Computer Communications 36 (2013) 721–735

Contents lists available at SciVerse ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Network of Information (NetInf) – An information-centric networking architecture ☆

Christian Dannewitz [a], Dirk Kutscher [b,*], Börje Ohlman [c], Stephen Farrell [d], Bengt Ahlgren [e], Holger Karl [a]

[a] University of Paderborn, Germany
[b] NEC Laboratories Europe, Germany
[c] Ericsson Research, Sweden
[d] Trinity College Dublin, Ireland
[e] Swedish Institute of Computer Science, Sweden

ARTICLE INFO

Article history:
Available online 27 January 2013

Keywords:
Data communication networks
Information-centric networking
ICN
Network of Information
NetInf

ABSTRACT

Information-centric networking (ICN) is a promising approach to networking that has the potential to provide better – more natural and more efficient – solutions for many of today's important communication applications including but not limited to large-scale content distribution. This article describes the Network of Information (NetInf) architecture – a specific ICN approach that targets global-scale communication and supports many different types of networks and deployments, including traditional Internet access/core network configurations, data centers, as well as challenged and infrastructure-less networks. NetInf's approach to connecting different technology and administrative domains into a single information-centric network is based on a hybrid name-based routing and name resolution scheme. In this article, we describe the most important requirements that motivated the NetInf design. We present an architecture overview and discuss the different architecture elements such as naming, message forwarding, caching, and a name resolution service (NRS) in detail. As efficient caching and a scalable NRS are two main success factors, we present an evaluation of both elements based on a theoretical analysis, complemental simulation results, and prototyping results. The results suggest that a scalable NRS for 10^{15} and more objects with resolution latencies (well) below 100 ms is possible, implying that a global Network of Information that removes the need for today's application-specific overlay solutions is feasible.

1. Introduction

According to recent predictions [1], global IP traffic will approach 966 exabytes per year in 2015. Much of this traffic stems from various forms of video (TV, video on demand (VoD), Internet Video, and peer-to-peer (P2P)) that continue to account for approximately 90% of global consumer traffic by 2015. Global mobile data traffic is expected to increase by a factor of 26 from 2010 to 2015. Likewise, machine-to-machine (M2M) traffic is expected to grow 22-fold from 2011 to 2016, with a compound annual growth rate of 86%, and the number of mobile-connected M2M modules is expected to grow 5.8-fold between 2011 and 2016, reaching 1,906 million.[1]

The best current practice to manage this growth in terms of data volume and devices is to employ application-layer overlays such as CDNs, P2P applications, and M2M application platforms that cache content, provide location-independent access to data, and optimize its delivery. In principle, such platforms provide a service model of *accessing named data objects (NDOs)* (replicated web resources, M2M data in data centers) instead of a host-to-host packet delivery service model.

However, since this functionality resides in overlays only, the full potential of content distribution and M2M application platforms cannot be leveraged as the network is not aware of data requests and data transmissions, leading to:

- data having to travel sub-optimal routes depending on the overlay, and not the Internet layer, topology;
- multicast and broadcast features of wireless networks cannot be leveraged, i.e., request and delivery for the same object have to be made multiple times;
- overlays typically require a significant amount of infrastructure support, e.g., authentication portals, content storage, and applications servers, making it often impossible to establish local, direct communication;

☆ The research leading to these results has been conducted in the SAIL project and received funding from the European Community's Seventh Framework Program (FP7/2007-2013) under Grant Agreement No. 257448.

* Corresponding author. Tel.: +49 5251 60 5385; fax: +49 5251 60 5377.
E-mail addresses: cdannewitz@upb.de (C. Dannewitz), Dirk.Kutscher@neclab.eu (D. Kutscher), Borje.Ohlman@ericsson.com (B. Ohlman), stephen.farrell@cs.tcd.ie (S. Farrell), bengta@sics.se (B. Ahlgren), hkarl@upb.de (H. Karl).

[1] <http://www.cisco.com/web/solutions/sp/vni/vni_mobile_forecast_highlights/index.html>.

0140-3664/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.comcom.2013.01.009

- the network not being aware of the nature of data objects and thus being unable to manage access and transmission (without layer violations); and
- many applications providing their own approach to caching, replication, transport, authenticity validation (if at all), although they all share the same model of accessing named data objects in the network.

Information-centric networking (ICN) is a networking approach that provides access to named data objects as a first-class networking primitive and leverages object naming and ubiquitous in-network caching to provide more efficient and robust networking services than current (overlay) approaches allow.

ICN can be seen as a generalization of CDN technologies – however, ICN is not limited to media distribution scenarios. Other scenarios could include, e.g., those where repeatedly accessing and modifying local caches gives advantages over remote server accesses. To illustrate, consider a group of collocated users (with good local connectivity, e.g. WiFi, and weak Internet connectivity) collaboratively editing documents stored in a version control repository (e.g., Subversion[2]) or a Wiki. Conventionally, all updates would go via the slow Internet link only to be redistributed locally again. With an ICN infrastructure, the relevant objects are cached and modified locally, with local copies being created opportunistically at nearby, even optimal, places, without the application having to worry.

Various proposals for such ICN systems have been made (cf. Section 6). In this paper, we give an introduction to the NetInf architecture that we describe to some level of detail. For more detailed descriptions we refer the reader to the SAIL project deliverables NetInf proposal [2–4]. Compared to other ICN proposals, NetInf offers the following combination of key features. The basic NetInf naming and security model does not require (but can use) naming authorities or a public key infrastructure (PKI). NetInf has a concrete content model based on the widely deployed MIME standard and also offers specific search primitives that provide a link from search terms to object names. NetInf offers a flexible object retrieval approach combining name resolution and name-based routing. NetInf can use either one separately in different parts of the network; it can even mix them into a hybrid scheme by switching between them on a hop-by-hop basis. By adjusting the object retrieval method, NetInf can adapt to environments with very different requirements like global connectivity, network mobility [5], and Delay-Tolerant Networking (DTN). This creates interesting scaling properties – NetInf can scale from an ad hoc network of two directly connected user nodes to a large infrastructure network with dedicated infrastructure nodes, e.g., for global name resolution.

NetInf simplifies deployment and migration as it can run as an overlay on top of the existing network infrastructure during the migration phase. First, the most beneficial applications like media distribution could be migrated to NetInf with other applications following later.

The rest of this paper is structured as follows: Section 2 presents a set of issues with today's Internet and a set of requirements that a future network architecture should meet. Section 3 provides an overview of NetInf's design principles and the NetInf architecture. Section 4 discusses details about the architecture elements, protocol stack, node architectures, and the network architecture. In Section 5, we evaluate our work, and in Section 6, we compare it to existing related work.

[2] <http://subversion.apache.org/>

2. Problem statement and requirements

In this section we describe the set of requirements that we see as fundamental for any ICN approach. In the following two sections we explain how NetInf fulfils them.

2.1. Scalable and efficient content distribution

The tremendous traffic increase described above necessitates content dissemination that scales in traffic, latency, cost, and energy usage. There are two main developments: content distribution networks (CDNs) and P2P networking, both moving towards an information-centric model. Yet there are a number of issues. P2P networks suffer from suboptimal peer selection that leads to expensive inter-provider traffic and heavy loads on weak access links (e.g., mobile and cable networks), lack the ability to effectively use in-network storage, and suffer from incompatibility of the many existing systems as P2P networks are generally an application-specific solution. CDNs extend the underlying network infrastructure by interpreting uniform resource identifiers (URIs) and Domain Name System (DNS) names to access cached copies of content. However, CDNs are limited to a set of explicitly created copies and cannot benefit from all available copies, e.g., on user devices or other servers. Moreover, CDNs are only an add-onto today's Internet; they solve a limited subset of the general problem and only for a restricted group of customers: Since a CDN requires dedicated setup, configuration, and customer relationships, typically only large players (e.g., large content providers) use it. Hence, small customers most in need of support (e.g., to protect their small infrastructure against flash crowd effects) are the least likely to benefit from this approach. On the contrary, a solution is needed that eases experiments with new services for new players.

2.2. Persistent, location-independent naming

Today's Internet architecture lacks a persistent, location-independent naming scheme for content. Most content URIs in today's network directly point to a specific location; they are *locators*. As a result, the name-object binding can easily break, for example, when an object is moved, the site changes domain, or the original site is unreachable for some reason. Moreover, replicas of an object at different web servers are accessible only under different URIs and essentially appear as different objects. This complicates, e.g., efficient network utilization and caching.

2.3. Data availability

In today's Internet, data access in challenging network conditions with spotty connectivity, network disruptions, or large network delays is difficult or impossible. This is often caused by inaccessibility of global infrastructure like DNS, inability to use locally available copies, and a required persistent connection between hosts. However, not all applications actually do require seamless end-to-end communication.

2.4. Security

Today's security is host-centric, i.e., based on securing channels between hosts via encryption, e.g., transport layer security (TLS) and trusting servers via authentication. As a consequence, a client cannot trust a copy that is received from an untrusted location although the copy might be authentic. To address this problem, workarounds like separately published fingerprints of the data (e.g., MD5) are sometimes used. A future security model should provide an architecturally sound approach that works for any data copy independent of its location. The model should enable ubiquitous caching while retaining data integrity and authenticity, something that the current model does not provide. In addition, it should not have to rely on trusted third parties for data integrity checking. In contrast, with today's model, trust is practically transfered to the software vendor that compiles the set of trusted authorities.

2.5. Mobility and multihoming

Due to the host-based nature of today's Internet, mobility and multihoming of nodes and networks is a challenging problem that requires managing end-to-end data flows (e.g., with handovers) and choosing which path and/or interface to use. Several solutions like mobile Internet protocol (IP) have been proposed and implemented, however, they suffer from disadvantages like increased stretch and complexity due to the inherent problems of the host-centric approach.

3. Architectural overview of the Network of Information

NetInf is a *networking approach* that provides *access to named data objects (NDOs)* as a first order networking primitive, i.e., the primary service of *NetInf nodes* is the forwarding of NDO requests and the transfer of the corresponding objects (or pointers to copies). This fundamental service model is common to many ICN approaches. In the following, we first describe a set of important NetInf design principles and then provide an architecture overview.

3.1. Design principles

3.1.1. Accessing named data objects

Accessing named data as a first order principle implies that NetInf nodes (the nodes in a NetInf network) do generally not communicate on the basis of network or host addresses, but instead use NDO names to identify objects independent of network location. The unique, location-independent naming enables ubiquitous replication and caching of NDOs in the network. A NetInf NDO is an Application Data Unit (ADU), i.e., it can be a complete object or any application-define fragment of an object, and it consists of its name in a common format and the actual object in a common data structure (see below). The main service is to forward NDO requests to appropriate copies and transfer objects back. Forwarding can be done using routing on the object names (so-called *name-based routing/forwarding*) or with the help of *name resolution services*.

3.1.2. Minimal common node requirements

To broadly apply to different types of networks and deployments, NetInf only makes minimal node requirements:

Naming format: There is one common NDO *naming* format that all nodes understand.

Object model: There is one format for *representing* NDOs and optional metadata. All nodes can process these structures. The format allows for, e.g., application-specific extensions; not all nodes need to understand all extensions or metadata formats.

NetInf protocol: There is one simple protocol that all nodes implement. The NetInf protocol is message-based; it provides requests and responses for `PUBLISH`ing, `GET`ting, and `SEARCH`ing for NDOs. These requests employ the common naming format and the common object model.

3.1.3. Generic NetInf nodes

Based on these minimal requirements, NetInf nodes can provide different functions such as forwarding requests and responses, caching, and name resolution. The same functionality can be provided by specialized infrastructure nodes for a global network as well as by user NetInf nodes, e.g., in a local context.

3.1.4. Flat namespace

NetInf employs a flat namespace for NDO names, i.e., there is neither topology- nor organization-related hierarchy in names. On the one hand, this limits the ability to aggregate names based on such a hierarchy, e.g., for routing or name resolution. However, methods like explicit aggregation [6] can be used to circumvent this problem. On the other hand, for many hierarchical namespaces, the hierarchy is based on aspects like organizational structures, folder structures, etc. Compared to such hierarchical namespaces, a flat namespace provides better name persistence as it eliminates such interdependencies. Otherwise, e.g., organizational changes would result in name changes. In addition, a flat namespace also has the advantage of separating tussle over trademarks from unique data naming [7].

With properly designed distributed algorithms for name construction, name management is significantly simplified even without a naming authority to assign names. Naming can rely on statistical uniqueness; rare name collisions can be handled as error, e.g., by the NRS.

3.1.5. Name-Data integrity

Name-data integrity validation is NetInf's fundamental object security service for both static and dynamic objects. The common naming format and object model enable data integrity validation by requesters (or any other node). Name-data integrity validation can be performed without infrastructure support like a PKI (employing message digests and/or public key digests as part of NDO names). Not all nodes are required to perform the validation.

NetInf provides additional object security services such as *owner pseudonymity* and *owner identification*, employing public-key cryptography.

3.1.6. Lower layer independence

The NetInf protocol specifies messages for node-to-node communication, their semantics, and corresponding node requirements. Different deployments will use different link layers and underlays – with a variation of services and properties. NetInf accommodates this by *convergence layers (CLs)* that map the conceptual protocol to specific messages, transactions, or packet exchanges in an existing, concrete protocol. A CL provides framing and message integrity for NetInf requests and responses for communication between two nodes as its main service, but specific CLs can provide additional services as depicted in Fig. 1.

For example, NetInf-over-IP would require a CL that encapsulates, and potentially fragments and reassembles, NetInf message for transfer in IP packets and validates message integrity (e.g., by CRC). NetInf-over-Ethernet, on the other hand, could be done with a slim CL, but that restricts choices within NetInf (see below). Fig. 1 depicts a schematic and sample NetInf stack for different CLs/underlays; it also shows that between applications and NetInf, additional functions like request scheduling (for implementing flow control, congestion control and other transport layer functions) can be inserted. With CLs, NetInf runs over quite different types of network links, including uni-directional or heavily delay-challenged links.

While specific CLs *can* provide transport protocol functions (reliability, flow control, congestion control), there is a need for transport layer functions across CL links. In NetInf, such functions

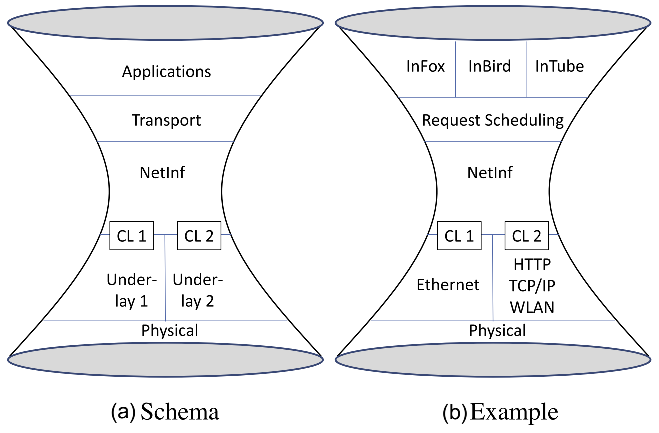

Fig. 1. NetInf protocol stack, assuming a node with two convergence layers over two different underlays.

are implemented on-top of the NetInf layer, e.g., by request scheduling/retransmission decisions.

3.1.7. Routing and name resolution

The NetInf protocol's request and response forwarding service requires routing information to decide to which next hop, over which interface a request should be forwarded. For GET requests, these decisions are generally based on the name of the requested NDO.

NetInf supports both name-based routing and name resolution, i.e., employing in-network name resolution services that can map NetInf names to network or possibly host identifiers in different namespaces, which we call *routing hints*.

Routing hints indicate where to find copies of the object. Different types of routing hints are supported. Routing hints can be lower-layer host locators that can be directly used to retrieve the object via the underlying network protocol (e.g. TCP/IP), protocol-specific routing hints that can subsequently be used to support name-based routing (e.g., by pointing to the next network that can provide the object), or another NetInf name that allows for indirection.

Name resolution plays a major role in NetInf, including NetInf's scalability. In responding to a NetInf GET request, any node may either return the requested object or routing hints that can help the requester to access the object, and any requesting nodes must hence be prepared to receive such routing hints.

Notably, NetInf supports a *hybrid* combination of name resolution and name-based routing where at each forwarding step for a given request, either scheme can be freely chosen.

Not all combinations of underlays, convergence layers, and forwarding mechanisms make sense, however. In particular, an underlay that does not provide a (virtually) fully connected graph combined with a functionally slim CL makes name resolution inapplicable and necessitates name-based routing.

For global connectivity, we suggest a Border Gateway Protocol (BGP)-like routing infrastructure combined with a global NRS infrastructure, i.e., a hybrid approach as described above.

3.1.8. On-path and off-path caching

NetInf supports both on-path caching (on the request/data path) as well as off-path caching. NetInf can integrate any available copy when retrieving data: the original server, redundant copies, as well as replicas stored on user devices (if so permitted). Thereby, NetInf can access the best available copy.

3.1.9. Heterogeneous networks

NetInf's flexibility in convergence layers and routing/forwarding schemes allows custom-tailored configurations for different technologies and different administrative domains. As long as NetInf's minimal node requirements are maintained, the NetInf protocol can be mapped to quite different types of network links – from bi-directional message/packet-based point-to-point links to unidirectional broadcast links and intermittently connected links. NetInf also does not assume anything about specific network system architectures – e.g., terminals, access networks, core networks are flexible notions and there is no assumption on *where a network ends*: NetInf user devices can appear as terminals, but they can also provide caching and store-carry-forwarding, thus extending the network or connecting NetInf networks.

3.1.10. Inherent mobility and multi-homing support

NetInf inherently makes mobility and multi-homing simpler. The main reason for this claim is that when a requester in NetInf makes a request for a NDO it is not requesting it from any specific source, it just a general request towards *the network*. For mobility this means that parts of the NDO can be delivered from different sources (holding copies of the requested NDO), as the requester moves, without a need for requester to be involved or even aware of the changing sources. Regarding multi-homing, the NetInf architecture puts no limits on the number of interfaces used in parallel to send requests and to receive responses. As all object copies are equal, the order of requests and responses is not critical. To use network resources in an optimized way, several request strategies can be used in NetInf. In some network scenario, e.g., a local setup, broadcasting requests on all interfaces might be optimal. In some other network scenario, an NRS-based solution using only a single interface might be preferred.

There are different types of mobility. For client mobility (i.e., moving requester), there is no need to continue using a specific object copy. Instead, alternative copies close to the client's new location can be used. For server mobility (i.e., moving content), updates of the routing information and/or name resolution information in the network are required to ensure that an origin NDOs stays accessible.

For traditional point-to-point services (e.g. a voice call) the need for mobility support will be very different in different parts of the network. As NetInf can support different name resolution mechanisms in different parts of the network, these can be selected taking the dynamics of respective network parts into account. While traditional mobility anchor point mechanisms can be used for some parts, very dynamic environments where whole networks are moving (e.g. trains) can rely on mechanisms like late locator construction (LLC) [5].

3.2. Architecture overview and sample setup

3.2.1. Sample message flow

Fig. 2 shows an example of name resolution, name-based routing, and the hybrid approach in NetInf. The name-based routing (steps A1–A4) forwards a GET request hop-by-hop between NetInf nodes until a cached copy of the NDO is found or the original server (or an alternative server) is reached. If the router does not have enough routing information to perform name-based routing (NBR) in step A2, it can perform a name resolution step (steps A1.1–A1.2) before forwarding the request (step A2) based on the retrieved routing hints, which illustrates the hybrid mode. On the return path, the object can be cached in intermediate nodes for subsequent requests. Alternatively, the initial requester can query an NRS (steps B1–B4) via a GET message to resolve the object name into a set of *routing hints*, in the example simply lower-layer host locators. Subsequently, the routing hints are used to retrieve the

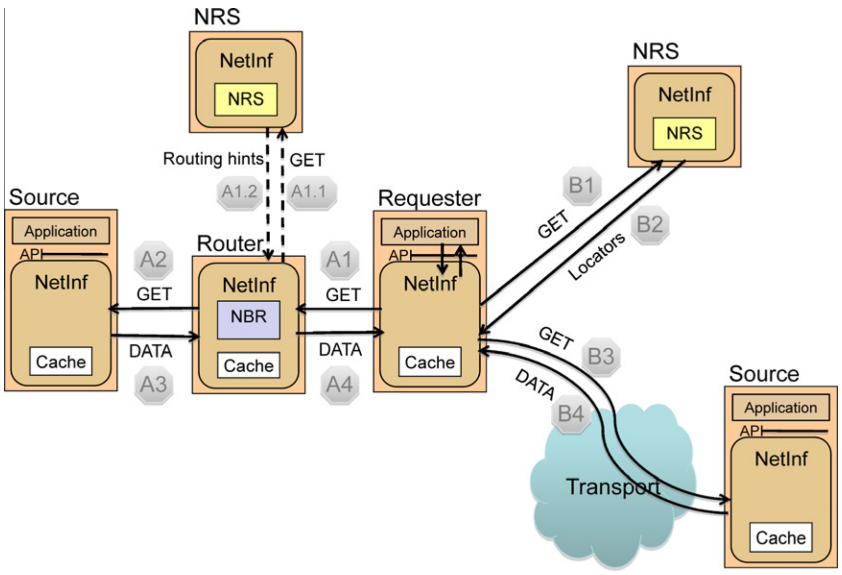

Fig. 2. Example message flow.

object via the underlying transport network, e.g., a legacy IPv4 network.

3.2.2. Sample network setup

Fig. 3 shows an example of a Network of Information. As a simple case, NetInf nodes can be interconnected in an infrastructure-less ad hoc way (Fig. 3, left side) or in a local infrastructure-based network (right side) with dedicated NetInf infrastructure nodes such as an NRS and a NetInf router, possibly specialized for the idiosyncrasies of the local network (e.g., DTN, network mobility). Such local networks can easily be connected to the global Network of Information. This requires updating the local NetInf router with routing information or connecting the local NRS node to a higher-level NRS (publishing any locally registered information at the higher-level NRS if desired).

Such setups are based on interconnecting multiple coexisting NRSes. We expect that network providers typically run their own NRS as this allows to better control the network-internal traffic flow and reduce inter-domain traffic. Each provider can set up a local hierarchy of NRS nodes that matches its network topology. Providers will typically also add NetInf-enabled (on-path and off-path) in-network caches to their networks. Off-path caches are typically connected to NRS nodes to retrieve information about local object popularity and to register cached objects. In addition to in-network caches, NetInf can continue to access today's content servers (e.g., web servers, etc.) as shown in Fig. 3.

Although NetInf can support multiple global NRSes, we expect that a single global NRS will evolve. Fig. 3 shows a hierarchical, topologically-embedded NRS which interconnects smaller, local NRSes. This global NRS consists of all/most other NRS nodes. We estimate [8] that such a global NRS would require about 10^6 NRS nodes to handle 10^{15} objects globally.

In an infrastructure-based Network of Information, user nodes which request objects have two alternative options. They can

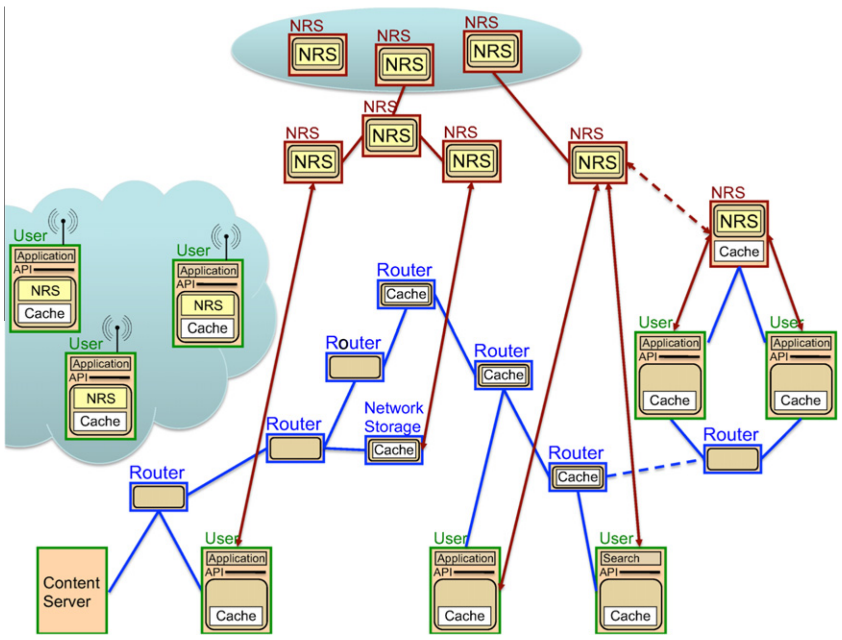

Fig. 3. Sample network setup.

either send the object GET request to a known NRS or use name-based routing to find the next hop NetInf router (which might possibly be well known and the only one for all requests). Both the NRS and the adjacent routers are typically preconfigured at the user node like in today's networks, e.g., via the dynamic host configuration protocol (DHCP) or manually. NetInf routers are placed inside provider networks to forward requests and data internally as well as between domains to perform inter-domain routing/forwarding. Thanks to NetInf's hybrid object retrieval approach, NetInf routers can also contact NRS nodes to resolve NetInf names into a set of routing hints.

4. Elements of the network of Information

The following sections provide a more in-depth discussion of the main elements of the NetInf architecture, including NDOs, naming, security, NetInf communication including convergence layers, the NetInf protocol, and routing and forwarding, caching, name resolution and name-based routing, and inter-domain communication. NetInf application programming interface (API). For a comparison of how these elements relate to those of other ICN approaches we refer the reader to a recently published ICN survey article [9].

4.1. Named data objects

To apply ICN broadly will require agreement on how to structure NDOs. For example, some name-data integrity validation approaches require additional cryptographic material (signatures, keys, certificates) to be attached to the actual object that is distributed and stored in a network. Moreover, objects may have some sub-structure, including object fragments, application-specific metadata, etc.

A well-known object structure as depicted in Fig. 4 and a common naming format (see below) represent cornerstones of NetInf – independent of how NDOs are transmitted and received, the NDO (i.e., its name and data structure) are sufficient for NetInf nodes to forward or otherwise process the NDO. The existence of application-specific extensions (specific data types, meta information, etc.) is explicitly signaled, and the common structure enables access to such extensions. Current NetInf prototypes use MIME as a concrete format, leveraging multi-part messages, message (content) type identification and security services (CMS).

4.2. Basic NDO naming and security

NetInf names serve three different purposes. First, names unambiguously identify NDOs. Second, NetInf names contain security-related information to enable name-data integrity and other advanced security features. Third, names act as keys (in the database sense) for name resolution and routing mechanisms. This section describes the basic NetInf naming and security concept, while the following section describes additional advanced security features of this naming approach.

To foster application development and simplify migration, we have developed a general URI scheme for named information: *the named information (ni) URI scheme* [10]. In the ni scheme, a basic NetInf name contains a hash algorithm and hash value like this: `ni:///sha-256;f40x... JtkGk.` [3]

The ni URI form enables NetInf nodes to optionally verify *name-data integrity*, which is based on the idea of verifying that the received data corresponds to the requested name, assuming that one obtained the correct name beforehand. In its simplest form,

[3] The hash value may be truncated but that is not discussed further here.

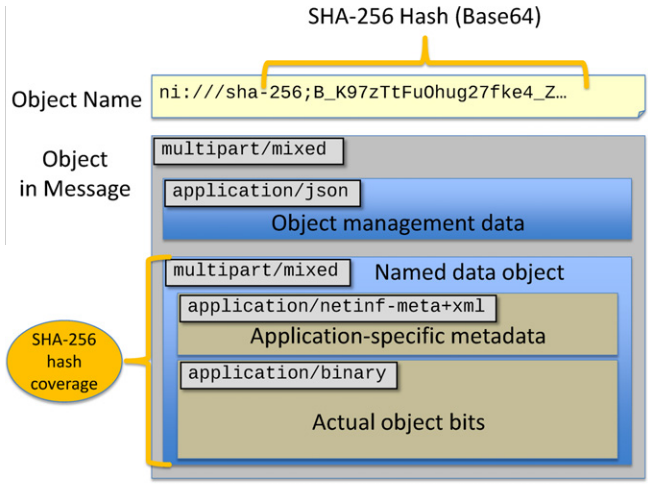

Fig. 4. Conceptual object model.

this is achieved if the hash value in an ni URI is calculated over the form of the NDO that will be delivered by the NetInf protocol. The major benefit of this simple scheme is that it requires no key management at all since there are no keys needed! Yet, basic name-data integrity provides enough security to enable the main ICN benefits (location independence) to be achieved.

Names can contain an authority field to assist in accessing the named object for routing requests or assisting NRS scalability. Names may additionally contain a query string that can hold routing hints or other values used in the NetInf protocol.

There are also binary and human-friendly forms of this name format defined for use, respectively, in more constrained environments and in case it is required to "speak" a name, e.g., over the phone. In addition, there is a well-defined way to map these ni URIs to and from HTTP URLs, via the "well-known" URL scheme [11].

4.3. Advanced naming and security

NetInf supports additional advanced security features, including owner identification and pseudonymity. Details can be found in reference [12]; we briefly cover some of these.

As described so far, the ni URI scheme only provides name-data integrity for *static data*. To do so for *dynamically changing data*, we include a hash of a public key in the ni URI rather than a hash of the NDO. Thus, anything signed by the holder of the corresponding private key can be verified by NetInf nodes. In addition, this use of signatures provides a concept of "ownership" and a means to manage owner pseudonyms.

Owner pseudonymity can express that multiple NDOs belong to the same owner/creator. An owner can build up trust in this pseudonym, allowing users to trust in subsequent content published under it. At the same time, owner pseudonymity allows NDO owners to remain completely anonymous by supporting an unlimited number of pseudonyms per user.[4]

Owner identification can bind the pseudonym to the owner's real-world identity. This optional feature can be done via the use of standard PKI mechanisms, e.g., a Trusted Third Party or a Web of Trust. They are both, however, external systems and not part of the core NetInf architecture.

[4] In a deployed system, incentives (e.g., associated costs) might be required to prevent users from "wasting" pseudonyms to remain a high likelihood of statistical name uniqueness.

4.4. NetInf communication

NetInf nodes use the NetInf protocol, forwarding messages over some lower-layer technology. This may be TCP/UDP/IP-based or may in future be based on non-IP networking technologies, e.g., being realized directly on top of point-to-point links without end-to-end networking capabilities. Semantic gaps to lower layers are bridged by "convergence layers" (using the almost identical term from the DTN architecture [13]; see Ref. [3] for differences between DTN and NetInf convergence layers).

4.4.1. Convergence layer approach

NetInf nodes communicate using convergence layers (CLs) (Fig. 5) that can be connection-oriented or not, uni- or multicast, may encapsulate, fragment and reassemble or even reformat higher-layer protocol data units (PDUs), etc. By abstracting the NetInf protocol from lower layers, CLs ensure that NetInf implementations can be deployed across technologies.

A convergence layer communication is always hop-by-hop between NetInf nodes, i.e., there is no communication over multiple NetInf hops *inside* any convergence layer. A convergence layer does not make use of any NetInf layer node identifiers.

Depending on deployment details, one or more CLs may be used in the overall network. There may be one common CL (that most nodes are expected to support) and a set of less common ones – for instance, for specific access network types.

A NetInf CL does not treat a message PDU as opaque data. It understands structure and nature of the message fields to improve efficiency but it never alters the content. The CL architecture provides marshalling higher "layer" PDUs, exactly reproducing that PDU on the other side of the CL hop; it does so efficiently given the used CL protocol, maintaining message integrity, name-data binding, and other security properties.

4.4.2. Conceptual protocol

A specific CL implements the conceptual NetInf protocol that provides the following fundamental messages and responses:

GET/GET-RESP	The GET message requests an NDO from the NetInf network. A node responds to the GET message if it has an instance of the requested NDO; it sends a GET-RESP that uses the GET message's msg-id as its own identifier to link those two messages with each other.
PUBLISH/PUBLISH-RESP	The PUBLISH message allows a node to push the name (e.g., to an NRS) and, optionally, a copy of the object data and/or object metadata (e.g., to a cache).
SEARCH/SEARCH-RESP	The SEARCH message contains search keywords. The response is either a status code or a multipart MIME object containing a set of metadata body parts, each of which must include a name for an NDO that is considered to match the query keywords.

The details of the conceptual protocol are described in [3].

4.4.3. Routing, forwarding, transport

NetInf forwards and resolves requests for objects as well as forwards the response messages. Different parts of the network can have different routing requirements and, thus, will need different routing protocols, just as we have multiple routing protocols for IP today. Hence, NetInf supports different request/response routing/forwarding mechanisms, such as Open Shortest Path First (OSPF) for local domains. NetInf implementations should, thus, provide a way to easily plug in new routing, name resolution, or forwarding schemes.

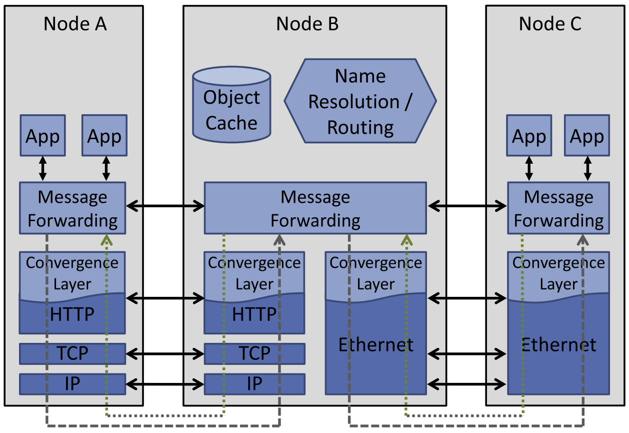

Fig. 5. Example NetInf convergence layers.

4.4.4. Request forwarding

NetInf uses a hybrid request routing/forwarding scheme, integrating pure name-based routing (in the sense of "routing on flat names") with name-resolution-based aspects. The name-based routing could, e.g., simply match names with default routes via prefix or pattern matching. For example, requests from inside a network can reach an "edge" with external access, at which point name resolution is likely to be used. The integration of name resolution and name-based routing also supports, e.g., *late binding*: In heterogeneous domains or in a NetInf DTN island, it is often not possible to resolve the object name *at the requester side* into a locator that *has meaning* at the requester side. Instead, the name might be resolved into some routing hint that leads only *towards* the final destination. In this case, the resolution into the final destination locator can be performed by an intermediate NetInf node along the path to the destination. This combination with name resolution enhances scalability and performance of name-based routing.

Routers performing name-based routing (or default route forwarding) can be configured to do request aggregation (similar to CCN's pending interest management).

At the global level, we expect only one routing scheme, akin to BGP for the Internet. Such a scheme adapted to the convergence layer protocol approach is described in Section 4.7.

4.4.5. Response forwarding

Responses need to get back to requesters. As with request routing, NetInf requires flexible response routing and forwarding and an implementation should make this pluggable like the request routing.

For response forwarding, the main issue is how to handle the state required to make sure the response gets back to the right place when the request has spanned more than one CL "hop." The maintenance of in-network state for a single CL "hop" is something that is handled by the specific CL, e.g., if a TCP CL is used, the response must be generally returned on the same socket from which the request was received. In the general case where a request has passed over many CL "hops," the issue boils down to how to associate a response coming from "downstream" (towards the requester/"source") with a request previously received from "upstream" (the requester).

The NetInf protocol allows different approaches depending on the deployment scenario and scalability constraints. Options are to maintain state in the routers or to annotate the request with some form of token or label. Labels can be locally significant identifiers and label stacks can be used to record a request route for returning responses along a path of NetInf nodes. Ref. [3] provides an example of a complete system using that approach.

4.4.6. Transport services

Since not all underlays have their own flow/congestion control or reliability, there are two options: (1) A NetInf CL implements this functionality. (2) If the CL only provides a basic, integrity-protected message delivery service, transport functions are needed above the CL. For this, NetInf employs a receiver-driven transport above the NetInf protocol layer [3].

4.5. Caching

Caching is critical for NetInf as it lays the basis for efficient content dissemination, load balancing, and operation in challenging network conditions. NetInf supports three different kinds of caching that address different scenarios. First, a NetInf router with built-in caching functionality can provide *on-path caching*. While forwarding GET responses, the router can cache the contained objects. Subsequently, it can answer requests for the same object directly from its cache. Second, NetInf also supports *off-path caching*, i.e., a cache that is not directly on the request/data path. An off-path cache is a dedicated cache that is typically placed in the network by the provider to reduce traffic (especially costly inter-domain traffic) and to reduce latency. As the off-path cache is not directly on the request/data forwarding path, it has to take additional steps to know which objects to cache and to advertise cached copies. Typically, an off-path cache is closely connected to one or more NetInf NRS nodes. The NRS tells the off-path cache which objects to cache based on object popularity and the cache advertises cached copies in the NRS in return. When the NRS receives a GET request for an object cached at a connected off-path cache, the NRS can return the cache's locator as routing hint to the requester, which can subsequently retrieve the object from the cache. In addition, the NRS can optimize locator selection by either filtering or sorting available locators based on additional network knowledge and server/cache load information. Thereby, NetInf can offer functionality similar to approaches that utilize network knowledge to improve peer selection in P2P networks (e.g., ALTO [14], P4P [15]). However, this functionality is tightly integrated in the NetInf architecture and does not require supplemental infrastructure or protocols. Similarly, an off-path cache can also be closely connected to one or more NetInf routers. Thereby, multiple routers can share a joined cache and only have to store an index of the cached objects' names instead of the objects themselves.

Finally, NetInf also supports *peer caching* in NetInf nodes on user devices. Peer caches can function as on- or off-path caches or a combination, i.e., a peer cache can advertise its cached copies in a local NRS and can also respond to GET requests (received, e.g., via local broadcast) with a cached copy. Peer caching is beneficial in challenging network conditions, and it can help to reduce inter-domain traffic and latency and to reduce load at origin servers.

4.6. Name resolution

Name resolution and name-based routing are essential complements and alternatives in NetInf. Especially during migration, name resolution has several advantages. It does not require global adoption right from the start but can grow from the edges by first providing NRS capabilities in some edge networks that can make objects available inside the edge networks. NetInf name resolution supports today's URLs, hence, any already existing object can be used in NetInf right away. URLs pointing to already existing copies can be registered as object locators in edge NRSes and objects can be retrieved via existing protocols like TCP/IP. NetInf routers are not required for a pure name resolution approach.

NDOs are advertized to an NRS using the PUBLISH message that contains the object name, a set of routing hints, and optionally some metadata. By supporting different kinds of routing hints like lower-layer locators (e.g., IP, URLs), indirection to other NetInf name, and other protocol-specific routing hints, this approach offers a flexible mechanism to adapt the data retrieval to heterogeneous underlying networks. To retrieve the advertized information from an NRS, the GET message is used.

NetInf does not dictate a single global NRS but can support multiple NRSes which all support the same NetInf interface. Different types of NRSes are possible, e.g., a local broadcast NRS for local ad hoc networks, a global NRS, and local network NRSes. Operating an NRS in their local network allows network providers to improve traffic engineering by selecting appropriate object copies during the resolution process to reduce and control network traffic and, potentially, load at the caches and data servers. This gives network operators a strong incentive to invest in NetInf NRSes, thereby supporting the NetInf migration process.

In general, there are three operational modes for routing hint selection. First, the NRS can return all its known routing hints and the requester selects one or more routing hints to retrieve the data. Second, the NRS can return a preselected set of routing hints based on its network knowledge, resulting inter-domain traffic, server load, internal policies, etc., giving the NRS full control of the selection process. In a hybrid approach, the NRS can return a prioritized list of routing hints. This leaves the final selection to the user while still making use of the internal knowledge of the NRS.

For a fully functional, world-wide Network of Information, at least one global NRS is required. NetInf offers multiple options to interconnect separate local NRSes into a global NRS infrastructure, including the multi-level distributed hash table (MDHT) system [8] and the Hierarchical SkipNet (HSkip) system [16]. Both systems form a global, hierarchical NRS that is topologically embedded in the underlying network to improve scalability, latency, and locator selection for efficient information dissemination. The hierarchical, topological embedding of both systems combined with their registration and retrieval scheme ensures that close-by locators are automatically preferred over farther away locators during name resolution. To reduce load at the global level, caching of NRS entries can be used at lower-level NRS nodes. Details can be found in reference [16].

The NRS can also collect statistics about object popularity. The NRS is well suited for that as it aggregates resolution requests by many users. In a topologically embedded NRS, dedicated NRS subsystems are responsible for separate subnetworks. Thereby, the statistics collected by a local NRS automatically represent local popularity, making caching even more efficient when combined with *local* caches.

4.7. Inter-domain communication

Fig. 6 shows an interconnection of different NetInf domains, connected via a central default-free zone (DFZ). Like in today's Internet, edge domains can decide internally on NetInf routing/forwarding, adapted to a domain's needs. In the DFZ, the NetInf BGP routing system is used, a variant of today's BGP combined with a global resolution system for aggregation. The DFZ is described in detail in reference [3].

Fig. 6 shows an example for inter-domain routing that relies on NetInf's hybrid, name-based/name-resolution-based object retrieval. The hexagons are NetInf routers. The messages exhibit type

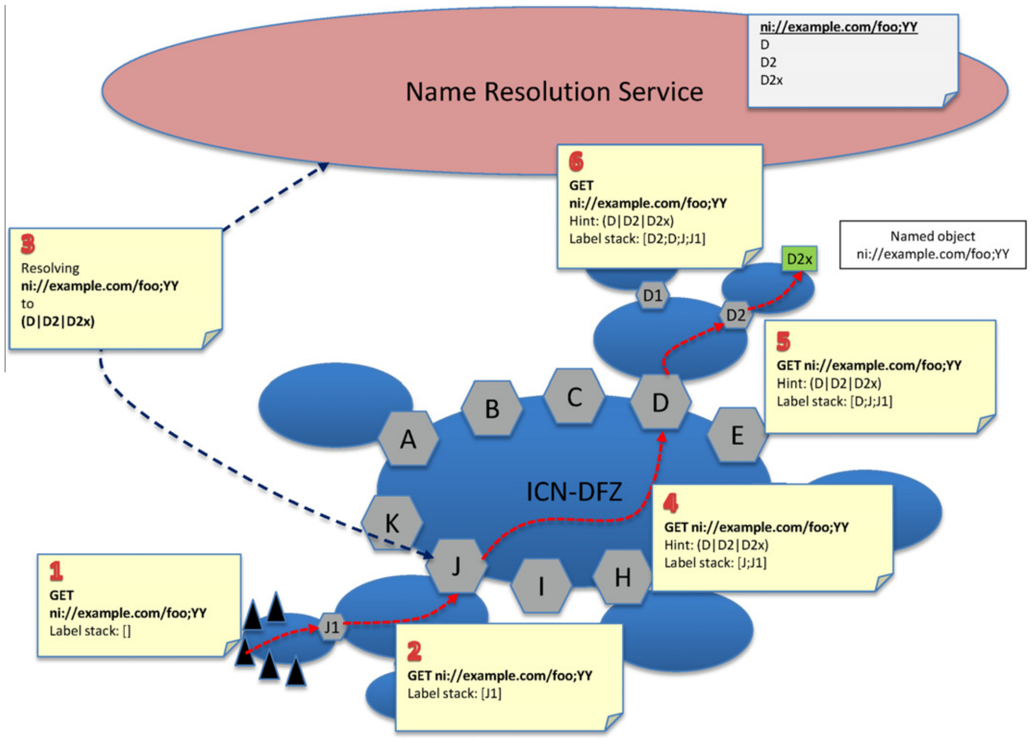

Fig. 6. NetInf inter-domain scenario.

and additional parameters (e.g., requested NDO name, label stack, routing hints). The example assumes that no cached copy is available at first. In step 1, a client in *J1*'s access network sends a *GET* request for the NDO `ni://example.com/foo;YY`. The request is forwarded to router *J* via a default route (step 2). While forwarding the request, the names of involved NetInf routers are added to the label stack in the request (here router *J1*) to construct a source route. Router *J* lacks routing information. Therefore, it consults an NRS, which resolves the NDO name into a set of routing hints (*D*—*D2*—*D2x*) (step 3), which are added to the request. The routing hints typically do not name the end-point destination node of the request but are rather used to forward to the next hop; node *D* in our example (step 4). Node *D* belongs to a different provider with its own routing/forwarding scheme, e.g., based on the provided routing hints (step 5). Finally, the request reaches node *D2x* that holds a copy of the requested NDO (step 6). Then, the information collected in the label stack is used to forward the NDO back to the requester (not shown in Fig. 6).

5. Evaluation

We have evaluated the NetInf architecture theoretically and by simulations, and different groups of researchers and companies have implemented several NetInf prototypes, including two open source implementations: the NetInf Software Project[5] and OpenNetInf.[6] In the following, we focus on the components we believe to be most critical for an effective Network of Information system and successful migration: *name resolution* and *caching*.

5.1. Name resolution

The NetInf architecture supports name resolution as well as name-based routing. While migrating towards NetInf, we assume that NetInf will run on top of today's IP network. Such a scenario provides a fully connected underlying network graph; name resolution then becomes the critical component of the NetInf architecture to find named data objects. NetInf's name resolution concept is based on distributed, topologically embedded, and interconnected name resolutions services. The resulting global scalability is a key performance metric.

To achieve this scalability, we have developed two name resolution systems: HSkip and MDHT. We present a detailed analysis and comparison of both systems in reference [16]. In the following, we discuss the scalability analysis results, using HSkip as an example. We focus on load distribution and request latency, i.e., the time from posing a request until the response arrives at the requester (excluding technology-dependent last mile effects). The following evaluation can only provide an overview. More details about our model and assumptions can be found in reference [16].

Our analysis in reference [8] has shown that 10^6 NRS nodes should be sufficient for a world-wide HSkip/MDHT system. To highlight HSkip's scalability, we evaluate an HSkip system with up to 12 million nodes, approximately matching today's number of world-wide DNS nodes. Our theoretical analysis has shown that the main factors influencing HSkip's average latency are the total number of NRS nodes,[7] the number of hierarchical levels, and the *neighborhood effect*. The neighborhood effect describes the influence of the requester's location on the popularity of requested content. Users typically show a strong interest in content with a local reference, e.g., the company's intranet, web pages in the local language, etc. In the following, we call the probability that a user requests an object registered at a certain NRS level (e.g., the company-wide or country-wide level) the *level probability (LP)*.

Fig. 7 plots the average latency as a function of the total number of NRS nodes in the system,[8] first *ignoring* the neighborhood effect.

[5] <http://sourceforge.net/projects/netinf/>.
[6] <http://www.netinf.org/>.

[7] Our analysis considers the total No. of NDOs indirectly via the No. of NRS nodes as an increased No. of NDOs requires an increased No. of NRS nodes so that each NRS node can handle its managed objects efficiently [8].
[8] For the detailed analysis itself, see reference [16].

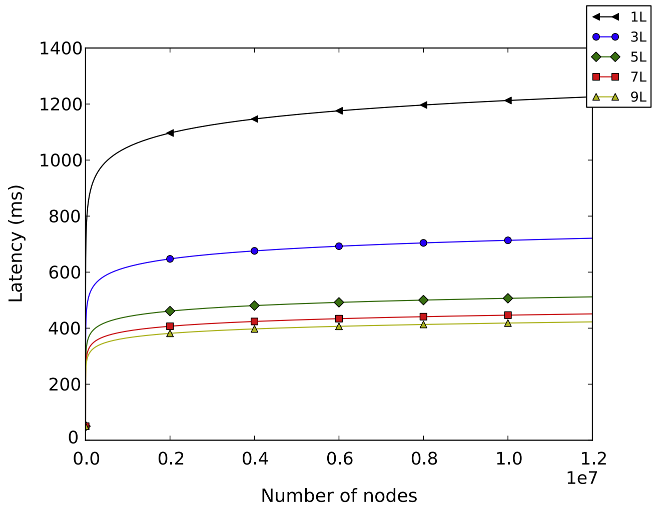

Fig. 7. Analysis: HSkip average latency with up to 12 million NRS nodes, 1–9 levels (L), LP = 0.0.

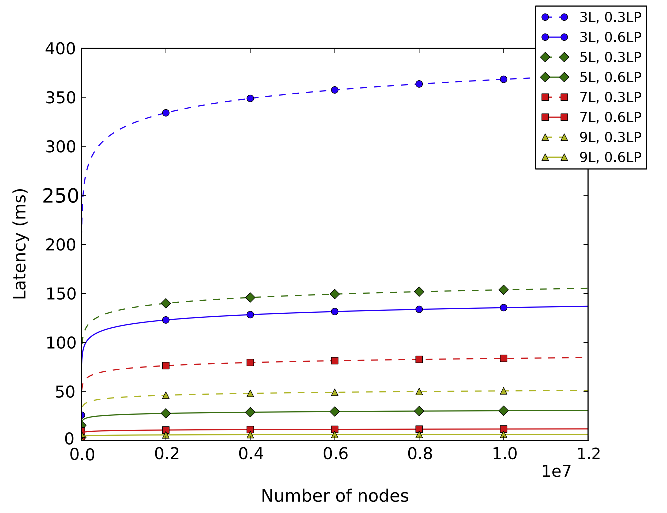

Fig. 8. Analysis: HSkip average latency with up to 12 million NRS nodes, 3–9 levels (L), LP = 0.3 and 0.6.

With 12 million nodes, an HSkip system with 7 levels ("7L") would have an average latency of approximately 450 ms. The latency decreases with an increasing number of levels. This is important as it allows a fine-grained topologically embedded NRS (i.e., reducing inter-domain traffic as described in Section 4.6) while reducing average latency. We assume that an NRS system with 5–9 levels is realistic, resulting in an average latency of approximately 420–510 ms when ignoring the neighborhood effect. This compares well against the latency of a flat distributed hash table (DHT) ("1L") that requires more than 1200 ms on average with 12 million nodes.

Fig. 7 is conservative as it ignores the neighborhood effect. A DNS traffic analysis [17] performed at University of Paderborn revealed that 40–70% of all DNS requests generated inside the university network are for locally-hosted domains and subdomains. These results are supported by related studies such as a study by McDaniel and Jamin [18] that shows a similar effect not only for university networks but also for the AT&T network. A study by Ager et al. [19] also reveals a similar effect at the continent level.

Based on these results, we assume a level probability of 0.3 and 0.6 in Fig. 8. The increased level probability significantly decreases the average HSkip latency. For example, an HSkip system with 9

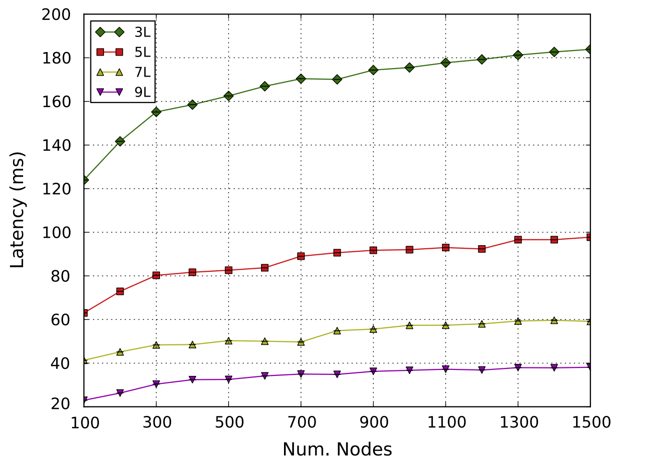

Fig. 9. Simulation: HSkip average latency with up to 1500 nodes, 3–9 levels (L), LP = 0.3.

levels, 12 million nodes, and LP = 0.3 has an average latency of only 50 ms, down from 450 ms when ignoring the neighborhood effect. In contrast, the latency of a flat distributed hash table (DHT) (not shown in Fig. 8 for a better ordinate scale) is not influenced by the level probability as it does not have multiple levels, i.e., it equals the graph "1L" shown in Fig. 7.

Our simulations implemented with OMNeT++[9] support our theoretical analysis. Fig. 9 shows the average latency for an HSkip simulation with 1500 NRS nodes and LP = 0.3. The simulation results show a behavior similar to the theoretical analysis with only logarithmically increasing latency as a function of the number of NRS nodes and average latencies in the same range. Similar to the theoretically analysis, the average latency is decreasing with an increasing number of levels.

The general influence of the neighborhood effect is illustrated in Fig. 10. It shows the significantly decreasing latency with an increasing level probability. For an HSkip system with 5–9 levels, 12 million nodes, and a level probability of 40–70%, this results in average latencies (well) below 100 ms.

For the overall scalability of the NRS system, the load distribution among the NRS nodes is also important. In our simulation, we assume a Zipf-like object popularity distribution similar to the popularity in today's web [20] with the parameter α between 0.7 and 1.0. Fig. 11 shows the load distribution of all requests processed by NRS nodes inside an HSkip system with 5 levels. The histogram shows a bell-shaped curve with most NRS nodes getting between 20 000 and 40 000 requests. Only a few nodes have to process more requests, with no node getting more than 62 500 requests.

In summary, our evaluation results indicate that a hierarchical NRS system like HSkip meets the scalability requirements for a world-wide NRS for NetInf, offering good average latencies and a reasonable load distribution among the NRS nodes. The MDHT system shows results similar to HSkip; reference [16] gives more details about both systems.

5.2. Caching

Caching is a critical component to efficiently distribute content and to increase data availability. We have evaluated the effectiveness of the NetInf caching concepts in a scenario using our MDHT implementation in the OpenNetInf prototype. As usual, quantitative results of prototype evaluations depend heavily on the setup.

[9] <http://www.omnetpp.org>.

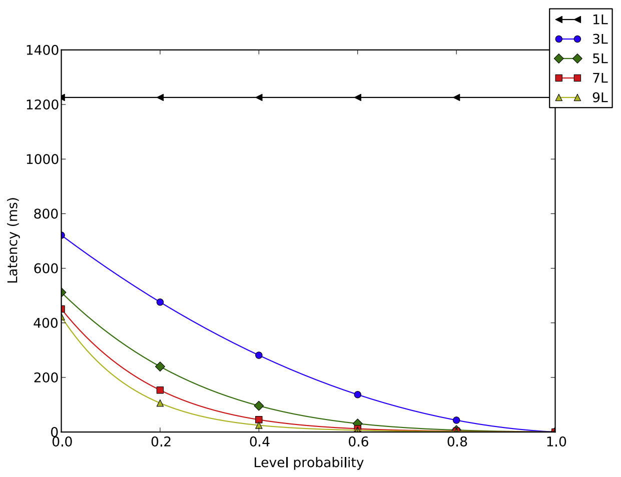

Fig. 10. Analysis: HSkip latency with 12 million NRS nodes, 1–9 levels (*L*).

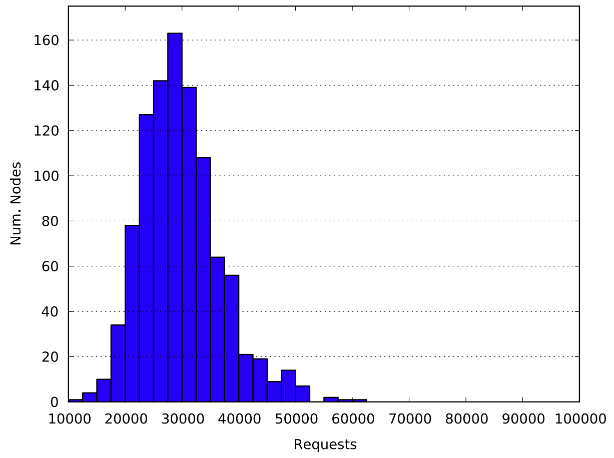

Fig. 11. Simulation: load distribution in HSkip system with 1000 nodes, 5 levels, LP = 0.3.

Nonetheless, they can give some intuition and can point out strengths and weaknesses of an architecture.

We evaluated the effects of peer-side caching, in-network caching, and the hierarchical MDHT NRS on inter-domain traffic by constructing a hierarchical network structure and measuring traffic between the levels. Reduced inter-domain traffic both represents reduced costs for providers as well as overall reduced network load. In addition, keeping traffic inside the local domain can increase the data availability in case of challenging network conditions as dependencies on the inter-domain connectivity are reduced. We also evaluate the general overhead of our prototype in terms of additional traffic (management traffic, request packets, etc.).

5.2.1. Measurement setup

Fig. 12 shows our setup. The network is hierarchical with 4 levels. We have 16 clients (*Cl.*), grouped into 4 subnetworks at level 4, that all request the same 1 MB file one after the other; the request order is random. We measure the traffic between level 1 and 2 (at *router 1*) and between level 2 and 3 (at *router 1.1* and 1.2). We measure all traffic passing through the respective router interfaces including any management traffic and other overhead. The *origin file server* serving the requested file is connected at the top level. Each network at level 1–3 has its own in-network cache (off-path).

We have used the following measurement settings: Peer-side caches serve other NetInf peers only in the same local network at level 4. In-network caches serve all nodes in the respective subtree (e.g., *cache 1.1* serves *clients 1–8*) and cache an object as soon as it is requested the first time in its subtree. Clients use sources in the following order: (1) Peer-side caches, (2) in-network caches at lower levels, and (3) in-network caches at higher levels.

5.2.2. Results

Fig. 13 shows the cumulative inter-domain traffic between levels 1/2 and levels 2/3. We have measured the inter-domain traffic of the OpenNetInf system with all caching turned off, peer-side *or* in-network caching turned on, and both peer-side *and* in-network caching turned on. For comparison, we have measured the same scenario with "pure" Hypertext Transfer Protocol (HTTP), i.e., without any NetInf nodes and NetInf infrastructure. All measurements have been performed with a varying number of up to 16 downloading clients. The subset of clients is chosen randomly from all clients. The first client always downloads from the origin server, and all caches are initially empty. In the NetInf case, subsequent clients can access available (in-network and peer-side) caches. Fig. 13 shows confidence intervals at a 95% confidence level, however, most confidence intervals are very small so that they are only visible as a single line.

The HTTP graph shows traffic increasing linearly with the number of downloading clients, with about 34 MB *cumulative* traffic at both inter-domain borders for 16 downloading clients. This is as expected as all clients have to download the 1 MB file from the origin server at the top level, hence, producing $2 \cdot 16$ MB plus about 5% overhead from packet headers, which is consistent with theoretical overhead calculations. NetInf without caching shows a sim-

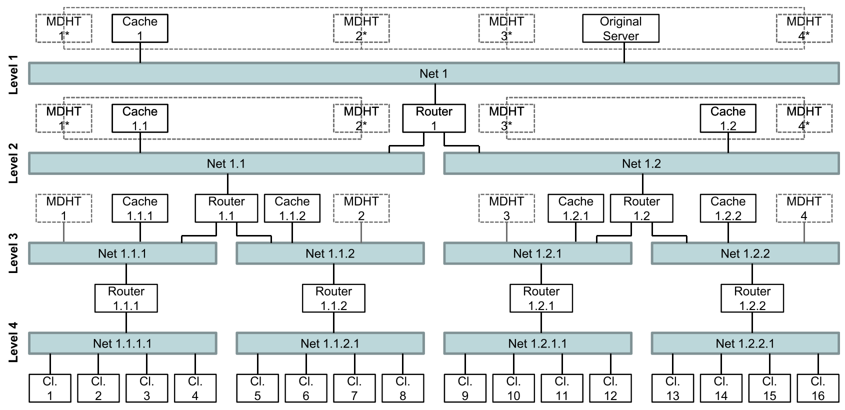

Fig. 12. Prototype: caching scenario with hierarchical MDHT.

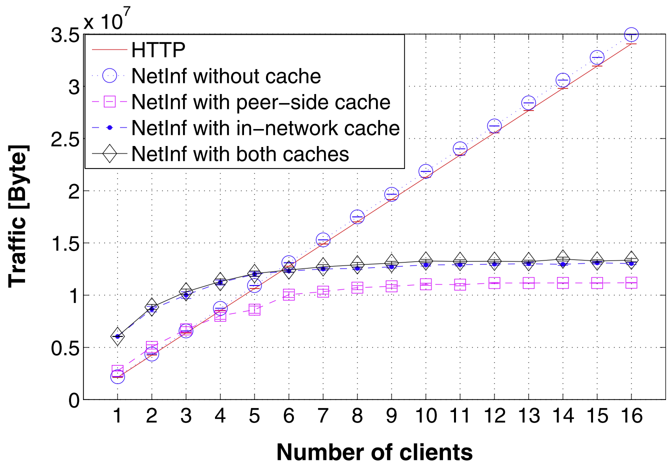

Fig. 13. Prototype: cumulative inter-domain traffic between levels 1/2 and 2/3.

ilar behavior. Comparing both cases shows that the (unoptimized) OpenNetInf implementation produces traffic overhead of only about 2.5% compared to HTTP for a 1 MB file size. This overhead is constant (in bytes) and mainly consists of management packets generated by the MDHT system and NetInf request/response messages that additionally contain metadata.

Both in-network caching and peer-side caching (combined as well as separately) result in a significantly reduced inter-domain traffic, keeping traffic as local as possible thanks to caching and the hierarchical, topologically embedded resolution approach. The traffic increases with the number of clients until at least one peer-side cache per subnetwork has cached the object or until the in-network caches contain the object, respectively. Thereafter, the inter-domain traffic remains stable as all following clients can retrieve the object from the caches without producing additional inter-domain traffic. The measurements with in-network caching turned on show higher traffic than the measurements which only use peer-side caching and higher traffic compared to HTTP for a small number of clients. This is due to the fact that filling the *off-path* in-network caches produces additional inter-domain traffic. However, this has only meaningful influence due to our setup with a limited number of clients.

In summary, the NetInf caching concept yields significant traffic reductions. These advantages can be achieved in many different scenarios thanks to the broad caching concept that includes peer-side caching, in-network caching, and on-path as well as off-path caching. Whether peer-side or in-network caching produces better results heavily depends on the scenario, e.g., on the number of clients per subnetwork, the number of subnetworks sharing a common in-network cache, local caching decisions by peers, and the ability/willingness of peers to cache. Additional details can be found in Dannewitz et al. [21].

6. Related work

Several projects have developed an ICN architecture in recent years. The architectures all center around the content itself and around efficient content distribution. However, the projects differ in their focus, design choices, and resulting solutions. In this section, we give a brief overview of these ICN projects and how they relate to NetInf. A more detailed comparison of some of the major projects with active ICN research, including DONA, CCN, NDN, PSIRP, and PURSUIT, can be found in our article by Ahlgren et al. [9].

The translating relaying internet architecture integrating active directories (TRIADs) project [22] has created one of the first next generation architectures based on the information-centric paradigm. TRIAD defines an explicit content layer that provides content routing, caching, connection setup, load balancing, and object naming. In contrast to NetInf, TRIAD uses a hierarchical namespace and performs aggregation based on the hierarchical names for name-based routing. The object namespace does not provide name-data integrity like in NetInf. In addition to content distribution, TRIAD also incorporates content transformation (e.g., adapting web pages for small screen resolutions), which sets it apart from other ICN projects.

The data-oriented network architecture (DONA) project [23] also focuses on information dissemination. Like NetInf, it uses a flat namespace that allows for name-data integrity checking. Objects can be replicated to several servers but servers have to be authenticated in order to be allowed to distribute content, which is not the case in NetInf. Based on the flat namespace, DONA performs name-based anycast routing to retrieve objects. The content-centric networking (CCN) [24] project—now named-data networking (NDN) [25]—project also used a name-based routing approach to retrieve objects. In contrast to NetInf, CCN's name-based routing approach purely relies on routing state in the network routers during data transport. Name-based routing is based on CCN's hierarchical namespace. Names can be human-readable but require a PKI for name-data integrity checking, which is not the case for NetInf. CCN performs caching of small data chunks in each network router with a focus on on-path caching. In addition to on-path caching, NetInf also supports off-path caching and peer-side caching based on its complementary name resolution.

The publish-subscribe internet routing paradigm (PSIRP) architecture [26]—now continued in the Publish-Subscribe Internet Technology (PURSUIT) project [27]—fundamentally builds on the publish/subscribe paradigm to find and retrieve data in the future Internet. A rendezvous mechanism matches publishers and subscribers. A name-based routing approach is used to forward data based on routing information stored in the packet, i.e., routers do not need to keep forwarding state. Content-based networking (CBN) [28] goes one step further than PSIRP. It tries to integrate dissemination of content with perpetual validity like a video file, which is the main focus of most ICN architectures, with publish/subscribe event notification for short ephemeral messages. Object naming and subscription is based on a set of predicates that are used to identify matches between subscriptions and publications. CBN builds on ideas from the SIENA [29] event notification architecture.

Several other projects relate to the ICN ideas or build on ideas from the above projects. The content-oriented networking: a new experience for content transfer (CONNECT) project [30] builds on CCN and aims to complement CCN with proposals in the area of traffic control, naming, routing/forwarding, replication and caching, and deployment strategies. The CONVERGENCE project [31] focuses on enhancing the Internet with a content-centric, publish/subscribe-based service model. A strong focus lies on the information model using containers for all kinds of digital content like audio data, video data. These containers are accessed via a publish/subscribe mechanism. The underlying information access is performed via a name-based routing mechanism. Unlike CCN and similar to NetInf, CONVERGENCE uses a stateless routing mechanism and source routing by storing information for the return path in the request packet. The COAST project [32] focuses on a content-aware delivery architecture with "on the fly" identification and distributed "on-line" discovery. It makes heavy use of deep packet inspection in the network. In addition, it supports content adaptation and enrichment similar to TRIAD. A Service and Information Oriented Network Architecture (SIONA) [33] builds on IP to construct an information-centric network architecture. IP addresses are explicitly made part of the object and service naming structure. Based on these names, SIONA performs name-based routing. The request packets leave "bread crumbs" in network routers, which are used to route responses back along the request path, closely related to CCN's approach. The content mediator architecture for content-aware nETworks (COMET) project [34,35] builds an overlay network that aims to become a global quality of service (QoS)-aware content access mechanism for all kinds of content. It can be deployed on top of today's Internet architecture as well as on top of future Internet architectures. COMET supports content distribution in a content- and network-aware way based on a name resolution approach. The eXpressive Internet Architecture (XIA) [36] aims to not limit the architecture to one kind of communication scheme (e.g., host-, content-, or service-centric) but supports multiple communication schemes in parallel. It also supports yet unforeseen communication schemes. This is achieved via flexible addressing and forwarding semantics. The scheme defines the desired *intent*, e.g., to retrieve a certain piece of content, and additionally several alternative *means* how to achieve this intent. For example, the means could be host-centric, i.e., via a specific web server running on a host in a defined network, or the means could be information-centric, i.e., via directly addressing the content in some information-centric underlying network mechanism. Hence, XIA can build on and integrate ICN mechanisms as developed, e.g., by NetInf. The MultiCache architecture [37] proposes a pure overlay solution. It has a strong focus on caching and multicast for an information-centric extension of today's Internet. The architecture is based on the Pastry [38] DHT overlay and on the Scribe [39] multicast overlay. In addition, providers deploy proxy overlay access routers and in-network caches at their Point of Presences (PoPs). Based on this setup, data is delivered in a hybrid approach either based on multicast (e.g., in case of flash crowds) or via unicast, making use of Pastry's inherent locality awareness to locate nearby caches. The Cache-and-Forward (CNF) architecture [40,41] uses a name resolution mechanism to resolve content identifiers into IP addresses, which is related to NetInf's NRS-based approach. CNF introduces dedicated routers in the network that can cache content and forward content hop-by-hop, quite similar to the CCN approach. However, forwarding is performed based on the previously resolved IP addresses. The project takes special measures for mobile nodes by introducing so called post offices at the network edges where participants with intermittent connectivity can deliver and fetch their content.

DTN [13] is a research area that is related to ICN in that it provides a hop-by-hop store-and-forward approach for data transfer. NetInf utilizes a Convergence Layer approach related to the Convergence Layer approach as proposed in the DTN context. While DTN is generally building on a traditional host-centric addressing paradigm using end-point identifiers, projects like DPSP [42] and Haggle [43] bridge the DTN and ICN areas by providing a kind of publish/subscribe system for DTN that uses a data-centric addressing approach. The addressing scheme is based on metadata in the form of key-value pairs. This approach is, however, closer related to publish/subscribe architectures like SIENA and CBN than to NetInf.

In addition to the afore described projects that aim for a full ICN architecture, there are several publications focusing on various specific ICN aspects, including the overall ICN paradigm [44–46], caching [47–51], routing and transport [52–55], and naming and security [56,6,57].

NetInf shares many goals and assumptions with other ICN approaches however we like to mention two key differentiators. NetInf combines name resolution and name-based routing that allows for both on path as well as off path caching. NetInf's flat, self-certifying namespace gives good security properties without the need for a trusted third party and allows for routing based on explicit aggregation.

7. Conclusions and future work

ICN is generally considered as a more adequate approach to content distribution and similar applications. In fact, it can be of much broader applicability. By providing access to NDOs as a first class networking service, ICN is beneficial to all applications and network interactions that can be modeled after this paradigm, including content distribution but also M2M applications, with comparatively small objects.

ICN puts accessing named data objects, name-based routing and name resolution, in-network storage, and data object security into the thin hour glass waist of the network – removing the need for application-specific overlays. This enables networks to provide much more efficient communications and enables operators to manage data object transport much better.

The specific NetInf approach to ICN is targeted at global scale communication with support for many different types of networks and deployments, including traditional Internet access/core network configurations, data centers, as well as challenged and infrastructure-less networks. NetInf's approach to connecting different technology and administrative domains into a single information-centric network is based on a hybrid name-based routing and name-resolution scheme.

The design is based on the idea of not limiting it too much by implicit assumptions how networks are built and what underlying communication services are available. A minimal set of node requirements for supporting the NetInf object model, naming format, and protocol messages provides the necessary common denominator. The convergence layer approach enables extensibility with respect to new underlying network technologies. Finally, the hybrid name-based routing and name resolution approach allows us to inter-connect such different domains.

In this article, we have put particular focus on the latter aspect, specifically the global NRS scalability and performance issues. Our analysis and simulation results indicate that the proposed NetInf NRS solutions, specifically MDHT and HSkip, provide a scalable solution that can handle more than 10^{15} NDOs while keeping resolution latency (well) below 100 ms.

Naturally, there are still interesting questions to look into and hard problems to address:

7.1. Access control

A global network where anybody has access to any published data object is not likely to prevail. We are investigating the possibility to apply scoping concepts to NetInf, leveraging NRS extensions.

7.2. NDO aggregation

NetInf has a concept of application layer framing [58] for NDOs. We are investigating ways to reduce the load for NRSes through NDO aggregation. NDO aggregation means that a set of NDOs are handled as a group. All NDOs in an aggregate share NRS bindings and routing information, resulting in that an NRS lookup can be cached and reused for all but the first NDO in the aggregate. This optimisation is important for efficient handling of small NDOs. Examples are objects on a web site and individual photos in a collection. Aggregation would also enable publishers to segment large objects into many small ones in order to facilitate more convenient access to parts of the object.

7.3. Transport performance

NetInf's convergence layer allows for inter-connecting different types of networks into a single ICN. With underlays providing really different communication services (from uni-directional, opportunistic message forwarding to flow- and congestion-controlled higher layer communication services; from delay-challenged to high-speed optical backbone networks) the interaction of NetInf transport functions with CL transport functions is interesting to investigate.

7.4. Interactive, real-time communication

The concepts of accessing named data objects and in-network caching in NetInf and ICN in general almost intuitively provide benefits for applications that are based on delivering requested named data objects. It's an interesting research question how far ICN can be taken to support different types of applications, especially those where the notion of *one request – one data object* does not hold or is not practical. Some of those applications such as VoIP today employ a session abstraction and explicit session initiation transactions, and we are investing limits but also extension possibilities for NetInf in this regard. The above mentioned aggregation of NDOs is one such extension.

Acknowledgments

The authors thank the partners from the 4WARD and SAIL EU projects for their contributions in many discussions on NetInf and Sebastian Dröge for his implementation support.

References

[1] Cisco Systems, Inc., Entering the zettabyte era, white paper, June 2011.
[2] D. Kutscher, B. Ahlgren, M. D'Ambrosio, E. Axelsson, L. Brown, C. Dannewitz, Z. Despotovic, A.E. Eriksson, S. Farrell, J. Frtunikj, M. Gallo, P. Pöyhönen, C. Imbrenda, B. Kauffmann, A. Lindgren, H. Lundqvist, A. Lynch, L. Muscariello, B. Ohlman, J.F. Peltier, K.-A. Persson, O. Strandberg, P. Truong, J. Tuononen, V. Vercellone, S. Weber, (D.3.1) The network of information: architecture and applications, deliverable, SAIL 7th FP EU-funded project, July 2011.
[3] D. Kutscher, B. Ahlgren, M. D'Ambrosio, E. Davies, A.E. Eriksson, S. Farrell, B. Grönvall, C. Imbrenda, B. Kauffmann, G. Kunzmann, A. Lindgren, I. Marsh, L. Muscariello, B. Ohlman, K.-A. Persson, P. Pöyhönen, M. Shehada, D. Staehle, O. Strandberg, J. Tuononen, V. Vercellone, (D.3.2) Content delivery and operations, deliverable, SAIL 7th FP EU-funded project, May 2012.
[4] D. Kutscher, B. Ahlgren, M. D'Ambrosio, E. Davies, A.E. Eriksson, S. Farrell, B. Grönvall, C. Imbrenda, B. Kauffmann, G. Kunzmann, A. Lindgren, I. Marsh, L. Muscariello, B. Ohlman, K.-A. Persson, P. Pöyhönen, M. Shehada, D. Staehle, O. Strandberg, J. Tuononen, V. Vercellone, (D.3.3) Final NetInf architecture, deliverable, SAIL 7th FP EU-funded project, November 2012.
[5] A. Eriksson, B. Ohlman, Dynamic internetworking based on late locator construction, in: 10th IEEE Global Internet Symposium, 2007.
[6] A. Ghodsi, T. Koponen, J. Rajahalme, P. Sarolahti, S. Shenker, Naming in content-oriented architectures, in: Proceedings of ACM SIGCOMM Workshop on Information-Centric Networking, ICN '11, ACM, New York, NY, USA, 2011, pp. 1–6.
[7] D.D. Clark, J. Wroclawski, K.R. Sollins, R. Braden, Tussle in cyberspace: defining tomorrow's Internet, in: Proceedings of Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM), ACM Press, New York, NY, USA, 2002, pp. 347–356.
[8] M. D'Ambrosio, C. Dannewitz, H. Karl, V. Vercellone, MDHT: a hierarchical name resolution service for information-centric networks, in: Proceedings ACM SIGCOMM Workshop on Information-centric Networking, ACM, New York, NY, USA, 2011, pp. 7–12.
[9] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, B. Ohlman, A survey of information-centric networking, IEEE Communications Magazine 50 (7) (2012) 26–36.
[10] S. Farrell, D. Kutscher, C. Dannewitz, B. Ohlman, A. Keranen, P. Hallam-Baker, Naming things with hashes, IETF Internet-Draft draft-farrell-decade-ni-06, version 06 (May 2012).
[11] M. Nottingham, E. Hammer-Lahav, Defining Well-known uniform resource identifiers (URIs), RFC 5785 (Proposed Standard) April 2010. <http://www.ietf.org/rfc/rfc5785.txt>.
[12] C. Dannewitz, J. Golic, B. Ohlman, B. Ahlgren, Secure naming for a network of information, in: Proc. 13th IEEE Global Internet Symposium 2010 in conjunction with IEEE Infocom 2010, San Diego, USA, 2010.
[13] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, H. Weiss, Delay-tolerant networking architecture, RFC 4838 (Informational) April 2007. <http://www.ietf.org/rfc/rfc4838.txt>.
[14] J. Seedorf, E. Burger, Application-layer traffic optimization (ALTO) problem statement, RFC 5693 October 2009. <http://www.rfc-editor.org/rfc/rfc5693.txt>.
[15] H. Xie, A. Krishnamurthy, A. Silberschatz, Y.R. Yang, P4P: explicit communications for cooperative control between P2P and network providers, P4PWG whitepaper, May 2007.
[16] C. Dannewitz, M. D'Ambrosio, H. Karl, V. Vercellone, Hierarchical DHT-based name resolution for information-centric networks, Elsevier Computer Communications, Special Issue on Information-Centric Networking, 2013.
[17] C. Dannewitz, H. Karl, A. Yadav, Report on locality in DNS requests – Evaluation and impact on future internet architectures, Tech. Rep. TR-RI-12-323, University of Paderborn, Paderborn, Germany, July 2012.
[18] P. McDaniel, S. Jamin, A scalable key distribution hierarchy, Tech. rep., University of Michigan, Dept. of Electrical Engineering and Computer Science, 1998.
[19] B. Ager, W. Mühlbauer, G. Smaragdakis, S. Uhlig, Web content cartography, in: Internet Measurement Conference (IMC '11), ACM, 2011.
[20] A. Williams, M. Arlitt, C. Williamson, K. Barker, Web Workload Characterization: Ten Years Later, Springer, 2005. Ch. 1, pp. 3–22.
[21] C. Dannewitz, M. Herlich, H. Karl, OpenNetInf – Prototyping an information-centric network architecture, in: Proceedings of IEEE LCN – Workshop on Architectures, Services and Applications for the Next Generation Internet (WASA-NGI), 2012.
[22] D.R. Cheriton, M. Gritter, TRIAD: A new next-generation internet architecture. <http://www-dsg.stanford.edu/triad/> July 2000.
[23] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K.H. Kim, S. Shenker, I. Stoica, A data-oriented (and beyond) network architecture, in: Proceedings of Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '07), ACM Press, New York, NY, USA, 2007, pp. 181–192.
[24] V. Jacobson, D.K. Smetters, J.D. Thornton, M. Plass, N. Briggs, R. Braynard, Networking named content, Communications of the ACM 55 (2012) 117–124.
[25] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J.D. Thornton, D.K. Smetters, B. Zhang, G. Tsudik, kc claffy, D. Krioukov, D. Massey, C. Papadopoulos, T. Abdelzaher, L. Wang, P. Crowley, E. Yeh, Named data networking (NDN) project, TechReport NDN-0001, PARC, October 2010. <http://www.named-data.net/ndn-proj.pdf>.
[26] M. Ain, D. Trossen, P. Nikander, S. Tarkoma, K. Visala, K. Rimey, T. Burbridge, J. Rajahalme, J. Tuononen, P. Jokela, J. Kjällman, J. Ylitalo, J. Riihijärvi, B. Gajic, G. Xylomenos, P. Savolainen, D. Lagutin, D2.3 – architecture definition, component descriptions, and requirements, Deliverable, PSIRP 7th FP EU-funded project, February 2009.
[27] PURSUIT – Publish–Subscribe Internet Technology. <http://www.fp7-pursuit.eu/>.
[28] A. Carzaniga, M. Papalini, A.L. Wolf, Content-based publish/subscribe networking and information-centric networking, in: Proceedings of ACM SIGCOMM Workshop on Information-Centric Networking (ICN-2011), Toronto, Canada, 2011, pp. 56–61.
[29] A. Carzaniga, Architectures for an event notification service scalable to wide-area networks, Ph.D. thesis, Politecnico di Milano, Milano, Italy, Dec. 1998.
[30] Content-oriented networking: a new experience for content transfer (connect). <http://www.anr-connect.org/>.
[31] S. Salsano, A. Detti, G. Tropea, N.B. Melazzi, L. Chiariglione, H. Castro, A.C. Anadiotis, A. Mousas, C. Patrikakis, T. Huebner, M. Tanase, L. Corlan, P. Gkonis, J. Ribas, D. Sequeira, System architecture, EU project deliverable D3.2, July 2011.
[32] Content Aware Searching retrieval and sTreaming (COAST). <http://www.coast-fp7.eu/>.
[33] M. Xu, Z. Ming, D. Li, C. Xia, SIONA: a service and information oriented network architecture, in: Proc. AsiaFI Summer School, August 2011.
[34] COntent Mediator architecture for content-aware nETworks (COMET). <http://www.comet-project.org>.
[35] W. Chai, N. Wang, I. Psaras, G. Pavlou, C. Wang, G. De Blas, F. Salguero, L. Liang, S. Spirou, A. Beben, et al., CURLING: content-ubiquitous resolution and delivery infrastructure for next-generation services, IEEE Communications Magazine 49 (3) (2011) 112–120.
[36] D. Han, A. Anand, F. Dogar, B. Li, H. Lim, M. Machado, A. Mukundan, W. Wu, A. Akella, D.G. Andersen, J.W. Byers, S. Seshan, P. Steenkiste, XIA: efficient support for evolvable internetworking, in: Proceedings of 9th USENIX NSDI, San Jose, CA, 2012.
[37] K. Katsaros, G. Xylomenos, G.C. Polyzos, MultiCache: an overlay architecture for information-centric networking, Computer Networks 55 (4) (2010) 936–947.
[38] A. Rowstron, P. Druschel, Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems, in: Proceedings of IFIP/ACM Conference on Distributed Systems Platforms, 2001, pp. 329–350.
[39] M. Castro, P. Druschel, A.M. Kermarrec, A.I.T. Rowstron, Scribe: a large-scale and decentralized application-level multicast infrastructure, IEEE Journal on Selected Areas in Communications 20 (8) (2002) 1489–1499.

[40] S. Paul, Postcards from the edge: a cache and forward architecture for the future internet, in: 2nd COST-NSF Workshop on Future Internet (NeXtWorking), Berlin, 2007.
[41] L. Dong, H. Liu, Y. Zhang, S. Paul, D. Raychaudhuri, On the cache-and-forward network architecture, in: Proceedings of IEEE Conference on Communications, ICC'09, IEEE Press, Piscataway, NJ, USA, 2009, pp. 2119–2123.
[42] J. Greifenberg, D. Kutscher, Efficient publish/subscribe-based multicast for opportunistic networking with self-organized resource utilization, in: The First IEEE International Workshop on Opportunistic Networking, 2008.
[43] J. Su, J. Scott, P. Hui, J. Crowcroft, E. De Lara, C. Diot, A. Goel, M.H. Lim, E. Upton, Haggle: seamless networking for mobile applications, in: Proceedings Conference on Ubiquitous Computing, UbiComp '07, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 391–408.
[44] A. Ghodsi, T. Koponen, B. Raghavan, S. Shenker, A. Singla, J. Wilcox, Information-centric networking: seeing the forest for the trees, in: Proceedings of 10th ACM Workshop on Hot Topics in Networks (HotNets-X), ACM Press, 2011.
[45] D. Trossen, A. Kostopoulos, Exploring the tussle space for information-centric networking, in: 39th Research Conference on Communication, Information and Internet Policy (TPRC), Arlington, 2011.
[46] D. Trossen, M. Särelä, K.R. Sollins, Arguments for an information-centric internetworking architecture, Computer Communication Review 40 (2) (2010) 26–33.
[47] Y. Wang, K. Lee, B. Venkataraman, R.L. Shamanna, I. Rhee, S. Yang, Advertising cached contents in the control plane: Necessity and feasibility, in: Proceedings of Workshop on Emerging Design Choices in Name-Oriented Networking (in conjunction with IEEE INFOCOM), Orlando, USA, 2012.
[48] I. Psaras, R.G. Clegg, R. Landa, W.K. Chai, G. Pavlou, Modelling and evaluation of CCN-caching trees, in: Proceedings of 10th Conference on Networking – volume Part I. Networking'11, Springer, Berlin, Heidelberg, 2011, pp. 78–91.
[49] L. Muscariello, G. Carofiglio, M. Gallo, Bandwidth and storage sharing performance in information centric networking, in: Proceedings of ACM SIGCOMM Workshop on Information-Centric Networking, ICN '11, ACM, New York, NY, USA, 2011, pp. 26–31.
[50] G. Carofiglio, M. Gallo, L. Muscariello, D. Perino, Modeling data transfer in content-centric networking, in: Proceedings of Teletraffic Congress, ITC '11, ITCP, 2011, pp. 111–118.
[51] J. Rajahalme, M. Särelä, P. Nikander, S. Tarkoma, Incentive-compatible caching and peering in data-oriented networks, in: Proceedings of ACM CoNEXT Conference – ReArch Workshop, 2008.
[52] G. Carofiglio, M. Gallo, L. Muscariello, ICP: design and evaluation of an interest control protocol for content-centric networking, in: IEEE INFOCOM Workshop on Emerging Design Choices in Name-Oriented Networking, Orlando, USA, 2012.
[53] S. Oueslati, J. Roberts, N. Sbihi, Flow-aware traffic control for a content-centric network, in: Proceedings of IEEE Infocom, 2012.
[54] J. Rajahal, M. Särelä, K. Visala, J. Riihijärvi, On name-based inter-domain routing, Computer Networks 55 (2011) 975–986.
[55] S. DiBenedetto, C. Papadopoulos, D. Massey, Routing policies in named data networking, in: Proceedings of ACM SIGCOMM Workshop on Information-Centric Networking, ICN '11, ACM, New York, NY, USA, 2011, pp. 38–43.
[56] M. Baugher, B. Davie, A. Narayanan, D. Oran, Self-verifying names for read-only named data, in: Proceedings of Workshop on Emerging Design Choices in Name-Oriented Networking, 2012.
[57] Privacy and Security Working Group, Identity in an information-centric internet, White paper, CFP Privacy and Security Working Group, MIT, April 2008.
[58] D.D. Clark, D.L. Tennenhouse, Architectural considerations for a new generation of protocols, SIGCOMM Computer Communication Review 20 (4) (1990) 200–208, http://dx.doi.org/10.1145/99517.99553.