Towards a Minimal Core for Information-Centric Networking

Kari Visala[1], Dmitrij Lagutin[1], and Sasu Tarkoma[2]

[1] Helsinki Institute for Information Technology HIIT / Aalto University School of Science
firstname.lastname@hiit.fi
[2] University of Helsinki
sasu.tarkoma@cs.helsinki.fi

Abstract. The question of whether there exists a minimal model for *information-centric networking* (ICN), that allows the bulk of features of various recent ICN architectures to be expressed as independent extensions to the model, is largely unexplored. Finding such *core* would yield more orthogonality of the features, better adaptability to the changing multi-stakeholder environment of the Internet, and improved interoperability between ICN architectures.

In this paper, we introduce the concept of *information space* as a potential solution, which is based on the sharing of information and late binding service composition of decoupled entities as the essence of information-centrism. The specified framework is an abstract model without dependencies to low-level implementation details and achieves minimality by leaving naming and content security outside the core. This approach makes the experimentation of new features above and their implementation below faster and provides a possible evolutionary kernel for ICN.

1 Introduction

There are many clean-slate designs for ICN that aim to fundamentally change the basic suite of protocols used in the current Internet or even replace IP as the waist of the hourglass shaped layered stack of dependent protocols. Such proposals include CCN [1], DONA [2], NetInf architecture [3], and PSIRP/PURSUIT [4]. A survey of ICN research can be found in [5].

These systems try to take advantage of the experience gained from observing the evolution of the current Internet and design features of the architectures not as an aftertought, which has been the case so far [6]. Features like *naming and addressing*, *security*, *searching and rendezvous*, *scoping of data*, *caching*, *routing*, and *forwarding* have been integrated in cohesive combinations in these designs.

However, it is difficult for this type of engineered and relatively complex solutions to become popular as the replacement for IP, because a large number of stakeholders have their own heterogeneous, changing requirements for the network. For example, IP can be boiled down to the hierarchical addressing and a packet format, which do not fix other aspects of the network, but have allowed a large ecosystem of protocols to be built around the basic framework provided by the simple core.

1.1 Advantages of a Minimal Core

Specifying a **minimal** model, a core for ICN, that allows the above-mentioned network functions to be independently developed as extensions to the core (while they already exploit the advantages of information-centrism), would yield the following three main advantages:

Orthogonality of network functions, which would work synergistically in any combinations. For example, a routing mechanism benefits from caching or a rendezvous function can utilize the security solution etc. A minimal core would also be easier to implement in different environments, including faster prototyping, and its optimizations would be maximally reused by the auxiliary functions depending on the core.
An alternative approach to the ubiquitous core would be to simply specify multiple modular functions independently, but this kind of solution would produce orthogonal concepts only as a coincidence. The separate functions would always be specified relative to something else and it would not be possible to combine them easily, because they have nothing in common. They would not benefit from the *network effect* provided by a core.
The core of the ICN architecture could be compared to the role of lambda calculus for functional programming languages: By mapping higher-level concepts to the minimal core instead of directly implementing them, we can get well-defined, tested, orthogonal semantics as the core specification will be reused and the features work together similar to compatible Lego bricks that can be assembled in various ways. Formal semantics for the core also provide a framework for specification of and reasoning about network functions.

Adaptability to changing environment and requirements by multiple stakeholders is a prerequisite for the formation of a stable evolutionary kernel [7]. Handley's observations on the Internet should not be viewed as a motivation for a new top-down design that tries to cater for everyone, but as an argument of what type of protocol can in general become the ubiquitous glue of the Internet. The view that evolution in open markets determines the deployment of protocols, is further supported by the findings in [8].

Interoperability between different ICN architectures becomes easier, if they can agree on a common core concepts without fixing different monolithic organizations for their functions.

1.2 Design Goals for the Core

In general, there seems to be a trend to move towards more **abstract** interfaces and protocols that allow the applications to modularly express their intent using stable concepts on the level of the application logic and use late binding to the actual protocols used [9]. The model exposed to higher layers should not have dependencies to arbitrary low-level implementation details such as the use of packet switching. For example, a database or a file system, which do not have the concept of a packet, could be used to implement ICN locally.

ICN is often motivated by the need to optimize information dissemination, content security, and the receiver-driven communication pattern that allows the tackling of unwanted traffic. In addition, the high-level functional goal of ICN can be stated to implement a network layer substrate that allows the *information-centric paradigm* (ICP) to be used on the application layer with little "impedance mismatch" between the layers.

We have so far given an argument for the need of an abstract, minimal core for ICN. The remaining contributions of this paper are:

1. The definition of our core in Sect. 4 based on our view on the essence of ICP briefly covered in Sect. 2 and the argument why certain functions such as naming and content security should be excluded from the core in Sect. 3,
2. a concrete, example node architecture explained in Sect. 5, which is not meant to be efficient and realistic implementation, but to support the claim that our approach is feasible, and
3. discussion on how to develop new features on top of the core and what advantages does this have compared to the monolithic approach in Sect. 6.

2 Information-Centrism

A typical ICN architecture supports a communication pattern, where the receiver needs to initiate the communication event by subscribing a named data publication. The granularity of the named entities is more fine grained than hosts identified by IP addresses and the names are location independent. After created, publications are typically immutable associations between their name and content. They do not have a destination like messages, and can be cached and received via multiple routes. ICN is in contrast to the message-based approach, where the sender triggers a (remote) continuation, which may change the state of the destination object and spawn a dialog between the two endpoints.

In many current protocols the roles of the communicating entities are typically coupled at the design time and global invariants are easy to maintain as the degrees-of-freedom of the architecture are limited. These invariants are often not explicitly documented in the code, but exist mostly in the head of the developer as assumptions about how the system works as a whole. For example, a connection abstraction can be built between the endpoints, but already giving an identity to data allows interpretation. The decoupling of the acts of receiving and sending of information in space and time can be taken further by allowing more indirection in the addressing [10] or even interaction between applications that do not share anything in common on the protocol level [11].

The variable part in a communication event, the data, is needed to convey information over the channel. The possible assumptions made about the communication channel form a partial order based on the entailment relation. At the very minimum, some assumptions must be shared about the data itself, e.g. typing information, to make the communication meaningful for the applications.

Even though ICP could be approached symmetrically to message-passing by naming data instead of continuations [12], we choose here to reduce the core functionality to the absolute minimum. The idea is that the core can be extended by new features that add assumptions about the communication channel analogous to the development of transport abstractions. The semantics provided by the network are reflected as metadata publications that can specify, for example, *scopes* under which publications adhere to more specific *distribution strategies*.

Weak core semantics encourage explicit "plumbing" of assumptions about the data as machine-readable metadata, which reduces coupling between communicating entities. This supports dynamic service composition, where loosely coupled applications are independently developed and may only indirectly share the knowledge of some data types. The result is not a top-down framework or a pattern of interaction, where each component has its well-defined place, but the late binding of new functionality based on shared metadata. The goal is to utilize all existing functionality in a system by allowing their combination in new ways not available in fixed configurations.

The components sharing information produce in parallel derivations based on the information already available, similar to the *blackboard* architecture, and solutions emerge dynamically. Information items shared can be interpreted as data or *intentions*, such as *subscriptions*, that instruct the subsequent execution.

For example, Alice with a digital wallet app for her mobile phone with Near Field Communication capability enters a store that does not support the same protocol for mobile payments. However, the store's WiFi network automatically shares metadata about the store with Alice's phone so that it is able to load from the Internet the necessary helper functions to semantically bridge the information provided by both the wallet app and the store.

We informally characterize the model of computation behind our core as:

1. Distributed, parallel execution based on
2. shared data, metadata, and instructions using
3. the metadata to guide dynamic service composition of decoupled functions.

In addition to the model exposed to higher layers (such as addressing), an internetworking architecture needs to define an interface between networks (e.g. packet format) to allow interoperability between heterogeneous implementations.

3 Naming and Content Security

Naming is a central design element in most of the ICN architectures as it pervades most aspects of the system. Typically, naming and routing designs are coupled. For example, the names may contain restrictions such as hierarchical structure or embedded location information, which make it possible to optimize routing

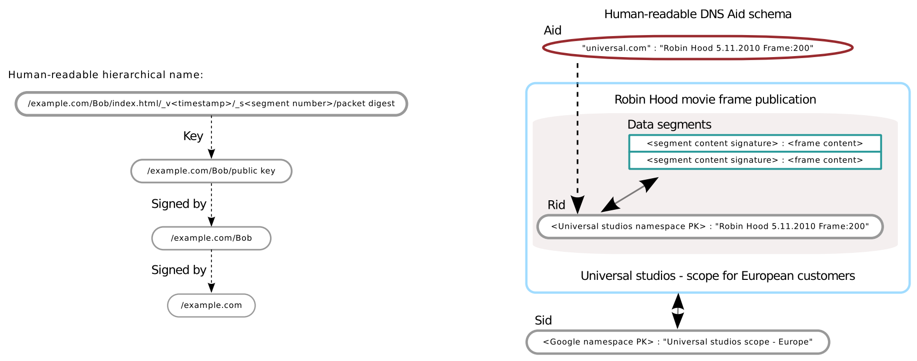

Fig. 1. CCN (on the left) and PURSUIT (on the right) naming side-by-side

based on them. There are also countless other dimensions in which different identifier structures and semantics can be compared such as the lifetime of the identifiers, (im-)mutability of objects, trust model, human-readability etc.

For example, CCN [1] uses opaque, binary objects that have an explicitly specied number of ordered components to name the content chunks hierarchically. These names allow subscriptions that match simple sets of publications and at the same time allow naming of data chunks that do not yet exist.

Another example is the realization of the PURSUIT functional model in [13]. It uses *self-certifying* DONA-like [2] *rendezvous identifiers* (Rids), which are (P, L) pairs. P is the public-key of the *namespace* owner of the identifier and L is a variable length label. CCN and PURSUIT names are shown in Fig. 1.

Both in CCN and PURSUIT, all content chunks can be authenticated with public-key signatures by storing a signature that binds the name and the content in each data packet. The integrity check is possible to do independently for each packet as the PURSUIT Rids contain the public key of the namespace the Rid belongs to. Such identifiers are called *self-certifying*. In CCN, on the other hand, signed data packets are *publicly authenticatable* by containing enough information to allow the retrieval of the public key necessary to verify it. In CCN, the trust model and names are contextual as there does not exist a single entity that everyone trusts for every application. Keys are just CCN data and content can be used to certify other content.

PURSUIT introduces the additional concept of *scope* and every publish operation also determines the scope in which the publication takes place. The scoping of information is orthogonal to the information structures formed by naming schemes and data items refering to other data items. Instead, the scope determines the distribution strategy of the publications inside it. This has the additional benefit that the subscriber can separate its trust on the distribution strategy from the trust on the content. For example, inside untrusted scopes, 3rd party data sources may falsely advertise data that they do not intend to serve in order to cause the data to be unavailable. CCN cannot solve this problem in a general way by using data sources with the credentials from the original publisher as this restricts the use of 3rd party data sources opportunistically.

The naming and content security solutions in the two ICN architectures covered here are different and have their own weaknesses. In PURSUIT, content security is coupled with the names, which makes it impossible to use the names as long-term identifiers. Rendezvous, routing, and forwarding are performed in phases for typical operations and the low-level model leaves little leeway for different types of interoperable implementations. These two aspects make the whole architecture monolithic and difficult to deploy. Also, a rendezvous phase is needed before the subscription of all dynamic publications to determine the correct data source.

In CCN, the structure of the hierarchical namespace is restrictive and the routing of interest packets directly based on the names does not scale well as the largest Internet routing table contains only $4 * 10^5$ routes while there are already $9.5 * 10^7$ generic top-level domains [14]. Also, the subscription of data based on the prefix matching cannot solve locally the problem of the most recent version. CCN tries to incorporate the trust model of SDSI/SPKI for the key management problem of long-term identifiers on the network level, but this type of single solution cannot work for many applications. The lack of global secure identifiers in CCN also increases the complexity of the management of the names at the higher layers. CCN security model does not differentiate the orthogonal concepts of trust to data and trust to the communication channel, which causes availability of data to become a problem, if data is falsely advertised.

Both of the architectures covered here have chosen the naming and content security to be the central design element, maybe following the success of IP, but as the above problems show, the offered solutions cannot satisfy all stakeholders. In the Sect. 4, we present a core for an ICN architecture that does not include these aspects, but allows multiple approaches to naming and content security, that possibly utilize the information-centric problem solving itself, on top of it.

4 Information Space

The abstract, minimal core for ICN, that we define here, does not require any specific API, protocol message format, or a node architecture such as blackboard, but specifies the concept of an *information space* (IS) and its semantics. If a concrete system can be mapped to the concepts of the IS and adheres to its axioms, we say that the system implements the IS. In Sect. 5 we specify an example API and a node architecture, but they are mostly implementation details from the point of view of the core.

4.1 Definition

An IS *trace* is defined to be a six-tuple $(P, R, I, d, i, <)$, where R is a countable set of *receive* events over the whole life time of the IS and I is a countable set of instances of interfaces to the IS. Receive events form the output of the IS. P is a countable set of *publish* events and a pair $(P_c \subset P, I_c \subset I)$ forms the external context of the IS. $d : E \to \mathbb{N}$ (, where $E = (P \cup R)$) is a function that assigns a natural number to each externally observable event of the IS. The contents of the publication related to the event are encoded as this number. Another function $i : E \to I$ maps events to the interfaces in which they took place.

We ground the IS model of distributed computation in the physical concept of spacetime following the *actor model of concurrency* [15]. We assume that each interface of the IS follows its world line and events $e \in E$ occuring at the interface can be assigned a point $L(e)$ on the path of the interface. However, we abstract the physical description into a *strict partial order* relation $<$ over the set of events E. $<$ is intensionally defined to respect the laws given below and it must agree with the causal structure of the spacetime, that is $\forall a, b \in E. a < b \Rightarrow$ there exists a future-directed non-spacelike curve from $L(a)$ to $L(b)$. Different structures for the spacetime can be used here as long as all observers agree on the order of causal connection between two events, which is the case for relativistic frames of reference.

Events must obey the *actor laws* [15]. That is, $\forall e_1, e_2 \in E$ the sets $\{e \mid e_1 < e < e_2\}$ and $\{e \mid e < e_1\}$ are finite. In addition, $\forall x \in I$ the set $\{e \mid i(e) = x\}$ is totally ordered by the relation $<$. For the bare IS, we add one more law:

$$\forall r \in R\ \exists p \in P\ (d(r) = d(p) \wedge p < r) \qquad \textit{(No publication from nothing (NPN))}$$

Finally, we define IS to be a system, whose possible behavior, when run in a given context C, is a set of IS traces. These axioms intensionally describe the IS with weak semantics. The idea is to define subtypes of IS that enrich the structure with additional constructs and axioms that can, for example, apply to a subset of publications, based on the functionality implemented in the network. For example, stronger consistency between publications could be guaranteed for publications inside a given scope.

4.2 Discussion

IS definition is compatible with actor semantics, which means that applications and network elements can be specified as actors attached to the IS. IS can be overlaid on top of multiple implementation technologies by using actors to bridge information relevant to satisfying global semantics between the systems. The creation of new interfaces is outside the scope of this paper, but it can easily be included in the model. Basic IS delivers data only using *best effort* semantics which allows independent failures of parts of the system. This can be seen as a feature for security functions, that may filter unwanted traffic intentionally. An IS does not have a global state, because defining one would impede the scalability of the implementations.

IS can be interpreted to have *one-time-creation* semantics for each publication. Immutability after creation allows cached information to be used safely locally. Implementations can store publications as soft state and automatically garbage collect old data. Whoever wants a publication to persist, must be responsible for keeping it alive. This adheres to the *fate sharing principle* [16].

There is no concept of subcription in the IS, but different kinds of subscriptions are just publications that are routed based on some search ontology specific strategy and resolved to result sets using the late-binding based problem solving strategy. A high-level query or subscription can be handled in multiple phases as functions joined to the IS translate publications to other publications of lower abstraction level, and at some point, the information can trigger an external side effect such as the forwarding of the information to another node. We claim that both the CCN and PURSUIT naming can be emulated on top of the IS and provide evidence for this in Sect. 6 by showing a partial concrete proposal.

The idea is that side-effects external to the IS, such as forwarding of information, are produced by components, that reflect their capabilities in the IS as metadata. For example, a network description language can be used to describe the topology and routing algorithm of the network. There is no separate concept of metadata, but it consists of publications, that refer to other publications.

Compared to Haggle [11], IS does not limit the data model and searching to information graphs and sets of attributes as we believe that these design choices will produce a bias in the architecture towards certain applications. Compared to the NetAPI [9], IS does not specify a concrete API, but an abstract model based on sharing of information. IS does not assume communication with named resources, but publications are simply published and received. Specifying an API instead of information structures is diametrically opposite to the idea of ICP: In object-oriented paradigm one fixes a stable interface that can have multiple implementations (construction) with hidden data and in ICP one fixes an algebraic data type, whose elements can be shared and consumed (destruction) in new ways by new functions. The lack of naming scheme for publications resembles Linda *tuplespace*, but we neither allow removal of data from the IS nor assume any structure for the data.

5 An Example Node Architecture

As an example, we show a possible language-neutral API that can also be interpreted as link-level protocol, that allow the access to an IS. We call this API an *Universal Information Interface* (UII) and it has the following two operations shown in Haskell-like syntax below:

```
class IS a where
   publish :: a -> Publication -> IO ()
   listen  :: a -> (Publication -> IO ()) -> IO ()
```

An application of the `publish` function with argument *data* is mapped to the publish operation $p \in P$ for which $d(p) = data$ and $i(p) = x$, where the API instance itself is mapped to $x \in I$. The `Publication` type here is simply a chunk of binary data. The `listen` operation registers a callback for receiving publications from the IS and each callback invocation is mapped to a receive operation of the IS similarly to the publish. All other functionality will be implemented on top of the IS by publishing and receiving publications. This allows, for example, extending the core to be able to handle publication of large sets of publications, when it would be impractical to advertise them individually. In this case, the application could publish the individual publications dynamically as a response to certain types of matching subscription publications received.

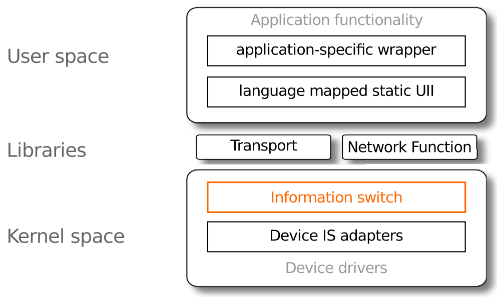

Fig. 2. New transports and network functions can be extended as dynamic libraries over a simple information switch with basic local forwarding semantics

In Fig. 2, an example node architecture, based on a kernel side *information switch* forwarding publications between local components, is shown. The switch itself understands only publications belonging to a simple forwarding ontology. A basic switch could, for example, understand *forwarding directives*, that instruct which interfaces should other publications be sent to. The switch itself could publish a simple map of its interfaces and external helpers would be responsible for mapping higher-level routing protocols to the forwarding directives executed by the switch. Alternatively, the switch can receive the forwarding publications along with the payload publications similar to packet headers. A caching component, for example, could publish a directive to forward everything through it. Even though IC will increase overhead by not exposing low-level optimizations, they would only help with a particular implementation technology.

Devices advertise their metadata and listen to directions via IS adapters attached to the central information switch. The node can be extended by new transports and network functions installed as dynamically linked libraries communicating with the information switch. Each network function implements a new abstraction to the IS, mostly in a layerless fashion and they can be installed on demand, on the fly based on the metadata describing their function. In the diagram, a statically linked, language mapped UII implementation is registered to the local information switch, which implements the kernel sided version of the UII. We assume that the raw UII will not be often exposed to application logic, but specific information ontologies are mapped to easy-to-use wrapper interfaces expressed using the natural idioms of the used language and application domain.

6 Development on Top of the Core

In the ICP, the development of new data vocabularies on top of the core is supposed to be mainly definition of 1. types and ontologies for data and metadata and 2. their semantics based on the framework provided by the IS definition. This is in contrast to the specification of protocol sequences between endpoints with specific roles. For example, instead of specifying a rendezvous architecture as interconnected nodes, we should fix the kind of publication types needed to exchange information about rendezvous.

Avoiding non-local design invariants is not always possible. For example, the efficient operation of a distributed hash table (DHT) is based on the top-down design of its routing algorithm. It is difficult to imagine how the individual nodes could be used as a part of a completely different scheme. Here the reusable pattern is the DHT routing as a whole and not the individual nodes. Therefore, it would be natural to parametrize the information structures used by the DHT to allow other applications to utilize the capabilities of the DHT.

The number of potential interactions grows quadratically as a function of the number of components, which were previously limited by the architectures they were part of. Turning the silent design-time invariants into explicit runtime metadata increases implementation effort upfront, increases overhead, and the modular functionality lacks the immediate security and optimization possibilities offered by monolithic solutions. However, as the amount of information-centric functionality grows past a critical mass, the network effect between the interworking components may dominate the additional initial cost incurred.

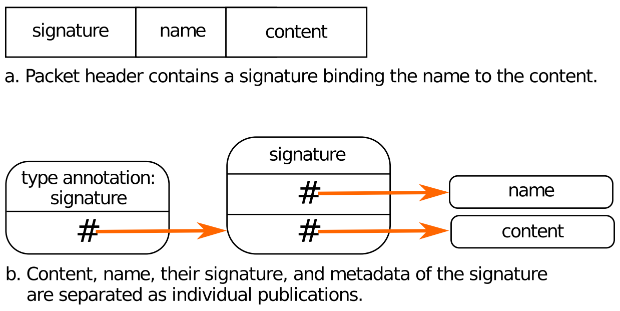
signature
name
content
a. Packet header contains a signature binding the name to the content.
type annotation: signature
#
signature
#
#
name
content
b. Content, name, their signature, and metadata of the signature are separated as individual publications.

Fig. 3. Low-level packet format vs. ontology

As a concrete example of how a low-level, non-information-centric, monolithic ICN architecture feature can be translated on top of our core, we use a simplified version of content integrity based on cryptographic signatures of the content. On the top in Fig. 3, we have a supposed packet format, that specifies that the signature of the content is carried in the header of the packet. Below, one possible mapping of this functionality to publication types on top of IC is given. A separate publication is used to store the signature of the content and it contains a cryptographic hash of the payload as a pointer. Similarly, a *run-time type annotation* publication describes the type of the signature publication by referring to it. Two of the above publications need to include the pointer to the data they talk about instead of the tight coupling of a packet format and the associated design-time assumption that the header and payload are connected.

We have now defined the needed publication types without fixing any low-level details such as the use of packet switching. The translated solution has the advantage that new, orthogonal security mechanisms can be simultaneously deployed as *aspects* or the signatures could be completely dropped in some environments. There is also more freedom in the implementation: For example, the signature publications could be mapped back to headers on the wire, or they could as well be cached or transported by some other means. Also, because the signatures are now ordinary publications, they can orthogonally benefit from other functions implemented such as error correction metadata and late-binding based information-centric problem solving. The core-based solutions are also more interoperable, as the security mechanism is specified abstractly and not entangled with the implementation of the core.

6.1 Managing Consistency with Scopes

In the object-oriented paradigm, when the remote state of a service is protected by an interface, the state is easy to keep consistent locally. In the information-centric setting, part of the system state can be shared in the IS as publications interpreted as *factoids*. In the Internet, the source of information can be a malicious party. These reasons contribute to the problem of managing consistency. Transactions are one possible solution to keep a shared state consistent by making the write operations potentially fail. However, this approach is too heavy to be required for all communication. Therefore, we assume that the publish operations always succeed and IC itself does not guarantee any inter-publication relationships. Different types of consistency models can be built on top of the core based on application needs using techniques such as automatic merging.

The basic IS can be enriched with scopes, that each contain a subset of the publications. Scopes can set constraints on their contents based on internal consistency of the data, trust, distribution strategy, or some application logic specific criteria. For example, a distributed algorithm operating on the the data in the IS can produce multiple potential solutions to a problem. Application could then search for the result in the IS and the result of this query could be viewed as one scope containing only the best solution found and related publications consistent with the selected solution. Thus, scopes can be dynamically created by limiting what information is relayed to the application by its request. Scopes can also be represented explicitly as information in the IS and generated by a 3rd party entity, such as a scope implementation in the PURSUIT architecture. For example, a metadata publication could mark which data items belong together.

7 Conclusion

ICN is not just about optimization of information dissemination, but a fundamental change in paradigm providing a natural substrate for information-centric computing. We have shown that basing ICN on a minimal, abstract core has advantages compared to the existing proposals: Using content security as an example of a feature extending the core, we achieved a solution, that does not depend on legacy concepts such as packets and is not entangled in a rigid architecture, but can immediately benefit from the information-centric model.

The benefits of the core still need to be balanced against the increased initial implementation effort, overhead, and the lack of optimization options offered by a monolithic architecture. The manageability of dynamic service composition and the incompatibility of confidentiality with the sharing of information remain open problems even though we sketched an initial solution based on scoping.

References

1. Jacobson, V., Smetters, D., Thornton, J., Plass, M., Briggs, N., Braynard, R.: Networking Named Content. In: ACM CoNEXT 2009 (2009)
2. Koponen, T., Chawla, M., Chun, B.G., Ermolinskiy, A., Kim, K.H., Shenker, S., Stoica, I.: A Data-Oriented (and Beyond) Network Architecture. In: SIGCOMM: Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, vol. 37, pp. 181–192 (2007)
3. Dannewitz, C., Golić, J., Ohlman, B., Bengt, A.: Secure Naming for a Network of Information. In: 13th IEEE Global Internet Symposium 2010 (2010)
4. Trossen, D., Parisis, G., Visala, K., Gajic, B., Riihijärvi, J., Flegkas, P., Sarolahti, P., Jokela, P., Vasilakos, X., Tsilopoulos, C., Arianfar, S.: PURSUIT Deliverable D2.2: Conceptual Architecture: Principles, patterns and sub-components descriptions. Technical report, PURSUIT (2011)
5. Ahlgren, B., Dannewitz, C., Imbrenda, C., Kutscher, D., Ohlman, B.: A Survey of Information-Centric Networking (Draft). In: Dagstuhl Seminar Proceedings (February 2011)
6. Handley, M.: Why the Internet only just works. BT Technology Journal 24(3), 119–129 (2006)
7. Dovrolis, C.: What would Darwin Think about Clean-Slate Architectures? ACM SIGCOMM Computer Communication Review 38(1), 29–34 (2008)
8. Akhshabi, S., Dovrolis, C.: The Evolution of Layered Protocol Stacks Leads to an Hourglass-Shaped Architecture. In: SIGCOMM 2011 (2011)
9. Ananthanarayanan, G., Heimerl, K., Demmer, M., Koponen, T., Tavakoli, A., Shenker, S., Stoica, I.: Enabling Innovation Below the Communication API. Technical report, University of California at Berkeley (October 2009)
10. Stoica, I., Adkins, D., Zhuang, S., Shenker, S., Surana, S.: Internet Indirection Infrastructure. In: SIGCOMM 2002 (2002)
11. Su, J., Scott, J., Hui, P., Crowcroft, J., de Lara, E., Diot, C., Goel, A., Lim, M.H., Upton, E.: Haggle: Seamless Networking for Mobile Applications. In: Krumm, J., Abowd, G.D., Seneviratne, A., Strang, T. (eds.) UbiComp 2007. LNCS, vol. 4717, pp. 391–408. Springer, Heidelberg (2007)
12. Filinski, A.: Declarative Continuations and Categorical Duality. Master's thesis, University of Copenhagen (August 1989)
13. Visala, K., Lagutin, D., Tarkoma, S.: Security design for an inter-domain publish/subscribe architecture. In: Domingue, J., et al. (eds.) Future Internet Assembly. LNCS, vol. 6656, pp. 167–176. Springer, Heidelberg (2011)

14. Narayanan, A., Oran, D.: NDN AND IP ROUTING - CAN IT SCALE? presentation in ICNRG side meeting, IETF-82, Taipei
15. Hewitt, C., Baker, H.: Laws for Communicating Parallel Processes. In: IFIP 1977 (1977)
16. Carpenter, B.: rfc1958: Architectural Principles of the Internet. Technical report, IETF (June 1996)