

cats
Internet-Draft
Intended status: Informational
Expires: 30 August 2026

C. Li, Ed.
Huawei Technologies
Z. Du
China Mobile
M. Boucadair, Ed.
Orange
L. M. Contreras
Telefonica
J. Drake
Independent
26 February 2026

A Framework for Computing-Aware Traffic Steering (CATS)
draft-ietf-cats-framework-20

Abstract

This document describes a framework for Computing-Aware Traffic Steering (CATS). Specifically, the document identifies a set of CATS functional components, describes their interactions, and provides illustrative workflows of the control and data planes.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 30 August 2026.

Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document.

Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

| | |
|---|----|
| 1. Introduction | 3 |
| 2. Terminology | 4 |
| 3. CATS Framework and Components | 7 |
| 3.1. Assumptions | 7 |
| 3.2. CATS Identifiers | 7 |
| 3.3. Framework Overview | 8 |
| 3.4. CATS Functional Components | 9 |
| 3.4.1. Service Sites, Service Instances, and Service Contact Instances | 10 |
| 3.4.2. CATS Service Metric Agent (C-SMA) | 11 |
| 3.4.3. CATS Network Metric Agent (C-NMA) | 11 |
| 3.4.4. CATS Path Selector (C-PS) | 12 |
| 3.4.5. CATS Traffic Classifier (C-TC) | 12 |
| 3.4.6. CATS-Forwarders | 12 |
| 3.4.7. Underlay Infrastructure | 13 |
| 4. CATS Framework Workflow | 13 |
| 4.1. Service Announcement | 14 |
| 4.2. Metrics Distribution | 14 |
| 4.3. Service Access Processing | 15 |
| 4.4. Service Contact Instance Affinity | 15 |
| 5. Operational Considerations | 16 |
| 5.1. Provisioning of CATS Components | 16 |
| 5.2. Supervision of CATS Components & CATS OAM | 17 |
| 5.3. Deployment Considerations | 18 |
| 5.4. Implementation Considerations on Using CATS Metrics | 19 |
| 5.5. Verifying Correct Operations | 20 |
| 5.6. Impact on Network Operations | 20 |
| 6. Security Considerations | 20 |
| 7. Privacy Considerations | 21 |
| 8. IANA Considerations | 21 |
| 9. Informative References | 21 |
| Appendix A. Deployment Examples | 24 |
| A.1. Distributed Model | 24 |
| A.2. Centralized Model | 26 |
| A.3. Hybrid Model | 28 |
| Appendix B. Acknowledgements | 29 |
| Contributors | 30 |
| Authors' Addresses | 31 |

1. Introduction

Computing service architectures evolved from a single service site to multiple, sometimes collaborative, service sites to address various issues such as long response times or suboptimal utilization of service and network resources (e.g., resource under-utilization or exhaustion).

The underlying networking infrastructures that include computing resources usually provide relatively static service dispatching, e.g., the selection of the service instances for a request. In such infrastructures, service-specific traffic is often directed to the closest service site from a routing perspective without considering the actual network state (e.g., traffic congestion conditions) or the service site state.

As described in [I-D.ietf-cats-usecases-requirements], traffic steering that takes into account computing resource metrics would benefit several services, including latency-sensitive services such as immersive services that rely upon the use of augmented reality or virtual reality (AR/VR) techniques. This document provides an architectural framework that aims at facilitating the making of compute- and network-aware traffic steering decisions in dynamic networking environments with variable computing service resources.

Today, organizations often distribute user services across on-premises and cloud service provider networks. To support both redundancy and responsiveness, the Computing-Aware Traffic Steering (CATS) framework supports single or multiple service instances providing one given service, which may exist in one or more service sites. Clients access service instances via client-facing service functions known as service contact instances. A single service site may host one or multiple service contact instances. A single service site may have limited computing resources available at a given time, whereas the various service sites may experience different resource availability issues over time. Therefore, steering traffic among different service sites can address resource limitations in a specific service site.

Steering in CATS aims to select the appropriate service contact instance to service a request according to a set of network and computing metrics. This selection may not reveal the actual service instance that a client will invoke, e.g., in hierarchical or recursive contexts. Therefore, the metrics of the service contact instance may be aggregate metrics from multiple service instances.

The CATS framework is an overlay framework for the selection of the suitable service contact instances from a set of candidates. A combination of networking and computing metrics determines the exact characterization of services as 'suitable' or not.

Furthermore, this document describes a workflow of the main CATS procedures (Section 4) executed in both the control and data planes.

This document assumes that CATS functional elements are hosted in a provider network. As such, it is out of scope to discuss deployment options where such elements are co-located with a client.

2. Terminology

This document makes use of the following terms:

Client: An endpoint that connects to a service provider network.

Flow: A logical grouping of packets during a time interval, identified by some fields from the packet header, such as the 5-tuple transport coordinates (source address and destination address, source and destination port numbers, and protocol).

Computing-Aware Traffic Steering (CATS): A traffic engineering approach [RFC9522] that takes into account the dynamic nature of computing resources (e.g., compute and storage) and network state to optimize service-specific traffic forwarding towards a given service contact instance. The CATS framework leverages various metrics to enable computing-aware traffic steering policies.

Metric: A quantitative measure that provides suitable input to a selection mechanism for CATS decision-making.

Computing metrics: Metrics specific to the computing resources in the underlying CATS systems as opposed to other metrics, such as network metrics. Examples of computing metrics are discussed in [I-D.ietf-cats-metric-definition].

Service: An offering that is made available by a service provider by orchestrating a set of resources (networking, compute, storage, etc.).

The service provider retains control of internal resources and the service logic. For example, these resources may be:

- * Exposed by one or multiple processes.

- * Provided by virtual instances, physical, or a combination thereof.
- * Hosted within the same or distinct nodes.
- * Hosted within the same or multiple service sites.
- * Chained to provide a service using a variety of means.

How a service provider structures its services remains out of the scope of CATS.

Service providers may provide the same service in many locations; each of them constitutes a service instance.

Computing Service: A service offered to a client by a service provider by orchestrating a set of computing resources.

CATS Service ID (CS-ID): An identifier representing a service, which the clients use to access it. See Section 3.2.

Service instance: A collection of running resources that are orchestrated following a service logic. When invoked by a client request, these resources will collectively provide the intended service.

A service provider may enable many service instances that adhere to the same service logic to provide the same service.

A service instance runs in a service site and one or more instances may service clients' requests.

Service site: A location that hosts the resources that implement one or more service instances.

A service site may be a node or a set of nodes.

Service contact instance: A client-facing function that is responsible for receiving requests in the context of a given service.

A service contact instance can handle one or more service instances.

Steering beyond a service contact instance is hidden to both clients and CATS components.

A service contact instance processes a client's service request

according to the service logic (e.g., handle locally or solicit backend resources).

A service contact instance is reachable via at least one Egress CATS-Forwarder.

Clients may access a service via multiple service contact instances running at the same or different locations (service sites).

A service contact instance may dispatch service requests to one or more service instances (e.g., a service contact instance that behaves as a service load-balancer).

CATS Service Contact Instance ID (CSCI-ID): An identifier of a specific service contact instance. See Section 3.2.

Service request: A request to access or invoke a specific service. CATS-Forwarders steer a service request to a service contact instance.

Clients generate service requests using service-specific protocols.

Clients send service requests to a service instance (identified by a CS-ID), without explicit knowledge of CATS-Forwarders.

CATS-Forwarder: A network entity that steers traffic specific to a service request towards a service contact instance according to forwarding decisions supplied by a CATS Path Selector (C-PS), which may or may not be part of a CATS-Forwarder.

A CATS-Forwarder may behave as an Ingress or Egress CATS-Forwarder. See Section 3.4.6.

Ingress CATS-Forwarder: An entity that steers service-specific traffic along a CATS-computed path that leads to an Egress CATS-Forwarder that connects to the most suitable service site that hosts the service contact instance selected to satisfy the initial service request.

Egress CATS-Forwarder: An entity located at the end of a CATS-computed path which connects to a service site.

CATS Path Selector (C-PS): A functional entity that selects paths

towards service sites and instances (and thus service contact instances) in order to accommodate the requirements of service requests. The path selection engine takes into account the service and network status information. See Section 3.4.4.

CATS Service Metric Agent (C-SMA): A functional entity that is responsible for collecting service capabilities and status, and for reporting them to a C-PS. See Section 3.4.2.

CATS Network Metric Agent (C-NMA): A functional entity that is responsible for collecting network capabilities and status, and for reporting them to a C-PS. See Section 3.4.3.

CATS Traffic Classifier (C-TC): A functional entity that is responsible for determining which packets belong to a traffic flow for a specific service request. It coordinates with the Ingress CATS-Forwarder so that such packets are placed onto a path computed by the C-PS that leads to the selected service contact instance. See Section 3.4.5.

3. CATS Framework and Components

3.1. Assumptions

CATS assumes that a service might be provided by one or multiple service instances. Such instances may be hosted within the same or distinct service sites. A given service is represented by the same service identifier (Section 3.2). CATS does not make any additional assumption about these instances other than that they are reachable via one or multiple service contact instances.

3.2. CATS Identifiers

CATS uses the following identifiers:

CATS Service ID (CS-ID): An identifier (ID) representing a service, which the clients use to access it. Such an ID identifies all the instances of a given service, regardless of their locations.

The CS-ID is independent of which service contact instance serves the service request.

Service requests are spread over the service contact instances that can accommodate them, considering the location of the initiator of the service request and the availability (in terms of resource/traffic load, for example) of the service instances resource-wise among other considerations like traffic congestion conditions.

CATS Service Contact Instance ID (CSCI-ID): An identifier of a specific service contact instance.

This document makes no assumptions about the structure or semantics of this identifier. One example of such an ID is a unicast IP address, which uniquely identifies the location of a service instance.

3.3. Framework Overview

A high-level view of the CATS framework, without expanding the functional entities in the network, is illustrated in Figure 1.

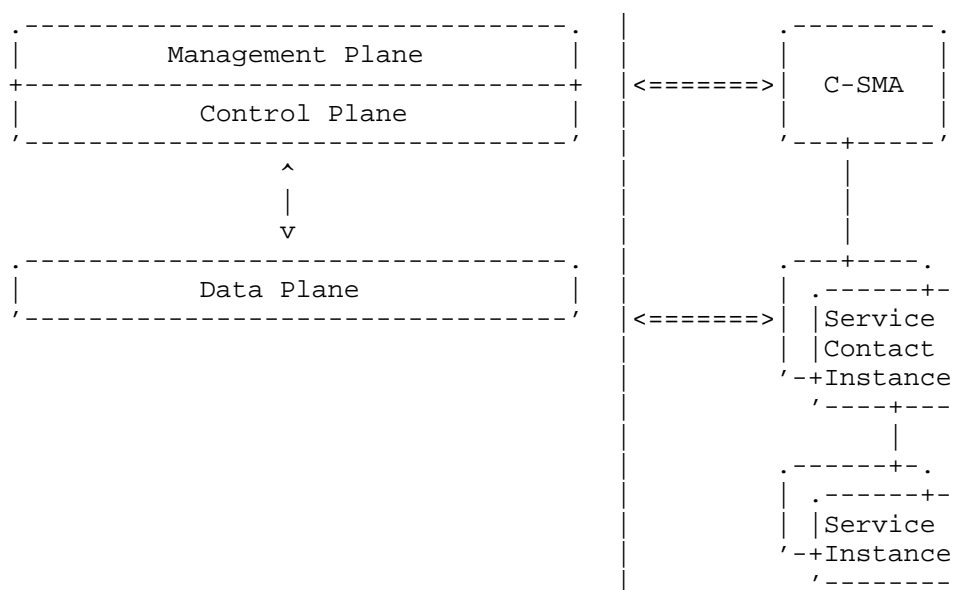


Figure 1: Main CATS Interactions

For the sake of illustration, "Service Instance" is shown as a single box in Figure 1. However, this does not imply that a service instance is hosted in a single node. Whether a service instance is realized by invoking resources within the same node or by chaining resources exposed by several nodes is deployment-specific.

The following planes are defined:

- * CATS Management Plane: Responsible for monitoring, configuring, and maintaining CATS network devices.

- * CATS Control Plane: Responsible for scheduling services based on computing and network information. It is also responsible for making decisions about how packets should be forwarded by involved forwarding nodes and communicating such decisions to the CATS Data Plane for execution.
- * CATS Data Plane: Responsible for computing-aware forwarding, including classifying packets, steering them onto chosen paths toward selected service contact instances, and forwarding the packets along the paths to the service contact instances.

Depending on implementation and deployment, these planes may consist of several functional components, and the details will be described in the following sections. For example, the control plane may consist of C-PS, C-NMA, etc. The data plane may consist of CATS-Forwarders, C-TC, etc.

3.4. CATS Functional Components

CATS nodes make forwarding decisions for a given service request that has been received from a client according to the capabilities and status information of both service contact instances and network. The main CATS functional components and their interactions are shown in Figure 2. These components are described in the subsections that follow.

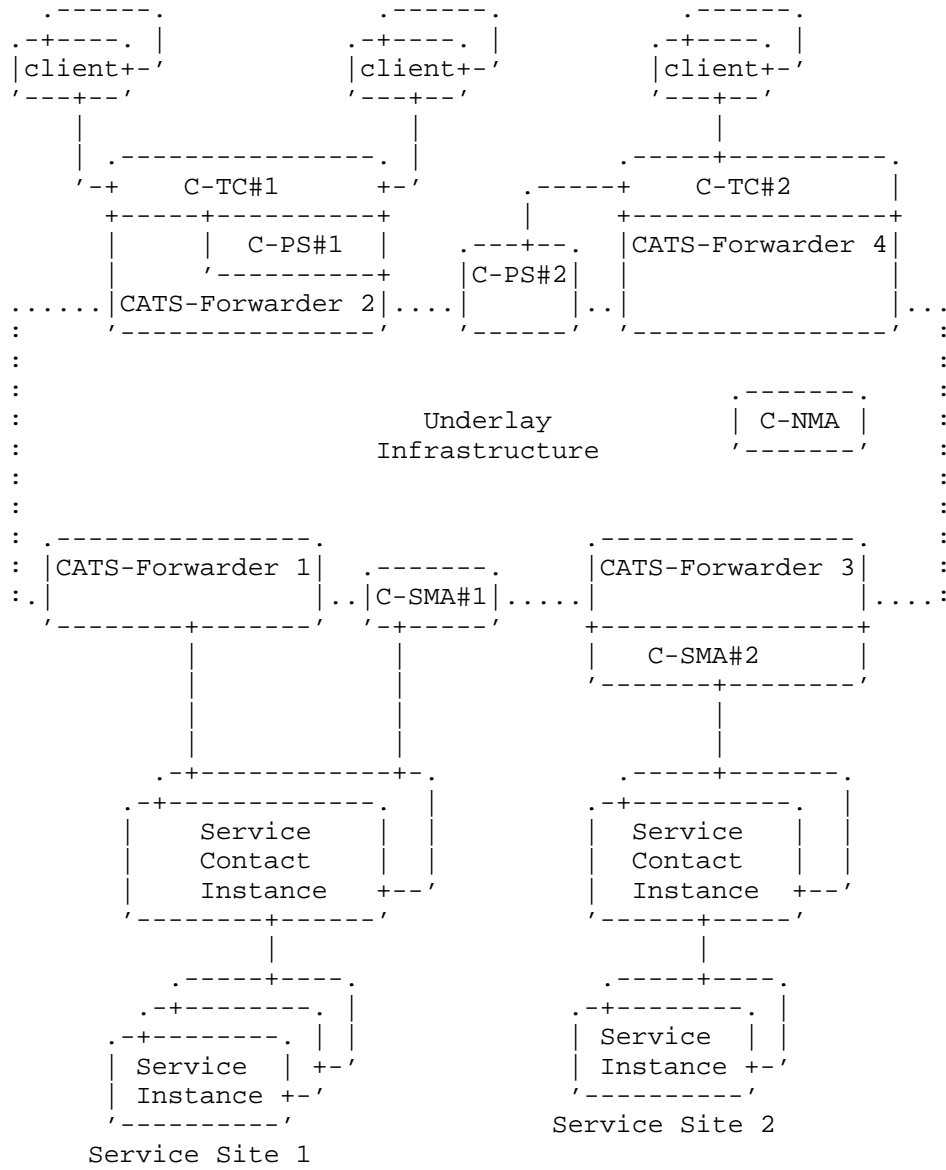


Figure 2: CATS Functional Components

3.4.1. Service Sites, Service Instances, and Service Contact Instances

Service sites are locations that host resources (including computing resources) that are required to offer a service.

A compute service (e.g., for face recognition purposes or a game server) is identified by a CATS Service Identifier (CS-ID). The CS-ID does not need to be globally unique, but must be sufficiently unique to unambiguously identify the service at all of the components of a CATS system.

A single service can be represented and accessed via several contact instances that run in the same or different regions of a network.

As service instances are accessed via a service contact instance, a client will not see the service instances but only the service contact instance.

Figure 2 shows two CATS nodes ("CATS-Forwarder 1" and "CATS-Forwarder 3") that provide access to service contact instances. These nodes behave as Egress CATS-Forwarders (Section 3.4.6).

Note: "Egress" is used here in reference to the direction of the service request placement. The directionality is called to explicitly identify the exit node of the CATS infrastructure.

3.4.2. CATS Service Metric Agent (C-SMA)

The CATS Service Metric Agent (C-SMA) is a functional component that gathers information about service sites and server resources, as well as the status of the different service instances. A C-SMA may be co-located or located adjacent to a service contact instance, hosted by or adjacent to an Egress CATS-Forwarder (Section 3.4.6), etc. There may be one or more C-SMAs in a deployment.

Figure 2 shows one C-SMA embedded in "CATS-Forwarder 3" and another C-SMA that is adjacent to "CATS-Forwarder 1".

3.4.3. CATS Network Metric Agent (C-NMA)

The CATS Network Metric Agent (C-NMA) is a functional component that gathers information about the state of the underlay network. The C-NMAs may be implemented as standalone components or may be hosted by other components, such as CATS-Forwarders or CATS Path Selectors (C-PSes) (Section 3.4.4).

C-NMA is likely to leverage existing techniques (e.g., [RFC7471], [RFC8570], and [RFC8571]).

Figure 2 shows a single, standalone C-NMA within the underlay network. There may be one or more C-NMAs for an underlay network.

3.4.4. CATS Path Selector (C-PS)

The C-SMAs and C-NMAs share the collected information with C-PSes that use such information to select the Egress CATS-Forwarders (and potentially the service contact instances) where to forward traffic for a given service request. C-PSes also determine the best paths (possibly using tunnels) to forward traffic, according to various criteria that include network state and traffic congestion conditions. The collected information is encoded into one or more metrics that feed the C-PS path selection logic. Such information also includes CS-ID and possibly CSCI-IDs.

There might be one or more C-PSes used to select CATS paths in a CATS infrastructure.

A C-PS can be integrated into CATS-Forwarders (e.g., "C-PS#1" in Figure 2) or may be deployed as a standalone component (e.g., "C-PS#2" in Figure 2). Generally, a standalone C-PS can be a functional component of a centralized controller (e.g., a Path Computation Element (PCE) [RFC4655]).

Refer to Section 4.2 for a discussion on metric distribution (including interaction with routing protocols).

3.4.5. CATS Traffic Classifier (C-TC)

The CATS Traffic Classifier (C-TC) is a functional component that is responsible for associating incoming packets from clients with service requests. C-TCs also ensure that packets that are bound to a specific service contact instance are all forwarded towards that same service contact instance, as instructed by a C-PS. To that aim, a C-TC uses CS-IDs (or their resolution of CS-ID to network locators) to classify service requests. Refer to Section 5.1 for more details about provisioning of classification rules.

Note that CS-IDs may be carried in packets if mechanisms such as TLS Server Name Indication extension (SNI) (Section 3 of [RFC6066]) are used.

C-TCs are typically hosted in CATS-Forwarders.

3.4.6. CATS-Forwarders

Ingress CATS-Forwarders are responsible for steering service-specific traffic along a CATS-computed path that leads to an Egress CATS-Forwarder. Egress CATS-Forwarders are the elements that behave as an egress for service requests that are forwarded over a CATS infrastructure.

A service site that hosts service instances may be connected to one or more Egress CATS-Forwarders (e.g., multi-homing design). If a C-PS has selected a specific service contact instance and the C-TC has marked the traffic with the CSCI-ID related information, the Egress CATS-Forwarder then forwards the traffic to the relevant service contact instance accordingly.

In some cases, the choice of the service contact instance may be left open to the Egress CATS-Forwarder (i.e., traffic is marked only with the CS-ID). In such cases, the Egress CATS-Forwarder selects a service contact instance using its knowledge of service and network capabilities as well as the current load as observed by the CATS-Forwarder, among other considerations. In the absence of an explicit policy, an Egress CATS-Forwarder must make sure to forward all packets that pertain to a given service request towards the same service contact instance.

Note that, depending on the design considerations and service requirements, per-service contact instance computing-related metrics or aggregated per-site computing-related metrics (and a combination thereof) can be used by a C-PS. Using aggregated per-site computing-related metrics appears as a preferred option scalability-wise, but relies on Egress CATS-Forwarders that connect to various service contact instances to select the proper service contact instance. An Egress CATS-Forwarder may choose to aggregate the metrics from different sites as well. In this case, the Egress CATS-Forwarder will choose the best site by itself when the packets arrive at it.

3.4.7. Underlay Infrastructure

The "underlay infrastructure" in Figure 2 indicates an IP and/or MPLS network that is not necessarily CATS-aware. The CATS paths that are computed by a C-PS will be distributed among the CATS-Forwarders (Section 3.4.6) and will not affect the underlay nodes. Underlay nodes are typically P routers (Section 5.3.1 of [RFC4026]).

4. CATS Framework Workflow

The following subsections provide an overview of a typical CATS workflow. In order to enable CATS in a given domain, some provisioning is needed; see more details in Section 5.1. Section 5.3 describes several deployment options (distributed, centralized, and hybrid models) to accommodate a variety of contexts.

4.1. Service Announcement

A service is associated by the service provider with a unique identifier called a CS-ID. A CS-ID may be a network identifier, such as an IP address. The mapping of CS-IDs to network identifiers may be learned through a name resolution service (e.g., DNS [RFC1034]). Note that CATS framework does not assume or preclude any specific name resolution service.

4.2. Metrics Distribution

As described in Section 3.4, a C-SMA collects both computing-related capabilities and metrics, and associates them with a CS-ID that identifies the service. The C-SMA may aggregate the metrics for multiple service contact instances, maintain them separately, or both.

The C-SMA then advertises CS-IDs along with metrics to related C-PSes in the network. Depending on the deployment choice, CS-IDs with metrics may be distributed in different ways. Refer to Section 5.4 for more deployment considerations.

The computing metrics include computing-related metrics and potentially other service-specific metrics like the number of clients that access the service contact instance at any given time, etc.

Computing metrics may change very frequently (e.g., see Section 5.3 of [I-D.ietf-cats-usecases-requirements] for a discussion). How frequently such information is distributed is to be determined as part of the specification of any communication protocol (including routing protocols) that may be used to distribute the information. Various options can be considered, such as (but not limited to) interval-based updates, threshold-triggered updates, policy-based updates, or using normalized metrics.

Additionally, the C-NMA collects network-related capabilities and metrics. These may be collected and distributed by existing measurement protocols and/or routing protocols, although extensions to such protocols may be required to carry additional information (e.g., link latency). The C-NMA distributes the network metrics to the C-PSes so that they can use the combination of service and network metrics to determine the best Egress CATS-Forwarder to provide access to a service contact instance and invoke the compute function required by a service request. Similar to computing-related metrics, the network-related metrics can be distributed using distributed, centralized, or hybrid schemes. This document does not describe such details since this is deployment-specific.

Network metrics may also change over time. Dynamic routing protocols may take advantage of some information or capabilities to prevent the network from being flooded with state change information (e.g., Partial Route Computation (PRC) of OSPFv3 [RFC5340]). C-NMAs should also be configured or instructed like C-SMAs to determine when and how often updates should be notified to the C-PSes.

4.3. Service Access Processing

A C-PS selects paths that lead to Egress CATS-Forwarders according to both service and network metrics that were advertised. A C-PS may be collocated with an Ingress CATS-Forwarder (as shown in Figure 3) or logically centralized (in the centralized or hybrid models (Section 5.3)).

This document does not specify any specific algorithm for path selection purposes to be supported by C-PSes so as not to constrain the CATS framework to one possible selection only. Instead, it is expected that a service request or local policy may feed the C-PS with appropriate information on that selection logic that takes the suitable metric information as input and the selected service contact instance as output. Such appropriate information may be utilized to differentiate selection mechanisms to enable service-specific selections.

In the example shown in Figure 3, the client sends a service request via the network through the "CATS-Forwarder 1", which is an Ingress CATS-Forwarder. Note that, a service request to access the service may consist of one or more service packets (e.g., Session Initiation Protocol (SIP) [RFC3261], HTTP [RFC9112], IPv6 [RFC8200], SRv6 [RFC8754], or Real-Time Streaming Protocol (RTSP) [RFC7826]) that carry the CS-ID and potential parameters. When a matching classification entry maintained by a C-TC is found for the packets, the Ingress CATS-Forwarder encapsulates and forwards them to the C-PS selected Egress CATS-Forwarder. When these packets reach the Egress CATS-Forwarder, the outer header of the possible overlay encapsulation will be removed and the inner packets will be sent to the relevant service contact instance.

4.4. Service Contact Instance Affinity

Service contact instance affinity means that packets that belong to a flow associated with a service request should always be sent to the same service contact instance. Furthermore, packets of a given flow should be forwarded along the same path to avoid mis-ordering and to prevent the introduction of unpredictable latency variations. A CATS framework implementation must ensure that service instance selection and path steering decisions remain consistent for a flow.

Specifically, the same Egress CATS-Forwarder needs to be solicited to forward the packets.

Ensuring service affinity for flows is a feature that can be configured on the C-PS when the service is deployed (i.e., all flows bound to a service) or determined at the time of newly formulated service requests (i.e., specific flow).

Note that different services may have different notions of what constitutes a 'flow' and may, thus, identify a flow differently. Typically, a flow is identified by the 5-tuple transport coordinates (source address and destination address, source and destination port numbers, and protocol). However, for instance, an RTP video stream may use different port numbers for video and audio channels: in that case, affinity may be identified as a combination of the two 5-tuple flow identifiers so that both flows are addressed to the same service contact instance.

Hence, when specifying a protocol to communicate information about service contact instance affinity to C-TCs in particular, the protocol should support flexible mechanisms for identifying flows. Or, from a more general perspective, there should be a mechanism to specify and identify the set of packets that are subject to a service contact instance affinity.

More importantly, the means for identifying a flow for ensuring instance affinity should be application-independent to avoid the need for service-specific instance affinity methods. However, service contact instance affinity information may be configurable on a per-service basis. For each service, the information can include the flow or packet identification type and means, affinity timeout value, etc.

This document does not define any mechanism for defining or enforcing service contact instance affinity.

5. Operational Considerations

5.1. Provisioning of CATS Components

Enabling CATS in a network can be done incrementally. That is, not all ingress routers (Provider Edges (PEs), typically) need to be upgraded to support CATS.

In addition to the CATS steering policies that are communicated by a C-PS to an Ingress CATS-Forwarder, some provisioning tasks are required. This includes, but is not limited to:

- * Provide C-PS elements with the locators of available Ingress CATS-Forwarder. Such locators may also be discovered from the network.
- * Supply information needed to connect C-PS elements with C-NMAs and C-SMAs.
- * Allocate identifiers CS-ID/CSCI-ID and bind them to specific service contact instances.
- * Provide C-PS elements with the set of optimization metrics (per service) and an optimization policy.
- * Configure specific encapsulation capabilities of CATS-Forwarders for use, including any credentials for mutual authentication between peer CATS-Forwarders.
- * Reset the classification table of C-TC elements.
- * Set the traffic counters at CATS-Forwarders to ease correlation between both Ingress and Egress CATS-Forwarders. Such a correlation is needed to help identify issues induced by the underlying encapsulation.

Provisioning includes configuration as well as distribution through protocols. Specifically, the above tasks can be enabled using a variety of means (NETCONF [RFC6241], IPFIX [RFC7011], RESTCONF [RFC8040], YANG-Push [RFC8639], etc.). It is out of scope to discuss required CATS extensions to these protocols.

5.2. Supervision of CATS Components & CATS OAM

Also, companion supervision and OAM tools are needed to drive CATS provisioning but also to assess the overall CATS operations. This includes, but is not limited to:

- * Expose classification capabilities of C-TC elements.
- * Expose encapsulation capabilities supported by CATS-Forwarders.
- * Retrieve the active classification table of C-TC elements.
- * Retrieve active steering rules in CATS-Forwarders.
- * Retrieve active installed policies in C-PSes.
- * Retrieve the traffic counters at CATS-Forwarders to ease correlation between both Ingress and Egress CATS-Forwarders.

- * Enable OAM tools to check the correct behavior of various entities (e.g., classification rules, steering rules, and forwarding behavior). See also Section 5.5.
- * Enable OAM tools for performance measurement.

5.3. Deployment Considerations

This document does not make any assumption about how the various CATS functional elements are implemented and deployed. Concretely, whether a CATS deployment follows a fully distributed design or relies upon a mix of centralized (e.g., a centralized C-PS) and distributed CATS functions (e.g., C-TCs) is deployment-specific, which may reflect the preferences and policies of the (CATS) service provider. The deployment can also be informed by specific use case requirements [I-D.ietf-cats-usecases-requirements].

For example, in a centralized design, both the computing-related metrics from the C-SMAs and the network metrics are collected by a (logically) centralized path computation logic (e.g., a PCE). In this case, the CATS computation logic may process incoming service requests to compute paths to service contact instances. More generally, the paths might be computed before a service request comes. Based on the metrics and computed paths, the C-PS can select the most appropriate path and then synchronize with C-TCs.

According to the method of distributing and collecting the computing metrics, three deployment models can be considered for the deployment of the CATS framework:

- * ***Distributed model***: Computing metrics are distributed among network devices directly using distributed protocols without interactions with a centralized control element (e.g., network controller). Service scheduling function is performed by the CATS-Forwarders in the distribution model, therefore, the C-PS is integrated into an Ingress CATS-Forwarder.
- * ***Centralized model***: Computing metrics are collected by centralized control elements. These elements then compute the forwarding path for service requests and syncs up with Ingress CATS-Forwarders. In this model, C-PS is implemented in a centralized control element.
- * ***Hybrid model***: Is a combination of distributed and centralized models.

A part of computing metrics are distributed among involved

network devices, and others may be collected by a centralized control element. For example, some static information (e.g., capabilities information) can be distributed among network devices since they are quite stable (i.e., change infrequently). Frequent changing information (e.g., resource utilization) can be collected by a centralized control element to avoid frequent flooding in the distributed control plane. Service scheduling function can be performed by a centralized control element, Ingress CATS-Forwarders (co-located with a C-PS), or both depending on the specific deployment policies.

When path computation is distributed, centralized control elements have to communicate the path information they collect to Ingress CATS-Forwarders (co-located with a C-PS) so that they take into account the full set of metrics for service scheduling.

Examples to illustrate these models are provided in Appendix A.

The framework covers only the case of a single service provider. Deployment considerations about the case of multiple service providers are out of scope.

5.4. Implementation Considerations on Using CATS Metrics

Advertising per-instance computing-related metrics instead of aggregating them into per-site advertisements has scalability implications on involved CATS elements. Special care should be considered by providers when enabling per-instance metric distribution.

Computing metrics need to be normalized (i.e., convert metric values with or without units into unitless scores), aggregated, or a combination thereof in order to soften the scalability impact while providing sufficient detail for effective CATS decision-making. See, e.g., [I-D.ietf-cats-metric-definition] for a discussion on metrics and distribution approaches.

Depending on the resources and processing capabilities of CATS components, the normalization and aggregation functions can be located in different CATS components. An approach is to implement the normalization and aggregation functions located away from C-PSes, especially when C-PSes are co-located with CATS-Forwarders. With this in mind, the normalization and aggregation functions of CATS metrics can be placed at service contact instances or C-SMAs.

When C-SMAs are co-located with CATS-Forwarders where there is limited resource for processing, the placement of normalization functions in a C-SMA may bring too much overhead and may influence the routing efficiency. Therefore, this document suggests implementing the normalization function at the service contact instances. Regarding the aggregation functions, it can be implemented in a C-SMA or the service contact instances.

In order to ensure consistent CATS decisions, the same normalization and aggregation functions must be enabled in all involved CATS components. Also, in the case of service contact instances and C-SMAs are provided by different vendors, it is needed to use the same common normalization function and aggregation functions, so that the service contact instance selection result can be fair among all the service contact instances. To that aim, a set of normalization and aggregation functions must be standardized. To accommodate contexts where multiple functions are supported, CATS implementations must expose a configuration parameter to control the activation of normalization and aggregation functions.

5.5. Verifying Correct Operations

A CATS implementation must log error events for better network management and operation. Means to assess the reachability and trace CATS paths should be supported.

5.6. Impact on Network Operations

Computing metrics are collected and distributed in CATS. A new function is needed to be deployed to manage the cooperation between network elements and computing elements. For example, this function may be provided by an orchestrator connecting with C-SMA and C-NMA. This might bring more complexity of the network management, especially if this function is not leveraged for other purposes beyond CATS.

6. Security Considerations

The computing resource information changes over time very frequently, especially with the creation and termination of service instances. When such information is carried in a routing protocol, too many updates may affect network stability. This issue could be exploited by an attacker (e.g., by spawning and deleting service instances very rapidly). CATS solutions must support guards against such misbehaviors. For example, these solutions should support aggregation techniques, dampening mechanisms, and threshold-triggered distribution updates.

The information distributed by the C-SMAs and C-NMAs may be sensitive. Such information could indeed disclose intelligence about the network and the location of compute resources hosted in service sites. This information may be used by an attacker to identify weak spots in an operator's network. Furthermore, such information may be modified by an attacker, resulting in disrupted service delivery for the clients, even including misdirection of traffic to an attacker's service implementation. CATS solutions must support authentication and integrity-protection mechanisms between C-SMAs/C-NMAs and C-PSes, and between C-PSes and Ingress CATS-Forwarders. Also, C-SMAs need to support a mechanism to authenticate the services for which they provide information to C-PS computation logics, among other CATS functions.

This document focuses on the scenario of a single service provider. Hence, security considerations relevant to deployment with multiple service providers are out of scope.

7. Privacy Considerations

CATS solutions must support preventing on-path nodes in the underlay infrastructure to fingerprint and track clients (e.g., determining which client accesses which service). More generally, personal data must not be exposed to external parties by CATS beyond what is carried in the packet that was originally issued by the client.

In some cases, the CATS solution may need to know about applications, clients, and even user identity. This information is sensitive and should be encrypted. To prevent the information leaking between CATS components, the C-PS computed path information should be encrypted in distribution. The specific encryption method may be applied at the network layer, transport layer, or at the application/protocol level depending on the implementation, so this is out of the scope of this document.

This document focuses on the scenario of a single service provider. Hence, privacy considerations relevant to deployment with multiple service providers are out of scope.

For more discussion about privacy, refer to [RFC6462] and [RFC6973].

8. IANA Considerations

This document makes no request for IANA action.

9. Informative References

[I-D.ietf-cats-metric-definition]

Yao, K., Li, C., Contreras, L. M., Ros-Giralt, J., and G. Zeng, "CATS Metrics Definition", Work in Progress, Internet-Draft, draft-ietf-cats-metric-definition-05, 2 February 2026, <<https://datatracker.ietf.org/doc/html/draft-ietf-cats-metric-definition-05>>.

[I-D.ietf-cats-usecases-requirements]

Yao, K., Contreras, L. M., Shi, H., Zhang, S., and Q. An, "Computing-Aware Traffic Steering (CATS) Problem Statement, Use Cases, and Requirements", Work in Progress, Internet-Draft, draft-ietf-cats-usecases-requirements-14, 2 February 2026, <<https://datatracker.ietf.org/doc/html/draft-ietf-cats-usecases-requirements-14>>.

[I-D.yao-cats-awareness-architecture]

Yao, H., wang, X., Li, Z., Huang, D., and C. Lin, "Computing and Network Information Awareness (CNIA) system architecture for CATS", Work in Progress, Internet-Draft, draft-yao-cats-awareness-architecture-02, 22 October 2023, <<https://datatracker.ietf.org/doc/html/draft-yao-cats-awareness-architecture-02>>.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<https://www.rfc-editor.org/rfc/rfc1034>>.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <<https://www.rfc-editor.org/rfc/rfc3261>>.

[RFC4026] Andersson, L. and T. Madsen, "Provider Provisioned Virtual Private Network (VPN) Terminology", RFC 4026, DOI 10.17487/RFC4026, March 2005, <<https://www.rfc-editor.org/rfc/rfc4026>>.

[RFC4655] Farrel, A., Vasseur, J.-P., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", RFC 4655, DOI 10.17487/RFC4655, August 2006, <<https://www.rfc-editor.org/rfc/rfc4655>>.

[RFC5340] Coltun, R., Ferguson, D., Moy, J., and A. Lindem, "OSPF for IPv6", RFC 5340, DOI 10.17487/RFC5340, July 2008, <<https://www.rfc-editor.org/rfc/rfc5340>>.

- [RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <<https://www.rfc-editor.org/rfc/rfc6066>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/rfc/rfc6241>>.
- [RFC6462] Cooper, A., "Report from the Internet Privacy Workshop", RFC 6462, DOI 10.17487/RFC6462, January 2012, <<https://www.rfc-editor.org/rfc/rfc6462>>.
- [RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <<https://www.rfc-editor.org/rfc/rfc6973>>.
- [RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <<https://www.rfc-editor.org/rfc/rfc7011>>.
- [RFC7471] Giacalone, S., Ward, D., Drake, J., Atlas, A., and S. Previdi, "OSPF Traffic Engineering (TE) Metric Extensions", RFC 7471, DOI 10.17487/RFC7471, March 2015, <<https://www.rfc-editor.org/rfc/rfc7471>>.
- [RFC7826] Schulzrinne, H., Rao, A., Lanphier, R., Westerlund, M., and M. Stiemerling, Ed., "Real-Time Streaming Protocol Version 2.0", RFC 7826, DOI 10.17487/RFC7826, December 2016, <<https://www.rfc-editor.org/rfc/rfc7826>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/rfc/rfc8040>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/rfc/rfc8200>>.

- [RFC8570] Ginsberg, L., Ed., Previdi, S., Ed., Giacalone, S., Ward, D., Drake, J., and Q. Wu, "IS-IS Traffic Engineering (TE) Metric Extensions", RFC 8570, DOI 10.17487/RFC8570, March 2019, <<https://www.rfc-editor.org/rfc/rfc8570>>.
- [RFC8571] Ginsberg, L., Ed., Previdi, S., Wu, Q., Tantsura, J., and C. Filsfils, "BGP - Link State (BGP-LS) Advertisement of IGP Traffic Engineering Performance Metric Extensions", RFC 8571, DOI 10.17487/RFC8571, March 2019, <<https://www.rfc-editor.org/rfc/rfc8571>>.
- [RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Notifications", RFC 8639, DOI 10.17487/RFC8639, September 2019, <<https://www.rfc-editor.org/rfc/rfc8639>>.
- [RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <<https://www.rfc-editor.org/rfc/rfc8754>>.
- [RFC9112] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP/1.1", STD 99, RFC 9112, DOI 10.17487/RFC9112, June 2022, <<https://www.rfc-editor.org/rfc/rfc9112>>.
- [RFC9522] Farrel, A., Ed., "Overview and Principles of Internet Traffic Engineering", RFC 9522, DOI 10.17487/RFC9522, January 2024, <<https://www.rfc-editor.org/rfc/rfc9522>>.

Appendix A. Deployment Examples

This section provides examples to illustrate examples of CATS metrics distribution. These examples are not deployment recommendations.

The following example mainly describes a per-instance computing-related metric distribution for illustration purposes. Such information may be aggregated into a single advertisement.

A.1. Distributed Model

Figure 3 shows an example of how CATS metrics can be disseminated in the distributed model.

There is a client attached to the network via "CATS-Forwarder 1". There are three service contact instances of the service with CS-ID "1": two service contact instances with CSCI-IDs "1" and "2", respectively, are located at "Service Site 2" attached via "CATS-Forwarder 2"; the third service contact instance is located at

"Service Site 3" attached via "CATS-Forwarder 3" and with CSCI-ID "3". There is also a second service with CS-ID "2" with only one service contact instance located at "Service Site 3".

The C-SMA collocated with "CATS-Forwarder 2" distributes the computing metrics for both service contact instances (i.e., (CS-ID 1, CSCI-ID 1) and (CS-ID 1, CSCI-ID 2)). Similarly, the C-SMA located at "Service Site 3" advertises the computing metrics for the two services hosted by "Service Site 3". The C-SMA may distribute the computing metrics to the Egress "CATS-Forwarder 3". Then, the computing metrics can be redistributed by the Egress CATS-Forwarder to the Ingress CATS-Forwarder. The C-SMA also may directly distribute the computing metrics to the Ingress CATS-Forwarder.

The computing metrics advertisements are processed by the C-PS hosted by "CATS-Forwarder 1". The C-PS also processes network metric advertisements sent by the C-NMA. All metrics are used by the C-PS to select the most relevant path that leads to the Egress CATS-Forwarder according to the initial client's service request, the service that is requested ("CS-ID 1" or "CS-ID 2"), the state of the service contact instances as reported by the metrics, and the state of the network.

In the case of distributing aggregated per-site computing-related metrics, the per-instance CSCI-ID information will not be included in the advertisement. Instead, a per-site CSCI-ID may be used in case multiple sites are connected to the Egress CATS-Forwarder to explicitly indicate the site from where the aggregated metrics come.

at "Service Site 3" advertises the computing metrics for the two services hosted by "Service Site 3" to the centralized C-PS as well. Furthermore, the C-PS receives the network metrics sent from the C-NMA. All metrics are used by the C-PS to select the most relevant path that leads to the Egress CATS-Forwarder. The selected paths will be sent from the C-PS to "CATS-Forwarder 1" to indicate traffic steering.

```

Service CS-ID 1, instance CSCI-ID 1 <computing metrics>
Service CS-ID 1, instance CSCI-ID 2 <computing metrics>
Service CS-ID 1, instance CSCI-ID 3 <computing metrics>
Service CS-ID 2, <computing metrics>

```

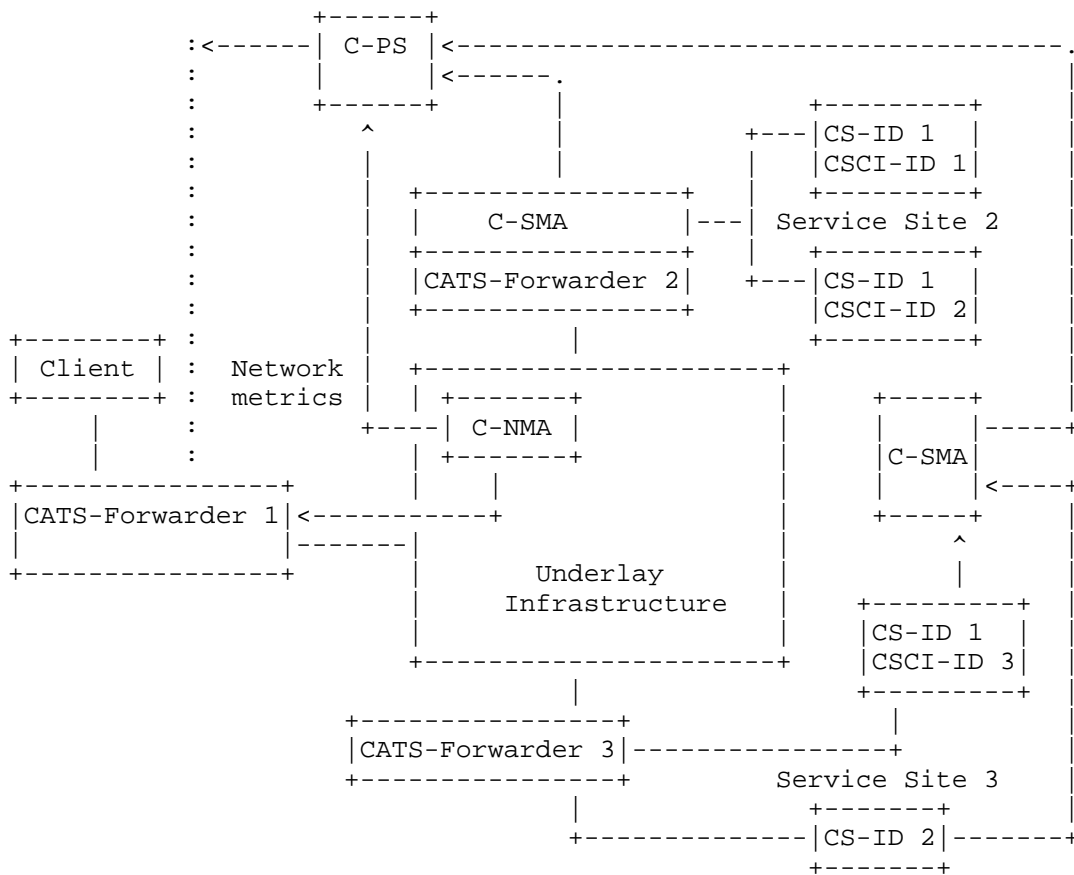


Figure 4: An Example of CATS Metric Distribution in the Centralized Model

A.3. Hybrid Model

An example of metrics distribution in the hybrid model is illustrated in Figure 5.

For example, the metrics 1, 2, and 3 associated with the "CS-ID1" are collected by the centralized C-PS, and the metrics 4 and 5 are distributed via distributed protocols to the Ingress CATS-Forwarder directly. For a service with "CS-ID2", all the metrics are collected by the centralized C-PS. The CATS-computed path result will be distributed to the Ingress CATS-Forwarders from the C-PS by considering both the metrics from the C-SMA and C-NMA. Furthermore, the Ingress CATS-Forwarder may also have some ability to compute the path for subsequent packets accessing the same service.

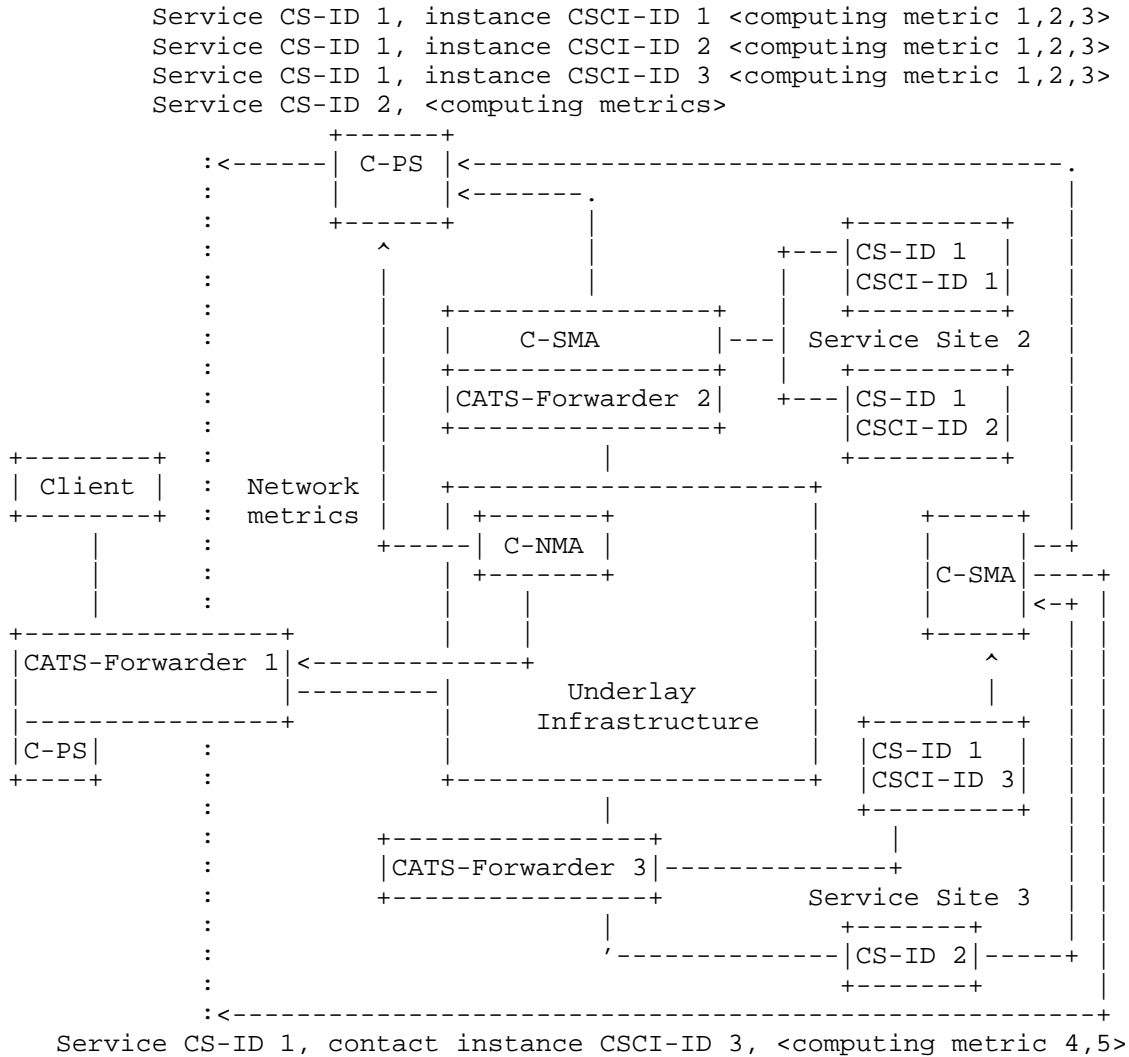


Figure 5: An Example of CATS Metric Distribution in the Hybrid Model

Appendix B. Acknowledgements

The authors would like to thank Joel Halpern, John Scudder, Dino Farinacci, Adrian Farrel, Cullen Jennings, Linda Dunbar, Jeffrey Zhang, Peng Liu, Fang Gao, Aijun Wang, Cong Li, Xinxin Yi, Jari Arkko, Mingyu Wu, Haibo Wang, Xia Chen, Jianwei Mao, Guofeng Qian, Zhenbin Li, Xinyue Zhang, Weier Li, Quan Xiong, Ines Robles, Nagendra Kumar, and Taylor Paul for their comments and suggestions.

Some text about various deployment models was originally documented in [I-D.yao-cats-awareness-architecture].

Special thanks to Adrian Farrel for the careful shepherd review and various suggestions that enhanced this specification.

Thanks to Thomas Fossati for the GENART review.

Contributors

Guangping Huang
ZTE
Email: huang.guangping@zte.com.cn

Gyan Mishra
Verizon Inc.
Email: hayabusagsm@gmail.com

Huijuan Yao
China Mobile
Email: yaohuijuan@chinamobile.com

Yizhou Li
Huawei Technologies
Email: liyizhou@huawei.com

Dirk Trossen
DaPaDOT Tech UG (haftungsbeschraenkt)
Email: dirk@dapadot-tech.eu

Luigi Iannone
Huawei Technologies
Email: luigi.iannone@huawei.com

Hang Shi
Huawei Technologies
Email: shihang9@huawei.com

Changwang Lin
New H3C Technologies
Email: linchangwang.04414@h3c.com

Xueshun Wang
CICT
Email: xswang@fiberhome.com

Xuewei Wang
Ruijie Networks
Email: wangxuewei@ruijie.com.cn

Christian Jacquenet
Orange
Email: christian.jacquenet@orange.com

Authors' Addresses

Cheng Li (editor)
Huawei Technologies
China
Email: c.l@huawei.com

Zongpeng Du
China Mobile
China
Email: duzongpeng@chinamobile.com

Mohamed Boucadair (editor)
Orange
France
Email: mohamed.boucadair@orange.com

Luis M. Contreras
Telefonica
Spain
Email: luismiguel.contrerasmurillo@telefonica.com

John E Drake
Independent
United States of America
Email: je_drake@yahoo.com