Computing-Aware Traffic Steering
Internet-Draft
Intended status: Informational
Expires: 8 January 2026

Y. Kehan
China Mobile
C. Li
Huawei Technologies
L. M. Contreras
Telefonica
J. Ros-Giralt
Qualcomm Europe, Inc.
H. Shi
Huawei Technologies

7 July 2025

CATS Metrics Definition
draft-ietf-cats-metric-definition-03

Abstract

Computing-Aware Traffic Steering (CATS) is a traffic engineering approach that optimizes the steering of traffic to a given service instance by considering the dynamic nature of computing and network resources. In order to consider the computing and network resources, a system needs to share information (metrics) that describes the state of the resources. Metrics from network domain have been in use in network systems for a long time. This document defines a set of metrics from the computing domain used for CATS.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the Computing-Aware Traffic Steering Working Group mailing list (cats@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/cats/.

Source for this draft and an issue tracker can be found at https://github.com/VMatrix1900/draft-cats-metric-definition.

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/.

Copyright Notice

Table of Contents

1. Introduction 3
2. Conventions and Definitions 3
3. Definition of Metrics 3
3.1. Design Principles - Why Three Metric Levels? 3
3.2. Level 0: Raw Metrics 4
3.3. Level 1: Normalized Metrics in Categories 5
3.4. Level 2: Fully Normalized Metric. 6
4. Representation of Metrics 7
4.1. CATS Metric Fields 7
4.2. Level 0 Metric Representation 10
4.2.1. Compute Raw Metrics 10
4.2.2. Communication Raw Metrics 11
4.2.3. Delay Raw Metrics 12
4.3. Level 1 Metric Representation 12
4.3.1. Normalized Compute Metrics 12
4.3.2. Normalized Communication Metrics 13
4.3.3. Normalized Composed Metrics 13
4.4. Level 2 Metric Representation 14
5. Comparison among Metric Levels 15
6. Implementation Guidance on Using CATS Metrics 16
7. Security Considerations 16
8. IANA Considerations 16
9. Informative References 16
Contributors 18
Authors' Addresses 18

1. Introduction

Service providers are deploying computing capabilities across the network for hosting applications such as distributed AI workloads, AR/VR and driverless vehicles, among others. In these deployments, multiple service instances are replicated across various sites to ensure sufficient capacity for maintaining the required Quality of Experience (QoE) expected by the application. To support the selection of these instances, a framework called Computing-Aware Traffic Steering (CATS) is introduced in [I-D.ietf-cats-framework].

CATS is a traffic engineering approach that optimizes the steering of traffic to a given service instance by considering the dynamic nature of computing and network resources. To achieve this, CATS components require performance metrics for both communication and compute resources. Since these resources are deployed by multiple providers, standardized metrics are essential to ensure interoperability and enable precise traffic steering decisions, thereby optimizing resource utilization and enhancing overall system performance.

Metrics from network domain have already been defined in previous documents, e.g., [RFC9439], [RFC8912], and [RFC8911], and been in use in network systems for a long time. This document focuses on categorizing the relevant metrics at the computing domain for CATS into three levels based on their complexity and granularity.

2. Conventions and Definitions

This document uses the following terms defined in [I-D.ietf-cats-framework]:

* Computing-Aware Traffic Steering (CATS)
* Service
* Service contact instance

3. Definition of Metrics

3.1. Design Principles - Why Three Metric Levels?

As outlined in [I-D.ietf-cats-usecases-requirements], the resource model that defines CATS metrics MUST be scalable, ensuring that its implementation remains within a reasonable and sustainable cost. Additionally, it MUST be useful in practice. To that end, a CATS system should select the most appropriate metric(s) for instance selection, recognizing that different metrics may influence outcomes in distinct ways depending on the specific use case.

Introducing a definition of metrics requires balancing the following trade-off: if the metrics are too fine-grained, they become unscalable due to the excessive number of metrics that must be communicated through the metrics distribution protocol. (See [I-D.rcr-opsawg-operational-compute-metrics] for a discussion of metrics distribution protocols.) Conversely, if the metrics are too coarse-grained, they may not have sufficient information to enable proper operational decisions.

Conceptually, it is necessary to define at least two fundamental levels of metrics: one comprising all raw metrics, and the other representing a simplified form— consisting of a single value that encapsulates the overall capability of a service instance.

However, such a definition may, to some extent, constrain implementation flexibility across diverse CATS use cases. Implementers often seek balanced approaches that consider trade-offs among encoding complexity, accuracy, scalability, and extensibility.

To ensure scalability while providing sufficient detail for effective decision-making, this document provides a definition of metrics that incorporates three levels of abstraction:

* *Level 0 (L0): Raw metrics.* These metrics are presented without abstraction, with each metric using its own unit and format as defined by the underlying resource.

* *Level 1 (L1): Metrics normalized within categories.* These metrics are derived by aggregating L0 metrics into multiple categories, such as network and computing. Each category is summarized with a single L1 metric by normalizing it into a value within a defined range of scores.

* *Level 2 (L2): Fully normalized metric.* These metrics are derived by aggregating lower level metrics (L0 or L1) into a single L2 metric, which is then normalized into a value within a defined range of scores.

3.2. Level 0: Raw Metrics

Level 0 metrics encompass detailed, raw metrics, including but not limited to:

* CPU: Base Frequency, boosted frequency, number of cores, core utilization, memory bandwidth, memory size, memory utilization, power consumption.

* GPU: Frequency, number of render units, memory bandwidth, memory size, memory utilization, core utilization, power consumption.

* NPU: Computing power, utilization, power consumption.

* Network: Bandwidth, capacity, throughput, bytes transmitted, bytes received, host bus utilization.

* Storage: Available space, read speed, write speed.

* Delay: Time taken to process a request.

L0 metrics serve as foundational data and do not require classification. They provide basic information to support higher-level metrics, as detailed in the following sections.

L0 metrics can be encoded and exposed using an Application Programming Interface (API), such as a RESTful API, and can be solution-specific. Different resources can have their own metrics, each conveying unique information about their status. These metrics can generally have units, such as bits per second (bps) or floating point instructions per second (flops).

Regarding network-related information, [RFC8911] and [RFC8912] define various performance metrics and their registries. Additionally, in [RFC9439], the ALTO WG introduced an extended set of metrics related to network performance, such as throughput and delay. For compute metrics, [I-D.rcr-opsawg-operational-compute-metrics] lists a set of cloud resource metrics.

3.3. Level 1: Normalized Metrics in Categories

L1 metrics are organized into distinct categories, such as computing, communication, and composed metrics. Each L0 metric is classified into one of these categories. Within each category, a single L1 metric is computed using an _aggregation function_ and normalized to a unitless score that represents the performance of the underlying resources according to that category. Potential categories include:

* *Computing:* A normalized value derived from computing-related L0 metrics, such as CPU, GPU, and NPU utilization.

* *Communication:* A normalized value derived from communication-related L0 metrics, such as communication throughput.

* *Composed:* A normalized value derived from an end-to-end aggregation function by levaraging both computing and communication metrics. For example, end-to-end delay computed as the sum of all delays along a path.

Editor note: detailed categories can be updated according to the CATS WG discussion.

L0 metrics, such as those defined in [RFC8911], [RFC8912], [RFC9439], and [I-D.rcr-opsawg-operational-compute-metrics], can be categorized into the aforementioned categories. Each category will employ its own aggregation function (e.g., weighted summary) to generate the normalized value. This approach allows the protocol to focus solely on the metric categories and their normalized values, thereby avoiding the need to process solution-specific detailed metrics.

3.4. Level 2: Fully Normalized Metric.

The L2 metric is a single score value derived from the lower level metrics (L0 or L1) using an aggregation function. Different implementations may employ different aggregation functions to characterize the overall performance of the underlying compute and communication resources. The definition of the L2 metric simplifies the complexity of collecting and distributing numerous lower-level metrics by consolidating them into a single, unified score.

TODO: Some implementations may support the configuration of Ingress CATS-Forwarders with the metric normalizing method so that it can decode the information from the L1 or L0 metrics.

Figure 1 provides a summary of the logical relationships between metrics across the three levels of abstraction.

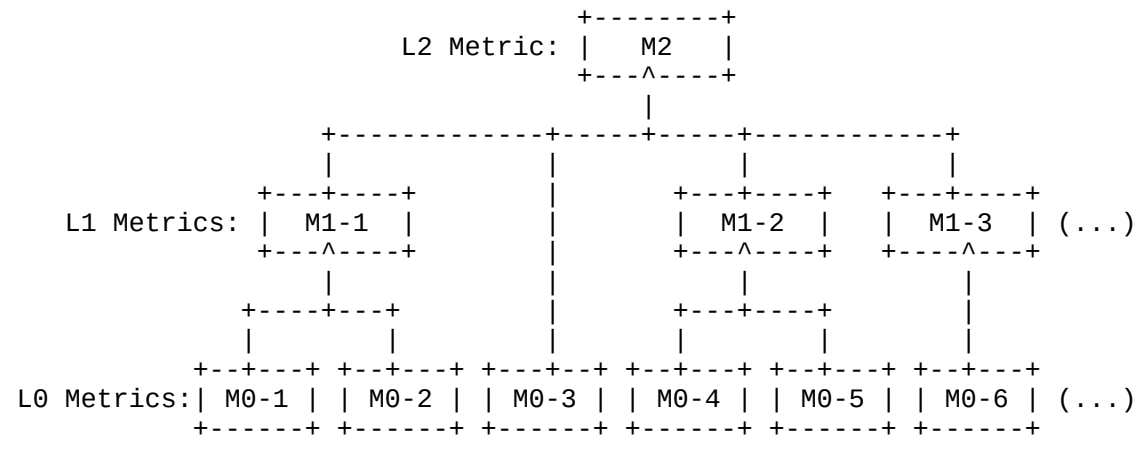

Figure 1: Logic of CATS Metrics in levels

4. Representation of Metrics

The representation of metrics is a key component of the CATS architecture. It defines how metrics are encoded and transmitted over the network. The representation should be flexible enough to accommodate various types of metrics along with their respective units and precision levels, yet simple enough to enable easy implementation and deployment across heterogeneous edge environments.

4.1. CATS Metric Fields

This section presents the detailed representation of CATS metrics. The design aligns with principles established in similar IETF specifications, such as the network performance metrics defined in [RFC9439].

A CATS metric is represented using a set of fields, each describing a property of the metric. This document introduces the following CATS metrics fields:

```
- Cats_metric:
      - Metric_type:
            The type of the CATS metric.
            Examples: compute_cpu, storage_disk_size, network_bw,
            compute_delay, network_delay, compute_norm,
            storage_norm, network_norm, delay_norm.
      - Format:
            The encoding format of the metric.
            Examples: int, float.
      - Format_std (optional):
            The standard used to encode and decode the value
            field according to the format field.
            Example: ieee_754, ascii.
      - Length:
            The size of the value field measured in octets.
            Examples: 2, 4, 8, 16, 32, 64.
      - Unit:
            The unit of this metric.
            Examples: mhz, ghz, byte, kbyte, mbyte,
            gbyte, bps, kbps, mbps, gbps, tbps, tflops, none.
      - Source (optional):
            The source of information used to obtain the value field.
            Examples: nominal, estimation, normalization,
            aggregation.
      - Statistics(optional):
            The statistical function used to obtain the value field.
            Examples: max, min, mean, cur.
      - Level:
            The level this metric belongs to.
            Examples: L0, L1, L2.
      - Value:
            The value of this metric.
            Examples: 12, 3.2.
```

Figure 2: CATS Metric Fields

Next, we describe each field in more detail:

* *Metric_Type (type)*: This field specifies the category or kind of CATS metric being measured, such as computational resources, storage capacity, or network bandwidth. It acts as a label that enables network devices to identify the purpose of the metric.

* *Format (format)*: This field indicates the data encoding format of the metric, such as whether the value is represented as an integer, a floating-point number, or has no specific format.

* *Format standard (format_std, optional)*: This optional field indicates the standard used to encode and decode the value field according to the format field. It is only required if the value field is encoded using a specific standard, and knowing this standard is necessary to decode the value field. Examples of format standards include ieee_754 and ascii. This field ensures that the value can be accurately interpreted by specifying the encoding method used.

* *Length (length)*: This field indicates the size of the value field measured in octets (bytes). It specifies how many bytes are used to store the value of the metric. Examples include 4, 8, 16, 32, and 64. The length field is important for memory allocation and data handling, ensuring that the value is stored and retrieved correctly.

* *Unit (unit)*: This field defines the measurement units for the metric, such as frequency, data size, or data transfer rate. It is usually associated with the metric to provide context for the value.

* *Source (source, optional)*: This field describes the origin of the information used to obtain the metric. It may include one or more of the following non-mutually exclusive values:

  - 'nominal'. Similar to [RFC9439], "a 'nominal' metric indicates that the metric value is statically configured by the underlying devices. For example, bandwidth can indicate the maximum transmission rate of the involved device.

  - 'estimation'. The 'estimation' source indicates that the metric value is computed through an estimation process.

  - 'directly measured'. This source indicates that the metric can be obtained directly from the underlying device and it does not need to be estimated.

  - 'normalization'. The 'normalization' source indicates that the metric value was normalized. For instance, a metric could be normalized to take a value from 0 to 1, from 0 to 10, or to take a percentage value. This type of metrics do not have units.

  - 'aggregation'. This source indicates that the metric value was obtained by using an aggregation function.

Nominal metrics have inherent physical meanings and specific units without any additional processing. Aggregated metrics may or may not have physical meanings, but they retain their significance relative to the directly measured metrics. Normalized metrics, on the other hand, might have physical meanings but lack units.

* *Statistics (statistics, optional)*: This field provides additional details about the metrics, particularly if there is any pre-computation performed on the metrics before they are collected. It is useful for services that require specific statistics for service instance selection.

  - 'max'. The maximum value of the data collected over intervals.

  - 'min'. The minimum value of the data collected over intervals.

  - 'mean'. The average value of the data collected over intervals.

  - 'cur'. The current value of the data collected.

* *Level (level)*: This field specifies the level at which the metric is measured. It is used to categorize the metric based on its granularity and scope. Examples include L0, L1, and L2. The level field helps in understanding the level of detail and specificity of the metric being measured.

* *Value (value)*: This field represents the actual numerical value of the metric being measured. It provides the specific data point for the metric in question.

4.2. Level 0 Metric Representation

Several definitions have been developed within the compute and communication industries, as well as through various standardization efforts---such as those by the [DMTF]---that can serve as L0 metrics. This section provides illustrative examples.

4.2.1. Compute Raw Metrics

This section uses CPU frequency as an example to illustrate the representation of raw compute metrics. The metric type is labeled as compute_CPU_frequency, with the unit specified in GHz. The format should support both unsigned integers and floating-point values. The corresponding metric fields are defined as follows:

```
   Basic fields:
         Metric Type: compute_CPU_frequency
         Level: L0
         Format: unsigned integer, floating point
         Unit: GHz
         Length: four octets
         Value: 2.2
   Source:
         nominal

   |Metric Type|Level|Format| Unit|Length| Value|Source|
       8bits    2bits  1bit  4bits  3bits 32bits  3bits
```

Figure 3: An Example for Compute Raw Metrics

4.2.2. Communication Raw Metrics

This section takes the total transmitted bytes (TxBytes) as an example to show the representation of communication raw metrics. TxBytes are named as "communication type_TxBytes" . The unit is Mega Bytes (MB). Format is unsigned integer or floating point. It will occupy 4 octets. The source of the metric is "Directly measured" and the statistics is "mean". Example:

```
   Basic fields:
         Metric type: “communication type_TXBytes”
         Level: L0
         Format: unsigned integer, floating point
         Unit: MB
         Length: four octets
         Value: 100
   Source:
         Directly measured
   Statistics:
         mean

   |Metric Type|Level|Format| Unit|Length| Value|Source|Statistics|
       8bits    2bits  1bit  4bits  3bits 32bits  3bits   2bits
```

Figure 4: An Example for Communication Raw Metrics

4.2.3. Delay Raw Metrics

Delay is a kind of synthesized metric which is influenced by computing, storage access, and network transmission. Usually delay refers to the overal processing duration between the arrival time of a specific service request and the departure time of the corresponding service response. It is named as "delay_raw". The format should support both unsigned integer or floating point. Its unit is microseconds, and it occupies 4 octets. For example:

```
Basic fields:
      Metric type: “delay_raw”
      Level: L0
      Format: unsigned integer, floating point
      Unit: Microsecond(us)
      Length: four octets
      Value: 231.5
Source:
      aggregation
Statistics:
      max

|Metric Type|Level|Format| Unit|Length| Value|Source|Statistics|
    8bits    2bits  1bit  4bits  3bits 32bits  3bits   2bits
```

Figure 5: An Example for Delay Raw Metrics

4.3. Level 1 Metric Representation

L1 metrics are normalized from L0 metrics. Although they don’t have units, they can still be classified into types such as compute, communication and composed metrics. This classification is useful because it makes L1 metrics semantically meaningful.

The sources of L1 metrics is normalization. Based on L0 metrics, service providers design their own algorithms to normalize metrics. For example, assigning different cost values to each raw metric and do weighted summation. L1 metrics do not need further statistical values.

4.3.1. Normalized Compute Metrics

The metric type of normalized compute metrics is “compute_norm” , and its format is unsigned integer. It has no unit. It will occupy an octet. Example:

```
   Basic fields:
         Metric type: “compute_norm”
         Level: L1
         Format: unsigned integer
         Length: one octet
         Value: 5
   Source:
         normalization


   |Metric Type|Level|Format|Length|Value|Source|
       8bits    2bits  1bit   3bits 8bits  3bits
```

Figure 6: An Example for Normalized Compute Metrics

4.3.2. Normalized Communication Metrics

The metric type of normalized communication metrics is “communication_norm” , and its format is unsigned integer. It has no unit. It will occupy an octet. Example:

```
   Basic fields:
         Metric type: “communication_norm”
         Level: L1
         Format: unsigned integer
         Length: one octet
         Value: 1
   Source:
         normalization

   |Metric Type|Level|Format|Length|Value|Source|
       8bits    2bits  1bit   3bits 8bits  3bits
```

Figure 7: An Example for Normalized Communication Metrics

4.3.3. Normalized Composed Metrics

The metric type of normalized composed metrics is “delay_norm” , and its format is unsigned integer. It has no unit. It will occupy an octet. Example:

```
   Basic fields:
         Metric type: “composed_norm”
         Level: L1
         Format: unsigned integer
         Length: an octet
         Value: 8
   Source:
         normalization

   |Metric Type|Level|Format|Length|Value|Source|
       8bits    2bits  1bit   3bits 8bits  3bits
```

Figure 8: An Example for Normalized Composed Metrics

4.4. Level 2 Metric Representation

A fully normalized metric is a single value which does not have any physical meaning or unit. Each provider may have its own methods to derive the value, but all providers must follow the definition in this section to represent the fully normalized value.

Metric type is “norm_fi” . The format of the value is unsigned integer. It has no unit. It will occupy an octet. Example:

```
   Basic fields:
         Metric type: “norm_fi”
         Level: L2
         Format: unsigned integer
         Length: an octet
         Value: 1
   Source:
         normalization

   |Metric Type|Level|Format|Length|Value|Source|
       8bits    2bits  1bit   3bits 8bits  3bits
```

Figure 9: An Example for Fully Normalized Metric

The fully normalized value also supports aggregation when there are multiple service instances providing these fully normalized values. When providing fully normalized values, service instances do not need to do further statistics.

5. Comparison among Metric Levels

Metrics are progressively consolidated from L0 to L1 to L2, with each level offering a different degree of abstraction to address the diverse requirements of various services. Table 1 provides a comparative overview of these metric levels.

Level	Encoding Complexity	Extensibility	Stability	Accuracy
Level 0	Complicated	Bad	Bad	Good
Level 1	Medium	Medium	Medium	Medium
Level 2	Simple	Good	Good	Medium

Table 1: Comparison among Metrics Levels

Since Level 0 metrics are raw and service-specific, different services may define their own sets—potentially resulting in hundreds or even thousands of unique metrics. This diversity introduces significant complexity in protocol encoding and standardization. Consequently, L0 metrics are generally confined to bespoke implementations tailored to specific service needs, rather than being standardized for broad protocol use. In contrast, Level 1 metrics organize raw data into standardized categories, each normalized into a single value. This structure makes them more suitable for protocol encoding and standardization. Level 2 metrics take simplification a step further by consolidating all relevant information into a single normalized value, making them the easiest to encode, transmit, and standardize.

Therefore, from the perspective of encoding complexity, Level 1 and Level 2 metrics are recommended.

When considering extensibility, Level 0 metrics allow new services to define their own custom metrics. However, this flexibility requires corresponding protocol extensions, and the proliferation of metric types can introduce significant overhead, ultimately reducing the protocol's extensibility. In contrast, Level 1 metrics introduce only a limited set of standardized categories, making protocol extensions more manageable. Level 2 metrics go even further by consolidating all information into a single normalized value, placing the least burden on the protocol.

Therefore, from an extensibility standpoint, Level 1 and Level 2 metrics are recommended.

Regarding stability, Level 0 raw metrics may require frequent protocol extensions as new metrics are introduced, leading to an unstable and evolving protocol format. For this reason, standardizing L0 metrics within the protocol is not recommended. In contrast, Level 1 metrics involve only a limited set of predefined categories, and Level 2 metrics rely on a single consolidated value, both of which contribute to a more stable and maintainable protocol design.

Therefore, from a stability standpoint, Level 1 and Level 2 metrics are preferred.

In conclusion, for CATS, Level 2 metrics are recommended due to their simplicity and minimal protocol overhead. If more advanced scheduling capabilities are required, Level 1 metrics offer a balanced approach with manageable complexity. While Level 0 metrics are the most detailed and dynamic, their high overhead makes them unsuitable for direct transmission to network devices and thus not recommended for standard protocol integration.

6. Implementation Guidance on Using CATS Metrics

<Authors Note: This part has been moved to [I-D.ietf-cats-framework], according to he chairs' sugguestion. Since this document is primarily on metric definition, rather than real implementations.>

7. Security Considerations

TBD

8. IANA Considerations

TBD

9. Informative References

[DMTF] "DMTF", n.d., <https://www.dmtf.org/>.

[I-D.ietf-cats-framework]
Li, C., Du, Z., Boucadair, M., Contreras, L. M., and J. Drake, "A Framework for Computing-Aware Traffic Steering (CATS)", Work in Progress, Internet-Draft, draft-ietf-cats-framework-10, 24 June 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-cats-framework-10>.

[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-07, 10 June 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-cats-usecases-requirements-07>.

[I-D.rcr-opsawg-operational-compute-metrics]
Randriamasy, S., Contreras, L. M., Ros-Giralt, J., and R. Schott, "Joint Exposure of Network and Compute Information for Infrastructure-Aware Service Deployment", Work in Progress, Internet-Draft, draft-rcr-opsawg-operational-compute-metrics-08, 21 October 2024, <https://datatracker.ietf.org/doc/html/draft-rcr-opsawg-operational-compute-metrics-08>.

[performance-metrics]
"performance-metrics", n.d., <https://www.iana.org/assignments/performance-metrics/performance-metrics.xhtml>.

[RFC8911] Bagnulo, M., Claise, B., Eardley, P., Morton, A., and A. Akhter, "Registry for Performance Metrics", RFC 8911, DOI 10.17487/RFC8911, November 2021, <https://www.rfc-editor.org/rfc/rfc8911>.

[RFC8912] Morton, A., Bagnulo, M., Eardley, P., and K. D'Souza, "Initial Performance Metrics Registry Entries", RFC 8912, DOI 10.17487/RFC8912, November 2021, <https://www.rfc-editor.org/rfc/rfc8912>.

[RFC9439] Wu, Q., Yang, Y., Lee, Y., Dhody, D., Randriamasy, S., and L. Contreras, "Application-Layer Traffic Optimization (ALTO) Performance Cost Metrics", RFC 9439, DOI 10.17487/RFC9439, August 2023, <https://www.rfc-editor.org/rfc/rfc9439>.

Contributors

Mohamed Boucadair
Orange
Email: mohamed.boucadair@orange.com

Zongpeng Du
China Mobile
Email: duzongpeng@chinamobile.com

Authors' Addresses

Kehan Yao
China Mobile
China
Email: yaokehan@chinamobile.com

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

L. M. Contreras
Telefonica
Email: luismiguel.contrerasmurillo@telefonica.com

Jordi Ros-Giralt
Qualcomm Europe, Inc.
Email: jros@qti.qualcomm.com

Hang Shi
Huawei Technologies
China
Email: shihang9@huawei.com