# Routing in Information Centric Networks

Maximilian-Dominik Robl, Stefan Liebald*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: ga94kow@mytum.de, liebald@net.in.tum.de

***Abstract*—ICN is a new approach to change the current Internet architecture. In terms of locating and retrieving it differs a lot from the IP based internet. This paper gives n insight into the architecture of ICN. Also, this paper provides explanations of routing and naming in two examples of ICNs, NDN and NetInf. At the end of this paper, both approaches will be compared , especially in terms of routing and naming.**

***Index Terms*—software-defined networks, named data network, network of information, information centric network, routing**

## 1. Introduction

The current Internet architecture is based on the IP protocol. In IP a client is building a connection to a server, through this tunnel the publisher can send the requested data. This approach of connecting is similar to the telephone service where two persons are connected to be able to talk with each other. The IP architecture is from the year 1981 and oriented on a client to client communication. Over the last decades, the internet shifted from the usage of two clients communicating with a few users to a worldwide interconnected network where everyone is searching for specific data and not user.
With services like YouTube, Google, Netflix, Spotify, and co. many users request the same data from the same source, e.g. a live stream 100.000 people are watching.
Information centric networking (ICN) is the idea to replace IP with an architecture covering today's use cases of the internet. Providing an architecture that provides a client to data network. ICN attempts to create an architecture that doesn't need to search for the publisher but simply for the name of the needed data.
This paper focuses on routing names of the requested data and its naming, which makes this possible. For this we give you a brief introduction of ICN, especially naming and routing. This paper provides the basics of ICNs and explains the architectures NDN and NetInf based on this knowledge.

## 2. Routing Difficulties in ICN

The main idea of ICN is routing by the names of the requested data instead of routing by the addresses of the hosts [1]. Routing in general has difficulties. For ICN some of these difficulties are naming of the data (section 2.1) and occurring packet loss. If you download a file you download many packets. It can be enough that one byte is corrupted in your packet in any routing point so that you have to download this packet again. The problem is that instead of just going back to the last healthy file it instead the routing goes completely back to the server. This problem can be better managed with an ICN architecture, since most ICN architectures provide caching for their network nodes. Caching in this case provides the option to retrieve the lost packet from the last network node instead of the complete network path to the host.

### 2.1. Naming

An ICN architecture needs a suiting naming scheme, because it's fundamentally for the routing. Routing resolves the request with the name of the data until a node matching this name. Naming has some difficulties to regard. The names of the data in the network must be globally unique. [2] If the user wants to resolve this data the name must be clearly determinable to ensure that the user gets exactly the requested data. Another difficulty is the forwarding strategy in combination with the naming. For some architectures like NDN the forwarding strategy depends on the naming. Naming can also be human-readable or not. This is a trade-off that an architecture or the software engineer has to determine. Names in ICNs must be location independent. [3] This means that the names don't change with the environment they are saved. Names in ICNs mostly follow two naming schemes, first a structured or hierarchical naming scheme and second a structure-less or flat naming scheme. There is also a hybrid naming scheme which combines both naming schemes. Hierarchical names have a structure that can be interpreted similar to directories in operating systems. Flat names are missing structure.

### 2.2. Routing and Forwarding

Routing is one of the most important parts of an ICN since it significantly distinguishes from the current Internet architecture. Routing highly depends on the naming of the given ICN approach. In this paper, we discuss two routing schemes, name-based routing, and name resolution. Name-based routing works by using the name of the requested data to resolve the next hop in the network. Name resolution mostly works with a name resolution service, which returns some possible next hops in the network based on the committed name. A name


doi: 10.2313/NET-2019-10-1_03

resolution service can have different approaches [4], [5]. Routing consists of two main steps, locate the data by forwarding request and sending the requested data back to the subscriber. Every architecture can freely choose the way to design these steps. Especially the second step could still use an IP protocol since in ICNs forwarding the request should be based on names.

## 3. ICN Architectures

### 3.1. NDN - named data networking

NDN first appearance was in a google tech talk in 2006 by V. Jacobson. NDNs routes are lying in the earlier project content centric network which also was accompanied by V. Jacobson [6]. As an ICN paradigm, NDN takes a prominent role within the broader field of all ICN architectures. ICN is using various networking technologies below the waist for connectivity, including, but not limited to, IP [7]. Furthermore, the NDN architecture can be an independent routing architecture but includes the support to be built on top of IP for better integration in today's internet [8].
An important part of the architecture is the hierarchical namespace that NDN provides. It helps with name-based routing through its URL like structure [9]. The rough routing works by a client requesting data by sending interest packets that include names of the desired data. The NDN network then routes the interest packet forward until a node in the network that holds a copy of the requested data is found. The data packet is then sent back by this node. NDN as like most ICNs supports caching of data [10]. This is essential to give the routing paradigm the option to just send back a cached copy instead of always resolving to the original publisher. Also, the data objects should be independent from the location they are cached, but due to the idea of ICN routing by names, this applies for NDN.

**3.1.1. Naming.** Names in the NDN architecture are organized hierarchical. The names have a tree-like structure similar to URL, organized similarly to the directories in operating systems. E.g. Michi has a directory which contains projects and in this directory are another two directories raspberry pi and Arduino. In NDN we can request all data in these directories by using the prefix /Michi/projects/raspberrypi if Michi chose this organizational structure.
Names in NDN can be human readable like in the example but doesn't have to. Each component can be every format, a hash value would also be possible. It's a trade-off between human-readable and length of the name. The decision is to be made when implementing an NDN network. A human-readable name can have the advantage for the client to keep track when reading the messages, but also can lead to undesirable overhead. Also, global uniqueness is a necessary requirement which will be affected by the decision of the trade-off.
NDN uses a named-based forwarding design where forwarding is dependent on the naming structure. For resolving the names NDN uses longest prefix matching. E.g I'm searching for the name /Michi/project in the network. The forwarding protocol would first search for the prefix /Michi/, then for the prefix /project/ and would resolve to the name that fulfills the most prefixes. Important is that it will just route to identical prefixes. In the case of NDN, there is some specialty if I search for /Michi/projects the network will route for this name and return every data which shares exactly this prefix like /Michi/projects/raspberrypi and /Michi/projects/arduino. The names in NDN are strings with a flexible length. This gives us many possibilities to name our objects ensuring less overlapping of names and thus likely securing names to be globally unique. NDN is using pending interest tables storing object names and their location. Due to the flexible names they can easily get very large, resulting in a slower look up and overall network [11], [12].

**3.1.2. Routing and Forwarding.** Routing in NDN works with forwarding packets containing interests or data in the network. An interest is a request message forwarded as a packet. E.g. a client requests an information object with the name "/Michi/projects/" this would be sent as interest and the network would return a packet consisting of the data. The routing itself is resolved with longest prefix matching in the network. An item is found if the interests name exactly consists of an arrived node or if a prefix of the interests name consists.
NDN network architecture consists of special routers named content routers (CR). The content router extends a common router by providing three data structures, the forwarding table (FIB), the pending content store (CS) and the interest table (PIT). The forwarding table saves all the data to forward an interest. This is done hop-by-hop, the FIB can only forward an interest one step to the next router. This structure at least holds a column with names and one with fitting routing points. The content store is the cache of the router. The router can cache data traveling through it. The pending interest table saves all incoming interests. Only active interests which are not currently resolved are saved. If the interest found it's requested data and returns to the CR it will be deleted from the table. If a client requests a name that is already in the list it will be added in the PIT but not forwarded. The usage of a PIT in combination with the CS allows several people requesting data without resolving it several times. This increases the speed of the network and decreases the workload on the host because the host has to deal with fewer requests.
Figure 1 is an example of the routing an interest that is requested [7]. The routing consists of locating the data and returning the data to the client.
Beforehand the network starts with an empty FIB in the very first start. Forwarding is impossible in NDN with an empty FIB. The CRs must first set up the FIBs by searching for other CRs and information about the network. This problem is called "bootstrapping" and is used in today's internet, e.g. the ARP protocol is a way to resolve to bootstrap in a local network. NDN supports different protocols one is OSPF [6].
Step 1 to 3 in figure 1 shows the sending of an interests message for the data named **/aueb.gr/ai/new.htm** to the network and locating the data. In the first step, the clients send his interest packet to the closest CR. If a CR is obtaining an interest it always executes the following steps.

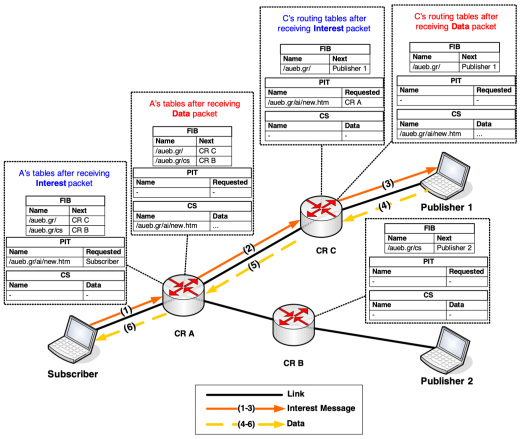
First, the CR checks it's CS containing an item with the exact name. If the item is found the CR just sends back a packet containing the data. If the item isn't in the CS the CR starts checking its PIT. In the PIT the CR checks if a request of the item already arrived independently from the requester's ID. If already a request for the exact item exists the CR aggregates both requests in the PIT and waits for the returning interest. If there is also no entry in the PIT the CR starts checking its FIB. Checking the FIB is done by using longest prefix match, e.g. the FIB entries for the following interest are /aueb.gr/ for CR2 and /aueb.gr/ai/ for CR3 the interest message would be forwarded to router CR3 because more prefixes match. This just applies if all prefixes are given a match. If there is just one prefix existing in the name of the FIB, but not in the interesting name the entry will be ignored.
In step 3 the CR finds the name of the interest in it's CS. Step 4-6 shows the retrieving of the data to the subscriber. In step 4 the current CR sends back the data to the last CR that has sent the interest. In the following steps, the CRs check their PIT every time they receive a data packet. The data is sent back to every client that requested the data. The CR checks all entries for the data objects name, sends the item to all subscribers linked to that name and deletes the interests in the PIT. NDN supports different types of caching strategies, thus the CR has the option to cache the data in every traveled CRs content store [10].

## 3.2. NetInf - Network of Information

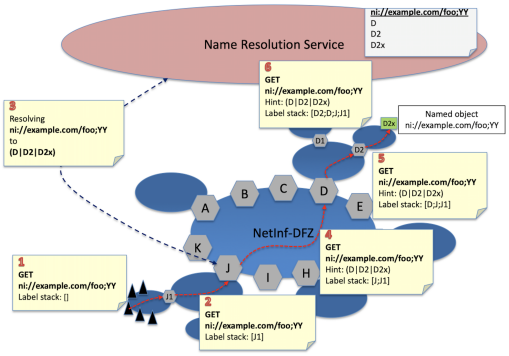
The network of information paradigm is one of the projects funded by the fp7 program of the EU and also part of the funded SAIL program. The principle of this paradigm is like in the most ICNs that the first order is accessing information via named data objects (NDO). NDOs are split into two parts one part is the name in a common format and the other part is the actual object in a common data structure [14].
This paradigm is said to have high support for migration. One part is playing the convergence layers (CL) which help the NetInf to be built on top of an existing routing or forwarding technology. Convergence layers are placed between the two layers and help them to communicate with each other. This helps to support a broad variety of different implementations and is also the idea of the creator, to create a paradigm which is very open in its implementation. Another way to achieve this is for the nodes to focus on minimal common node requirements to also be broadly applied to different types of networks. There is one naming format that all nodes understand and one format for representing the NDOs and optional metadata [13].
NetInf has its own protocol which is based on a few different messages. These messages are GET, PUBLISH and SEARCH. The GET message is used if I exactly know the name of the requested item, the PUBLISH message is used to advertise my information object and the SEARCH message is used to find an item by using keywords. With the SEARCH message, NetInf supports searching for information objects with keywords. NetInf nodes can implement the same request and response forwarding logic, transport and caching strategies for different networks that they are attached to [13]. Due to the ICN architecture and routing by names the routing is location independent and late-binding is supported, which results in the capability of caching information objects in the nodes of the network. NetInf supports on-path and off-path caching. The architecture of NetInf combines some design elements that are present in the NDN [6] and the PURSUIT [15] architecture. E.g. the possibility of routing on object names like in NDN and the idea of using a name resolution service like in PURSUIT. This will be described more precisely in section 3.3.2.

**3.2.1. Naming.** The namespace in NetInf is flat-ish. This means NDO names in NetInf can be flat, but names can also contain a hierarchy in their authority part. The authority part comes from the URI structure that names in NetInf fulfill. This structure was set by the SAIL project (the project with all papers and deliverables can you find here: https://sail-project.eu/deliverables/index.html) and has been registered as permanent URI schemes. A name in NetInf could be named like **ni://example.com/foo;YY** Considering the comparison of names NetInf names are flat [13].
An advantage is that a flat namespace provides better name persistent, due to its independence from organizational structure. If my object name in a hierarchical namespace is something like /de/user/data and I change this structure the name should be changed as well. This case won't happen with a flat namespace. A flat namespace also has the advantage of separating tussle over trademarks from unique data naming [13]. Uniqueness can be a problem in a hierarchical structure if two users are quite similar with also a similar organizational structure. With a flat namespace which is based on hashes, the naming can rely on statical uniqueness. In the case of a rare name collision, this can be handled as an error by the NRS. A disadvantage of a flat namespace can be the capability of aggregating names based on a hierarchical name. Yet naming in NetInf isn't completely flat, in the case of routing the names can be considered to be hierarchical. So it comes that in terms of routing NetInf supports name-based routing as well as naming resolution, which will be part of section 3.3.2. For name-based routing, it is also supported to use longest prefix match like in NDN.
Regarding naming scheme NetInf is using a common naming scheme which supports multiple "pluggable" cryptographic algorithms and representations [13]. The SAIL project registered two URI naming schemes that were designed for the NetInf paradigm and they are available to use for this architecture. The two registered naming schemes are "ni" and "nih". The scheme "ni" allows the inclusion of hashes in URIs in a structured manner [13]. On the other hand, "nih" is derived from "ni" by removing all optional features and ensuring the remaining structure was unambiguous when spoken [13].

**3.2.2. Routing and Forwarding.** As mentioned earlier the NetInf protocol is based on messages. These messages are used for request and response forwarding. As a small recap the messages are GET, SEARCH and PUBLISH, all of these also have a response part GET-RESP, SEARCH-RESP, and PUBLISH-RESP that are described in the protocol of NetInf. The routing of the message requires




(a) Figure 1: The NDN architecture. CR stands for Content Router, FIB for Forwarding Information Base, PIT for Pending Interest Table, CS for Content Store. [7]




(b) Figure 2: NetInf inter-domain scenario [13]

routing information to decide how to forward on each hop. The GET request is routing by the name of the requested NDO.

In NetInf name-based routing and name resolution is supported, which results in the possibility of a hybrid approach where either scheme can be freely chosen. Name resolution in NetInf works with routing hints. Routing in NetInf supports a component which is called routing hint. Routing hints indicate where to find copies of the object [13]. Routing hints are locators for lower layer hosts. The routing in NetInf forwards and resolves the request for objects as well as response messages. Here I also separate the routing in three different parts 1) bootstrapping 2) locate the NDO and 3) Return the information object. NetInf supports a big network like IP which exists of each other different network. All these networks also can have different routing requirements and thus need different routing protocols.

NetInf uses a hybrid request routing/forwarding scheme, which combines the possibility of using a name-based routing and name resolving. This is implemented by integrating pure name-based routing with name resolution aspects. That means that the routing paradigm tries to use name-based routing to forward the request and switches to a name resolution service if it can't find a hint for forwarding the next hop. Routers in a NetInf network also support a data structured called label stack similar to the PIT in the NDN architecture. The label stack stacks every name of the involved routers to easily travel back the route for the response message afterward. The name-based routing can use a pattern matching or as in NDN a prefix-matching approach. The NRS is then the system with a freely chosen algorithm that returns a set of routing hints.

For illustrating the routing scheme it needs a network to run in. In NetInf the creators assumed a network which is quite similar to today's internet. It is expected to have one global network which is expected of using just one routing scheme. This global network consists of different edge domains binding their network with the global network (The global network here takes the part of being the DFZ). Every edge domain can internally decide on NetInf routing/forwarding, adapted to the domains need. The created network relies on the hybrid routing approach. If a client will now request the object **ni://example.com/foo;YY** in the network the routing consists of 6 steps (Figure 2):

- Step 1: The clients sends a GET message to it's the closest network with the content ni://example.com/foo;YY as the name of the NDO.
- Step 2: The request gets forwarded to the next node. This can be done by name-based routing. In every step, the last router will be saved in the label stack.
- Step 3: The node is lacking routing information. It consults an NRS and gets a set of routing hints back. These will be added to the GET message.
- Step 4: Following the set of routing hints by performing the next hop the request reaches the next node. The set can also be just one element or there could be several nodes in between which are forwarded hop by hop with help of the routing hints.
- Step 5: The current node belongs to a different provider and thus has it's own routing/forwarding scheme. Our set also possesses all the necessary nodes to route forward based on it.
- Step 6: The requested node reaches a node holding a copy of the requested NDO.

Returning the NDO is done by the node holding the NDO sending a RESPONSE message containing the NDO and the label stack. This message is then routed step by step dismantling the label stack. While routing backward NetInf allows caching the NDOs [13].

## 4. Architecures differences and conclusion

In this paper we summarize the ICN approach and two of its current architecture. Both architectures can be implemented completely independent to the current IP infrastructure, still, both architectures support to be built on top of an IP network, the NDN architecture with the option to be built on top of existing routing architectures and NetInf with its CLs.

In terms of routing NDN provides a hierarchical structure


doi: 10.2313/NET-2019-10-1_03

which has an enormous impact to its routing. NetInf has a flat-ish naming approach, which combines the hierarchical naming scheme with a flat one. Due to the flat-ish naming structure and hashing support, NetInf ensures global uniqueness with statistical uniqueness, while NDN uses its prefixes to ensure this global uniqueness.
The NDN naming approach though can due to its prefixes have very long names. Also the approach of routing with names create more entries to look up. With IP adresses a host had one adress and contained several data objects. In ICN we route for the data objects, which are in a larger number. This can result in the FIB and PIT to be very large, consuming storage space and decrease the lookup speed. The NetInf architecture uses an NRS which tends to be the bottleneck of the network. Since the NRS will be called by several access points to request routing information and has to store more entries in the table than the NDN approach. Also the NRS is the location were some sort of table like the FIB and PIT is used to resolve the next hops for the request. And the NRS is potentially linked to a larger network than a CR, which can also lead to a larger table.
Both architecture support caching in their nodes, which in most cases can reduce the number of traveled nodes and thus the lookup speed, because the data can easily retrieved by these nodes. Also, packet loss can be handled way better in both architectures. If packet loss occurs the client can request the data from the last node holding a copy. In case of packet loss, the data packets mostly have to travel fewer nodes as in the first request making the network faster.

## References


[1] M. Awais and M. A. Shah, "Information-centric networking: a review on futuristic networks," in *2017 23rd International Conference on Automation and Computing (ICAC)*. IEEE, 2017, pp. 1–5.

[2] M. S. Akbar, K. A. Khaliq, R. N. B. Rais, and A. Qayyum, "Information-centric networks: Categorizations, challenges, and classifications," in *2014 23rd Wireless and Optical Communication Conference (WOCC)*. IEEE, 2014, pp. 1–5.

[3] M. F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and B. Mathieu, "A survey of naming and routing in information-centric networks," *IEEE Communications Magazine*, vol. 50, no. 12, pp. 44–53, 2012.

[4] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, 2012.

[5] X. Jiang, J. Bi, G. Nan, and Z. Li, "A survey on information-centric networking: rationales, designs and debates," *China Communications*, vol. 12, no. 7, pp. 1–12, 2015.

[6] 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.

[7] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A survey of information-centric networking research," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 1024–1049, 2014.

[8] 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.

[9] H. Yuan, T. Song, and P. Crowley, "Scalable ndn forwarding: Concepts, issues and principles," in *2012 21st International Conference on computer communications and networks (ICCCN)*. IEEE, 2012, pp. 1–9.

[10] W. Shang, A. Bannis, T. Liang, Z. Wang, Y. Yu, A. Afanasyev, J. Thompson, J. Burke, B. Zhang, and L. Zhang, "Named data networking of things," in *2016 IEEE first international conference on internet-of-things design and implementation (IoTDI)*. IEEE, 2016, pp. 117–128.

[11] Y. Yu, A. Afanasyev, and L. Zhang, "Name-based access control," *Named Data Networking Project, Technical Report NDN-0034*, 2015.

[12] N. L. Van Adrichem and F. A. Kuipers, "Globally accessible names in named data networking," in *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2013, pp. 345–350.

[13] D. Kutsher *et al.*, "Final netinf architecture," https://sail-project.eu/deliverables/index.html, 2013.

[14] C. Dannewitz, D. Kutscher, B. Ohlman, S. Farrell, B. Ahlgren, and H. Karl, "Network of information (netinf)–an information-centric networking architecture," *Computer Communications*, vol. 36, no. 7, pp. 721–735, 2013.

[15] N. Fotiou, P. Nikander, D. Trossen, and G. C. Polyzos, "Developing information networking further: From psirp to pursuit," in *International Conference on Broadband Communications, Networks and Systems*. Springer, 2010, pp. 1–13.