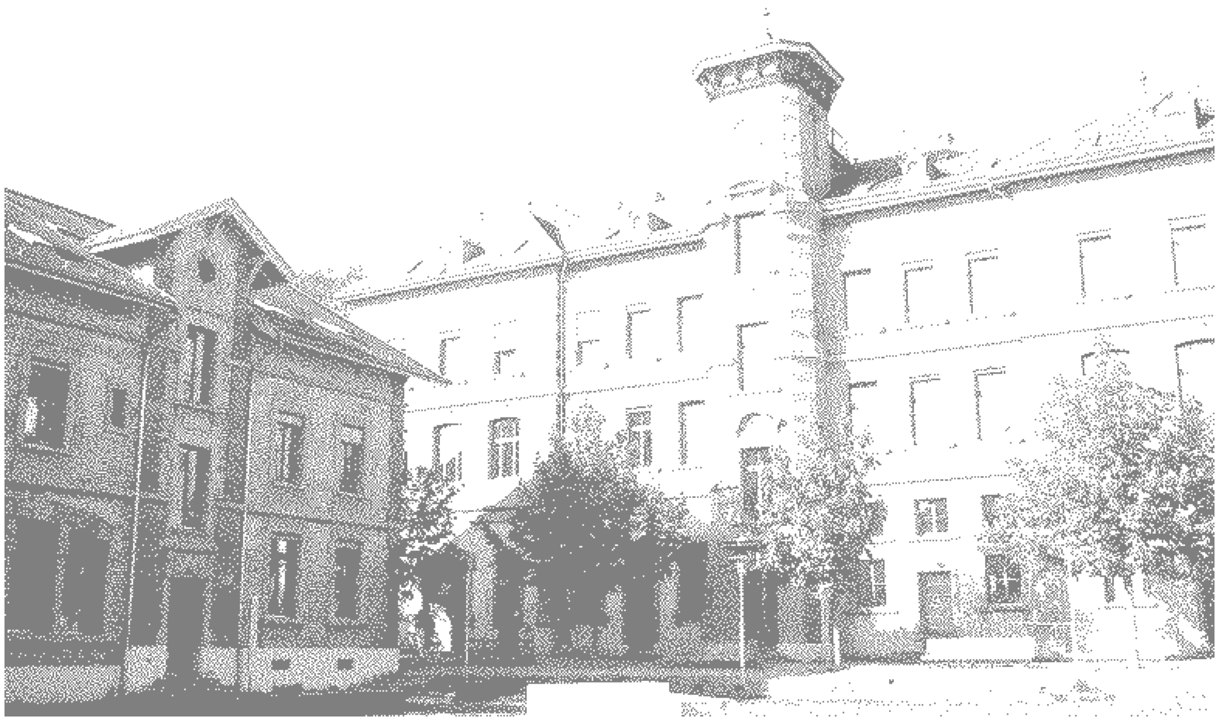
Survey in Service-Centric Networking

D. Mansour, T. Braun

Technical Report IAM-14-004 , 28. November 2014

Institut für Informatik und angewandte Mathematik, www.iam.unibe.ch



Survey in Service-Centric Networking

Dima Mansour, Torsten Braun

Technical Report IAM-14-004 , 28. November 2014

CR Categories and Subject Descriptors:
C.2.1 [Computer-Communication Networks]: Network Architecture and Design;
C.2.2 [Computer-Communication Networks]: Network Protocols;
C.2.3 [Computer-Communication Networks]: Network Operations;
C.2.4 [Computer-Communication Networks]: Distributed Systems;

General Terms:
Design, Management, Measurement, Performance

Additional Key Words:
Service-Centric Networking, Content-Centric Networking, Systematic Survey

Institut für Informatik und angewandte Mathematik, Universität Bern

Abstract

Information-Centric Networking (ICN) considers the content as the building block of the network instead of hosts. This allows clients to request content objects by their names but not by the addresses of their providers. One limitation of ICN is that it does not naturally support services. Service-Centric Networking considers services as first-class citizens in the network shifting the current Internet architecture towards a service-centric style. In this paper, we discuss and compare several approaches in Service-Centric Networking with respect to goals, architectures, naming schemes, and design decisions. We demonstrate the results and the future challenges of those approaches in general to be considered as guidelines for future research in the Service-Centric Networking field.

Contents

1 Introduction

Today's Internet architecture was mainly designed to allow hosts to remotely access resources on other hosts. It follows a host-to-host or host-centric communication model where each host has a name (identity) and an address (location).

This architecture is well-suited for the early use case scenarios of the Internet, but it introduces many limitations in the current use cases [1], which are manifested by the development of the new web applications, the huge increase in the delivered content (e.g. YouTube, BitTorrent), and the need to address mobility and security issues.

Many research trends focus on moving from the current Internet architecture to a new one to meet the new usage requirements of the Internet [2]. All these research works argue that content is what matters in the current Internet, but not where it resides. To approach this vision, the communication framework is changed by binding the client to the content itself instead of the content providers'IP address.

Upon this idea, Information-Centric Networking (ICN) is proposed. It is a networking paradigm that centers around distributed content to be the main building block of the new Internet architecture. The argument here is that named content has become a better abstraction than named hosts for solving today's communication problems. Many projects develop this approach further, e.g., CCN [3], DONA [4], NetInf [5], and PURSUIT[6].

The idea of ICN is a revolution in the networking world. But there is a limitation behind this idea, which is manifested when the client requests services, but not content. ICN supports content, but it does not really support services.

Service-Centric Networking is a new and promising paradigm for the future Internet architecture, which is built on the fact that *services* are better abstraction for current Internet applications than *content*. Content is just a subset of services and what applies to services can easily apply to content, not the other way around. There are some projects concerned with services like SoCCeR [7], CCNxServ [8], and Serval [9]. We are going to give a thorough overview of those approaches, and discuss the pros, cons, and design decisions behind each one to reach a well-built conclusion of what any Service-Centric Networking project should address in the future.

In Section 2, we give a short technical background for some projects in the field of Information-Centric Networking and related naming schemes. In Section 3, we discuss in more detail each of the surveyed Service-Centric Networking projects. In Section 4, we discuss the design decisions behind the architectures and naming schemes of the surveyed projects. Then, we provide guidelines for further research in Service-Centric Networking and we address future challenges. In Section 5, we review and conclude this article.

	DONA	CCN	CBCB
Naming Scheme	Flat	Hierarchical	Attribute-values
Human-readable Names	No	Yes	Yes
Authentication	Self-Certifying	Signature	Signature
Communication model	Find/Data	Interest/Data	Publish/Subscribe
Caching responsibility	RH	Content Router	Limited to the scope

Table 1: Comparison between some ICN projects

2 Information-Centric Networking

Information-Centric Networking (ICN) aims to change the Internet architecture in to make it more efficient and suitable for nowadays communication requirements. We present a short overview of several ICN projects. Table 2 summarizes the main naming and architectural features of those projects.

2.1 Data-Oriented Network Architecture

Data-Oriented Network Architecture (DONA) [4] replaces DNS name resolution with a name-based anycast primitive that sits above the IP layer. The content is associated by a Principal (publishing entity), which is associated with a public key pair. There is a Resolution Handler (RH) for each domain, which is a register for all nodes that are authorized to serve data. The DONA communication model relies on two packets: FIND and DATA. A FIND packet handles the name of the content requested by the client. Client requests are routed by name to find the closest copy in a hierarchical style, which is supported by the RH infrastructure. The DATA packet takes the reverse path of the corresponding FIND packet handling the data.
DONA proposes a flat and self certifying naming scheme based on hierarchically organized name resolution. Each content object in DONA has a name of the form P:L, where P is the cryptographic hash of the owner's public key and L is an owner assigned label. The owner is responsible for the uniqueness and granularity of L. This flat naming scheme satisfies the need of authentication and integrity. When a client asks for a content object with the name P:L, it receives the triple <data, public key, signature>. Then, it can immediately verify that the data did indeed come from the publisher by checking that the public key hashes to P (authentication) and that the signature was generated using that public key (integrity).

2.2 Content-Centric Networking

Another important project is Named-Data Networking (NDN) or Content-Centric Networking (CCN) [10], which is implemented under the name CCNx [11]. In CCN, content objects are identified by names, which follow a hierarchical structure. Content providers publish their content objects by announcing them to the content routers. Those content routers maintain a special type of table called Forwarding Information Table (FIB). This table maps content names to the next hops (faces). FIB tables are similar to ordinary routing tables in current IP network routers, but routing is based on content names rather than host's IP addresses.

In CCN, the client sends an Interest message containing the requested content name, which content routers use to select the appropriate face or faces to forward the request. This Interest packet leaves trails as bread crumbs. When the Interest packet reaches the content provider it returns a Data packet, which goes back in the reverse path to the requester following the aforementioned bread crumbs.

Content routers also maintain another type of tables called Content Store (CS), which serves as a buffer memory (cache) for the retrieved content objects. This is possible due the fact that content objects are self-identified and self-authenticated and might be useful for many clients in a given time window. So content routers first lookup the cache for the requested content before forwarding the Interest packets to next faces.

The hierarchical names in CCN are composed of multiple components arranged in a hierarchy. A component can be any string of arbitrary length. CCN only proposes the structure and anticipates that naming standards will emerge and become standardized through the development of different types of applications. Names also contain information like versions and segment numbers. To provide content authenticity and integrity, name to content mappings are digitally signed and delivered with the content. CCN names are human friendly due to the hierarchical structure (similar to Internet URLs) with support of versioning and chunking data. For example, the name could be: *ccn://SlideShare/ Presentations/Dima/scn/v3/part1*. This name means that we ask for the first part of the third version of the content *ccn://SlideShare/ Presentations/Dima/scn*.

2.3 Combined Broadcast and Content Based Routing

Combined Broadcast and Content Based routing or (CBCB) [12][13] has a publish/subscribe architecture, where publishers publish their contents using messages and subscribers advertise their interests using predicates.The published messages are propagated over a broadcast tree from their sources. The message is delivered to all the client nodes that advertised predicates matching the message using predicates themselves to prune the branches of the broadcast tree to ensure delivery of the message to interested nodes only. Each message (content name)in CBCB *name* is a group of type-attribute-value triplets. For example: suppose that there is a content name: *Wiley.com/Laurie/Info_Cent_Net.pdf*, the corresponding attributes for this name are: *[FileType <String>= pdf, Title <String>=Information-Centric Networking, Author <ListOfString> = Laurie, Organization <String> = Wiley, Year <Integer>=2011]*. These names are used by content providers to publish the content. Clients subscribe to content by sending *predicates* that have a query form. A *predicate* is basically a disjunction of conjunctions on individual attributes. Each *predicate* describes what the client wants by specifying constraints on the content attributes. For example, the request for this content could be:
$[FileType = pdf \wedge Year >= 2011 \wedge Organization = Wiley]$. Content names and predicates are used by the routers to match subscriptions to publications. This naming scheme is unique and helpful since it allows for the description of content to be embedded within the name itself. It enables automatic content discovery and filtering. But, on the downside, this scheme neither ensures name uniqueness nor enforces secure content names.

2.4 Discussion

There are three main naming schemes in ICN: Attribute-Value Pairs Based [13], Flat [4][5][6], and Hierarchical [3]. All three schemes can be aggregated to some extent, to improve routing table scalability. CCN/NDN performs prefix aggregation on hierarchical names.
CBCB uses attribute-value pairs to prune unnecessary branches of the broadcast tree. Predicates with common attributes at a CBCB router can be combined to reduce the number of routing table entries. Flat names used by DONA in the form P:L can be aggregated at the publisher level or can use DHT based lookup services, where storage load is distributed uniformly between the resolution nodes.

Another point to note here is that the use of a cryptographic hash in a content name hides the underlying content semantics from human users and makes the names difficult to remember. This fact makes self-certifying flat names less human friendly. On the other hand, hierarchical and attribute-value pair based naming schemes are more human friendly, because they are easier to remember and they provide more information about the content semantics. But this human friendliness comes at the cost of some challenges: ensuring global uniqueness, security binding, and authenticity[14].

3 Major Research Projects in Service-Centric Networking

There are few ongoing research projects that consider services instead of content as the core principal of the new Internet architecture based on the fact that services are better abstractions than content for Internet communication.
The projects discussed hereafter, represent the current state of the art in the field of service-centric networking. They share many objectives but differ in details such as naming and architecture design decisions. The main goal is the same, which is to achieve a new Internet architecture to better support the deployment, fault tolerance, load balancing, and mobility of services.
We will present objectives, architecture, and naming scheme for each project with some discussions to extract the best requirements in order to build a suitable and strong infrastructure for future works in this field, especially regarding architecture and service names.

3.1 CCNxServ

3.1.1 Objectives

CCNxServ[8] is built on top of a CCN implementation (CCNx) to make CCN support dynamic services. Most of the CCNx functionality and implementation are kept intact (minimal changes) but new components are added to provide service support.

3.1.2 Naming Scheme

Since CCNxServ is based on CCN, it adopts the same hierarchical structure of the names as CCN. The naming scheme is *ContentName + ServiceName*, where *ContentName* is exactly like in CCN. *ServiceName* is the name of the service module that should be invoked on the requested content. For instance, suppose that the client asks for a video content object (BugsLife.mp4) and an advertisement service (ads), which takes the content as a parameter to display an advertisement for five minutes inside the video. The request looks like: *ccnx://BugsLife.mp4 + ad*.

3.1.3 Architecture

The CCNxServ architecture has two essential elements as shown in Figure 1: The first one is the controller CCNxServiceProxy, which is responsible for intercepting Interest messages and making two separate Interest messages: one of them with the content name and the other with the service name. Then, it forwards the files of the content (content file) and the service module (JAR file) to the service framework (NetServ[15]), which is the second element that is responsible for applying the service on the content.

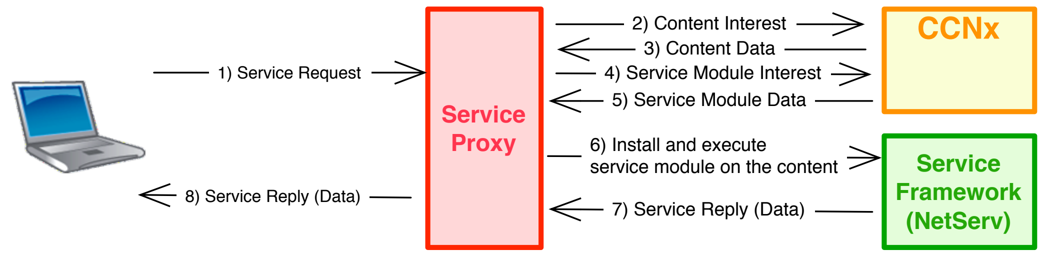


Figure 1: General overview of the CCNxServ architecture in its main elements.

To describe the approach, we follow the process in Figure 1 for the last example (ccnx://BugsLife + ad). We can summarize the process as follows:

1. The client sends the service request to the CCNx network. This request contains the content name and the service name.
2. The control element (ServiceProxy) intercepts the request and creates two Interest packets. It sends the first Interest to the CCNx network. This Interest contains the content name.
3. ServiceProxy fetches the corresponding content file (possibly from the cache).
4. ServiceProxy sends the second Interest containing the service name to the CCNx network to get the service module.
5. ServiceProxy gets the service module as a Jar file of the requested service.
6. Now the ServiceProxy has the content file and the service module. So it installs them on the service framework (NetServ) to execute the service on the content.

7. The result from NetServ is sent back to the CCNx network.
8. The result is sent back from the ServiceProxy to the client via the CCNx network.

Basically CCNxServ was developed as a proof of concept that services can be implemented over CCNx. That is why the implementation of the "framework" is actually specific for the examples written in [8] and the implementation of the "framework" is actually based on the *FileProxy* example that is shipped with the CCNx project [11].

3.1.4 Discussion

There are many advantages of the CCNxServ implementation:

- This architecture allows CCNxServ to benefit from all CCNx functionalities like routing and caching.
- With the adopted naming scheme, it is possible to aggregate services easily. We can invoke a service on the result of another service as a parameter (e.g, *ccnx://content + service1 + service2*). In this case, the result of *ccnx://content + service1* goes as a parameter to service2.
- Merging CCNx and NetServ provides dynamic deployability of services.

But there are some limitations with CCNxServ as well:

- The combination of the CCNxServ and NetServ implementations might be a problem, because the authors did this combination decision with the hope that NetServ, which is initially built for IP networks, can be reworked to support CCN. In other words, CCNxServ project can be fully-implemented only if NetServ supports CCNx.
- The CCNxServ architecture relies on NetServ. This introduces a problem with service implementation diversity. CCNxServ forces all services to be implemented in a unified style that suits service virtualization. That means any CCNxServ services should be implemented in Java, implement a very specific interface, and be available in a jar file. Services cannot be implemented in any language other than Java and all the code of the services should be encapsulated in a Jar file making CCNxServ almost not realistic, because usually services are interfaces to entire systems (*e.g.*, e-banking, Google Maps, Amazon e-store).

- There might be performance issues with the CCNxServiceProxy and the Service Framework components. Each service module needs to be migrated, deployed, and then executed in the NetServ framework. This process will introduce significant performance overhead.

- As noted in [14], there are some legal issues since service code needs to be migrated and executed on different nodes. Not all vendors allow the execution of their services on hosts out of their control.

- With the current CCNxServ implementation, services can accept only one parameter, which is a content object in the form of a file. This is a very simple case and real-world scenarios are much more complicated.

- Both content and service have the same name *prefix*. In other words, a client can not ask for a service on a certain content object unless the service comes from the same content owner.

3.2 SoCCeR

3.2.1 Objectives

The Service Over Content-Centric Routing (SoCCeR) [7] is a Service-Centric Networking project trying to tackle some of the limitations of CCN that stand in the way of adding service support. In CCN there might be redundant content retrievals when the router FIB table has no entry for that specific content object and uses broadcast to find the content object or the content object is not in the router Content Store (CS) and the router FIB table has many route entries for the content object. In those cases the Interest packet might reach many content object replicas, all of which will respond with content packets but only one will reach the client.
This redundancy becomes very expensive (network traffic, memory consumption, CPU time, and energy) when we deal with services, because when a client sends an Interest message requesting a service, all service replicas will execute the service but only one replies to the client. When requesting content, this redundancy is not a big problem because there is no processing involved, but with services it might create significant overhead.
SoCCeR aims to extend CCN to support services by incorporating Ant Colony Optimization (ACO) [16] in service routing. This allows routers to select the best face from the list of FIB faces (satisfying service requests)

by providing them with some parameters like available bandwidth and service load traffic. Briefly, the goal of SoCCeR is to forward service requests to the best available service replica.

3.2.2 Naming Scheme

SoCCeR extends the CCN protocol to support services. The authors' first step was to work on the routing protocol and service instance selection using Ant Colony Optimization ACO. The project supports the same naming scheme as CCN. In this sense, services in SoCCeR do not accept parameters and service names are exactly like content names.

3.2.3 Architecture

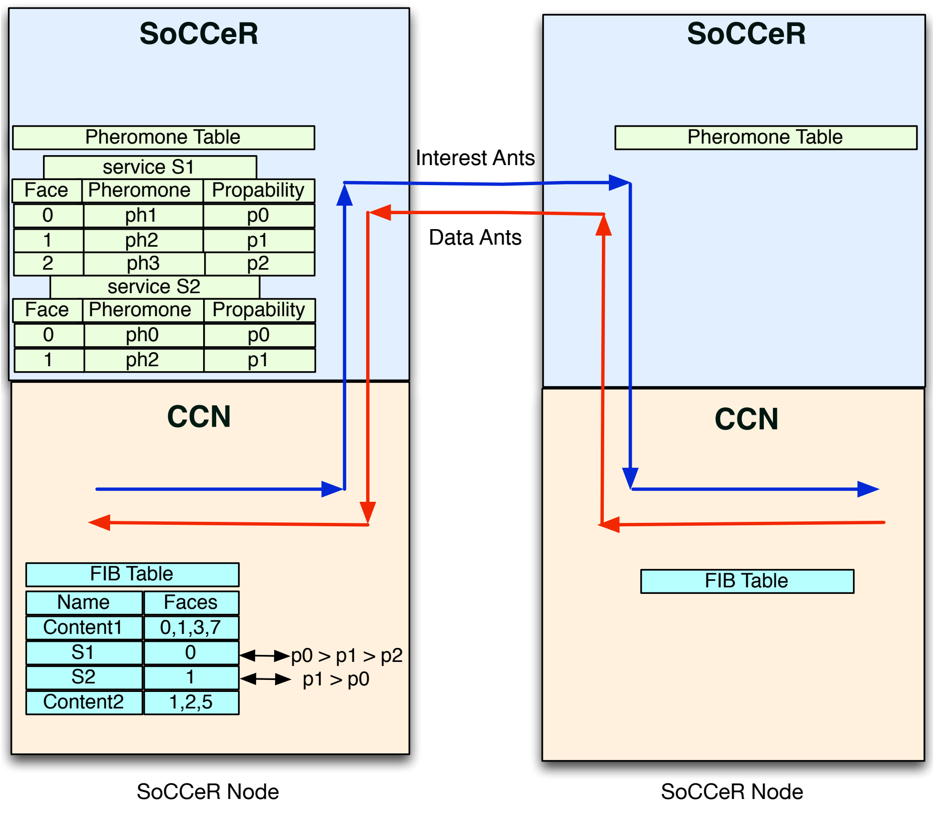


Figure 2: General overview of SoCCeR architecture and the corresponding routing tables.

The main idea is to make routers able to select the best face in the list of FIB faces according to the requested service by providing them with some parameters like available bandwidth and service load traffic. Each node in the network contains a SoCCeR control layer above the CCN layer. The SoCCeR control layer has access to the announced services and each SoCCeR node has a pheromone table, which consists of the service name, the associated faces, the corresponding pheromone values, and the probabilities to determine the best face according to calculated pheromone values as shown in Figure 2. The content entries in the FIB are not changed, while service entries are changed by adding the highest probability values (in the pheromone table) for each face. Consequently, the service request will be sent to the face with the highest probability. In Figure 2 there are three faces for Service1 (S1) but the probability of face0 is bigger than of face1 and face2. So, the request will be sent to the closest instance over face0. At the same time, the request for service2 (S2) is sent over face1, because it has the biggest probability in the pheromone table.

Actually, SoCCeR requires modifying the CCN Interest and Data packets to enable the SoCCeR control layer to work. Each SoCCeR node creates periodically an Interest Ant containing the current timestamp. Then it sends this Interest to all faces. This process is repeated until the Interest Ant finds the target service. When the service node receives the Interest Ant it creates a Data Ant with a copy of the current timestamp and some status information like memory and service load. Then, it sends it back to the requesting SoCCeR node, tracking the bread crumb of the Interest Ant. In this case each node calculates the round trip time between itself and the service node using the status information and the time in the Data Ant. In addition, it refreshes the pheromone value for the face on which the data arrived. In this way, the pheromone table of each SoCCeR node is updated by its own Data Ant as well as any Data Ant generated by other nodes.

Each face probability represents the attractiveness of that face being traversed by an Interest Ant. And this probability is calculated from the pheromones that serve as the parameters for the probability formula. The pheromone updates are an inverse function of path metrics and service status. Currently, only two pheromones are considered in calculating the face probabilities, which are the network delay and the service replica's workload. But still, the approach allows considering additional factors easily, because the formulas, which calculate the probabilities based on the pheromones, are normalized and weighted.

3.2.4 Discussion

The advantages of SoCCeR can be summarized as follows:

- Routing is a decentralized probabilistic optimization heuristic, eliminating the problem of single point of failure.
- The approach is scalable by nature and inherently implements load balancing.
- Routers are very responsive to service failures and migration due to the fact that Ant Interests, Ant Data, and evaporation effects are always updating the pheromone tables.

But still, SoCCeR has some limitations:

- SoCCeR does not support sessions or stateful services, where consequent service requests need to go to the same service instance. This also complicates authentication and authorization mechanisms.
- There are many control messages (Ants) traversing the network constantly. This might increase traffic overhead.

3.3 Serval

3.3.1 Objectives

Nowadays, we have plenty of services over the Internet provided by data centers that contain thousands of machines. Currently, when a client looks up a service using DNS, this client binds to the service provider location very early in the connection establishment phase. Actually, the client might be bound to a special machine called the load balancer. This load balancer is responsible for choosing the best service replica that should serve the client. In this way, the load balancer becomes a single point of failure. Also service fault tolerance, replica migration, replica addition, and upgrading are very challenging tasks. Moreover, this situation introduces a problem on the client side. The problem is that when the client machine changes between interfaces (WiFi, 3G, Ethernet, etc), the flow also breaks and the connection should be reestablished.
Serval [9] is trying to tackle those problems by providing better abstraction for services in the TCP/IP protocol stack. The Serval protocol stack provides special interfaces to deal with service allocation and connection.

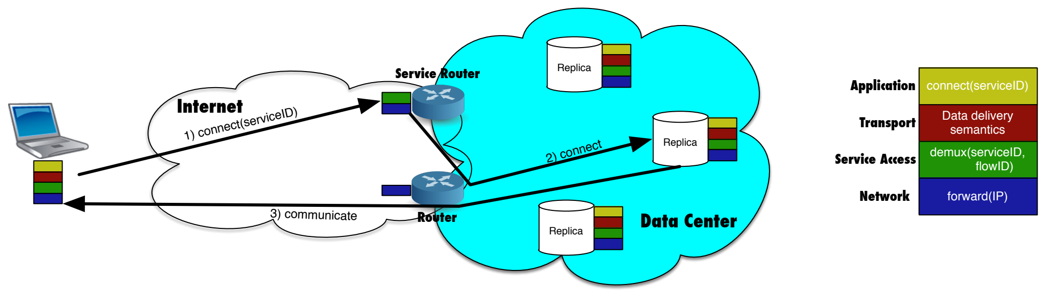


Figure 3: General overview of Serval architecture with the new Serval protocol stack.

3.3.2 Naming Scheme

In Serval, every service has an ID (name), which consists of three parts: *Provider-prefix + Provider-specific + Self-certifying*. The provider prefix (like Google for example) is obtained by an authorized Internet body like IANA. The provider-specific part comprises to the specific service name (like calendar or GMail). The self-certifying part represents a hash of the public key and the service prefix allowing the services to be self-authenticating without relying on a central certifying authority. The naming scheme in Serval combines a hierarchical structure with a flat naming structure. An example service name could be: *google.gmail.153AB119*.

3.3.3 Architecture

Serval introduces some changes in the TCP/IP protocol stack. Serval introduces a new layer called service access layer (SAL) between the Internet and the transport layers as shown in Figure 3. The transport layer responsibility of demultiplexing is moved to the SAL. The application layer has the responsibility for connection establishment with the serviceID. The SAL does the demultiplexing based on the serviceID and the flowID but not based on the IP address and the port number. The service access layer (SAL) has two types of tables: service table and flow table. The service table maps serviceIDs to service replicas and the flow table maps flowIDs to sockets or interfaces.

Serval changes the TCP/IP stack of clients and data centers. It adds a new element, which is a service router, to move the functionality of load balancers and proxies to the service router. Each service provider announces the availability of the service and all the replicas to the service

router. Suppose that there is a client request for a Gmail service. Then, the service name has the form (google+gmail+534AB3). The client asks the service router about a certain service. The service router chooses the best available service replica and forwards the request to it through a network (IP) router. The replica then answers back to the client, bypassing the service router. This is just for the first packet, then all remaining packets can be demultiplexed by the flow table based on destination flowIDs, without requiring the SAL header. This means that the remaining packets do not go through the service router anymore as shown in Figure 3.
Serval makes it easy for data centers to implement load balancing, migration, upgrading, and fault tolerance because of the late binding between the serviceID and the service replica and also because of the idea of a flowID, which makes it possible for another replica to carry out the session in case of a failure in the original replica. Because Serval supports sessions using flowIDs, clients can move among connectivity interfaces (WiFi, 3G, *etc.*) without interrupting the flow, which can be re-established and resumed easily.

3.3.4 Discussion

Serval has many advantages:

- The Serval service naming provides uniqueness, allows for prefix matching for scalability, and provides a way for self-authentication.
- The way Serval is implemented provides data centers with easy load balancing, fault tolerance, and service maintenance (service dynamism).
- The idea of keeping a table for flowIDs, provides a way to implement un-interruptible flows and sessions.
- Serval is a fully implemented project (The Serval website [17] works using Serval infrastructure).

A few design decisions resulted in some disadvantages:

- There is a compatibility issue due to the fact that the TCP/IP stack is dramatically changed. This can cause problems with deployability and cost-efficiency, because all service providers and service clients need to change their networking stacks.
- No caching mechanism is implemented in Serval.

3.4 Other Approaches in Service-Centric Networking

Service-Centric Networking is capturing the attention of many research groups that have already many interesting projects with great potentials.

3.4.1 Named-Function Networking

Named Function Networking (NFN) aims to find a naming scheme to support services using CCN infrastructure. Service Naming in NFN [18] is inspired by the λ-expression language. The authors use this expression to formalize hierarchical service names. Actually, naming in NFN is similar to CCNxServ with the advantage of allowing the service and the content to have different naming prefixes (publishers).
The corresponding grammar for the λ-expression:
f(g(dat)), has the following NFN name: *[ccn:nfn|/name /of/data | /name/of/g | /name/of/f]*. Suppose that we have this example in λ-expression: */util/compress/zip /codec/mpeg4 (/disney/BugsLife))* applies two functions (mpeg4 and zip) on the parameter (disney/BugsLife). The corresponding NFN name for this example will be: *[ccn:nfn|/disney/BugsLife|/codec/mpeg4|/util/compress /zip].* The limitation of this approach is that the service name can get complicated when there are many parameters and service calls combined when we deal with more complex real-world scenarios.

3.4.2 Object-Oriented Service-Centric Networking (SCN)

In [19], the authors discuss how to name services in ICN and how to carry out name resolution in the case of coupled and decoupled ICN approaches. They also discuss how to define and support routing metrics. Besides, they focus on what the requirements of service management are by extending ICN forwarding concepts like where and when to deploy services in the network and how we can adapt and terminate services in a Service-Centric Networking environment.
Another proposed design for Service-Centric Networking in [20] is built based on an object-oriented approach. In this approach, called SCN, the requested content or service are named as objects in a hierarchical structure in the CCN naming scheme. The authors proposed to list the service

object names in the routing header and then the service request is forwarded to the first object name, then to the second and so on. In this approach, servers are selected according to two parameters: the distance between the client and the server and the distance between the server and the required data.
In [21], the authors confirm the capability of CCN to support services. They introduce a method to invoke a service, which could be requested by CCN clients with the same name structure used by CCN. In their work, the authors discuss three approaches to implement services over CCN: to implement the service in the core of CCNx, at the publisher side, or as a separate application. They inspect the advantages and disadvantages of each approach to reach the conclusion that implementing the service as a separate application is easier than others. The authors implemented a prototype service as a proof of concept. Also, the authors note that there is a disadvantage in this scenario. The problem is that when a client requests a service on a content object (as a parameter), the reply to the client will be signed by the service publisher but not by the content publisher.

	CCNXServ	Serval	SoCCeR	NFN
Number of parameters	One	No params	No Params	Multiple
Types of parameters	Content	-	-	Any
Human-readable Names	Easy	Hard	Easy	Hard
Naming scheme	Hierarchical	Flat+Hierarchical	Hierarchical	Hierarchical+Lambda
Architecture dependencies	CCN+NetServ	Modified TCP/IP stack	CCN	CCN
Service code migration	Yes	No	No	No
Changes in TCP/IP stack	No	Yes	No	No
Routing	CCN Routers	Service Routers	CCN Routers	CCN Routers
Session support	No	Yes	No	No
Scalability	No	Yes	Yes	Yes

Table 2: Comparison among the different SCN projects

4 Discussion

The main goal of the surveyed projects in Section 3 is the same. It is to change the current Internet architecture and provide better abstractions that suit the current Internet use. This abstraction is the notion of services instead of hosts. Table 2 summarizes and compares the main aspects among the surveyed projects.
CCNxServ and SoCCeR are built on top of CCN trying to extend its functionality to support services. Serval has been built from scratch, significantly modifying the TCP/IP protocol stack. Even though, we can see the similarity of the three architectures. They all have services with many replicas, they all have service routers, and they all route service requests based on service names or IDs. From the discussed advantages and disadvantages of the surveyed projects, we can provide some recommendations for successful future research works in the field of Service-Centric Networking.

- A Service-Centric Networking architecture should be distributed as much as possible. So, routing, service allocation [22], and service execution should be handled in a distributed manner.

- The naming structure should be hierarchical (to be human readable) with a self-authenticating mechanism to avoid using a central certifying authority.

- Any Service-Centric Networking approach should integrate easily with the current Internet architecture to ease the deployment phase and decrease costs.

- Sessions should be supported based on the fact that most services require sessions.
- Usually, services are interfaces for entire systems. For example, Google Maps service has a huge system behind, which is composed of many complex components like maps, images, GPS, advertisements, and transportation. In this sense, service code should not be migrated or at most can be migrated to replicas authorized by the service provider only. In other words, services should still be executed within the control of their providers.
- The Service-Centric Networking approach should not interfere with how services are implemented but only with how services are published.
- Services should be treated like methods in any programming language. They accept any number of parameters, can be aggregated, take parameters of any data type, and return results of any data type. Also, aggregating services should be supported [22]. Any limitation on those requirements makes the approach infeasible.
- Service routers should be stateful and provide caching capabilities.

Whereas Service-Centric Networking is a new trend, there are still many research challenges that should be addressed:

- Scalability: We are dealing with named services and content instead of named hosts. The number of services and content items are much bigger than the number of hosts. This leads to the fact that scalability is a big issue in routing and naming.
- Legal Issues: Caching service implementation or results might introduce some legal issues that should be studied further. For example, YouTube does not legally allow the ISPs to cache videos.
- Costs: Since Service-Centric Networking is revolutionizing the Internet architecture, deploying any new approach will introduce tremendous costs. This poses some questions about the deployability of Service-Centric Networking.
- Security: Security should be on the level of services and content but not on the level of hosts. This should trigger many research activities about authentication, authorization, and data secrecy.

5 Conclusions

In this paper, we gave a thorough introduction about Information-Centric Networking in general and Service-Centric Networking in particular. We motivated how Service-Centric Networking is a promising networking paradigm for the future Internet. Services in this paradigm are first-class citizens. Service replicas can be added, removed and updated easily because client communication with services is not bound to locations but rather to names and identities. Then we surveyed three major Service-Centric Networking projects. We examined and explored their implementations, limitations, and advantages. We demonstrated the lessons learned from our study in the form of general guidelines and what should and should not be done. This study can serve as a guidance for future research works in Service-Centric Networking.

References

[1] T. Zahariadis, D. Papadimitriou, H. Tschofenig, S. Haller, P. Daras, G. Stamoulis, and M. Hauswirth, “Towards a future internet architecture,” in *The Future Internet*, vol. 6656 of *Lecture Notes in Computer Science*, pp. 7–18, Springer Berlin Heidelberg, 2011.

[2] J. Pan, S. Paul, and R. Jain, “A survey of the research on future internet architectures,” *Communications Magazine, IEEE*, vol. 49, pp. 26–36, July 2011.

[3] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, “Networking named content,” in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, CoNEXT ’09, (New York, NY, USA), pp. 1–12, ACM, 2009.

[4] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, “A data-oriented (and beyond) network architecture,” *SIGCOMM Comput. Commun. Rev.*, vol. 37, pp. 181–192, Aug. 2007.

[5] C. Dannewitz, J. Golic, B. Ohlman, and B. Ahlgren, “Secure naming for a network of information,” in *INFOCOM IEEE Conference on Computer Communications Workshops , 2010*, pp. 1–6, 2010.

[6] N. Fotiou, D. Trossen, and G. Polyzos, “Illustrating a publish-subscribe internet architecture,” *Telecommunication Systems*, vol. 51, no. 4, pp. 233–245, 2012.

[7] S. Shanbhag, N. Schwan, I. Rimac, and M. Varvello, “Soccer: services over content-centric routing,” in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, ICN ’11, (New York, NY, USA), pp. 62–67, ACM, 2011.

[8] S. Srinivasan, A. Singh, D. Batni, J. Lee, H. Schulzrinne, V. Hilt, and G. Kunzmann, “Ccnxserv: Dynamic service scalability in information-centric networks,” in *Communications (ICC), 2012 IEEE International Conference on*, pp. 2617–2622, 2012.

[9] E. Nordström, D. Shue, P. Gopalan, R. Kiefer, M. Arye, S. Y. Ko, J. Rexford, and M. J. Freedman, “Serval: an end-host stack for

service-centric networking," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, NSDI'12, (Berkeley, CA, USA), pp. 7–7, USENIX Association, 2012.

[10] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, CoNEXT '09, (New York, NY, USA), pp. 1–12, ACM, 2009.

[11] "Ccnx project." http://www.ccnx.org/. Visited December 06, 2013.

[12] A. Carzaniga and A. L. Wolf, "Content-based networking: A new communication infrastructure," in *NSF Workshop on an Infrastructure for Mobile and Wireless Systems*, no. 2538 in Lecture Notes in Computer Science, (Scottsdale, Arizona), pp. 59–68, Springer-Verlag, Oct. 2001.

[13] A. Carzaniga, M. Rutherford, and A. Wolf, "A routing scheme for content-based networking," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 2, pp. 918–928 vol.2, 2004.

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

[15] J. W. Lee, R. Francescangeli, W. Song, J. Janak, S. R. Srinivasan, M. S. Kester, S. A. Baset, E. Liu, H. G. Schulzrinne, V. Hilt, Z. Despotovic, and W. Kellerer, "Netserv framework design and implementation 1.0," technical report, Columbia University, http://academiccommons.columbia.edu/catalog/ac:135424, May 2011.

[16] T. Stützle and M. Dorigo, "Ant colony optimization," in *Evolutionary Multi-Criterion Optimization* (M. Ehrgott, C. Fonseca, X. Gandibleux, J.-K. Hao, and M. Sevaux, eds.), vol. 5467 of *Lecture Notes in Computer Science*, pp. 2–2, Springer Berlin Heidelberg, 2009.

[17] "Serval project." http://www.serval-arch.org/. Visited November 26, 2013.

[18] C. Tschudin and M. Sifalakis, “Named functions for media delivery orchestration,” in *Packet Video Workshop (PV), 2013 20th International*, pp. 1–8, Dec 2013.

[19] T. Braun, A. Mauthe, and V. Siris, “Service-centric networking extensions,” in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC ’13, (New York, NY, USA), pp. 583–590, ACM, 2013.

[20] T. Braun, V. Hilt, M. Hofmann, I. Rimac, M. Steiner, and M. Varvello, “Service-Centric Networking,” pp. 1–6, June 2011.

[21] E. Cheriki, “Design and implementation of a conversion service for content centric networking,” Master’s thesis, Institute of Computer Science and Applied Mathematics University of Bern, 2012.

[22] X. Huang, “Protocol and system design for a service-centric network architecture,” Master’s thesis, University of Massachusetts, 2010.