A Naming Scheme for Identifiers in a Locator/Identifier-Split Internet Architecture

Christoph Spleiß, Gerald Kunzmann[1]
Technische Universität München
Department of Communication Engineering
Munich, Germany
{*christoph.spleiss, gerald.kunzmann*}*@tum.de*

***Abstract*—Many researchers agreed that splitting the IP-address into a locator and an identifier seems to be a promising approach for a Future Internet Architecture. Although this solution addresses the most critical issues of today's architecture, new challenges arise through the mapping system which is necessary to resolve identifiers into the corresponding locators. One interesting question is how the naming of identifiers is achieved. In this work we give an overview of a naming scheme for identifiers based on the HiiMap locator/ID split Internet architecture. The naming scheme supports user-friendly identifiers for hosts, content and persons and does not rely on DNS. We furthermore give a possible solution for a lookup algorithm that can deal with spelling mistakes and typing errors.**

***Keywords*-Locator/ID split; Future Internet; Naming schemes; Content Addressing**

I. Introduction

Today's Internet architecture has been developed over 40 years ago and its only purpose was to interconnect a few single nodes. No one expected that the Internet and the number of connected devices would grow to the current size. Measurements show that the Internet continues growing at a tremendous high rate [1]. The address space of the current IPv4 addresses is already too small to address every single node in the Internet and the growth of BGP routing tables sizes becomes critical for the Internet's scalability [2]. While IPv6 is a promising solution for the shortage of addresses, it will probably increase the BGP routing table problem. Besides that, more and more devices connected to the Internet are mobile, such as smart phones or netbooks. However, the current Internet architecture has only very weak support for mobility as the IP address changes whenever a device roams between different access points.

Separating the current IP address into two independent parts for reachability and identification is a possible solution to many problematic issues with today's Internet [3]. With this approach a known identifier (ID) can always be used to reach a specific host, no matter where it is currently attached to the network. However, not only the number of hosts has developed differently than initially expected, but also the way people use the Internet. Today, the focus of the Internet is on accessing a specific piece of information and the host that stores the information os of minor interest. Furthermore, the emergence of social networks, Web 2.0 applications, Voice over IP (VoIP) and instant messaging applications additionally put the person in the focus of interest.

The split of locator and ID thereby offers perfect prerequisites for the support of addressing schemes for content, information and persons. Using this paradigm, an ID is assigned for every host, content and person. A highly scalable and flexible mapping system translates IDs into the corresponding locators. Note that the mapping system is mandatory a part of each locator/ID split architecture.

A crucial question is how to name and assign IDs. As IDs are used as control information in communication protocols and packet headers, they are mostly fixed-length bit strings that can be hardly memorized by humans. In this work we present a flexible and adaptable naming scheme for IDs that can be used to identify hosts, content, persons and is open for future extensions. Our approach is based on the HiiMap Internet architecture [4]. HiiMap provides a highly scalable and customizable mapping system. It does not rely on the Domain Name System and allows each entity to calculate the requested ID on its own.

The paper is structured as follows. In Section 2 we discuss related work and different concepts of locator/ID split architectures. Section 3 describes our approach of a new naming scheme for IDs while Section 4 deals with a lookup algorithm that tolerates spelling mistakes. Section 5 summarizes the results.

II. Related Work

Many proposals dealing with the split of locator and ID have been published so far, but only a few of them discuss how to name IDs. However, almost all of them use a bit-representation of constant length as ID.

A. Host-based approaches

LISP: In contrast to other architectures that are examined in this work, LISP [5] does not separate the identifier from routing purposes. Within an edge network, the normal IP-address still serves as so called *Endpoint Identifier (EID)* and routing address at the same time. While the EID is only routable inside a LISP-domain, an additional set of addresses is used for the routing between different LISP-domains,

[1]G. Kunzmann is now working for DOCOMO Communications Laboratories Europe GmbH, Landsberger Strasse 312, Munich, Germany.

 ISBN:978-1-61208-113-7

which are called *Routing Locators (RLOC)*. RLOCs are the public IP-addresses of the border routers of a LISP-domain, globally routable, and independent of the nodes' IP addresses inside the domain. Whenever a packet is sent between different LISP-domains, the packet is first routed to the *Ingress Tunnel Router (ITR)*, encapsulated in a new IP packet, and routed to the *Egress Tunnel Router (ETR)* according to the RLOCs. The ETR unpacks the original packet and forwards it to the final destination. A mapping system is necessary to resolve foreign EIDs (EIDs that are not in the same domain) to the corresponding RLOCs. However, as in normal IP networks, the EID changes whenever a node changes its access to the network. Furthermore, DNS is still necessary to resolve human readable hostnames to EIDs.

HIP: The Host Identity Protocol [6] implements the locator/ID split by introducing an additional layer between the networking and the transport layer. For applications from higher layers the IP address is replaced by the *Host Identity Tag (HIT)*, which serves as identifier. The IP address is purely used as locator. The main focuses of HIP are security features. This is why the HIT is a hash value from the public key of an asymmetric cryptographic key pair. Encryption, authenticity and integrity can be achieved in this way. However, the coupling of ID and public key is a major drawback, as the ID changes whenever the key pair changes. Furthermore, HIP solely is a host based protocol and is not suitable for addressing content or persons.

HIMALIS: Like HIP, the HIMALIS (Heterogeneity Inclusion and Mobility Adaption through Locator ID Separation in New Generation Network) [7]approach realizes the locator/ID split by introducing an extra layer between network and transport layer, the so-called Identity Sublayer. HIMALIS can use any kind of addressing scheme for locators and supports security features based on asymmetric keys. However, it does not burden the ID with the semantic of the public key. HIMALIS uses domain names as well as host IDs to identify hosts. In contrast to other approaches, a scheme how to generate host IDs out of the domain name using a hash function is shown. However, they use multiple databases for resolving domain names and hostnames to IDs and locators. Furthermore, it is again only a host based protocol.

B. Content-based approaches

Contrary to the host based approaches, the **NetInf** (Network of Information) architecture shows how locator/ID separation can be used for content-centric networking [8]. By introducing an information model for any kind of content, NetInf allows fast retrieval of information in the desired representation. Thereby, each information object (IO) includes a detailed description of the content and its representations, with locators pointing to the machine that stores the information. The ID is assigned to the IO and is composed out of hash values of the content creator's public key and a label created by the owner. In order to find a specific IO, the creator's public key and label must be known exactly.

Another Future Internet Architecture focusing on content is **TRIAD** [9]. One key aspect of TRIAD is the explicit introduction of a content layer that supports content routing, caching and transformation. It uses character strings of variable length as content IDs and uses the packet address solely as locator.

C. Hybrid approaches

A proposal for a Next Generation Internet architecture that supports basically any kind of addressing scheme is the **HiiMap** architecture [4]. Due to the locator/ID separation and a highly flexible mapping system, HiiMap allows for addressing hosts as well as content and is still open for future extensions and requirements. In the following we use the term *entity* for any addressable item.

The HiiMap architecture uses never changing IDs, so called UIDs (unique ID) and two-tier locators. One part of the locator is the LTA (local temporary address) that is assigned by a provider and routable inside the provider's own network. The other part is the gUID (gateway UID). This is a global routable address of the provider's border gateway router and specifies an entrance point into the network.

HiiMap splits the mapping system into different regions, whereby each region is its own independent mapping system that is responsible for the UID/locator mappings of entities registered in this region. The mapping system in each region consists of a one-hop distributed hash table (DHT) to reduce lookup times. As DHTs can be easily extended by adding more hosts, the mapping system is highly scalable. In order to query for UIDs which regions are not known, a region prefix (RP) to any UID is introduced. This RP can be queried at the so-called Global Authority (GA), which resolves UIDs to RPs. The GA is a centralized instance and acts as root of a public key infrastructure, thus providing a complete security infrastructure. As RP-changes are expected to be rare, they can be cached locally.

Like other approaches, HiiMap uses fixed length bit strings of 128 bits as UID. As plaintext strings are not feasible as UIDs due to their variable length, a naming scheme is necessary to assign UIDs to all kinds of entities. Thereby, the existing Domain Name System is to be replaced by the more flexible HiiMap mapping system.

III. New naming scheme for Identifiers

In this section we introduce a naming scheme for IDs that is suitable to address basically any entity and that can be generated out of human friendly information. Although we use the HiiMap architecture exemplarily, this approach can also be adapted to other locator/ID split architectures.

 ISBN:978-1-61208-113-7

Type	input to Hash(name)	Ext 1	Ext 2
static host	plain text domain name	hash of local hostname	service
non-static host	global prefix assigned by provider	hash of local hostname	service
content	plain text content name	child content	version number
person	first + last name	random	communication channel

Table I: Content of UID fields corresponding to different types

A. General Requirements for Identifiers

When introducing a Future Internet Architecture based on locator/ID separation, the ID has to fulfill some mandatory demands. In the following we sum up general requirements for IDs proposed by the ITU [10]:

- The ID's namespace must completely decouple the network layer from the higher layers.
- The ID uniquely identifiers the endpoint of a communication session from anything above the transport layer.
- The ID can be associated with more than one locator and must not change whenever any locator changes.
- A communication session is linked to the ID and must not disconnect when the locator changes.

In addition to the ITU we add further requirements:

- An ID must be able to address any kind of entity, not only physical hosts.
- Every communication peer can generate the ID of its communication partner out of a human readable and memorable string.
- The ID is globally unique, but it must be possible to issue temporary IDs.
- The registration process for new IDs must be easy.
- IDs must be suitable for DHT storage.

While some of these aspects mainly affect the design of a Future Internet Architecture based on a locator/ID split, some issues are directly related with the naming of IDs.

B. Generalized Identifier

As IDs are used in the transport layer protocol to determine the endpoint of a communication, we cannot avoid using fixed-length bit strings to realize packet headers of constant size. In combination with DHTs, which also require fixed-length bit strings, the usage of a hashing function is obvious. In contrast to other approaches, which compose the ID of one hash value only, we split the ID in several predetermined fields whose purposes are known to all entities.

In the following we introduce a generalized scheme how to compose global unique IDs (UID) for any entity and give concrete examples how to name hosts, content and persons. Our scheme allows storing all these IDs in the same mapping database and is yet flexible enough to support different databases for different types of IDs.

Figure 1 shows the generalized structure of an ID, which is composed of a region prefix (RP) and an UID. The UID consists of a type field (T), the hash value of a human friendly name for the entity to be identified as well as two extension fields (Ext 1 and Ext 2). The UID is stored in the mapping system of a specific region, denoted by the RP.

The type field T denotes to which type of entity the UID belongs to. As T contains the most significant bits (MSB) in the UID, it is possible to map different ID types to different databases in the mapping system. We suggest 128 bits for the UID, whereby 4 bits are used to determine the type, 76 bits are assigned for the hash value, 32 bits for Ext 1 and 16 bits for Ext 2. In the following we show realizations for applying UIDs to different types: host, content and persons. Table I gives an overview how the UID is composed according to the type of entity. Each part is described in detail in the following subsections. Note that our scheme is not limited to these types, but can easily be extended.

Figure 1: UID with regional prefix RP

C. Identifiers for Hosts

IDs for hosts are the most common use case today and DNS is used to resolve hostnames to IP addresses in order to access a specific machine. The hostname, or FQDN (full qualified domain name), which specifies the exact position in the tree hierarchy of the DNS, can be roughly compared to the ID in a locator/ID separated Internet architecture. However, the FQDN is not present in any lower layer network protocol and is solely used in the application layer.

Similar to today's hostnames, we introduce a hierarchy to our UIDs. However, contrary to FQDNs, our scheme is limited to two hierarchical levels: a global part and a local part. While the global part is used to identify a whole domain, e.g. a company or an institute at a university, the local part is used to identify single machines within this domain. Note that the term *domain* does not refer to a domain like in today's DNS hierarchy. A domain in our solution has a flat hierarchy and simply defines an authority for one or more hosts. We differentiate between two different types of host UIDs:

1) Static Host Identifier: Static host UIDs are never changing IDs that can be generated by hashing a human readable hostname. Their main purpose is for companies or private persons that want to have a registered UID that is always assigned to their host or hosts.

Hash: The domain name part of the plain text hostname is used to generate the hash field of the UID.

 ISBN:978-1-61208-113-7

Ext 1: The hash value of the local host name is used to generate this field. A local hostname is unique inside a specific domain. An example for a local hostname could be *pc-work-003* and *mydomain.org* as its domain name.

Ext 2: The Ext 2 field is used to identify a specific service on the host. It can be compared to today's TCP or UDP ports. However, specifying a value in Ext 2 is not necessary when requesting the locator for a specific host from the mapping system and can therefore be set to zero. As the host is precisely identified with the global and local UID part, it is not necessary to store identifiers for each service of the host as they would all point to the same locator. Instead, Ext 2 is set to zero in the UID when querying the mapping system and filled with the specific service identifier when actually accessing the node.

For privacy reasons it is possible not to publish the UID for a private host in the global mapping system but only in a local database. For a single point of contact it is possible to use an UID with Ext 1 set to zero, which points to e.g. a login server, router or load balancer which forwards incoming requests to internal hosts.

Note that the host has to update its mapping entry pointing to new locator(s) upon a locator change.

2) Non-static Host Identifier: Contrary to static host IDs and the basic idea of never changing UIDs there will always be the need for non-static host UIDs, i.e. IDs that do not have to be registered, that are assigned to a host for a specific time and that are returned to the issuer if no longer needed. An example can be a private household with a DSL or dial-up Internet connection and a few hosts connected through a router. Each host needs its own, distinct UID to make connections with other hosts in the Internet, but it does not need to have a registered, never changing UID if no permanent accessibility is needed.

Hash: The global part is assigned to the router or middlebox that provides Internet access to the other hosts during the login process. It can be compared to the advertisement of an IPv6 prefix. The global part is valid as long as the customer has a contract with its provider. A new global part is assigned if the customer changes its provider. Yet, the transfer of a global UID part between different providers should be possible. In order to assign non-static UIDs to customers, each provider holds a pool of global UID parts. The mapping entry for a specific dynamic host UID is generated by the corresponding host immediately after assignment and whenever its locator changes. However, each host with no static UID assigned must proactively request a non-static host UID, either by its provider or router and middlebox, respectively. Note that the global part of a non-static host UID does not consist out of the hash value of a plaintext string and can therefore not be computed.

Ext 1: The local part of the dynamic UID is generated from the local hostname of a machine.

Ext 2: Identical to the static host UIDs.

D. Identifiers for Content

As the focus of the users in the Internet is shifting from accessing specific nodes to accessing information and content, different approaches towards a content-centric network have been made as shown in Chapter II. By applying the idea of information models, like the NetInf approach, to our naming scheme, each content, which can be e.g. a webpage or an audio or video file, gets its own distinct UID. Hereby, the UID does not point to the data object itself but to the information model of the content that has a further description and metadata stored.

Hash: For generating the UID of content we have to use a meaningful name that can describe the corresponding content or information. While this is indeed quite a difficult task, possible solutions could be e.g. the name of a well-known newspaper like *nytimes* which refers to the front page of the New York Times online version. Similar, the name of an artist could refer to an information object where albums or movies are linked.

As the spelling of the content description is not always exactly known, we suggest a lookup mechanism that can cope with minor spelling mistakes in the next chapter. In our proposal this plaintext name is used as input to a known hash function to generate the hash part of the UID.

Ext 1: This field is optional and can be used to access some more specific parts of the content or information that is directly related with the main object. This can be e.g. one specific article from a newspaper site or one specific album or piece of music from an artist. Ext 1 can help to avoid downloading a maybe bigger object description of the main content to gather the desired information. Another benefit is that each child object has its own locator and therefore can be stored on different locations while still being accessible through its parent UID. This is not possible today as e.g. the URL of a newspaper article is directly coupled with the host storing the information.

Ext 2: This field can be used to access a specific version of the desired content or information. Like in a versioning system, the Ext 2 field allows the user to easily access any earlier version and the changes made to the information. The actual version can be obtained by setting Ext 2 to zero.

Unlike with host addressing, we cannot simply connect to a locator returned by the mapping system. As the information object is a description of content or information, the requesting application or user has to evaluate the information object and select the desired representation according to the users needs. Thus, the network stack will not evaluate the data received from the mapping system for a content UID query but forward it to the corresponding application.

Note, n case a name, e.g. *nytimes*, refers to both a host (company) as well as content (webpage), the type field is used to differentiate whether a host or a content locator is returned.

 ISBN:978-1-61208-113-7

E. Identifiers for Persons

With the emergence of social networks, Internet-capable devices, Voice-over-IP (VoIP), etc., the need for personal IDs arose, as the *person* inself is moving in the focus of interest. Whenever somebody wants to contact a specific person he is interested in the communication with that person and does not want to care about the device, e.g. which phone or computer the person is currently using for communication. However, the user must have the possibility to choose the communication channel. That can be an email, a phone call, a message on a mailbox, a chat with an instant messenger or a message in a social network and so on.

Hash: The main part of a person's UID consists of a hash value calculated from the person's full name, i.e. first name plus last name. As many people have the same first and last name, the hash value is ambiguous and we need further information to distinguish between different persons.

Ext 1: For this purpose we use a random number for Ext 1 when initially generating a person's UID [11]. This initial generation is not done by the person itself, but is issued by a federal authority and valid for lifetime.

Ext 2: This field is used to specify the communication channel to the corresponding person and has a set of predetermined values, e.g. for email, VoIP, or instant messaging. Note, there are still enough unused values for future needs. According to each Ext 2 value, different locators can be stored in the mapping system, i.e., the Ext 2 value referring to the VoIP account can point to the locator of a VoIP provider or directly to a VoIP phone, the value referring to the mailbox can point to a mail server. The mapping entry for Ext 2 set to zero includes the person's full name and, depending on the person's privacy settings, further details about the person like birth date or current residential address. Ext 2 set to one is used to get the locator of the machine the person is currently working on if the corresponding person agreed to publish this information. Thereby, the communication channel can be signaled in a higher layer.

However, to contact a specific person, not only the person's name but also Ext 1 must be known. There are two possibilities: First, the initiating person knows the correct UID of its communication partner because they have exchanged it (like email addresses today). Second, the holder of a personal UID can agree to be indexed in a directory that is accessible through a personal UID with Ext 1 and Ext 2 set to zero. This directory can be compared to a phone book and stores additional information about all persons that have the same name including their random Ext 1 values.

IV. Identifier assignment and lookup

As each UID is globally unique by definition, it must be ensured that only one entity at a time has a specific UID assigned. It must be further prevented that any entity is hijacking an UID for malicious purposes.

A. Registration and assignment

The registration process for a static host UID can be compared to today's domain names. Whenever a new static host UID shall be registered, the corresponding mapping region checks, like the NIC today, if a specific UID is already registered. If the UID is unused, the mapping region creates an initial entry in the mapping system, including the host's public key. From now on, the host can update its mapping entry at any time, e.g. when it changes its access point. The update message to the mapping system must be signed with the host's private key, thus avoiding the UID to be hijacked. The UID at the node is configured via a system file like /etc/hostname. The owner of a host must proclaim changing the key-pair of a node at the mapping region.

The purpose of non-static UIDs is that they do not need a registration process, as their prefixes are assigned by a provider and therefore belong to that provider. However, it must be possible for hosts with non-static UIDs to change their mapping entries due to roaming although they have not been individually registered. Therefore, whenever a new non-static host UID is assigned, the provider creates the initial mapping entry on behalf of the corresponding host using the host's public key. Then, the host can directly update its locator at any time in the mapping system. However, if a host wants to change its key pair, the node must directly be connected to the provider. Only the provider can verify that this host is allowed to update the public key because the provider can verify the login data. If a host is permanently relocated to another provider, it has to request a new non-static host UID at its new access point. After that it has to initiate the clearance of its old UID.

The procedure for content UIDs is basically the same like for static host UIDs. The content creator has to initially register the hash part of the UID at the mapping system. However, it does not need to register each single content that is provided. Then the content provider can freely create new content that only differs in Ext 1 and Ext 2. A special case is content that is free to public changes like Wikipedia. Here, everybody is allowed to create a new version of the corresponding content that differs in Ext 2 but changes must be verified with the person's key pair.

Unlike with UIDs for hosts or content, UIDs for persons are assigned by an authority of the state. As the personal UID can be used to make transactions and legal contracts, it has to be guaranteed that the UID cannot be abused. Furthermore, it has to be guaranteed that values for Ext 1 are unambiguous. That would not be the case if everybody would generate its own random value for Ext 1. Thus, during the registration process, an authority creates the mapping entry for the person requesting a UID and deposits the person's public key. Then, the person can update and create any entry for Ext 2 on its own. Changing the key pair must be accomplished through the issuing authority.

 ISBN:978-1-61208-113-7

B. Lookup mechanism

The idea of our naming scheme for IDs is based on the fact that each UID can be generated out of a known plain text string with a known hash function and without an additional naming system like DNS. However, as the main part of any UID consists of a hash function, the desired entity can only be found if the plain text string that builds the UID is exactly known and no spelling mistake or typing error occurred. To overcome this drawback, we suggest a lookup mechanism that is based on n-grams in addition to the pure UID lookup.

1) n-gram generation: Although DHTs only support exact-match lookups, it is possible to use n-grams to perform substring and similarity searches. Hereby, each plaintext string is split up into substrings of length *n*, which are called *n-grams*. The n-gram and its hash value is then stored as key/value pair in the DHT [12].

A typical value for n is two or three. With $n = 3$, the content name `nytimes` e.g. is split up into $I = 5$ trigrams h_i with $i = 1, ..., I$: `nyt`, `yti`, `tim`, `ime`, `mes`. Additional to the actual mapping entry indexed by the UID, the hash value $H(h_i)$ of each n-gram h_i is inserted in the mapping system together with the corresponding plain text name P. Thereby, the mapping entry for an n-gram consists of the tuple $\langle H(h_i); plaintext\ string \rangle$ [11]. Although these tuples are stored in the same mapping system like the UID, we suggest using a different database within the mapping system for performance reasons. Whenever the entity changes its location, no updates of the n-grams are necessary, as they do not contain any locator information but only the entitie's plaintext name.

2) Querying UIDs: Whenever querying the mapping system for a specific UID, the first step in the lookup process is using the precalculated (or already known) UID as query parameter. Only if the mapping system is not able to find a mapping entry to the corresponding UID, e.g. because of a spelling mistake, the n-gram lookup is executed. It is up to the user or application if an n-gram based query request is initiated.

In doing so, the second step is to calculate the corresponding n-grams out of the plaintext string and query the mapping system for each n-gram. The mapping system sorts all matching n-grams according to the frequency of the plaintext string and returns the list to the user. With high probability, the desired plaintext has a high rank in the returned list. By further correlating the input string with each returned plaintext string, the result is even more precise [13].

As the user must evaluate the results returned by an n-gram query, the network stack will forward that data directly to the application, which is responsible for correct representation. However, although this feature is similar to Google's "Did you mean...?", the mechanism is not suitable to handle complex queries with semantically coherent terms as Google can do.

V. Conclusion

In this work we presented a new naming scheme for IDs in locator/ID separated Future Internet Architectures. The generalized ID scheme is suitable for basically addressing any kind of entity. We showed examples for hosts, content and persons. Because each UID can be computed out of a human readable plaintext string, an additional naming system like DNS is not necessary any more. Due to the extendible type field, we have the possibility to assign ID-types for e.g. mobile phones, sensors or even cars or abstract services that provide any functionality to a user. Because IDs are independent from locators, a communication session is not interrupted upon an access point change. Furthermore, by introducing an n-gram based extended lookup mechanism we are able to cope with spelling errors and typing mistakes, thus improving the quality of experience for the user.

References

[1] ISC, "The ISC Domain Survey," http://www.isc.org/solutions/survey, Internet System Consortium, 2010.

[2] A. Afanasyev, N. Tilley, B. Longstaff, and L. Zhang, "BGP routing table: Trends and challenges," in *Proc. of the 12th Youth Technological Conference High Technologies and Intellectual Systems¨*, Moscow, Russia, April 2010.

[3] B. Quoitin, L. Iannone, C. de Launois, and O. Bonaventure, "Evaluating the benefits of the locator/identifier separation," in *Proc. of 2nd ACM/IEEE Internat. Workshop on mobility in the evolving Internet architecture*. ACM, 2007, pp. 1–6.

[4] O. Hanka, G. Kunzmann, C. Spleiss, and J. Eberspächer, "HiiMap: Hierarchical Internet Mapping Architecture," in *1st Internat. Conf. on Future Information Networks*, 2009.

[5] D. Farinacci, V. Fuller, D. Oran, D. Meyer, and S. Brim, "Locator/ID separation protocol (LISP)," Draft, 2010. [Online]. Available: http://tools.ietf.org/html/draft-ietf-lisp-07

[6] R. Moskowitz and P. Nikander, "Host identity protocol (HIP) architecture," RFC 4423, Tech. Rep., May 2006.

[7] V. Kafle and M. Inoue, "HIMALIS: Heterogeneity Inclusion and Mobility Adaptation through Locator ID Separation in New Generation Network," *IEICE TRANSACTIONS on Communications*, vol. 93, no. 3, pp. 478–489, 2010.

[8] C. Dannewitz, "NetInf: An Information-Centric Design for the Future Internet," *Proc. 3rd GI/ITG KuVS Workshop on The Future Internet*, May 2009.

[9] D. Cheriton and M. Gritter, "TRIAD: A new next-generation Internet architecture," Tech. Rep., 2000. [Online]. Available: http://www.dsg.stanford.edu/triad

[10] ITU, *Draft Recommendation ITU-T Y.2015: General requirements for ID/locator separation in NGN*, 2009.

[11] G. Kunzmann, "Performance Analysis and Optimized Operation of Structured Overlay Networks," Dissertation, Technische Universität München, 2009.

[12] M. Harren, J. Hellerstein, R. Huebsch, B. Loo, S. Shenker, and I. Stoica, "Complex queries in DHT-based peer-to-peer networks," *Peer-to-Peer Systems*, pp. 242–250, 2002.

[13] L. Mangu, E. Brill, and A. Stolcke, "Finding consensus among words: Lattice-based word error minimization," in *6th European Conf. on Speech Communication and Technology*, 1999.

 ISBN:978-1-61208-113-7