Evolving Network Management Architecture: Integrating CORECONF with NETCONF for Efficient Telemetry and Management

Manoj Gudi, Alexander Pelov, Laurent Toutain, Jean-Marie Bonnin
Department of Network Systems, Cyber Security and Digital Law (SRCD), IMT Atlantique, IRISA,
35576 Rennes, France
Email: manoj.gudi@imt-atlantique.net, {alexander.pelov, laurent.toutain, jm.bonnin}@imt-atlantique.fr

***Abstract*—This paper proposes an evolved network management architecture that integrates CORECONF alongside existing NETCONF deployments to optimize telemetry operations, particularly in constrained and resource-limited environments. Adoption of CORECONF can be done progressively in stages by using plugins or middlewares as shown in these architectures. By leveraging the efficiency of CORECONF with Concise Binary Object Representation (CBOR) encoding and Schema Item Identifiers (SIDs), the architecture enhances performance, scalability, and sustainability in network operations. This integration is especially beneficial for applications such as energy-efficient networking, IoT deployments, high-density telemetry scenarios, and unique use cases involving secondary connectivity technologies and asymmetrical connectivity.**

***Index Terms*—NETCONF, CORECONF, Network-Management, Constrained-Devices, IoT-Networks**

I. Introduction

The proliferation of internet services in the early 2000s gave rise to complex networks of routers, switches, and firewalls to manage home and enterprise networks. Network Configuration Protocol, also known as NETCONF, is a widely adopted network management protocol for these devices first standardized in RFC 4741, then updated by RFC 6241 by the Internet Engineering Task Force (IETF) [1]. However, as the networks have expanded to encompass several constrained devices from the Internet of Things (IoT) domain, NETCONF protocol has failed to adapt to these lightweight devices. The overhead associated with NETCONF, particularly for telemetry data collection, can severely drain the resources such as CPU and network-bandwidth of these constrained devices making it unsuitable[2].

RESTCONF protocol was introduced as an alternative way to perform network configuration based on RESTful principles first standardized in RFC 8040 [3]. Unlike NETCONF, RESTCONF is web centric and utilizes Hypertext Transfer Protocol (HTTP) for the network management with solid support for transaction based interactions (such as commit, lock, and rollback). This paper will use the term “NETCONF” in a broad sense to mean a NETCONF and/or a RESTCONF system. Jethanandani *et al.* The primary focus of this paper will be NETCONF systems, which predates RESTCONF in YANG-based network management [4].

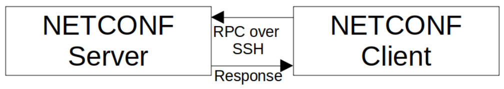

Fig. 1: Basic NETCONF Model

CoAP Management Interface (CORECONF) is a promising protocol for network management of such constrained devices being standardized by the IETF Constrained RESTful Environments (CoRE) Working Group [5]. Moreover, incorporating CORECONF together with NETCONF systems can allow administrators to benefit from collecting telemetry data with very low overhead yet maintaining compatibility with existing NETCONF systems.

II. Motivation and background

A. NETCONF Deployments

NETCONF utilizes a server-client architecture where the client sends commands to the server to configure the network, retrieve the data, and execute other network-management operations. Typically, these commands are implemented as remote procedure calls (RPCs) on the NETCONF servers, encoded in extensible markup language (XML) and communicated over secure socket shell (SSH) [1] as shown in figure 1. Additionally, Yet Another Next Generation (YANG) data modelling language is used to standardize NETCONF messages across all the devices built by various vendors [6].

Although NETCONF systems seem straightforward to build and deploy, they are not well suited for constrained devices as shown in A. Sehgal *et al.* [7] due to their relative large system requirements. There have been variants of NETCONF protocols, such as NETCONF light proposed by Schönwälder *et al.* [8] specifically for constrained devices. However, Mavromatis *et al.* [9] demonstrated that even NETCONF light requires more processing power, device energy and provisioning time than lightweight networks can afford.

B. Overview of CORECONF

CORECONF is a RESTful application protocol used by constrained devices to exchange data over CoAP/UDP as the combination is known to be relatively less complex with low overhead [2, 5]. The protocol allows us to encode CoAP payload into a terse representation by assigning integers to YANG identifiers of the data models, also known YANG Schema Item iDentifiers (YANG SIDs) [10]. Furthermore, this data is serialized into Concise Binary Object Representation (CBOR) form, further reducing the message footprint [11]. Thus, unlike NETCONF, CORECONF allows constrained devices to send messages even on low bandwidth networks [2]. Furthermore, as shown by Toutain *et al.* in their ietf-draft, YANG models can assist with rapid prototyping of CORECONF systems by automated code generation thereby reducing development time and effort [12].

Contrary to NETCONF, which supports secure communication over SSH or Transport Layer Security (TLS), CORECONF data is communicated using CoAP over Datagram Transport Layer Security (DTLS) or Object Security for Constrained RESTful Environments (OSCORE). These encryption schemes are designed for constrained environment and hence would make CORECONF ideal choice for telemetry data.

C. Proposed Architecture

This section discusses various possible architectures to integrate NETCONF with CORECONF and potential opportunities and drawbacks with them.

1) Basic NETCONF deployments: Existing NETCONF based systems can be modelled as a two-actor communication system described in figure 2 where actor #1 is a server (a router or switch) and actor #2 is a client (controller) which intends to perform network management and collect telemetry data from the server.

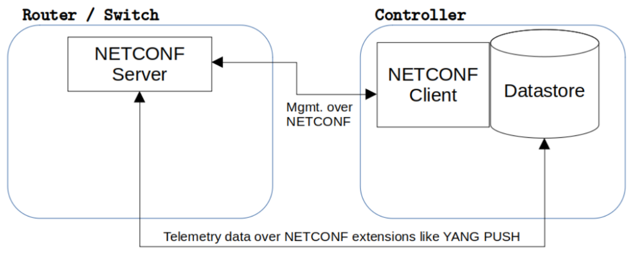

Fig. 2: Scenario 1: Basic NETCONF deployments

2) CORECONF for telemetry only: In this scenario shown in figure 3, the controller acts as a node only collecting telemetric data and archiving it in a datastore, which can be used for further analysis.

The router/switch is pre-installed with NETCONF server ready to send telemetry data to any NETCONF client which requests it. Thus, we add a middleware which has two main components. First component is an emulated NETCONF client running on the same device. It makes appropriate calls to NETCONF server to receive telemetry data. This data is later encoded into CORECONF-YANG/CBOR format and sent over the network to the controller.

This is particularly advantageous for two reasons:

- Leveraging strength of each protocol, namely — CORECONF protocol to send telemetry data without large overhead and NETCONF's ability to architect and manage complex networks.
- Existing clients will continue to be operable with the servers for management functions, the CORECONF middleware can be an implemented as a different application.

Controller accepts this telemetry data in CORECONF-YANG/CBOR form, decodes it and accumulates in a datastore.

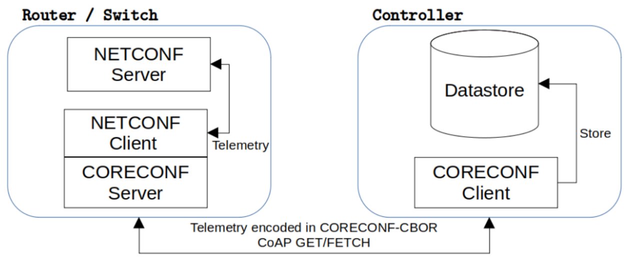

Fig. 3: Scenario 2: CORECONF for telemetry only

3) CORECONF-centric Integration: In this particular scenario as shown in figure 4, both the tasks: network management and telemetry reporting happens over the CORECONF middleware.

With this approach, all the requests are streamlined through CORECONF middleware, and due to efficient encoding of CORECONF, we should see reduced payload sizes between switch and the controller.

On the flip side, the CORECONF Middleware becomes single point of failure with no possibility to perform even network management in case of a failure.

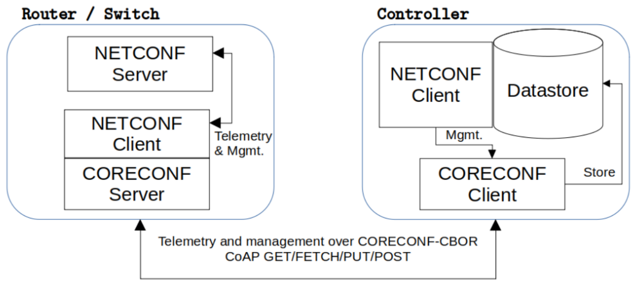

Fig. 4: Scenario 3: CORECONF-centric Integration

4) Fully Integrated Server: In a fully integrated architecture, the network management can be performed over either NETCONF or CORECONF protocols. However, if the client is a legacy NETCONF client, it must continue to use CORECONF middleware to help exchange data. This is illustrated in figure 5. The obvious advantage of this architecture is that the NETCONF-CORECONF translation is handled by server solely and directly, eliminating the need for middlewares. However, it may increase the development time and effort for such a tight integration on the server side.

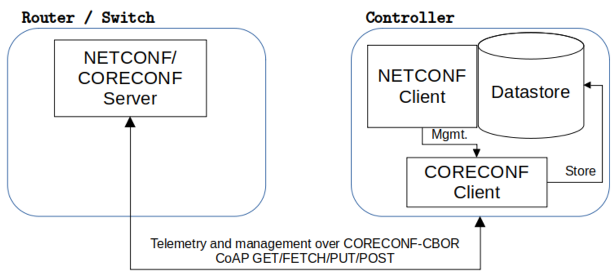

Fig. 5: Scenario 4: Fully Integrated Server

III. Applications / Use cases

Applications in several areas can benefit from low overhead and efficient message encoding provided by CORECONF protocol such as IoT for smart cities and industrial networks, environmental monitoring, vehicle-to-infrastructure (V2I) amongst others. Following section has a deeper look on some particularly interesting use-cases.

A. Energy-Efficient Networking as specified by the GREEN WG

One of the mandates of the Getting Ready for Energy-Efficient Networking Working Group (or GREEN WG), is to identify opportunities to optimize energy consumption of networking devices by collecting telemetry data [13]. These models to collect and generate energy-related statistics must be developed in YANG instead of Management Information Base (MIBs) [13]. Heterogenous networks using NETCONF for management can be easily configured to use CORECONF and profit from its low overhead by using one of the proposed architectures in section II-C.

For instance, cloud data centres can deploy CORECONF protocol internally to collect real-time data on power consumption or to configure energy-saving modes on the devices allowing their infrastructure to adapt to capacity utilization without significant overhead associated with network management.

B. Unique Use Cases Involving Secondary Connectivity and Asymmetrical Networks

1) Secondary Connectivity for Device Wake-Up: For IoT devices, the majority of energy consumption results from radio communication [14]. To conserve their energy, these devices are equipped to use low-power/sleep modes and transmit data over Low Power Wide Area Networks (LPWAN). LPWAN technologies like LoRaWAN and NB-IoT enable such devices to receive wake-up signals and switch to a faster primary networking interfaces such as ethernet or wi-fi to perform more complex configuration tasks.

Communicating on LoRaWAN is severely constrained with typical payload sizes of 51 bytes (for SF 12 on EU868 band) [15]. CORECONF protocol deployed on top of header compression frameworks like Static Context Header Compression and Fragmentation (SCHC), would be suitable for such constrained environment [16].

2) Highly Asymmetrical Connectivity: Systems such satellite are characterized by highly asymmetrical connectivity where the downlink bandwidth is much larger than its uplink to optimize for its disproportionately higher use of downloads than uploads. In such situations, devices may use popular specifications like Sensor Measurement Lists (SenML) to intermittently upload their telemetry data [17]. CORECONF protocol can be extended to such systems by creating YANG models for such specifications to further reduce message sizes and improve interoperability [18].

3) Telemetry via Broadcast over Bluetooth Beacons: Similar to constrained devices on LPWAN, CORECONF can be used to broadcast telemetry messages over Bluetooth Low Energy (BLE) beacon messages, which can be further compressed using SCHC [19]. Some examples can be:

- **Smart Environments**: Sensors broadcast environmental data to any listening devices without establishing connections.
- **Tiny embedded devices**: Suitable for tiny embedded devices that cannot support full protocol stacks required by NETCONF or RESTCONF.

IV. Comparative Analysis of Protocol Efficiency

1) Payload Size Reduction with SIDs: Using SIDs in CORECONF replaces string identifiers with compact integers, significantly reducing data sizes.

Using the methodology described in Toutain *et al.* [12], the following JSON message (139 bytes) containing information about temperature, power consumption and energy saving mode can be reduced into CORECONF-YANG/CBOR form (49 bytes) as follows:

```
{
  "energy-saving": {
    "device": [
      {
        "device-id": "router-01",
        "metrics": {
          "power-consumption": 125.50,
          "energy-saving-mode": "active",
          "temperature": 45.3
        }
}]}}
```

The corresponding SID file for the data model can be generated using pyang tool as follows:

```
{
  "assignment-range": [
    {
      "entry-point": 60000,
      "size": 100
    }
  ],
  "module-name": "energy-saving",
  "module-revision": "unknown",
  "item": [
    {
      "namespace": "module",
      "identifier": "energy-saving",
      "sid": 60016
      },
      {
        "namespace": "data",
        "identifier": "/energy-saving:energy-
            saving",
        "sid": 60006
      },
      {
        "namespace": "data",
        "identifier": "/energy-saving:energy-
            saving/device",
        "sid": 60004
      },
      {
        "namespace": "data",
        "identifier": "/energy-saving:energy-
            saving/device/device-id",
        "sid": 60000
      },
      {
        "namespace": "data",
        "identifier": "/energy-saving:energy-
            saving/device/metrics",
        "sid": 60003
      },
      {
        "namespace": "data",
        "identifier": "/energy-saving:energy-
            saving/device/metrics/power-
            consumption",
        "sid": 60002
      },
      {
        "namespace": "data",
        "identifier": "/energy-saving:energy-
            saving/device/metrics/energy-saving-
            mode",
        "sid": 60001
      },
      {
        "namespace": "data",
        "identifier": "/energy-saving:energy-
            saving/device/metrics/temperature",
        "sid": 60005
      }
    ]
}
```

Using a tool like pycoreconf, the JSON message is then transformed to its CORECONF-YANG/CBOR diagnostic form as follows:

```
{60006: {-2: [{-4: "router-01",
            -1: {-2: "active",
                -1: 125.5,
                2: 45.3}}]}}
```

Further, this can be serialized into CBOR and represented in hexadecimal as follows:

```
a119ea66a12181a22369726f757465722d303120
a320fb405f600000000000216661637469766502
fb4046a66666666666
```

Thus, JSON message is reduced from 139 bytes to 49 bytes without losing an information, achieving a compression of ≈ 65%.

V. Traffic Calculations Over a Day

To illustrate the impact of using CORECONF with SIDs on network traffic, we present calculations comparing NETCONF and CORECONF for telemetry data transmission over a day.

Assumptions

Protocol	**Payload Size**	**Reduction Compared to NETCONF**
NETCONF/XML	3̃50 bytes	N/A
CORECONF/CBOR without SIDs	1̃20 bytes	6̃5.7% reduction
CORECONF/CBOR with SIDs	3̃8 bytes	8̃9.1% reduction

TABLE I: Comparison of payload size for a sample XML message

- **Number of Devices**: 10,000
- **Telemetry Interval**: Every 5 minutes
- **Messages per day per Device**:

$$\frac{24 \text{ hours} \times 60 \text{ minutes}}{5 \text{ minutes}} = 288$$

- **Total Messages per day**:

$$288 \times 10,000 = 2,880,000$$

A. Data Transmission per day

Protocol	**Payload Size per Message**	**Total Data per day (MB)**	**Data Reduction vs. NETCONF**
NETCONF/XML	350 bytes	960 MB	N/A
CORECONF/CBOR without SIDs	120 bytes	329 MB	6̃5.7% reduction
CORECONF/CBOR with SIDs	38 bytes	104 MB	8̃9.1% reduction

TABLE II: Data Transmission Comparison

Calculations

- **NETCONF/XML**:

$$2,880,000 \times 350 \text{ bytes} = 1,008,000,000 \text{ bytes} \approx 960 \text{ MB}$$

- **CORECONF/CBOR without SIDs**:

$$2,880,000 \times 120 \text{ bytes} = 345,600,000 \text{ bytes} \approx 329 \text{ MB}$$

- **CORECONF/CBOR with SIDs**:

$$2,880,000 \times 38 \text{ bytes} = 109,440,000 \text{ bytes} \approx 104 \text{ MB}$$

Data Savings

- **NETCONF vs. CORECONF with SIDs**:

$$960 \text{ MB} - 104 \text{ MB} = 856 \text{ MB} \ (\approx 89\% \text{ reduction})$$

- **CORECONF without SIDs vs. CORECONF with SIDs**:

$$329 \text{ MB} - 104 \text{ MB} = 225 \text{ MB} \ (\approx 68\% \text{ reduction})$$

Thus, with reduced payload transmission without loss of information, the system should save on network bandwidth and the devices should save on energy, improving scalability of the network.

VI. Future Development

CORECONF protocol lacks a rich software ecosystem. For instance, there are software libraries like *pycoreconf* [20] and *ccoreconf* [21] to generate and manipulate CORECONF data embedded systems, but they lack tighter integration with embedded operating systems, header-compression libraries and cryptographic libraries amongst others. Certainly this can be developed with more interest from the industry.

Additionally, CORECONF can leverage group communication feature from CoAP as described in RFC 7390 [22] to deliver messages in multicast mode in lieu of several point-to-point messages. For example, instead of individually updating power mode of each device, CORECONF can send a single multicast message to simultaneously update a group of interested devices. Consequently, the network traffic should significantly reduce while allowing devices to confirm readiness individually over their primary interfaces. Such features will find ready applications in areas of Deep Space communications and Delay Tolerant Networks (DTNs).

VII. Conclusion

For YANG modelled data, CORECONF-YANG/CBOR clearly provides very efficient encoding suitable for energy conscious and constrained devices. Moreover, the proposed architecture provides ways to integrate the CORECONF protocol with existing NETCONF systems. Going forward, CORECONF aims to be the de facto standard for network management for wide range of networks and devices.

With real-life implementations, the proposed architectures provide a progressive path to adoption of CORECONF, allowing to reap its benefits while being interoperable with existing NETCONF environment.

References

[1] Rob Enns, Martin Björklund, Andy Bierman, et al. *Network Configuration Protocol (NETCONF)*. RFC 6241. June 2011. DOI: 10.17487/RFC6241. URL: https://www.rfc-editor.org/info/rfc6241.

[2] Mahesh Ganesh Bhat, Sushmit Bhattacharjee, Cenk Gündoğan, et al. “CORECONF, NETCONF, and RESTCONF: Benchmarking Network Orchestration in Constrained IIoT Devices”. In: *IEEE Internet of Things Journal* 11.7 (2024), pp. 13082–13090. DOI: 10.1109/JIOT.2023.3338470.

[3] Andy Bierman, Martin Björklund, and Kent Watsen. *RESTCONF Protocol*. RFC 8040. Jan. 2017. DOI: 10.17487/RFC8040. URL: https://www.rfc-editor.org/info/rfc8040.

[4] Jürgen Schönwälder, Martin Björklund, and Phil Shafer. “Network configuration management using NETCONF and YANG”. In: *IEEE Communications Magazine* 48.9 (2010), pp. 166–173. DOI: 10.1109/MCOM.2010.5560601.

[5] Michel Veillette, Peter Van der Stok, Alexander Pelov, et al. *CoAP Management Interface (CORECONF)*. Internet-Draft draft-ietf-core-comi-19. Work in Progress. Internet Engineering Task Force, Nov. 2024. 48 pp. URL: https://datatracker.ietf.org/doc/draft-ietf-core-comi/19/.

[6] Martin Björklund. *The YANG 1.1 Data Modeling Language*. RFC 7950. Aug. 2016. DOI: 10.17487/RFC7950. URL: https://www.rfc-editor.org/info/rfc7950.

[7] Anuj Sehgal, Vladislav Perelman, Siarhei Kuryla, et al. “Management of resource constrained devices in the internet of things”. In: *IEEE Communications Magazine* 50.12 (2012), pp. 144–149. DOI: 10.1109/MCOM.2012.6384464.

[8] Jürgen Schönwälder, Kent Watsen, Mehmet Ersue, et al. “Network configuration protocol light (netconf light)”. In: *Working Draft, IETF Secretariat, Internet-Draft draft-schoenw-netconf-light-01* (2012).

[9] Alex Mavromatis, Carlos Colman-Meixner, Aloizio P. Silva, et al. “A Software-Defined IoT Device Management Framework for Edge and Cloud Computing”. In: *IEEE Internet of Things Journal* 7.3 (2020), pp. 1718–1735. DOI: 10.1109/JIOT.2019.2949629.

[10] Michel Veillette, Alexander Pelov, Ivaylo Petrov, et al. *YANG Schema Item iDentifier (YANG SID)*. RFC 9595. July 2024. DOI: 10.17487/RFC9595. URL: https://www.rfc-editor.org/info/rfc9595.

[11] Carsten Bormann and Paul E. Hoffman. *Concise Binary Object Representation (CBOR)*. RFC 8949. Dec. 2020. DOI: 10.17487/RFC8949. URL: https://www.rfc-editor.org/info/rfc8949.

[12] Laurent Toutain, Manoj G, and Javier Alejandro FERNANDEZ. *SID Extension to efficiently manipulate YANG Data Models*. Internet-Draft draft-toutain-t2t-sid-extension-00. Work in Progress. Internet Engineering

Task Force, Mar. 2024. 21 pp. URL: https://datatracker.ietf.org/doc/draft-toutain-t2t-sid-extension/00/.

[13] Internet Engineering Task Force (IETF). *Green Charter*. https://datatracker.ietf.org/doc/charter-ietf-green/. Accessed: 2024-11-15.

[14] Onel L. A. López, Osmel M. Rosabal, David E. Ruiz-Guirola, et al. "Energy-Sustainable IoT Connectivity: Vision, Technological Enablers, Challenges, and Future Directions". In: *IEEE Open Journal of the Communications Society* 4 (2023), pp. 2609–2666. DOI: 10.1109/OJCOMS.2023.3323832.

[15] Albert Pötsch and Florian Hammer. "Towards End-to-End Latency of LoRaWAN: Experimental Analysis and IIoT Applicability". In: *2019 15th IEEE International Workshop on Factory Communication Systems (WFCS)*. 2019, pp. 1–4. DOI: 10.1109/WFCS.2019.8758033.

[16] Ana Minaburo, Laurent Toutain, Carles Gomez, et al. *SCHC: Generic Framework for Static Context Header Compression and Fragmentation*. RFC 8724. Apr. 2020. DOI: 10.17487/RFC8724. URL: https://www.rfc-editor.org/info/rfc8724.

[17] Cullen Fluffy Jennings, Zach Shelby, Jari Arkko, et al. *Sensor Measurement Lists (SenML)*. RFC 8428. Aug. 2018. DOI: 10.17487/RFC8428. URL: https://www.rfc-editor.org/info/rfc8428.

[18] Manoj G, Laurent Toutain, Alex Fernandez, et al. *SenML is CORECONF (almost)*. Internet-Draft draft-gudi-t2trg-senml-as-coreconf-00. Work in Progress. Internet Engineering Task Force, Oct. 2024. 15 pp. URL: https://datatracker.ietf.org/doc/draft-gudi-t2trg-senml-as-coreconf/00/.

[19] Laurent Toutain, Ana Carolina Minaburo, Dominique Barthel, et al. "Static Context Header Compression (SCHC) and fragmentation for LPWAN, application to UDP/IPv6". In: 2019. URL: https://api.semanticscholar.org/CorpusID:199018876.

[20] Alex Fernandez. *pycoreconf*. Accessed: 2024-11-16. 2024. URL: https://github.com/alex-fddz/pycoreconf.

[21] Manoj Gudi. *CCORECONF*. Accessed: 2024-11-16. 2024. URL: https://github.com/manojgudi/ccoreconf/.

[22] Akbar Rahman and Esko Dijk. *Group Communication for the Constrained Application Protocol (CoAP)*. RFC 7390. Oct. 2014. DOI: 10.17487/RFC7390. URL: https://www.rfc-editor.org/info/rfc7390.

[23] Mahesh Jethanandani. "YANG, NETCONF, RESTCONF: What is this all about and how is it used for multi-layer networks". In: *2017 Optical Fiber Communications Conference and Exhibition (OFC)*. 2017, pp. 1–65.