# NDNconf: Network Management Framework for Named Data Networking

Alex Afanasyev
*Florida International University*
aa@cs.fiu.edu

Sanjeev Kaushik Ramani
*Florida International University*
skaus004@fiu.edu

***Abstract*—The rapid growth of the Internet is, in part, powered by the broad participation of numerous vendors building network components. All these network devices require that they be properly configured and maintained, which creates a challenge for system administrators of complex networks with a growing variety of heterogeneous devices. This challenge is true for today's networks, as well as for the networking architectures of the future, such as Named Data Networking (NDN).**

**This paper gives a preliminary design of an NDNconf framework, an adaptation of a recently developed NETCONF protocol, to realize unified configuration and management for NDN. The presented design is built leveraging the benefits provided by NDN, including the structured naming shared among network and application layers, stateful data retrieval with name-based interest forwarding, in-network caching, data-centric security model, and others. Specifically, the configuration data models, the heart of NDNconf, the elements of the models and models themselves are represented as secured NDN data, allowing fetching models, fetching configuration data that correspond to elements of the model, and issuing commands using the standard Interest-Data exchanges. On top of that, the security of models, data, and commands are realized through native data-centric NDN mechanisms, providing highly secure systems with high granularity of control.**

## I. Introduction

The rapid growth of the Internet is, in part, powered by the broad participation of numerous vendors building network components to provide a large variety of services, from simple layer-2 and layer-3 switching to complex data center processing boxes. To address the needs for a unified network management, in 2013 IETF chartered Network Configuration workgroup [1] to develop a modern standard that addresses all SNMP issues [2] and introduces new features needed in complex network deployments. While the NETCONF provides a substantial advantage to unification of the network management plane, they inherit the "host-centric" nature of today's networking and cannot be directly applied in data-centric environments. In particular, (1) all configuration (RPC, state check, and others) is based on point-to-point model of channel communication; (2) data models while being retrieved, are specific to the conversation with the specific device; (3) security is offloaded completely to the lower-layer mechanisms and is essentially point-to-point (i.e., relying on SSH or TLS). To really view management as a task to be done at network-as-a-whole granularity, we need to apply data-centric model of communication, transport, and security that goes beyond point-to-point thinking.

Named Data Networking (NDN) [3], [4], [5], [6] is a proposed network architecture, developed as part of the NSF Future Internet Architecture program, featuring data-centric networking as a fundamental part of the network layer. Data consumers request named items of content, and the network responds with the requested data using the nearest available copy. Every data packet bears a cryptographic signature from the content creator, and highly expressive signature validation mechanisms allow security policies to be tailored to the needs of diverse applications. NDN provides critically important features for in-network caching, data authenticity, multicast data delivery, and multipath forwarding. To support these features, NDN forwarders require proper management of their internal state (e.g., cache capacity, cache replacement policies, forwarding strategies per namespace, various parameters of network interfaces). Moreover, not all features are required by every single NDN forwarder, with simple forwarders running on highly constrained Internet-of-Things platforms (e.g., using NDN-RIOT [7]) supporting a bare minimum set of capabilities. This effectively motivates the need for a flexible native NDN-based network management framework and corresponding tools.

In this paper, we explore the design of a protocol called NDNconf, largely inspired by NETCONF, that provides flexible management capabilities for NDN at the network-as-a-whole granularity. One of the main goals in this exploration is to create a protocol with "a new way" of data-centric thinking, which would ensure leveraging of the benefits provided by the NDN architecture. In other words, we explore the use of the structured naming shared among network and application layers, stateful data retrieval with name-based interest forwarding, in-network caching, and data-centric security model, and others.

Our contribution in this work is three-fold. First, we analyzed the existing NETCONF protocol and evaluated its features that can be leveraged in the data-centric networking environments (Section II). Second, we identified necessary configuration elements of NDN node and discovered an alignment between data-centric nature (the models) of the NETCONF protocol and NDN architecture (Section III). Third, we designed an initial version of NDNconf protocol and analyzed its security features (Section IV).

## II. Background

### A. Introduction to NETCONF

NETCONF is a recently designed network management protocol [1] provides a flexible mechanism to install, manipulate, and delete the configuration of network devices. It uses an XML-based data encoding for the configuration data as well as the protocol messages. NETCONF uses Yang modeling language to provide a fully customizable remote procedure call (RPC) mechanism for communication between servers (agents running on network devices) and clients (management agents). Conceptually, NETCONF is divided into four layers: secure transport, remote procedure call (RPC), operations, and content layers.

The transport layer provides low-level reliable and secure channel between clients and server. The base specification of NETCONF requires that devices at least support Secure Shell (SSH) transport protocol, allowing the use of other transports, such as Block Extensible Exchange Protocol (BEEP), TLS, as well as SOAP for management of IoT devices. The RPC layer provides a transport-independent mechanism to encode RPC messages, notifications, and their input and output parameters. This layer provides a building block to execute any command defined by the devices' data model. The operation layer defines several standard and a number of model-customized operations, including `get`, `get-config`, `edit-config`, `lock`, `unlock`, `copy-config` etc., which can be used to do configurable management and full or partial information retrieval. The content layer defines the semantics of operational data, configuration data, notifications, and operations using a data modeling language called Yang. The content of NETCONF operations can be represented using XML and JavaScript Object Notation (JSON).

One of the important characteristics of the NETCONF framework is that it embeds configuration data models inside the protocol. Specifically, whenever the management client connects to the NETCONF agent running on a device to be managed, the device "advertises" which configuration models it supports: *names* of the supported models returned as part of initial session setup. If the management client does not know the supported model or knows a different version, it can directly request the model by its name, effectively gaining the exact knowledge which parameters of the network device exist, what are the valid input parameters, as well as all RPCs that the device supports.

NETCONF separates all information on the network devices into two distinct classes: the state and configuration data. The state data represents the actual state of the device and can include the active device settings and various statistical data (e.g., interface byte counts). One or more sets of configuration data define sets of settings that can define the operational state of the device when activated. In other words, network administrators, can provision one or more configuration state and activate a specific configuration when needed, potentially using a transactional approach where devices roll back to previous working configuration if administrator-defined conditions are not met. This transactional feature is critically important in complex network deployments to ensure network operations even when configuration data includes accidental errors.

### B. Yang Data Modeling Language

Yang (Yet Another Next Generation) is a language for modeling network management data, with syntactic structure largely borrowed from SMIng [8]. Yang is an extensible language and extensions can be defined based on the needs by standard organizations, device vendors, developers, network administrators. Yang defines data models of network devices, available state and configuration elements, their data types and allowed values, possible RPC and their parameters, available notification streams, etc. Given a Yang model for a particular device, the client NETCONF application can easily understand what and how can be configured [9].

Yang model represents the data in a hierarchical structure in which each node has a particular name and either a value or a set of child nodes (see Figure 1). Each data model is named, versioned, and is structured in the form of modules and submodules. Each (sub-)module contains the collections of related definitions containing three types of statements: module header statement, revision statements and definition statements.

```
module acme-system {
    namespace "http://acme.example.com/system";
    prefix "acme";
    organization "ACME Inc.";
    contact "joe@acme.example.com";
    description "The module for entities implementing
                the ACME system.";
    revision 2007-11-05 {
        description "Initial revision.";
    }
    container system {
        leaf host-name {
            type string;
            description "Hostname for this system";
        }
        leaf-list domain-search {
            type string;
            description "List of domain names to search";
        }
        list interface {
            key "name";
            description "List of interfaces in the system";
            leaf name { type string; }
            leaf type { type string; }
            leaf mtu  { type int32;  }
            ...
        }
        ...
    }
}
```

Fig. 1: Excerpt from Yang model

## III. NDN Node Management and Alignment with Yang Modeling

The operations of an NDN node requires a set of management capabilities, including (1) ability to monitor statistical information about Interest, Data, and NACK packet forwarding, the number of entries in data structures, etc. and (2) ability to control parameters of node modules (create faces, routes, manage certificates, etc.). In addition, NDN forwarder provides several "streams" of notification about internal events, such as created/destroyed faces, FIB, and RIB entries.

Management for the prototype NDN forwarder NFD [10], [11] is defined for each logical module of the forwarder (Face, FIB, Content Store, Strategy Choice, and RIB management) in terms of three high-level mechanisms based on interest/data exchanges: (a) status dataset for general retrieval of statistical information; (b) notification stream for receiving real-time updates of internal forwarder states; and (c) control command to update internal states. In all cases, the naming structure and parameters are defined at the level of specification. For example, to create a face, the specification defines a verb "`create`" with the corresponding set of required and optional input/output parameters.

The control command parameter encoding provides limited provisions for backward compatibility: each newly defined element can be declared "critical" or "non-critical" [12]. This allows existing implementations to safely ignore (or raise an error) when encountering an unrecognized field. However, in general, any significant modification of the command specification, such new input/output parameters, definition of new or deprecation/removal of the existing commands, would require the corresponding updates in the libraries and supporting management applications. In practice, NDN team has already encountered this problem with a third-party management library written in Java [13]: the updated release of NFD caused crashes when using in conjunction with Java-based management; highlighting the urgent need for flexible management structure of the forwarder.

In addition to traditional data structures to support packet forwarding, NDN node model requires additional configuration, including the trust anchor(s) and sets of identities and public key certificates [14]. The existing work in this area addressed, in part, the configuration from the perspective of individual nodes: NDNCERT protocol [15] designed to automate managing of NDN certificates; Initial work on secure bootstrapping [16], [17] aims to automate the provisioning of trust anchors on IoT devices using specialized interest/data exchanges based on secure out-of-band initialization (e.g., QR code scanning). It is still highly critical to provide a coherent identity and certificate management mechanism, enabling enterprise-level management of NDN nodes.

### *A. Alignment with Yang Modeling*

The hierarchical structure of the Yang model gives a perfect alignment with the hierarchical naming and data-centric nature of NDN architecture. Yang models are effectively named objects that are exchanged within secure channels (Figure 2), each model defining the namespace of all management values and operations. Each value and operation within the model namespace is also named, potentially nested, and associated with named (potentially hierarchically) set of accepted values.

One significant difference that the proposed work plans to address is management of security and the granularity of the security. The IP-based NETCONF, inherits channel-based model, where point-to-point transport sessions between management clients and device agents are secured as a whole,

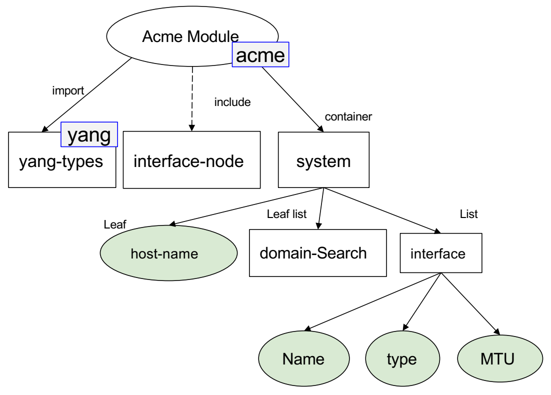

Fig. 2: Hierarchical structure of the excerpt

potentially giving different privileges at the application-level.[1] At the same time, the expressiveness of NDN names allows security permissions to be associated directly *at the networking level* with the management items of Yang models.

## IV. NDNconf: NDN Network configuration management based on NETCONF

The general operations of the proposed NDNconf design involve communications between the NDNconf server (potentially multiple independent or synchronized entities) and NDNconf agents running on network devices. These communications include obtaining the supported model by network elements, performing RPC, obtaining network state, and configuration of parameters (Figure 3). The important feature of NDNconf is not requiring the dedicated secure sessions nor point-to-point in general, as the data-centric communication will not be limited to point-to-point nature of TCP/IP. Specifically, by applying multi-party communication via dataset synchronization [18], [19], [20], NDNconf will provide native network-as-a-whole configuration capabilities. NDNconf's data models will be directly named and **secured**, so any interested party can fetch them either from the agents or anywhere else in the network: the unique name, integrity, and authenticity protections of NDN ensure that the network returns the right model.

The following management operations, including configuration changes, activation of configuration, and retrieving of statistical information, will be also based on the standard NDN interest/data exchanges, with security being directly associated with the data and/or request for the data. In other words, whenever NDNconf interest is requesting a state change or aims to obtain confidential management information, it must be signed by the proper key. The most significant detail here is that the name of the management interest, as highlighted in Section IV-A, will be fully derived directly from the data model, i.e., the model would not only define the supported management elements, management operations,

[1]As the most common transport channel for NETCONF is SSH, the privilege levels can be defined per user using any OS-specific mechanisms, e.g., user-level and SELinux permissioning when managing Linux-based boxes.

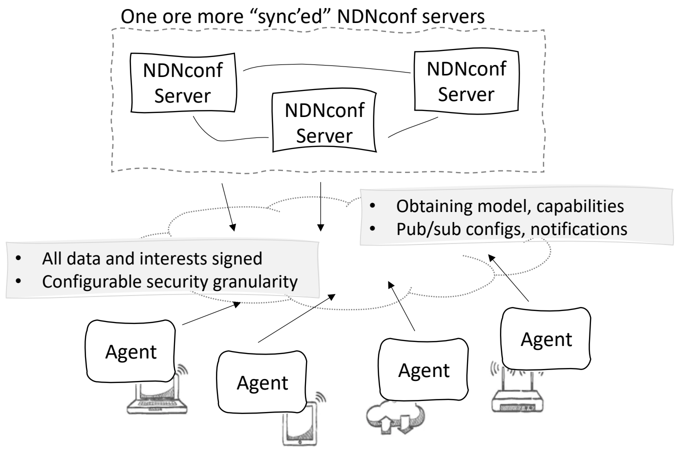
One ore more "sync'ed" NDNconf servers
NDNconf Server
NDNconf Server
NDNconf Server
Obtaining model, capabilities
Pub/sub configs, notifications
All data and interests signed
Configurable security granularity
Agent
Agent
Agent
Agent

Fig. 3: NDNconf overview

and data types, but also how to construct requests to execute management operations. With this, when the agent receives an NDNconf interest, it can decide the specific action(s) requested by merely looking at the specific portions of the interest name (see Section IV-A). If this operation is a valid control command defined in the device-supported data model, NDNconf agent then can check the signature and validate whether the requester has the permission to execute this command by applying the appropriate trust schema (see Section IV-B). If the validation succeeds, the requested action is performed by the NDNconf agent and response is generated as a data packet with information about the success or an error message is sent back to the requester.

## *A. Data and Interest Naming*

Name is a fundamental element of any NDN-based application, because the same name is being shared between applications and the network, with parts of the name defining paths the requested data can be retrieved (routing—the network function), how to dispatch the request to the producer applications (multiplexing and demultiplexing—the transport function), and how to identify the application-level information.

Generally following NDN naming structure and inspired by features of NETCONF, we identified the need for the following components to be exposed as part of the names:

- a prefix to identify the network and/or individual device to be managed;
- a component to indicate what operation needs to be performed (e.g., get-config, set-config , edit-config, get, set, etc.);
- a component to identify the scope of the operation, such as which datastore (running, candidate, start-up);
- components to identify subject of the operation (data model, module, elements, etc.); and
- auxiliary components to secure, version, and fragment Interest/Data as needed.

This structure results in names like:

"`<prefix...>/NDNconf/<operation>/<datastore>`
`/<subject...>/Parameters:<leaf>=<value>/...`"

Given the alignment of YANG data model and NDNconf, the "`subject...`" part of NDNconf naming is directly defined in the model. For example, as graphically highlighted in Figure 5, for the face system of an NDN forwarder shown in Figure 4), the subject components for obtaining information about the face will look like:

"`.../NDNCore/faces/face/Show/...`"

and the overall name of the Interest to get information about face with ID 300 from a router on NDN testbed will be:

"`/ndn/spurs/NDNconf/get-config/candidate/NDNCore`
`/faces/face/Show/FaceID=300/<aux...>`"

Note that the "`<aux...>`" part in the name are versioning (to ensure data immutability properties), fragmentation (to retrieve status information that does not fit in a single Data packet), and can include signature when accessing protected sections of the node configuration.

```
module faces {
  /* ... */
  list face {
    key "faceId";  description "List of faces";
    leaf faceId {
      type string; description "Face ID to identify face uniquely";
    }
    leaf uri     { type string; }
    /* ... */
    leaf nInInterests { type uint64_t;  }
    leaf nInData      { type uint64_t;  }
    /* ... */
  }

  /* RPC */
  rpc create {
    input {
      leaf faceId { type string; }
      leaf uri    { type string; }
      /* ... */
    }
    output {
      leaf faceId { type string; }
      leaf uri    { type string; }
      /* ... */
    }
  }
  rpc destroy {
    description "";
    input {
      leaf faceId { type string; }
    }
    output {
      /* ... */
    }
  }
}
```

Fig. 4: Excerpt of YANG model to manage face system of an NDN forwarder

## *B. NDNconf Control Granularity*

Given the exposure of the data model naming in the Interest and Data of NDNconf, the same naming can be leveraged to realize flexible access control. In particular, by applying security schemas [21] with a proper use of the corresponding key hierarchies, one can define any level of the access to management information and control. This is illustrated in the example shown in Figure 6 where the user is authorized to configure only the face module of candidate configuration datastore identified by "`<prefix>`". The user is not authorized to perform any operation on another component except for

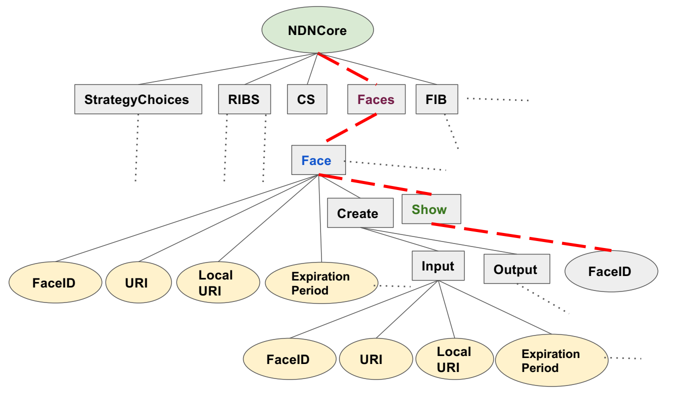

Fig. 5: Hierarchical structure of YANG data model for NDN

the faces of the particular device identified by "`<prefix>`". For example, in the example shown in figure 6 user is authorized to perform any operation on the interest packet with prefix "`<prefix>/NDNconf/edit-config/candidate/NDNCore/faces`". But if the same user wants to erase some entry from the content store using the interest packet "`<prefix>/NDNconf/edit-config/running/NDNCore/cs/erase/Name=sno`", he is not authorized to perform this operation. In the figure 6, "`<><>*`" is used to represents one or more component of interest packet name. Figure 6 shows that the prefix in interest packet name is fixed for a particular set of user, whereas "`<><>*`" can take one or more value according to the operation user wants to perform.

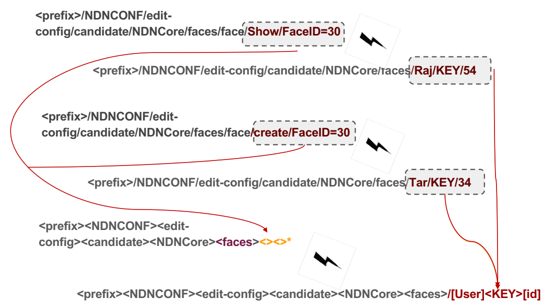
<prefix>/NDNCONF/edit-config/candidate/NDNCore/faces/face/Show/FaceID=30
<prefix>/NDNCONF/edit-config/candidate/NDNCore/faces/Raj/KEY/54
<prefix>/NDNCONF/edit-config/candidate/NDNCore/faces/face/create/FaceID=30
<prefix>/NDNCONF/edit-config/candidate/NDNCore/faces/Tar/KEY/34
<prefix><NDNCONF><edit-config><candidate><NDNCore><faces><><>*
<prefix><NDNCONF><edit-config><candidate><NDNCore><faces>/[User]<KEY>[id]

Fig. 6: Example of NDNconf control granularity

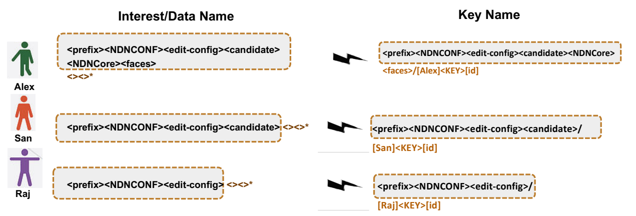
Interest/Data Name
Key Name
Alex
<prefix><NDNCONF><edit-config><candidate><NDNCore><faces><><>*
<prefix><NDNCONF><edit-config><candidate><NDNCore><faces>/[Alex]<KEY>[id]
San
<prefix><NDNCONF><edit-config><candidate><><>*
<prefix><NDNCONF><edit-config><candidate>/[San]<KEY>[id]
Raj
<prefix><NDNCONF><edit-config><><>*
<prefix><NDNCONF><edit-config>/[Raj]<KEY>[id]

Fig. 7: Generalized relation between data and key names

Figure 7 generalize the relation between interest/data and key names. First user, Alex is authorized to configure any parameters of faces of candidate configuration datastore of the device identified by "`<prefix>`". Second user, San is authorized to configure any component of the candidate configuration of the device identified by "`<prefix>`". Third user, Raj is authorized to manage any parameter of the device identified by "`<prefix>`". The user names (Alex, San, Raj) used in the Figure 7 represents a particular set of users with specific privileges. We can assume that Alex represent the group of network operators, San represents the group of network administrators and Raj represents the group of network architects.

Figure 8 shows the example of trust schema for NDNconf protocol. As depicted in the figure 8, there is a trust anchor or namespace owner "`<prefix>/NDNconf/KEY/1`" who signs the key of the network architects "`<prefix>/NDNconf/edit-config/Network_Architect_Raj/KEY/23`" who in turn signs the key of the network administrators "`<prefix>/NDNconf/edit-config/candidate/Network_Admin_San/KEY/11`". Network administrator signs the key of the network operators "`<prefix>/NDNconf/edit-config/candidate/NDNCore/faces/Network_Operator_Alex/KEY/54`". This signing process forms the trust chain.

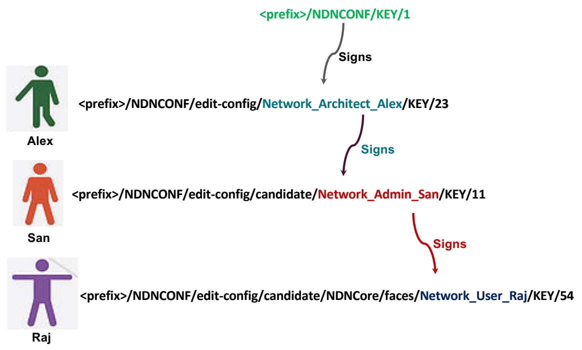
<prefix>/NDNCONF/KEY/1
Signs
Alex
<prefix>/NDNCONF/edit-config/Network_Architect_Alex/KEY/23
Signs
San
<prefix>/NDNCONF/edit-config/candidate/Network_Admin_San/KEY/11
Signs
Raj
<prefix>/NDNCONF/edit-config/candidate/NDNCore/faces/Network_User_Raj/KEY/54

Fig. 8: Example of authentication path for NDNconf protocol with prefix/NDNconf/KEY/1 as the trust anchor

## C. *Notification in NDNconf*

Notifications are asynchronous messages which are announced by the NDNconf agent when a particular event occurred. NDNconf protocol utilize the NDN sync-based publish/subscribe mechanism to implement the notification support. To understand the notification support in NDNconf protocol, we first need to understand the concept of data stream and the publish/subscribe mechanism of NDN. A data stream is the set of data which have the same name prefix but different sequence numbers. Each device in the NDN network (or NDNconf agent) generates a set of data streams and NDNconf manager is interested in a subset of the data streams. Suppose each NDNconf agent generates a data stream about the percentage of utilized memory with the name "`<prefix>/NDNconf/Percentage_Memory`" and NDNconf manager is interested in the data stream, it can subscribe to the corresponding name prefix using NDN publish/subscribe mechanism. NDNconf manager will be informed every time when a new data point is generated in the subscribed data stream. NDNconf manager can subscribe to different data streams of their choice using the data stream's name prefixes. NDNconf protocol uses the interest and data packets for notification support between manager and agent. It is the NDNconf manager responsibility to send an interest packets in order to get notifications from agents. These interest packets can have special component called filter which enable the NDNconf manager to select when to receive notifications. For example, NDNconf manager can specify to receive notification only when percentage of memory utilized by NDNconf agent is more than 90 percent. This filter can be specified by the NDNconf manager when the subscription is created. These event notifications will continue to be sent until the subscription terminated by the NDNconf manager. NDN PSync protocol [22] can be utilized to implement the notification support for the NDNconf protocol.

## V. Conclusion

Named Data Networking (NDN) as a new networking architecture requires standardized mechanisms to perform router and network-as-a-whole management tasks. In this paper, we introduced an initial version of a NDNconf management protocol that is aimed to address this need, leveraging advances of the recent developments in the traditional network managements, as well as data-centricity and security of NDN. Currently, we are working towards the implementation and integration of the NDNCONF protocol with the existing NDN codebase.

## References

[1] IETF, "Network configuration WG charter," Online: https://datatracker.ietf.org/doc/charter-ietf-netconf/, Last updated 2018.

[2] J. Case, R. Mundy, and D. Partain, "Introduction and applicability statements for Internet standard management framework," RFC 3410, December 2002.

[3] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*. ACM, 2009, pp. 1–12.

[4] NDN Team, "Named Data Networking (NDN) Project," Named Data Networking Project, Technical Report NDN-0001, October 2010.

[5] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, P. Crowley, C. Papadopoulos, L. Wang, B. Zhang *et al.*, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.

[6] A. Afanasyev, T. Refaei, L. Wang, and L. Zhang, "A brief introduction to Named Data Networking," in *Proc. of MILCOM*, October 2018.

[7] W. Shang, A. Afanasyev, and L. Zhang, "The design and implementation of the NDN protocol stack for RIOT-OS," in *Proceedings of GLOBECOM Workshop on Information Centric Networking Solutions for Real World Applications (ICNSRA)*, Dec. 2016.

[8] J. Schonwalder, M. Bjorklund, and P. Shafer, "Network configuration management using netconf and yang," *IEEE communications magazine*, vol. 48, no. 9, 2010.

[9] M. Bjorklund, "The yang 1.1 data modeling language," Tech. Rep., 2016.

[10] A. Afanasyev, J. Shi, B. Zhang, L. Zhang, I. Moiseenko, Y. Yu, W. Shang, Y. Li, S. Mastorakis, Y. Huang, J. P. Abraham, E. Newberry, S. DiBenedetto, C. Fan, C. Papadopoulos, D. Pesavento, G. Grassi, G. Pau, H. Zhang, T. Song, H. Yuan, H. B. Abraham, P. Crowley, S. O. Amin, V. Lehman, M. Chowdhury, and L. Wang, "Nfd developer's guide," NDN, Technical Report NDN-0021, Revision 8, Feb. 2018. [Online]. Available: http://named-data.net/techreports.html

[11] [Online]. Available: https://redmine.named-data.net/projects/nfd/wiki/Management

[12] NDN Team, "NDN packet format specification," http://named-data.net/doc/NDN-packet-spec/current/, Accessed: Novermber 10, 2018.

[13] "jndn-management," Online: https://github.com/01org/jndn-management, Last accessed: November 8, 2018.

[14] H. Zhang, Y. Li, Z. Zhang, A. Afanasyev, and L. Zhang, "NDN host model," *ACM SIGCOMM Computer Communication Review*, vol. 48, no. 3, pp. 35–41, July 2018.

[15] Z. Zhang, Y. Yu, A. Afanasyev, and L. Zhang, "NDN certificate management protocol (NDNCERT)," NDN, Technical Report NDN-0050, April 2017.

[16] L. Pi and L. Wang, "Secure bootstrapping and access control in NDN-based smart home systems," in *IEEE INFOCOM WKSHPS*, 2018.

[17] R. Tourani, T. Mick, S. Misra, and G. Panwar, "Security, privacy, and access control in information-centric networking: A survey," *IEEE Communications Surveys Tutorials*, vol. 20, no. 1, pp. 566–600, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8027034/

[18] T. Li, W. Shang, A. Afanasyev, L. Wang, and L. Zhang, "A brief introduction to NDN dataset synchronization (NDN Sync)," in *Proc. of MILCOM 2018*, October 2018.

[19] W. Shang, Y. Yu, L. Wang, A. Afanasyev, and L. Zhang, "A survey of distributed dataset synchronization in Named Data Networking," NDN, Technical Report NDN-0053, May 2017.

[20] Z. Zhu and A. Afanasyev, "Let's ChronoSync: Decentralized dataset state synchronization in Named Data Networking," in *Proceedings of the 21st IEEE International Conference on Network Protocols (ICNP)*, 2013.

[21] Y. Yu, A. Afanasyev, D. Clark, kc claffy, V. Jacobson, and L. Zhang, "Schematizing trust in Named Data Networking," in *Proceedings of 2nd ACM Conference on Information-Centric Networking*, Sep. 2015. [Online]. Available: http://dx.doi.org/10.1145/2810156.2810170

[22] M. Zhang, V. Lehman, and L. Wang, "Scalable name-based data synchronization for named data networking," in *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*. IEEE, 2017, pp. 1–9.