Controller-based Routing Scheme for Named Data Network

João Vitor Torres, Lyno Henrique G. Ferraz, Otto Carlos M. B. Duarte
Universidade Federal do Rio de Janeiro - GTA/COPPE/UFRJ
Rio de Janeiro, Brazil
Email: {jvitor, lyno, otto}@gta.ufrj.br

***Abstract*—Named Data Network (NDN) routing schemes must learn routes to named data locations, so routers know where to send interest packets. These routing schemes are based on IP routing schemes, therefore they inherit characteristics such as prefix dissemination and routing based prefix longest match. As the amount of named data and non-aggregated prefixes increase, routers store more routes and exchange more control messages, which results in high control overhead and possible Forwarding Information Base explosion. We address these issues with a novel Controller-based Routing Scheme (CRoS) for NDN. CRoS introduces special controllers which have two main functions: i) acquire topology and calculate routes, and ii) store named data locations. Named data locations are registered in controllers. On interest packets to an unknown prefix, routers request controllers for installation of a new route. CRoS controllers implement distributed hash tables to distribute the storage of named data locations efficiently. Furthermore, as CRoS runs on top of the NDN, it preserves NDN features such congestion control, network problem detection and path diversity.**

I. Introduction

The concepts introduced with the advent of Content Centric Network (CCN) [1] and Named Data Networking (NDN) [2] drastically change networking principles, as the focus is content delivery instead of host-to-host communication. Nevertheless, this model is already used in current networks running on top of a host-to-host-based network. NDN proposes a new network layer which uses unique names to identify addressable content. The network layer uses two types of packets, the interest and data packets. The interest packet expresses consumers will for content and also leaves information on each hop to reach the consumer back. For each interest packet, the network replies with a data packet containing the desired content. The NDN ensures efficient communication, load balance, energy efficiency, and flow control through popular content storage and data packet replies from any content cache copy [1], [3].

As named data addresses content directly, the NDN must learn how to route interest packets based on content names. Therefore, NDNs use routing schemes to announce named data prefixes and make them addressable. However, NDN routing schemes such as Open Shortest Path First for Named-data (OSPFN) [4] inherit IP characteristics as they focus on prefix dissemination and routing. These schemes suffer with the amount of named data which is intrinsically higher than IP prefixes. In addition, mobility introduces non-aggregated prefixes which increase the number of routes. In these situations, the routing schemes should store more routes and exchange more control messages to announce all the addressable content, which results in high control overhead and possible Forwarding Information Base (FIB) explosion.

With this in mind, we propose the Controller-based Routing Scheme (CRoS) for NDN. CRoS introduces special network elements called controllers that are responsible for the named data location storage and routing. The controllers acquire the topology in a bootstrap phase and calculate routes to all routers. In this phase, the router-controller routes are installed in routers, but the routes to named data are not. After the bootstrap phase, new named data is registered in the controllers, which store the named data location. Hence, a router can request the controllers for installation of a new route to an unknown prefix. Since all named data locations are registered in the controllers, they can calculate routes to any valid named data.

CRoS runs on top of the NDN. Thus it uses only NDN packets and it preserves NDN features such as congestion control, network problem detection and path diversity. Still, CRoS uses special interest packets with semantically meaningful names to reduce control overhead. To avoid explosion of named data location storage in the controllers, we use distributed hash tables (DHT). Hence, storage is balanced between controllers and new controllers can be added with minimal impact.

The rest of this paper is structured as follows. In Section II, we describe the main related work. We present the routing scheme in Section III. In Section IV we discuss the impact of mobility, the security of the scheme, network partition and inter-domain routing. Finally we conclude and present future work in Section V.

II. Related Work

Jacobson *et al.* proposed Content Centric Network (CCN) and introduced the interest-data packet pair model [1], [5]. This proposal resulted in a project that aims in building a new network architecture, the Named Data Network (NDN) [2]. NDN model allows fast detection of network problems and use of alternative paths according to a strategy layer [6]. However, current NDN routing schemes construct named data forwarding rules based on OSPF and OSPF floods non aggregated updates to all network nodes, imposing serious scalability limitations on the supported number of distinct prefixes and their mobility [4]. Baid *et al.* uses a two level indirection scheme to diminish memory requirements and

message exchange, mapping named data prefixes to a reduced set of flat identifiers and these identifiers to network addresses. It employs a DHT system to provide this indirection [7].

Some proposals also address content, but use a different approach based on a publisher-subscriber architecture [8]. Carzaniga *et al.* compares NDN on-demand content retrieval to subscription approaches. It presents a hybrid solution with selective and reduced use of packet flow state at routers [9]. Packet flow balance is a fundamental feature to provide adaptability to NDN and it is not a publisher-subscriber feature.

Besides, proposals such as Software Defined Networks (SDN) employ a centralized controller to install on demand flows forwarding rules in the network [10], [11], [12]. Controller based solutions alleviate general packet forwarding nodes from control message processing and fit well for next generation networks [13], [14]. The centralized controller function reduces message exchange, memory and processor requirements in forwarding nodes overcoming centralized criticism. Furthermore, this single failure point is in general redundant and it takes charge for limited subset of nodes [15].

III. Scheme

Our Controller-based Routing Scheme (CRoS) is based on NDN interest and data packets. Thus, packet forwarding follows default router processing through Content Store (CS), Pending Interest Table (PIT) and Forwarding Information Base (FIB) as detailed in [6]. The routing scheme expands the default processing and uses special interest packets with semantically meaningful names as SIP (Session Initiation Protocol) invite messages of Voice over CCN (VoCCN) [5]. The meaningful name interest packet reduces control overhead, because it is possible to send information directly in interest packets.

In our architecture, we consider that the network elements have two roles: router and controller. The router role is the basic network element role which forwards packets to destination and also registers the named data, and the controller role calculate routes and stores named data locations. We consider only one controller in this section for ease the scheme description, and we describe the scheme with multiple controllers in a separate subsection.

In this paper, we refer as named data any addressable and reachable data such as file, services and network elements. All routers and controllers have an ID, so they are also addressable in the network.

We use special prefixes reserved for the routing mechanism: `"/*"`, `"/router"`, `"/route"`, `"/controller"` and `"/register"`. The prefix `"/*"` means any node in the network, therefore when sent in a link, it is addressed to all neighbors; `"/router"` is normally followed by the router ID to forward packets to a specific router; `"/route"` is a route installation command that routers should process; `"/controller"` forward packets to any controller and if followed by a controller ID, it forwards packets to a specific controller; and finally the prefix `"/register"` is used to request registration of new named data.

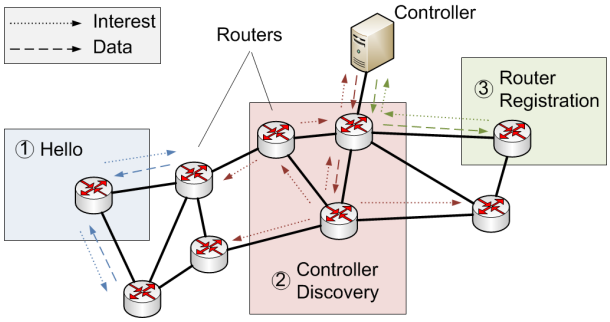

Figure 1. The bootstrap phase CRoS runs three sub protocol: 1) Hello Protocol, in which routers announce their presence to neighbors; 2) Controller Discovery, when routers run a flooding protocol to discover the controller location; an 3) Router Registration, when routers send their neighborhood to controller which assemble the topology.

Routers initially do not have any forwarding rule in FIB, except the rules that address the routers themselves for local processing such as `"/router/routerID"`, `"/route"`, `"/*"` and `"/register"`. Then, the routers initiate a bootstrap phase to enable controllers install routes.

A. Bootstrap

In the bootstrap phase, routers find the controllers to register themselves, and controllers get the topology and calculate routes. After this phase, the controllers know the routes to any router in the network and can install the routes on routers. The bootstrap phase has three sub-protocols: the Hello protocol, the Controller Discovery and Router Registration as Figure 1 demonstrates. Although these sub-protocols are essential for the network bootstrap phase, they run periodically to maintain routing information updated. Next we describe the sub-protocols and algorithms to discover routes.

1) Hello Protocol: All routers run a Hello protocol to inform their neighbors about their presence. Figure 1 item 1 illustrates the message flow. Routers send periodic interest packets

```
Interest("/*", nonce)
```

on all interfaces. Every router that receives this interest packet, replies with a data packet which contains its own ID and a sequence number

```
Data("/*",
ID:  routerID
SEQ: sequenceNumber).
```

The sequence number is used to topology coherence, thus it is updated whenever the router detects changes in the neighborhood, such as links or neighbor failures. When the router receives the data packet, it installs a FIB entry to prefix `"/router/routerID"` via incoming interface and stores the neighbor routerID and sequence number in a neighbors table. The router also stores the interface from the neighbor is connected to, and the metric for the link. The link metric can be the round trip time between the dispatch of interest packet

and the reception of the data packet, or other criteria. Using this protocol, all nodes have an updated neighbors list.

2) Controller Discovery: Routers must register themselves in the controller, so the controller assembles the network topology and calculates the network routes. However, as they initially do not have any forwarding rule to the controller, they first discover routes to the controller. The routers use a flooding protocol that ensures multipath, because all routers participate in route discovery and they use forwarding policies to choose the best path such as "the sooner the better".

The controller discovery flooding protocol initiates with the routers sending controller discovery interest packets asynchronously

```
Interest("/controller", nonce)
```

on all interfaces. When a router receives the interest packet, it adds to its PIT and forwards the interest packet to all interfaces. When the controller receives the interest, it replies with a data packet containing its ID and a sequence number

```
Data("/controller",
ID:  controllerID
SEQ: sequenceNumber),
```

where the sequence number is updated whenever the controller sends data packets regarding interest in `"/controller"` prefix. The router that receives the data packet stores it in its CS and forwards to the downstream path. Figure 1 item 2 illustrates the message flow. The router also install a FIB entry to prefix `"/controller"` and `"/controller/controllerID"` via the incoming interface and stores the controller ID with sequence number in a controllers table. As all routers receive the data packet because they sent or retransmitted the interest packet, all routers know a path to the controller. Routers store the data packet for a small period to reply late interests packets and limit the overhead of the flooding protocol.

Periodically, routers send new controller discovery interest packets update the routes to controller, but if in reply it receives a data packet which contains less or equal sequence number than it has stored in controllers table, the data packet is old and the router does not update the FIB entry.

3) Router Registration: When the router already knows the controller, it registers itself in the controller. Figure 1 item 3 illustrates the message flow. The router sends an interest packet with meaningful name which indicates a registration

```
Interest("/controller/controllerID
          /registerRouter/routerID
          /seq/sequenceNumber", nonce)
```

where `"controllerID"` is the known controller ID in controllers table, `"registerRouter"` is a reserved word that indicates the request for registration the `"routerID"`, and `"seq/sequenceNumber"` is the last announced sequence number in Hello Protocol. Each router that forwards the upstream path of interest packet, install a FIB entry to prefix `"/router/routerID"` via the incoming interface. When the controller receives this interest packet, it creates a router entry in the routers table and replies with an acknowledgement data packet.

After the registration of the router, the controller sends an interest packet to acquire the router's neighbors

```
Interest("/router/routerID
          /seq/sequenceNumber", nonce).
```

As the routers in the path already installed a FIB entry, the interest packet is forwarded directly to the router. When the router receives the interest packet, it replies with a data packet containing its own ID and sequence number and its neighbors' list

```
Data("/router/routerID
      /seq/sequenceNumber",
ID:  routerID
SEQ: sequenceNumber
NEIGHBORS: [
    [routerID, sequenceNumber,
     metric, interface],
    ...]).
```

Upon reception of the data packet, the controller updates the router entry adding the neighbor's list. To ensure that all data is updated and coherent, the controller compare the sequence numbers received with other router entries, and if detects different sequence numbers it sends interest packets to routers that announced old sequence numbers.

Whenever a router detects a topology change, the registration procedure is repeated, so that the controller has always the updated topology.

Route Calculus Algorithm: After the router registration procedure, the controllers know all routers in the network and their respective neighbors list. Therefore, it can infer the network topology and calculates the routes between any two routers using Dijkstra algorithm. The controller recalculates the routes whenever it perceives any topology change. The controller can also calculate alternative routes with higher costs to distribute the bulk traffic in the network and improve overall performance [16].

B. Routing

After the bootstrap phase, all routers can route to the controller, the controller has the topology and the calculated routes between all routers. However, the controller does not know the location of named data. Therefore, producers must register the named data in the controller, so it installs routes to named data in routers as depicted in Figure 2.

1) Named Data Registration: When the producer has a new unregistered named data, it sends an interest packet with meaningful name

```
Interest("/register/myprefix", nonce)
```

where `"register"` indicates the `"myprefix"` registration intention. When the connected router receives this packet, it adds a PIT entry and also adds a FIB entry to `"myprefix"`

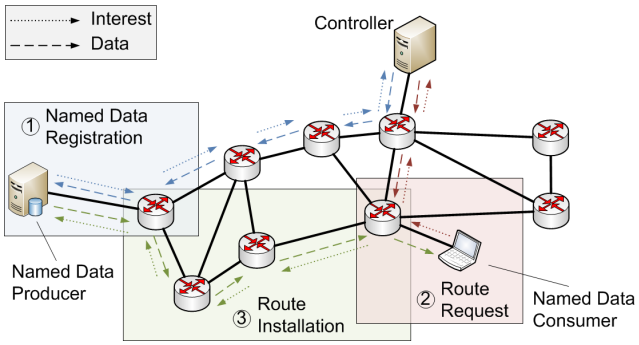

Figure 2. Routing in CRoS has three steps: 1) Named Data Registration, when producers register new named data in the controller; 2) Route Request, when routers ask the controller for routes to unknown prefixes; an 3) Route Installation, when routers install the requested route to the producer.

via the incoming interface. Following, instead of forwarding the interest packet to one of its interfaces, the router generates a new interest packet to indicate the named data location and sends it directly to the controller

```
Interest("/controller/controllerID
         /registerNamedData/routerID
         /myprefix", nonce).
```

When the controller receives this interest it stores `routerID` as the location of `"/myprefix"` in named data locations table. If the controller already received a registration request from part of that prefix at the same location, it can optionally aggregate the prefixes in a single entry. The controller replies an acknowledgment packet to the router, which replies an acknowledgment packet to the producer. Figure 2 item 1 illustrates the message flow.

2) Route Installation: When any node wants a named data, it sends the network an interest packet

```
Interest("/wantedprefix", nonce).
```

The first router receives this packet and adds to its PIT. If the router has no forwarding rule in its FIB, it should discover the route. Then, the router generates a route request interest packet with meaningful name

```
Interest("/controller/controllerID
         /routeFrom/sourceRouterID
         /wantedprefix", nonce).
```

where `"routeFrom/sourceRouterID"` indicates the source router of the route request interest. As `"wantedprefix"` is already registered, the controller knows the route destination. Then, the controller can search the best calculated route from source to destination and generate the route reply data packet.

The route reply data packet contains a string to be used in route installation, which contains the whole route from the source router until the destination. The controller can include also alternative routes in the route reply data packet.

```
Data("/controller/controllerID
      /routeFrom/sourceRouterID
      /wantedprefix",
ROUTE :
"/route/installRouteAndForwardInterest
 /sourceRouterID/routerID2
 /destinationRouterID/endroute
 /routingPreference/1
 /toprefix/wantedprefix"

ALTERNATIVE_ROUTES : [
"/route/installRoute
 /sourceRouterID/routerID3
 /destinationRouterID/endroute
 /routingPreference/2
 /toprefix/wantedprefix",
 ...])
```

The strings include the reserved words `"installRouteAndForwardInterest"` and `"installRoute"` to indicate the beginning of the route, `"endroute"` as end of the route, `"routingPreference"` as the route preference, and `"toprefix"` as an indication that the following is the wanted prefix route. Figure 2 item 2 illustrates the route request and reply message flow.

When the source router receives the route reply data packet, it searches for its ID in the string and gets the next hop. Next, the router add a FIB entry to `"/wantedprefix"` via the same interface it reaches the next hop router.

After that, the source router creates a special interest packet with received string as prefix to install the route. The next hop router receives the interest packet and uses the string to create the FIB entry. This process is repeated until the interest packet reaches the destination router, which already has a FIB entry to `"/wantedprefix"`. If the string has the reserved word `"installRouteAndForwardInterest"`, each router in path add a PIT entry to `"/wantedprefix"`, but does not add an entry regarding the special interest packet. Figure 2 item 3 illustrates the route installation message flow.

The destination router already has a FIB entry to the wanted prefix, so the route is now fully installed. If the string has the reserved word `"installRouteAndForwardInterest"`, the destination router also adds an entry to `"/wantedprefix"` in its PIT, and create and forwards an interest packet to `"/wantedprefix"`. If the string has `"installRoute"`, the destination router simply sends an acknowledgement data packet.

C. Multiple Controllers

The multiple controllers divide the named data location storage task, as well as the network topology vision. Each controller knows a routers subset called zone and router paths between routers in its own zone. Still, each controller has the entire controllers' topology view, so it can calculate the next zone hop to any other zone in the network. The controllers

create a distributed hash table (DHT) using the hash of its ID as the DHT node identifier, and the hash of named data prefix as the DHT key. The DHT value is the router ID for the prefix. Key distribution in DHT nodes follows an index rule based on DHT node ID. Hence, the controller can exchange data and lookup for data.

When the routers run the flooding protocol, they discover one controller and register themselves. With multiple controllers, each controller has a set of registered routers that form a zone in the network. Then, the controllers exchange their vision of its neighbor controllers to form the entire zone topology map. Controllers also exchange DHT information such as named data registration, named data location requests and DHT management messages.

1) Topology Exchange: After the bootstrap phase, each router is registered in a controller and stores the controller ID in the controllers table. We extended the Hello Protocol, so that routers include the known controller in Hello protocol messages, and neighbor routers detect the existence of different controllers. When this is the case, routers add a FIB entry to the prefix `"/controller/neighborControllerID"` via the interface of incoming neighbor Hello packet. Next, the router sends its own controller a special registration interest, which registers neighbor's controller at the router location

```
Interest("/controller/controllerID
          /registerController/routerID
          /neighborControllerID/RTT", nonce)
```

The controller replies with an acknowledgement data packet. As the same procedure runs to the neighbor's controller, both can install routes to the other controller.

Following, the controllers send each other an interest packet to acquire the other controller's topology vision to the prefix `"/controller/controllerID/topology"`. The first router requests the installation of the route which is installed until the border router. If multiple routers register the same neighbor controller, the controller can install multiple paths and use the one with minimum RTT. Then, the interest packet is forwarded until the other controller, which replies with its list of know controllers and their adjacency. This view enables controllers to calculate the best route to other zones. Controllers also inform already known controllers about the new controller with an interest packet to prefix `"/controller/controllerID/topologyUpdate"`. The controllers reply the interest and request the topology.

2) DHT Operations: After the topology exchange, controllers can access other controllers and exchange any DHT message. When controllers detect other controllers, they check if the named data keys it stores in its DHT should be transferred to other controllers and vice-versa. They send interest packets to prefix `"/controller/controllerID/dht/keyExchange"`, whose reply contains the named data keys that should be transferred. In addition, controllers forward to the correct controller all named data requests they do not know, and registration of named data that should be stored in other controllers.

3) Path Composition: When a controller receives a new named data registration whose key should be stored in other controller, it sends to the proper controller a registration message including also its ID. Thus, controllers know both the router which registered the data and also the controller of the zone. In installation of routes that the controller does not have the key stored, it consults the proper controller in DHT . The reply includes the destination router and controller's zone. When the named data is other zone, the controller installs a route to the border router in direction of the destination zone. The border router of the other zone consults its own controller for the installation of the route and so on.

IV. Discussion

In this section we discuss some issues that can affect the scheme operation.

A. Mobility

Mobility can affect the operation of content delivery because both producer as well as consumer can change its location. When a consumer changes its location, it should send new interest packets, so that the routes are installed in the new location. The producer mobility is more problematic because the consumer cannot foresee the producer mobility and resend interest packets to install new routes. In this case, producers have to inform the new locations and delete previous installed route to old location. In our proposal, the controller acts as a dynamic domain name system which knows the current location of producer. When a producer moves to another location, it reregisters prefixes in the controller. At the same time, when the producer old location router cannot find the producer it sends an Interest NACK No Data that forces the removal of the route up to consumer.

B. Security

In NDN, every producer has its own pair of keys that can be used to sign messages. Therefore, data validity, provenance and relevance can be verified. However, the use of semantically meaningful name interest packet could possibly harm the network operation. These interest packets change the network elements behavior and normally have no signed information. In this case, the generated interest packets could contain the signatures as in VoCCN [5]. Then, routers can verify that the interest packets were generated by the controllers.

C. Network partition

The network partition could harm the network, because some or all controllers may be unreachable. If there is still a controller in the partition, router can find it by the controller discovery sub-protocol. The controller acquires the new topology, and new routers register themselves. Since the routers always have FIB entries for the registered named data, the controller ask them to resend the registered named data in a data packet via a special interest packet. If the partition has no controller, routers elect one router to assume the controller

role. The procedure occurs as a new bootstrap phase, except that the controller request the routers to resend the registered named data.

D. Inter-domain Routing

Another issue relates to FIB and PIT memory scalability requirements. In the network core, named data packet rate and diversity increases. Furthermore, mobility increases FIB size with additional specific longer entries. These factors impose serious challenges to implement named data FIB and PIT with today technology [7].

Even a controller based strategy implementing memory reuse through FIB entry replacement would impose a significant message exchange overhead between routers and controllers. An encapsulation approach can be the strategy to overwhelm these limitations. In core network segments, router FIB stores only entries to other routers. Border core network routers encapsulate packet name with its own and the destination router IDs, as illustrated below.

```
Interest("/encapsulated/destRouterID
/srcRouterID/wantedprefix", nonce).
```

A core router receiving an Interest packet with the special prefix `"/encapsulated"` forwards the packet based on `"/destRouterID"` name component. For Data packets, core router forwards the packet based on `"/srcRouterID"` name component. Core routers do not depend on PIT to Data packets reverse path construction and PIT maintains only the balance flow for prefixes in the form `"/encapsulated/destRouterID/srcRouterID/"`. Besides, we can improve scalability using hierarchical distributed hash tables [17], [18].

V. Conclusion

In this paper we propose the Controller-based Routing Scheme (CRoS) for NDN. CRoS uses multiple controllers to preserve scalability. The controllers store the network topology and calculate the routes, and store named data locations, so that they can install route to any named data in the network. The controllers decentralized operation redistributes working load upon controllers failures, recoveries, substitution and new controllers inclusion. Controllers function allows router memory reuse storing only the mostly and the lastly used forwarding rules. Furthermore, controllers divide the network topology in zones, which limit the route calculus and flooding overhead.

NDN Router structure allows expanded route caching above FIB memory limits using its own Content Store. This feature alleviates the named data diversity impact on route request message overhead between routers and controllers.

For future work, we will analyze the scheme performance and compare with related routing approaches such as OSPFN. We will also examine tradeoffs between route caching and route expiration patterns. We will implement the scheme in ndnSIM [19] simulator, and we will test it using CCNx [20] distribution in Future Internet Testbed with Security (FITS) [14].

Acknowledgment

This work was supported by FINEP, FUNTTEL, CNPq, CAPES, FUJB, FAPERJ and UOL.

References

[1] 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*, ser. CoNEXT '09. ACM, 2009, pp. 1–12.

[2] L. Zhang *et al.*, "Named Data Networking (NDN) Project," Oct. 2010. [Online]. Available: http://www.named-data.net/

[3] G. Carofiglio, V. Gehlen, and D. Perino, "Experimental Evaluation of Memory Management in Content-Centric Networking," in *Communications (ICC), IEEE International Conference on*, june 2011, pp. 1 –6.

[4] L. Wang, A. Hoque, C. Yi, A. Alyyan, and B. Zhang, "OSPFN: An OSPF Based Routing Protocol for Named Data Networking," University of Memphis and University of Arizona, Tech. Rep., Jul. 2012.

[5] V. Jacobson, D. K. Smetters, N. H. Briggs, M. F. Plass, P. Stewart, J. D. Thornton, and R. L. Braynard, "VoCCN: Voice-over Content-Centric Networks," ser. ReArch '09. ACM, 2009, pp. 1–6.

[6] C. Yia, A. Afanasyevb, I. Moiseenkob, L. Wangc, B. Zhanga, and L. Zhangb, "A Case for Stateful Forwarding Plane," University of Arizona, University of California, Los Angeles and University of Memphis, Tech. Rep., 2012.

[7] A. Baid, T. Vu, and D. Raychaudhuri, "Comparing Alternative Approaches for Networking of Named Objects in the Future Internet," in *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conferenceon*, march 2012, pp. 298 –303.

[8] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A Data-Oriented (and beyond) Network Architecture," in *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '07. ACM, 2007, pp. 181–192.

[9] A. Carzaniga, M. Papalini, and A. L. Wolf, "Content-based Publish/Subscribe Networking and Information-centric Networking," in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, ser. ICN '11. ACM, 2011, pp. 56–61.

[10] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S., and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, Apr. 2008.

[11] D. Mattos, N. Fernandes, V. da Costa, L. Cardoso, M. Campista, L. Costa, and O. Duarte, "OMNI: OpenFlow MaNagement Infrastructure," in *Network of the Future (NOF), 2011 International Conference on the*, nov. 2011, pp. 52 –56.

[12] A. Bianco, R. Birke, L. Giraudo, and M. Palacin, "OpenFlow Switching: Data Plane Performance," in *Communications (ICC), IEEE International Conference on*, may 2010, pp. 1 –5.

[13] N. Fernandes, M. Moreira, I. Moraes, L. Ferraz, R. Couto, H. Carvalho, M. Campista, L. Costa, and O. Duarte, "Virtual Networks: Isolation, Performance, and Trends," *Annals of Telecommunications*, vol. 66, pp. 339–355, 2011.

[14] "Future Internet Testbed with Security," 2012. [Online]. Available: http://www.gta.ufrj.br/fits/

[15] C. E. Rothenberg, M. R. Nascimento, M. R. Salvador, C. N. A. Corrêa, S. Cunha de Lucena, and R. Raszuk, "Revisiting Routing Control Platforms with the eyes and muscles of Software-Defined Networking," ser. HotSDN '12. ACM, 2012, pp. 13–18.

[16] J. Mudigonda, P. Yalagandula, M. Al-Fares, and J. C. Mogul, "Spain: Cots data-center ethernet for multipathing over arbitrary topologies," in *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, ser. NSDI'10, 2010, pp. 18–18.

[17] M. D'Ambrosio, C. Dannewitz, H. Karl, and V. Vercellone, "MDHT: a Hierarchical Name Resolution Service for Information-centric Networks," in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, ser. ICN '11. ACM, 2011, pp. 7–12.

[18] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, and I. Stoica, "ROFL: ROuting on Flat Labels," ser. SIGCOMM '06. ACM, 2006, pp. 363–374.

[19] A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM: NDN simulator for NS-3," University of California, Los Angeles, Tech. Rep., 2012.

[20] "CCNx Project ," 2011. [Online]. Available: http://www.ccnx.org/