }
  // prefer connection with lowest latency
  return con_stats.latency().as_ms();
}
```

Listing 1. Based on the connection statistics and the current time a programmed metric function returns a weight.

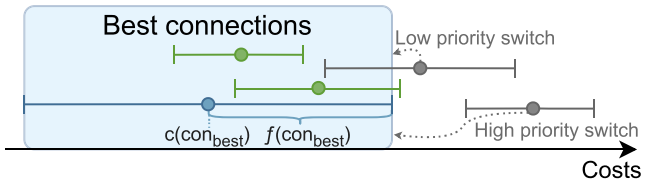

Fig. 6. All connections with costs inside the fluctuation of the "best" connection $c(con_{best}) \pm f(con_{best})$ are included into the best connection class. If the currently used connection is outside this class, the inter-site policy should be switched to a class member over time (dotted lines). Depending on the current cost difference, the switch is scheduled soon or later.

policy should be evaluated at the point where the decision is made.

Either way, the programmed metric functions used to weight the statistics should be easily changeable to provide high flexibility for administrators. This enables the adjustment of existing steering properties or the introduction of new QoS requirements. Further, robustness (e.g., limiting resource consumption) and security must be ensured, e.g., by isolating the metric function execution. To fulfill these requirements, we utilize the binary format WebAssembly (Wasm) [18], which not only is executed with near native speed inside Web browsers, but also in isolated execution environments called WebAssembly System Interface (WASI).[1] It allows a user to write `costs()` functions in any supported language,[2] compile them into a textual representation and append it to the global and subsequently to the inter-site policies. The binary is then loaded into the gateway's WASI execution environment and used to calculate the weight of a connection for a specific policy.

A. Making Stable Decisions

After receiving the weights of all connections, each inter-site policy leader needs to select one distinct connection for their policy. Like any traffic control algorithm, we aim to increase the quality of all flows while also ensuring stable end-to-end connections. Yet, optimizing the quality of inter-site policies presents several challenges that the selection algorithm must address:

1) Selecting the best available connection for all inter-site policies might decrease the quality of that connection.
2) Load balancing different inter-site policies onto the n metrical best connections might decrease quality if n is chosen too small (e.g., bandwidth exhaustion), yet also if chosen too high, e.g., higher delay for flows routed over n^{th} best connection.
3) Switching traffic of multiple inter-site policies simultaneously might (negatively) affect each other.
4) Inter-site policies might continuously toggle between connections since switching could decrease the quality of the selected connection while the previous connections quality might recover due to reduced utilization.
5) If one connection consistently offers slightly better costs over an extended period of time for an inter-site policy, it should be used to improve overall quality. The connection switch should neither be too fast (e.g., only temporary low costs) nor too slow (poor quality until switch is performed).

These challenges do not only occur because of dynamically programmed metric functions, but also because of varying WAN quality as well as variations in the number and size of traffic flows. Furthermore, objectives can conflict with each other, like reducing toggles between two connections and preferring the better connection in the long run. As a result, relying on a static threshold per metric and switching when it is exceeded will not adequately address the challenges described. Instead, a dynamic threshold is necessary. Its evaluation has to be efficient in computation but also in memory consumption. Hence, incremental calculations are preferred.

As a solution for the presented challenges, we propose a *class of best connections*. This class includes all connections whose current metric value falls within the fluctuation range of the best available connection $f(con_{best})$ (see Fig. 6). As with the handling of outliers when collecting statistics, the calculated metric values are smoothened, e.g., with a moving average, and a fluctuation is calculated.

When employing the first absolute central moment for fluctuation calculation, we observed that the fluctuation of the best connection will quickly converge to a value close to 0 if the metric weight is nearly static, e.g., the delay on an unutilized link. However, in case of extremely short quality degradations, the fluctuation also stays close to 0. Thus, our approach to calculate the fluctuation may not be the perfect choice for all scenarios. For example, weighting statistics in rapidly fluctuating network situations quicker than in stable situations may be more suitable. The evaluation of different fluctuation metrics and their application in different scenarios is, however, left for future work.

Switching an inter-site policy to another connection within the *class of best connections* does not offer a statistically relevant advantage, since all metrics within the class are insignificantly different from the best connection for the given metric. Yet, if the used connection is not part of the class, a (random) connection within the class (see Fig. 6) should be selected to transfer the corresponding traffic over a path with higher quality. To prevent instabilities due to many switches at the same time, the decision to switch to a different connection is decided in each round using a

[1] https://wasi.dev/
[2] https://github.com/appcypher/awesome-wasm-langs

Bernoulli distribution independently from the previous round. Considered over multiple rounds, it corresponds to a geometric distribution. The probability p whether to switch to another connection in this decision period/round is determined by the difference between the current metric value $c(con_{cur})$ and the upper limit of the *class of best connections* $c(con_{best}) + f(con_{best})$.

$$p = \max\left(\alpha, 1 - \frac{c(con_{best}) + f(con_{best})}{c(con_{current})}\right) : \alpha \in (0, 1]$$

If the previous connection is only slightly better than the best connection, p will diverge to the minimal switching probability α. Otherwise, the value of p will be between α and 1, depending on the difference between the connections. Therefore, α influences how long an inter-site policy stays on a connection that is nearly similar to the best connection but not part of the *class of best connections*. In Section V-A, we evaluate the duration required for a connection switch, taking into account different values of p.

B. Limiting the Required Communication

As outlined in the previous sections, all connected SD-WAN gateways have to directly exchange information regarding its established connections, WAN uplinks, policies as well as device-specific information like the CPU load (detailed list in Table I). This information must be directly exchanged among all connected gateways. To also support low bandwidth uplinks, the required bandwidth should be reduced. The subsequent paragraphs will outline various approaches to achieve this reduction.

1) Compression: Lossless compression can be applied to the data being transmitted, trading off some bandwidth for extra computation time in compression and decompression. For measured data modern compression algorithms typically achieve compression ratios ranging from 1.5 to 3 [19]. Yet, it is important to note that this process is most effective when different statistic values and/or decisions are transmitted as groups rather than individually.

2) Selective Send: By design, the leading site requires statistics from all established connections to select a suitable connection for each inter-site network policy. Hence, it is not possible to only send statistics for connections that are currently in use. Instead, the statistic values may be limited to those required by the leading site, e.g., by annotating each metric function with the required values. Then, non-leading gateways can spare out irrelevant information from the control messages. A simple and practical approach could be to categorize metric functions into different classes based on the statistics values they use. This way, a local gateway can easily determine which statistics to send. For smaller setups with only a limited number of distinct metric functions, this approach can lead to reduced bandwidth usage. Nonetheless, based on the statistics listed in Table I, even with slightly more metric functions, nearly all statistic values are likely to be used. In such cases, the benefit remains marginal.

3) Reduced Statistic/Decision Frequency: Changing the interval of sending statistics and making decisions directly impacts the amount of data to send but also the system's responsiveness (see Section V-A for more details). Thus, the interval should be tailored to the concrete scenario's requirements. For certain scenarios, connection statistics might require more frequent updates than uplink or device information. As a result, multiple independent sending intervals could be introduced for the different statistic types, to reduce the bandwidth required.

4) Reducing Redundant Information: If a gateway has multiple connections to the same remote gateway, it should send per-device and per-uplink information only once. Further, if a gateway with a low-bandwidth uplink has set up multiple connections to a cluster, it can send per-device and per uplink information to just one of the cluster gateways. The gateways of the remote site can then distribute the information within the cluster. This approach can also be employed for the decision-sending process. Note, that while this approach reduces the number of control messages, the following sites must be notified of the current cluster members of a leading site to detect cluster partitions. Otherwise, per-device and per-uplink statistics may not be available in all partitions resulting in partitions lacking the required statistics.

5) Statistic Reflectors: To reduce the strain at low-bandwidth gateways, particular well-connected sites can be designated as statistic reflectors that would forward statistics as needed. The assignment can be either static or dynamic. If a single reflector was used, the concept would remain relatively straightforward. Yet, to increase robustness and balance load across multiple sites, it is beneficial to have multiple reflectors.

6) Distributing Control Messages Over Time: The process of sending statistics to the different remote sites should be distributed over time. While this method does not reduce the required bandwidth it helps avoiding burst of control messages. This method could also be applied to any data sent to one site, yet this increases header overhead.

C. Multi-Flow Policies

The described distributed SD-WAN solution facilitates policy-based routing for specific traffic flows. Nonetheless, in certain scenarios, there may be the desire to direct multiple flows over a single pair of gateways or even the same connection, e.g., when using intrusion detection systems, or multi-flow applications expecting similar latency across all flows. If multiple flows share the same source and destination subnet, disjunctive traffic descriptions can be used within a single inter-site policy, e.g., all TCP traffic with destination ports 443 or 8443 between 10.0.0.0/24 and 192.168.0.0/24. Any flow matching the description will be treated as a single flow and will always use the same connection.

However, grouping of multiple flows with different subnets, e.g., all HTTP flows destined to 172.168.0.0/24 or 182.168.0.0/24 or 192.168.0.0/24, is not possible out of the box, since the inter-site policies only define flows between exactly one subnet at each site (see Section III-A). Thus, such a global policy would result in distinct inter-site policies for every pair of subnets, each deciding independently which connection to choose. Consequently, multiple different connections could be utilized. Forcing a set of inter-site

policies to only use connections between a pair of gateways or even one specific connection, requires additional coordination between these inter-site policies, especially if partitioning scenarios within in cluster shall be considered. To avoid this additional coordination, we introduce the concept of virtual multi-flow groups, which aggregate a set of inter-site policies that should be treated uniformly. Configuration options allow specifying whether this "uniform treatment" means using a single connection (no load-balancing wanted) or to utilize connections between one pair of gateways (load-balancing over different connections still possible).

To implement the preference of making uniform decisions, we modify the process of selecting a connection from the class of best connections. The objective is not to impose a decision, but rather to guide independent inter-site policies toward a cohesive resolution, eliminating the necessity for explicit coordination. Instead of choosing a class member randomly, we enhance the likelihood of selecting a connection to the gateway with the highest number of subnets associated with the multi-flow group. In case of multiple candidates, connections to the gateway with the lower gateway ID take precedence. In configurations mandating a single connection, this choice is further refined by incorporating the connection ID.

To reduce the number of used connections in the long term, these specialized inter-site policies are not obliged to remain on a previous selected connection that is still part of class of the best connection. Instead, an inter-site policy can sporadically check if a different connection within the class of best connections exists with a gateway announcing more of the associated subnets. Thus, if a pair of gateways announces all required subnets and its connections are part of the class of the best connections, all inter-site policies part of the multi-flow group will eventually transition to that connection. It is important to note that if a connection announces all needed subnets, but does not fall within the class of best connections, it will not be used. In case, no pair exists offering all required subnets, multiple pairs are used. When a consistent decision over a single connection is required, it is imperative to utilize only one metric program. Yet, if the sole requirement is for the pair of gateways to be identical, multiple different metric programs can be employed.

In essence, the introduction of multi-flow groups provides a straightforward method of uniformly routing specific traffic flows without the need for additional complex synchronization methods. This approach is applicable in cluster scenarios where members are connected to different sets of local subnets and is also capable of handling cluster partitioning.

V. Evaluation

The evaluation of the distributed SD-WAN solution follows the structure of the requirements in Section I. The proposed fully distributed solution (N1, N6) is able to implement global traffic policies. Matched flows (F2) are steered based on programmable metric functions (N5), which reflect the current QoS properties of the used WAN connections (F1). The decisions made by one intent leader do not only facilitate steering but can also block specific traffic flows (F6). Additionally, load balancing among a site's cluster members is attainable (F4), yet symmetric routing is ensured at all time (F3).

In the following, we first evaluate the time required to respond to changing network conditions (N3). We will then discuss robustness (N2) and support of high availability, especially in case of geo-redundant gateways (F5), and its scalability (N5).

A. Reaction Time

The evaluation of the response time is structured as follows: First, we discuss the reaction delay at sudden network changes, e.g., a failing connection. Next, we estimate the duration for which a connection switch can be artificially delayed enhancing the overall network stability, if the currently selected connection is only marginally outside the *class of best connection*.

1) Minimal Reaction Time: The routing decision for a specific policy involves several communication steps that impact the minimum reaction time. Initially, an inter-site policy leader must be selected. Next, the network conditions, in form of connection, uplink and device statistics, are regularly sent to the leader. The leader then calculates the appropriate decision and distributes it. In the event of a network or device failure, the failure must first be detected, and a new leader may need to be determined. The process of recalculating and distributing a decision is then repeated.

The selection of the leading site is done implicitly when setting up a connection. The identification of the correct policy leader happens implicitly when decisions are exchanged within a site. Consequently, both selection mechanisms do not add anything to the reaction time required for regular network condition changes.

Each gateway regularly collects local information and sends the statistics every t_{timer} to the leader. Updates from a remote site take $1/2\ \mathrm{RTT}_{inter}$ before being received at the leading site. Data measured at the leading site takes $1/2\ \mathrm{RTT}_{intra}$ to reach the inter-site policy leader. Every t_{timer}, the policy leader calculates a decision based on the received data and sends it to all gateways participating in the policy. If a remote gateway has no direct connection to the leader, updates to and decisions from the leader have to be forwarded by other gateways of the leading site adding $1/2\ \mathrm{RTT}_{intra}$ per direction. All in all, the reaction time sums up to:

$$t_{react} \leq \mathrm{RTT}_{intra} + \mathrm{RTT}_{inter} + 2 \times t_{timer}$$

In local clusters, members are expected to communicate without significant delay resulting in $\mathrm{RTT}_{intra} \approx 0$. Further, both the statistics and decision timer have an expected delay of $1/2\ t_{timer}$ until the next trigger. Consequently, the expected reaction time is:

$$E(t_{react}) = \mathrm{RTT}_{inter} + t_{timer}$$

The gateway that realizes the inter-site policy's routing automatically becomes its leader. Hence, a failure of the currently used connection can be handled immediately by

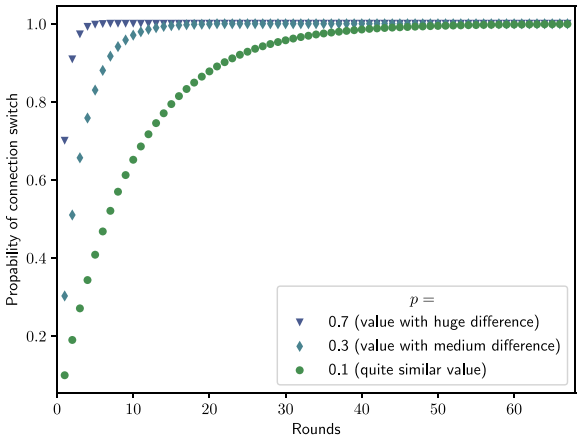

Fig. 7. Probability of switching into the class of best connections for different values of p corresponding to a geometric distribution.

calculating an appropriate response and informing both the local and remote gateways (orange color in Fig. 5):

$$t_{con\_failure} = \frac{1}{2}\,\mathrm{RTT}_{intra} + \frac{1}{2}\,\mathrm{RTT}_{inter}$$

Gateways at the leading site detect the failure of a policy leader when decisions are not renewed within its lifespan t_{exp}. If a gateway detects a failure, it instantly assumes itself as the new leader, calculates a new decision, and distributes it. As multiple gateways may have elected themselves, a uniform routing decision is reached after all cluster members distributed their decisions and a single leader is chosen using the rules described in Section III-B. Thus, the reaction time after a failure of the leading gateway is:

$$t_{lead\_failure} < t_{exp} + \frac{1}{2}\,\mathrm{RTT}_{intra} + \frac{1}{2}\,\mathrm{RTT}_{inter}$$

2) Artificial Delay of Reaction Time: To maintain a high robustness and not immediately switch a connection to a supposedly better one, Section IV-A introduced the concept of artificially extending the reaction time. Nonetheless, even if a connection switch is delayed, eventually all inter-site policies should use a connection inside the class of best connections. Fig. 7 displays the probability of switching a connection when *n* decision intervals (rounds) have passed. The different values of p describe the difference between the current metric and the *class of best connections*.

If the decision interval is configured to be 10 seconds, inter-site policies with a comparatively poor metric value ($p = 0.7$) have a 99.9% chance for switching into the *class of best connections* in under 1 minute/6 rounds. For policies with quite similar metric values ($p = 0.1$) it takes 11 minutes/66 rounds to reach the same probability. Thus, a connection that briefly leaves the class of best connections will not lose the policies assigned to it. Nonetheless, every policy whose metric value is not inside the *class of best connections* for a long time will eventually switch into it. Yet, if the currently used connection fails, a new decision is made without any artificial delay.

B. Robustness and High Availability Support

Section III describes how to fulfill the stated functional SD-WAN requirements without requiring any kind of exposed infrastructure component. By using inter-site policies instead of global policies and selecting an inter-site policy-specific leader, only one instance controls the routing of a specific traffic flow. Thus, if two gateways can communicate, they are also able to coordinate decisions to realize the optimal paths according to the deployed SD-WAN policies. The proposed algorithms in Section III-B ensure that independent from any previous network failure scenario eventually ($1/2$ RTT after connectivity is regained) a single leader persists per policy. The algorithms not only support a connection between single-gateway sites but also clustered gateways. If a gateway fails, all its decisions time out, and the appropriate inter-site policy leader roles will be taken over by other gateways of the leading cluster. The described algorithm ensures that only one gateway persists as the leader for a specific policy.

To ensure high robustness and availability, even in the case of geo-redundant clusters, the architecture must be capable of handling high-latency intra-cluster connections. Section V-A shows that the reaction time on changing network conditions and failures depends on the RTT_{intra}. Yet, the impact is only additive and does not depend on the number of cluster gateways. In contrast to the local cluster, the delay between members of a geo-redundant clusters is subject to its physical distance. Hence, RTT_{intra} can be expected similar to RTT_{inter} resulting in:

$$E(t_{react\_geo}) = 2 \times \mathrm{RTT}_{inter} + t_{timer}$$

The formulas for calculating the reaction time after network and gateway failures remain the same between geo-redundant and non-geo-redundant scenarios. In case, two gateways of a (geo-redundant) cluster do not reach each other because of a network failure, the cluster falls into two partitions. Nodes that do not reach their current policy leader start the process of determining a new policy leader using the rules stated in Section III-B. Nevertheless, each individual inter-site policy has at most one inter-site policy leader, as each cluster partition manages distinct and non-overlapping subnets. As outlined in Section IV-C, for multi-flow groups a partition might result in a routing over different connections of the separate flows if the corresponding leading gateways belong to different partitions. Yet, it is crucial to note that if flows belong to different subnets that cannot reach each other, it is inherently impossible to route them over the same connection. If connectivity is regained, the presented reassignment rule will eventually route all flows over the same connection, if possible.

C. Scalability

In Section V-A, we observed that the reaction time of the concept is not a critical factor for scalability. This is because it remains constant regardless of the number of sites, gateways, or gateways per site. Instead, the scalability of the overall solution is primarily determined by three key factors:

1) The execution time and memory usage of the programmable metric functions.

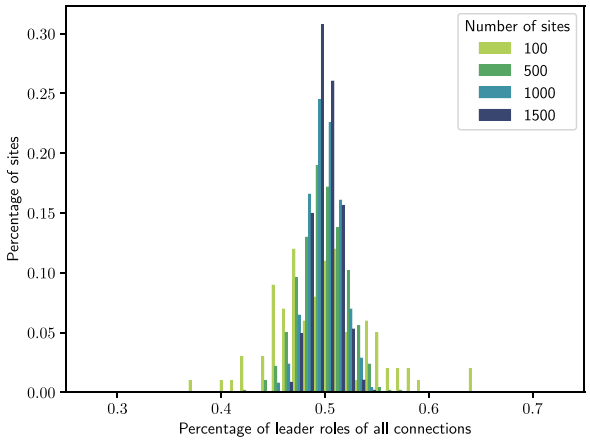

Fig. 8. Distribution of percentage of leader roles between varying number of sites with a standard library *SipHash* algorithm.

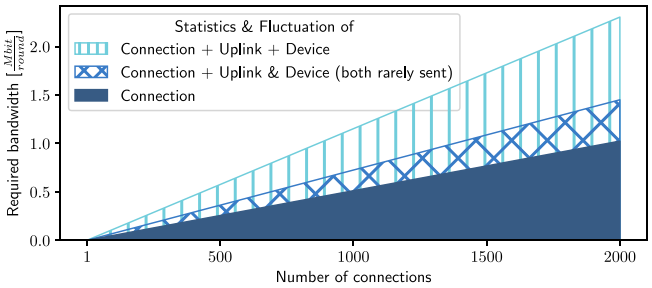

Fig. 9. Bandwidth requirements for sending statistics and its fluctuation of a single gateway with one WAN uplink. Optimization strategy of rarely sending uplink and device statistics (every third interval) is depicted.

2) How evenly the evaluation tasks can be distributed among multiple gateways.
3) The amount of data that needs to be transferred between gateways and a leader.

When building a prototype, we discovered that the first two factors pose no significant challenge:

1) CPU and Memory Usage: The CPU usage scales linear with the number of metric calculations. This corresponds to the number of inter-site policies $\times$ the number of connections. In our prototype, a simple metric function requires 775 kB of memory and takes approximately 3.4 μs to calculate on one 2.20 GHz (Xeon Gold 5120) core.

For example, if one CPU core is reserved for the distributed SD-WAN software, configured with a 10 second interval time, a gateway could handle up to $\sim$2800000 metric calculations (including $\sim$2.0% base utilization). Breaking down the 2800000 calculations into 2000 sites connected over two WANs, each gateway could be configured with 700 inter-site policies with individual metric functions. Assuming 700 metric functions are used, about 550 MB of memory would be needed for the metric function handling. Note that if metric functions are reused by multiple inter-site policies, connections between two sites must be weighted only once per metric. Hence, utilizing a shared metric function among multiple policies for the same pair of sites will reduce the required CPU resources.

2) Distributing the Calculations: Distributing the leader roles equally among all sites reduces the number of calculations required for an individual gateway by half. Fig. 8 shows that by utilizing the deterministic selection of a leader site (as described in Section III-B), an equal distribution will be achieved if the number of gateways is sufficiently high. Note that even for a low number of gateways the skew is negligible, since the additional load is low.

Additionally, if the calculations are distributed among multiple gateways, i.e., inside a cluster, the load is further distributed. Yet, the intra-cluster distribution depends on the realization of the concrete inter-site policies. One well connected gateway might handle all policies and become the leader for all policies. Thus, for cases where all metric functions are quite similar or one cluster member offers a comparably good connection, appropriate computing resources should be provisioned.

Note that distributing calculations over multiple gateways will not necessarily reduce memory usage because the metric functions must be stored on all gateways to enable rapid response to a policy leader change. However, like discussed in the previous section, this does still not pose an issue.

3) Bandwidth Overhead: The following evaluation of the bandwidth overhead of the payloads is partially based on an initial prototype that we had built for conducting feasibility tests. Yet, some initial assumptions from our previous work [1] led to a varying scalability of different factors. Thus, we leverage the linearity of the different factors to extrapolate the behavior for high numbers of gateways, connections, and policies in the following. Although this method may not be the most precise way to evaluate the required bandwidth, it provides an insight into the concept's capabilities as well as potential challenges when implementing it.

As the leading roles are equally distributed among all sites, each site transmits and receives statistics (see Table I) as well as decisions. Since the statistics must be exchanged directly among all connected gateways, certain data will be transferred multiple times. However, because decisions are made exclusively between two sites, connection-specific data must only be sent once between two connection endpoints (SD-WAN gateways). The quantity of data required for this purpose is denoted by the term *connection* in Fig. 9 and corresponds exactly to the volume of connection information that has to be sent in centralized scenarios. Still, the gateway's per-uplink and device-specific information must be disseminated to all (leading) gateways, in contrast to a single transmission to the controller. For ease of representation, Fig. 9 shows the worst-case (one-to-one) connection scenario, where deduplication of uplink and per device statistics is not possible. The plot reveals that with a single uplink and no bandwidth optimization, the uplink and device statistics transmitted to all remote sites consume roughly the same bandwidth as the connection statistics. In scenarios where the anticipated fluctuations of uplink or device statistics are minimal, the sending interval could be adapted (*rarely sent* in Fig. 9) and dynamically increased in response to significant changes. If additional connections were established to remote leading gateways, device-specific statistics would only need to be sent once. Consequently, the required bandwidth for device statistics

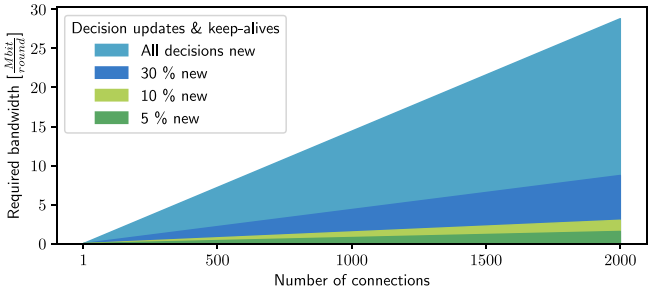

Fig. 10. Bandwidth requirement for different percentages of decision updates per interval across 100 policies. All updates involve gateway changes. If no new decision is made, a keep-alive packet is sent.

would remain unchanged. The same principle applies to uplink statistics when multiple connections share an uplink to the same leading site.

After evaluating and choosing a connection using the class of best connections, two cases can occur. First, the remote gateway of the newly selected connection may differ from the previously chosen one. In this situation, the leading gateway must notify both remote gateways by sending the policy ID (8 byte) and whether the policy is to be activated or deactivated (1 byte) over the corresponding connection, requiring a total transmission of 18 bytes. Second, when only a different connection to the same remote gateway is selected, only a single gateway has to be informed, resulting in a transmission of 9 bytes. The scenario where no prior decision was made aligns with the second case.

Fig. 10 illustrates that, analogous to centralized scenarios, this demand is multiplied by the number of connections. This multiplication results in a substantial bandwidth requirement when all decisions are updated in every round. Yet, in stable networks, the frequency of connection changes is typically low, resulting in a considerably reduced demand for bandwidth. For instance, in a scenario with a 10-second decision interval and a 5 % change rate, every single decision is renewed approximately every 3 minutes, indicative of a highly unstable network. Further, the plot assumes that decision updates consistently entail a switch of the remote gateway, necessitating the transmission of two update messages. In practice, we assume that the majority of decision updates merely involve switching to a different connection with the same remote gateway. Thus, only one message needs to be sent, effectively halving the required bandwidth.

In summary, for 2000 connections with uplink and device information rarely sent and a 5 % change rate, each gateway is expected to approximately transmit and receive 4 Mbit per round. With a configured round interval of 10 seconds, this translates to a transmit and receive bandwidth of 0.4 Mbit/s at each site. Yet, it is worth noting that commercial solutions employ much larger intervals, such as the 10-minute interval used in Cisco SD-WAN [20], which would result in an average bandwidth requirement of 7 kbit/s.

VI. Conclusion & Future Work

This paper presented a novel approach to fully distribute the control plane of SD-WANs, which enables automatic, fine-grained traffic control without introducing exposed entities. As a result, the network control plane remains highly available and robust, even in the event of device failures or network partitioning. This is achieved by three methods. The first is to split global network policies into local inter-site policies. This eliminates the need for complex consent algorithms when deciding how to route traffic. It facilitates distributed decisions over policy-based routes, even if gateways are placed in a geo-redundant clustered setup with high communication latency. Second, to enable a simple and comprehensible way of programming the distributed control plane, we propose to attach programmable metric functions to the policies. These functions are executed inside the distributed instances to evaluate the quality of appropriate connections. Third, the paper introduced a method for dynamically selecting an optimal path for routing without introducing instabilities to the network. This is achieved by considering not only the current quality of a connection, but also its fluctuations. The quantitative evaluation shows that this solution can manage several thousand nodes and inter-site policies in terms of computation. By avoiding a central controller, statistics and decisions have to be exchanged between the gateways. Consequently, non-connection-specific information must be transferred multiple times. Yet, with some of the suggested bandwidth optimizations in place, the increased need for bandwidth is only marginal compared to a centralized solution.

It should be noted that the described methods may not only be used in combination but could also be applied separately. Considering the fluctuation of an input value instead of using only thresholds adds network stability to any SDN/SD-WAN controlled network. In future work, we plan to evaluate the scalability and robustness in large real-world setups. Additionally, we aim to develop effective methods for managing and monitoring the distributed environment, particularly in scenarios involving partitioning. Further, the interaction between the proposed distributed SD-WAN control plane and the site-local routing has to be assessed.

In summary, this paper introduced new methods for achieving a highly robust and available traffic engineering solution that can be used to implement fine-grained global policies while remaining responsive even in the event of device failures, network failures, or network partitioning.

References

[1] F. Altheide, S. Buttgereit, M. Rossberg, and G. Schaefer, "Increasing resilience of SD-WAN by distributing the control plane," in *Proc. 14th Int. Conf. Netw. Future (NoF)*, 2023, pp. 10–18.
[2] W. Stallings, *Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud*. Boston, MA, USA: Addison-Wesley Prof., 2015.
[3] V. Paxson, "End-to-end routing behavior in the Internet," *IEEE/ACM Trans. Netw.*, vol. 5, no. 5, pp. 601–615, Oct. 1997.
[4] S. Vissicchio, O. Tilmans, L. Vanbever, and J. Rexford, "Central control over distributed routing," in *Proc. ACM SIGCOMM*, 2015, pp. 43–56.
[5] V. Yazici, M. O. Sunay, and A. O. Ercan, "Controlling a software-defined network via distributed controllers," 2014, *arXiv:1401.7651*.
[6] A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, and R. R. Kompella, "ElastiCon; an elastic distributed SDN controller," in *Proc. ANCS*, 2014, pp. 17–27.
[7] M. F. Bari et al., "Dynamic controller provisioning in software defined networks," in *Proc. CNSM*, 2013, pp. 18–25.
[8] L. Lamport, "The part-time parliament," *ACM Trans. Comput. Syst.*, vol. 16, no. 2, pp. 133–169, May 1998.

[9] T. Zhang, P. Giaccone, A. Bianco, and S. De Domenico, "The role of the inter-controller consensus in the placement of distributed SDN controllers," *Comput. Commun.*, vol. 113, pp. 1–13, Nov. 2017.
[10] E. Sakic and W. Kellerer, "Impact of adaptive consistency on distributed SDN applications: An empirical study," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 12, pp. 2702–2715, Dec. 2018.
[11] A. Tootoonchian and Y. Ganjali, "HyperFlow: A distributed control plane for OpenFlow," in *Proc. INM/WREN*, 2010, pp. 10–5555.
[12] T. Koponen et al., "Onix: A distributed control platform for large-scale production networks," in *Proc. 9th USENIX OSDI*, 2010, pp. 351–364.
[13] S. H. Yeganeh and Y. Ganjali, "Kandoo: A framework for efficient and scalable offloading of control applications," in *Proc. 1st Workshop Hot Topics Softw. Defin. Netw.*, 2012, pp. 19–24.
[14] Y. Fu, J. Bi, K. Gao, Z. Chen, J. Wu, and B. Hao, "Orion: A hybrid hierarchical control plane of software-defined networking for large-scale networks," in *Proc. 22nd ICNP*, 2014, pp. 569–576.
[15] D. Marconett and S. B. Yoo, "Flowbroker: A software-defined network controller architecture for multi-domain brokering and reputation," *J. Netw. Syst. Manag.*, vol. 23, no. 2, pp. 328–359, 2015.
[16] E. Brewer, "CAP twelve years later: How the 'rules' have changed," *Computer*, vol. 45, no. 2, pp. 23–29, 2012.
[17] P. Bailis and A. Ghodsi, "Eventual consistency today: Limitations, extensions, and beyond," *Commun. ACM*, vol. 56, no. 5, pp. 55–63, 2013.
[18] A. Haas et al., "Bringing the Web up to speed with Web Assembly," in *Proc. ACM SIGPLAN*, 2017, pp. 185–200.
[19] A. Gupta, A. Bansal, and V. Khanduja, "Modern lossless compression techniques: Review, comparison and analysis," in *Proc. 2nd ICECCT*, 2017, pp. 1–8.
[20] *Policies Configuration Guide for Vedge Routers, Cisco SD-WAN*, Cisco Syst. Inc., San Jose, CA, USA, 2022.

Friedrich Altheide received the B.Sc. and M.Sc. degrees in computer science from Technische Universität Ilmenau, Germany. In 2022, he joined Telematics/Computer Networks Research Group, where he worked on intent-driven distributed control planes for software-defined wide area networks and virtual private networks. He further researched on approaches to secure VPNs by combining asymmetric key exchanges with symmetric in-band and out-of-band key exchanges, such as quantum key distribution (QKD) and multipath key reinforcement (MKR). He is currently working with secunet Security Networks AG in the fields of VPNs, distributed SDN, PQC, QKD, and MKR.

Simon Buttgereit received the B.Sc. and M.Sc. degrees in computer engineering from Technische Universität Ilmenau, where he is currently pursuing the Ph.D. degree, focusing on automatically verifying security properties of critical infrastructure network architectures. Over the past few years, he has played an active role in various research projects with Telematics/Computer Networks Research Group with partners spanning the security, telecommunications, and network supply industries. His research interests primarily revolve around addressing the complexities of ensuring security and robustness in computer networks, all while harnessing the potential of modern networking technologies, such as software-defined networking, network function virtualization, and user workload virtualization.

Michael Rossberg (Member, IEEE) received the degree in computer science from Technische Universität Ilmenau, Germany, in 2007, and the Ph.D. degree in 2011 on the automatic configuration of large-scale VPN. In 2007, he joined Telematics/Computer Networks Research Group. Since then, he continued research in this field following academic and practical questions and broadening the scope to cloud technologies and SD-WAN applications.