Towards a Monitoring Protocol Over Information-Centric Networks

Dima Mansour  
University of Basel  
Switzerland  
dima.mansour@unibas.ch

Christian Tschudin  
University of Basel  
Switzerland  
christian.tschudin@unibas.ch

ABSTRACT
Managing data centers remains an important challenge despite the extensive research in this area. As the future prospective of networking is location-agnostic networks as in Information Centric Networking (ICN), some problems are solved but new challenges arise. When a data center runs over an ICN network, how can it discover failing replicas, monitor the load on running replicas, or decide when and where to instantiate new replicas? All these questions are hard to answer when the network is purely location-agnostic. In this paper, we discuss the problem of pinpointing replicas and locations on ICN networks. We propose a draft of a protocol, Name-Centric Monitoring Protocol (NCMP), the use of which makes it possible to scan for replicas serving a named entity (content or service) and communicate with each one individually. A main application of NCMP is to monitor data centers and pull load information from service replicas.

CCS Concepts
• Networks → Network protocols; Network monitoring; Network services;

Keywords
Information-Centric Networking, Monitoring Protocol.

1. INTRODUCTION
Recently, Information-Centric Networking (ICN) has been a hot networking research topic. ICN tries to glue the gap between user needs and the fundamentals of the current communication model. At its core, ICN is a location-agnostic network architecture where content is accessed by name (e.g., URLs), not by location (i.e., IP). Many projects implement ICN principles to retrieve content by name, such as DONA [13], NDN[11], PURSUIT[9] and CCN-Lite[1]. Other projects focus on invoking services by name[15][19][citeSCN and implement ICN using Software-Defined Networking (SDN) [8][7].

One of the main applications of ICN networks is data centers. One main challenge in this use case is how to control and monitor the data center without being able to communicate with servers individually. Jun Ko et al. [12] introduce Information-Centric Data center Network (IC-DCN) which aims to achieve better utilization of the network resources, managing the name space effectively, and routing scalability in data centers. However, IC-DCN isolates control-plane functionality from data-plane functionality and assumes that communication between planes utilizes IP-based networks. Although this is necessary to communicate with replicas\(^1\) individually, it makes IC-DCN not a pure ICN solution. Having multiple communication paradigms for different purposes in data centers complicates management activities and burn out the benefit of having simple network primitives.

In this paper, we propose a draft of a protocol, Name-Centric Monitoring Protocol (NCMP), to overcome this challenge. NCMP is pure ICN and makes use of the ICN primitives to scan different replicas serving the same named entity (content or service)\(^2\) to be able to collect load and liveness information. NCMP can be used to monitor the performance of a data center, discover failures, and dynamically kill old replicas or initiate new replicas as needed.

The rest of this paper is organized as follows: In section 2 we motivate this study by demonstrating the challenges of applying ICN on data centers. Then we explain the main steps of NCMP in section 3. In section 4 we discuss the main implementation and application issues of NCMP. Finally, we conclude this paper in section 6 detailing our future directions.

2. CHALLENGES OF APPLYING ICN ON DATA CENTERS
Throughout this paper, we assume that data centers work purely based on ICN design principles. We also assume that there is a dedicated machine or server, we call it the controller, that needs to collect run-time information from different replicas.

The features of a network monitoring protocol mainly include the ability to retrieve information from a specific net-
\(^1\) The terms “replica” and “instance” are used interchangeably in this paper.
\(^2\) Since NCMP can be used for content providers as well as service providers, we do not differentiate between the two in the paper. Often we use the term “named entity” to refer to both.
work entity about the entity’s health and the ability for en-
tities to send alerts (aka traps) to notify a management cen-
ter of exceptional events. In an ICN-based data center, the
main challenge is that the controller does not know where
data and services are distributed, how many replicas are de-
ployed for each named entity, or how to gather information
about the physical machines that hold the replicas. The only
thing it knows is the names. So practically, all traditional
monitoring protocols over IP-based data center cannot be
applied on ICN-based data centers We discuss here some
challenges of applying ICN on data center:

- How could the controller gather information about the
  physical machines of replicas while the only thing it
deals with is the name, not the the end point identifiers
  (such as IP addresses).
- In ICN, there are two kinds of packets: the Interest
  and the data packets. For each Interest (request) there
should be at most one data packet that satisfies it. In
this context, when the controller sends a request, it
does not know which replica replis. When the con-
troller sends a request, all replicas receive it, according
to the default forwarding strategy in ICN, and all repli-
cas reply, but only one answer reaches the controller.
- How could the controller discover failure of replicas?
- How could the controller distribute the load between
  replicas ?

In the following section, we introduce a simple protocol to
establish individual connections between the controller and
the controlled replicas. We also show how this protocol can
inherently solve many of the mentioned challenges.

3. NAME-CENTRIC MONITORING PROTO-
COL (NCMP)

To describe our protocol NCMP, we assume the ICN topol-
ogy shown in Figure 1. It contains three replicas for a service
S which is published under the name: /ch/unibas/register,
and one controller. The replicas S1, S2 and S3 do not know
about the controller address, and the controller in turn does
not have any information about the replicas, except for the
published service name which is common between all the
instances.

NCMP has two phases. The first phase is called init_ids
where the controller instructs all replicas to initialize their
IDs. The second is called pull_info where the controller asks
a specific replica for information using the replica ID. The
procedure is described as follows (Figure 2):

1. The controller sends an Interest packet with a com-
   pound name that contains two parts. The first part
   (prefix) is the named entity. In our case, it is /ch/uni-
bas/register. The second name has the reserved name
   component init_ids. So the whole name for the con-
troller Interest in our case is /ch/unibas/register/init_ids.
   We assume that every content/service provider should
   implement the functionality to respond to this request,
as we describe in the coming steps.

2. Every replica has a temporary table which matches
   the public key (PK) of the controller with its tempo-
   rary ID. When the replicas receive the request, each of
   them generates an ID, inserts the tuple (controller_PK,
   replica_ID), and sends the ID back to the controller.
   Although all replicas reply, only one reply will reach
   the controller at the end, the ID (=7) of S3 for in-
   stance, as shown in Figure 1. If the tuple already ex-
ists, the replica just replies with its own ID, without creating a new one or inserting any new tuple in the table.

3. As soon as the controller receives the ID=7 from S3, it saves it in a special table where all received IDs are saved. Then, it sends a request asking for some information from S3 about its physical machine (CPU, memory, overhead, number of requests per second, etc.) using an Interest name that has the entity name as a prefix and the name component pull info as a suffix. In our case the full Interest name is: /ch/unibas/register/pull_info.

However, this request has a parameter which is the ID of S3. This ID can be embedded in the name itself, as in NextServe [14] or NFN [19][2] or in one of the fields that construct an Interest packet.

4. This Interest will be received by all replicas. Each replica checks its own table if it has the corresponding ID for this public key in the Interest (of the controller). Only the one that has the same ID replies. So, S3 prepares a Data packet containing the run-time information and sends it back to the controller.

5. The controller needs to get information from other replicas as well. It sends an Interest with the name /ch/unibas/register/get ID with the list of already received IDs as a parameter. In our case, the received ID is 7. All replicas receive the Interest but only the ones with their IDs in the parameters do not reply (S3) while the rest of them reply with their IDs (S1 and S2). At the end, one reply reaches the controller, as in step 2. In our case let’s assume it is the ID (=12) of S1.

6. The controller repeats step 2, 3, 4, and 5 until its requests time out. By then, the controller knows how many replicas there are and has the needed run-time information.

The controller needs to repeat this procedure periodically to collect fresh data and keep the logs up-to-date.

4. DISCUSSION

4.1 On the Implementation

_instance IDs_

There are many of implementation issues that are worth discussing. The first is replica IDs. IDs do not exist in ICN networks. Content/service providers have public keys (PKs). However, there is a unique public key per provider or at most per named entity, but not per replica. For instance, in the example we have, the service ch/unibas/register can have its own public/private key pair or it can share the public/private key pair of the provider with all other named entities provided by the university of Basel. Thus, public keys cannot serve as IDs. One possibility is the use of random number generation. When a replica receives an init ID Interest, it generates a random number and considers it as its ID. This solution is simple and easy to implement but at the risk of generating two identical IDs. A second solution can be to use time. When a replica receives the init ID Interest, it takes the current time. Most systems represent time as the number of milliseconds since Jan. 1st 1970. The fact that the replicas cannot possibly receive the Interest exactly at the same time and the fact that the clocks of different machines cannot be exactly synchronized make it impossible to have identical IDs based on time. A third solution is to leave the implementation of ID generation to the provider as it might have a policy of assigning IDs to server instances.

get ID Parameters

When there is a large number of replicas, the parameter list of get ID can grow large. Suppose we have a thousand replicas. At the last iteration of the protocol, the controller sends the get ID Interest to get the last ID of the last replica. This interest has a parameter list of 999 items (Received IDs). One solution to this problem is to use the Bloom Filter [3]. In this case, the get ID has only one parameter which is the binary representation of the set of received IDs. When a replica receives the get ID Interest it can check if its ID is in the set via simple binary operations. In this case,
no matter how many replicas there are, there is only one parameter in the get ID Interest.

**Failure Discovery**

When a replica fails, the controller stops receiving replies from it and discovers that there is a replica down. However, it cannot define specifically which one that is.

One solution is that every time a replica receives an init IDs, it sends back the new and the previous one to the controller. The controller stores the history of the IDs of each replica. This way, the controller can decide which replica is the one that stops responding when it discovers that there is a failing replica.

**Security**

Replicas cannot possibly reply to all Interests requesting run-time information. One solution is public key authentication, as a popular mechanism in ICN [17][5]. Every server instance has list of authorized controllers (public keys) and it only answers to Interests when the public key is on the list. Then the Data packet is encrypted using the public of the controller. The data packet should also be signed with the private key of the provider. In this case, only the controller can decrypt Data packets only if it is authorized. Then the controller can verify the signature and make sure it comes from the trusted provider. Following this procedure ensures that the controller and the replicas trust each other and that sensitive data is protected.

**Pull vs Push**

NCMP has a pull architecture. It is similar to the get command in the Simple Network Management Protocol (SNMP) [6]. As explained in section 3, the whole procedure is instantiated by the controller and the server instances are passive. However, we plan to extend the commands of NCMP to have asynchronous notifications (push architecture) similarly to SNMP traps [6].

**4.2 On the Application**

NCMP allows the controller to collect information about the running replicas. This information can be used to build dashboards for network administrators and DevOps to monitor named entity replicas. However, it can also be used to dynamically and automatically discover failures when the replicas stop replying to NCMP requests. Indeed, the controller can also, based on the load information, discover that a specific replica is not needed and decides to kill it as it is a waste of resources.

The main challenge here is when the controller decides to instantiate a new replica in a specific part of the network when it finds that the replica there is overloaded with requests. We plan to investigate how this can be done in a pure ICN network.

**5. RELATED WORK**

To our knowledge, we are the first to tackle the problem of pinpointing server instances over ICN in a purely ICN manner. However, there are some approaches to routing over ICN [18][10][16] that share some of the challenges and goals of NCMP. For instance, SoCCeR [16] tries to route service requests to service replicas with lighter loads considering some metrics (service load, path congestion, etc.) to be able to select the best available replicas in a network. SoCCeR uses Ant Colony Optimization (ACO) and constantly sends Interest “ants” to determine the best face serving a specific name in content routers. NCMP can achieve the same goal but on the level of server instances and not router faces. Besides, the goal of SoCCeR is to route Interests to the best server replica, whereas the goal of NCMP is to collect run-time information about all server replicas in a central location to facilitate the management of the network.

Another closely related work is Information-Centric Data Center Network (IC-DCN) by Jun Ko et al. [12]. IC-DCN shares the same concern as NCMP on how to monitor and manage ICN service instances in a data center. However, IC-DCN implements a hybrid approach where it uses IPs to send and receive control commands, whereas NCMP relies solely on ICN primitives.

**6. CONCLUSIONS AND FUTURE WORK**

Information-Centric Networking (ICN) is a promising networking paradigm. It comes with the benefits of inherent solutions to many problems with the current IP-based networks. However, it also comes with some challenges coming mainly from the inability to communicate with replicas individually. We propose Name-Centric Monitoring Protocol (NCMP) that makes it possible to pull information from a specific replica in a pure ICN environment. NCMP allows a special machine, the controller, to scan all service/content replicas and pull run-time information, which in turn can be used to build monitoring dashboards, discover failures, and dynamically instantiate or kill replicas as needed. We believe that such a protocol is a necessary step towards smarter and self-managing networks. In the future we intend to implement NCMP and explore its applications concretely.

**7. REFERENCES**


