Network layer solutions for a content-centric Internet

Andrea Detti and Nicola Blefari-Melazzi

Abstract Nowadays most people exploit the Internet to get contents such as web pages, music or video files. These users only value ”what” they download and are not interested about ”where” content is actually stored. The IP layer does the opposite and cares about the ”where” and not about the ”what”. This contrast between the actual usage of the Internet and the service offered by the IP layer is deemed to be the source of several problems concerning usability, performance, security and mobility issues.
To overcome this contrast, research on the Future Internet is exploring novel so-called content-centric architectures, where the network layer directly provides users with contents, instead of providing communication channels between hosts.
In this paper, we identify the main functionalities of a content-centric network (CONET), we discuss pros and cons of literature proposals for an innovative, content-centric network layer and we draw our conclusions, stating some general requirements that, in our opinion, a CONET should satisfy.

1 Introduction

There is a growing consensus in the recent literature that the central role of the IP address poorly fits the actual form of Internet usage. A typical user does not type IP addresses; she gets data or services by using application tools (e.g., Google, YouTube, Facebook, Skype), which operate on the basis of a description of the desired content. This means that users actually exploit the Internet in a content-centric way; indeed, they are not interested in knowing from ”where” contents are provided,

Andrea Detti
Electronic Engineering Dept., Univ. of Rome ”Tor Vergata”, e-mail: andrea.detti@uniroma2.it

Nicola Blefari-Melazzi
Electronic Engineering Dept., Univ. of Rome ”Tor Vergata”, e-mail: blefari@uniroma2.it

they are only interested in the fact that they can get ”what” they want. Conversely, the underlying IP communication model is still address-centric (or host-centric); that is, the network layer has to be fed by IP addresses, which are used to ascertain from ”where” contents have to be taken. Therefore, there is a mismatch between the content-centric usage model of the Internet and the address-centric service model offered by the IP layer. Such a mismatch gives rise to several problems that would not exist if the network layer were a content-centric one [2, 4, 5]. Some of these issues are:

- **Persistence of the Names**: once a user gives a name to a content, she wishes that the name remains valid even if the provider that makes available the content on the Internet changes. Today, naming is based on the WEB URL structure, which has a ”where/what” format. The URL structure is perfectly tailored for an address-centric network layer, but it implies the change of the name in case the provider (i.e., ”where”) changes. If the network-layer were a content-centric one, it would be able to route on the basis of the content (routing-by-name), thus avoiding the need of including ”where” in the name of the content.
- **Content distribution**: today the reliability of a content and the time required for retrieving a content are improved by distributing (caching or pre-fetching) replicas of the content on different servers, geographically distributed. Furthermore, content replication strongly limits the amount of traffic traversing the backbone, since a lot of data sessions are locally handled by these servers. Since the IP layer is unable to ”understand” contents, content distribution is achieved by means of proprietary application-layer systems [10], like Akamay CDN, or WEB proxies. While the success of these systems is undeniable, they do not cooperate with each other, often they are not free of charge and they are not available everywhere; thus, their potential effectiveness is reduced. If the network layer were a content-centric one, it would be aware of which contents are traversing the network and could autonomously and natively implement replication strategies, everywhere and for all [12].
- **Lookup delay**: today the real delivery of a content begins only after that the client application has queried the DNS server, in order to discover the IP address of the related server. The additional DNS request-response delay is quickly becoming a dominant delay factor [2]; indeed Internet data-rates are faster growing, reducing the time needed to transfer content. If the network layer were content-centric, it would route directly the request toward the content, thus avoiding the mediation of a DNS server.
- **Mobility**: an IP address identifies both the location and the identity of an end-point; this location-identity tie is deemed to be the source of many mobility-related weaknesses. A content-centric network overcomes these limitations by basing the routing directly on the identity of an end-point; in this case the end-point is the actual content and its identity (e.g., a name) does not include any reference to its location.
- **Security**: users are interested in receiving trusted contents; today this is achieved in an indirect way: a user trusts ”who” provides the content, rather than trusting the content itself. This approach could be risky as there are a lot of actors

to trust, in the process of content retrieval [6]. For instance, a user believes that she is trading with Amazon because she clicks on a link with the name ”www.amazon.com”, which is provided by a search-engine tool, like Google. In doing so, she is trusting both the search-engine and the DNS server providing the name-to-address translation. Moreover, she is also trusting the mirror server where she could be redirected by a content distribution system [10]. The higher the number of actors to trust, the more critical is the overall security of the system. In a content-centric network, security information travels with the content, so that even if content is provided by an un-trusted server or is transferred on an un-trusted network, it can be validated by the consumer. Moreover, such a content-based security also enables the replication of secure contents by any network node, which is very important as users can get contents not only from the original content creator or source node but also from any node/user that has already downloaded that content.

Currently, several researchers and research projects are proposing their network architectures and protocol stacks aimed at implementing the content-centric paradigm. In terms of deployment, the proposed solutions can be classified as evolutionary [2, 4, 7] or clean-slate [5, 8, 11, 9]. The evolutionary architectures implement content-centric functionalities by enhancing the actual Internet. The enhancement consists in deploying new entities that form a content-centric ”overlay” network. Conversely, clean-slate architectures are alternative to the TCP/IP one. A clean-slate architecture is devised to operate directly over the link-layer and, therefore, implies a redesign of network layer functionalities from scratch. Obviously, a clean-slate solution can also be implemented at the overlay level, e.g. by visualizing IP tunnels as link-layer interface; however, in this case some functionalities could be duplicated in TCP/IP and content-centric layers.

In this paper, we outline the principles of a content-centric network layer and discuss pros and cons of literature approaches. Of course, a complete system requires also functionalities provided by higher protocol layers, in addition to the network ones; however, as of today, the main research effort focuses on the network layer, considered as the main pillar of a future content-centric Internet.

2 Principles of a content-centric network layer

In this section we discuss the main principles of a content-centric network layer, i.e. we deal with the protocol functionalities that should be implemented in each router of a clean-slate content-centric network architecture. A content-centric network layer should:

- address contents, rather than hosts, adopting an addressing scheme based on names, which do not include any reference to their location;
- route a user request, which includes a ”destination” content-name, toward the closest copy of the content with such a name (name-based anycast routing);

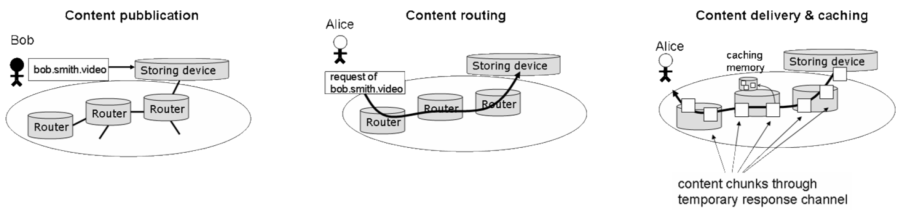

Fig. 1 Use-case

- deliver contents to address-less hosts;
- provide a native, in-network caching to achieve efficient content delivery both in fixed and mobile environments;
- exploit security information embedded in the content so as to avoid the diffusion of fake versions of contents.

For the sake of clarity we describe all these functionalities in a use-case, depicted in Fig. 1. Bob Smith is a young reporter that has recorded a video and wishes to publish it on a content-centric network. Bob prepares a data-package that embeds video and security information, he names this package as ”bob.smith.video” and puts it in a storing device, connected with one or more network routers. Such a data-package is the actual content. Alice wishes to retrieve Bob’s video and submits a request to the network layer, where the ”destination” content-name is bob.smith.video. Each network router forwards Alice’s request toward the closest storing device that holds a copy of Bob’s content (exploiting only the name ”bob.smith.video”); we name this phase ”content-routing”. When Alice’s request reaches a storing device, the network layer delivers Bob’s content to the address-less device of Alice; we name this phase ”content-delivery”. During the content-delivery phase, some intermediate routers may decide to cache and then forward [12] the content. When the content is received by Alice’s device, the whole security information is checked to verify, for instance, the authenticity of the embedded video, the right of Alice to play it and so forth.

This scenario involves a set of functionalities like naming, routing, delivery and caching, which we discuss in the next subsections by presenting some recent literature proposals.

2.1 Content-naming

In a content-centric network, routing is based on the name of the content; therefore, choosing the structure of names is an important aspect of the overall architecture.

From the user point of view it is better not to have any protocol restriction, so as to freely name contents. From the routing protocol point of view, this translates in flat-labels [3], i.e. names are a sequence of unstructured bytes.

Zooko's Triangle [13] identifies possible tradeoffs among three desirable properties of a naming scheme of a network protocol. Zooko's Triangle states that names can enjoy simultaneously at most two of the following characteristics:

- **Secure**: a name surely addresses only a given content; i.e., names can not be forged for addressing fake contents
- **Memorable**: a human can remember a name for a couple of hours since the first time she has seen it on the side of a moving bus (moving bus test).
- **Decentralized**: a name can be chosen in a distributed way, i.e. without the need of a centralized naming authority.

For instance, actual DNS names are secure (if we trust the DNS system) and memorable. Nick-names used by proprietary applications (e.g., Skype) are memorable and decentralized. So-called *self-certifying names* [4, 14] are secure and decentralized.

As regards the latter solution it is worth recalling that a self-certified name is cryptographically constructed so that one can securely verify if the associated content is the original one. For instance, in [4] the authors propose that a publisher P of a content may autonomously label the content as L and the resulting self-certified name is the Digest(Pubkey$_P$):L, where Digest(Pubkey$_P$) is a digest of the public key used by the publisher to sign the data. If that the publisher ensures the uniqueness of label L among her published contents, a self-certifying naming schemes is secure and decentralized.

We argue that the security of names is a necessary property for a content-centric network. In the actual Internet, we can be (almost) sure to obtain an original content, e.g. the CNN homepage, because we trusts the DNS system, which provides the IP address of the original WEB server (or of an authoritative one). In a content-centric network, the retrieval of the content from the original server may be a rare event. Indeed, any *untrusted* router or device may make available its cached copy of the CNN homepage, and the network layer provides the user with the closest data named "CNN homepage". Therefore, it is fundamental that all contents named "CNN homepage" are equal to the original one; i.e., the name "CNN homepage" must not be forge-able (security property).

Since security is a must, given the Zooko's Triangle, we have to choose between decentralization and memorability. We think that this choice depends on requirements of applications, therefore we could have both secure-memorable and secure-decentralized [4] content-names. The memorability property is desirable for contents that need to be advertised, like the home-page of a business company or of a newspaper. Conversely, memorability may be not desirable for contents that have a very narrow interest, like a content describing the status of a DHL shipment, or the value measured by a sensor. Finally, we point out that implementing a secure-memorable naming for a content-centric network is currently an open issue (the DNS approach is not applicable in a content-centric network that directly routes contents by name); also an open issue is how to allow the coexistence of secure-decentralized and secure-memorable naming schemes in the same network.

2.2 Content-routing

The aim of content-routing is to forward a content request issued by an host toward a storage device that can provide such a content. With this regard, recent research proposals on future Internet [5, 4, 8, 2, 3] focus on devising a network layer anycast routing, which is ”simply” based on names of contents, rather than on locations (i.e., IP address); consequently, in this paper, the term content-routing is synonymous with anycast routing-by-name. We observe that, before this wave of renewed interest, content-based routing has been widely studied for P2P overlay networks (e.g., [18]), improving also the simple routing-by-name with semantic functionality. Nevertheless, the inclusion of semantic functionality strongly increases the complexity of the architecture and may make critical its scalability at the Internet level.

Currently, the literature proposes two kinds of architectures, which implement the content-routing paradigm; we name these alternatives *advertise-based* and *rendezvous-based* architectures. In case of advertise-based architecture [5, 4, 2], routing protocols are practically the same of the actual IP ones (e.g., BGP), with the only difference that instead of advertising IP subnets, routers advertise the names of the contents that they can provide. Routing tables do not contain anymore IP addresses but content-names. Forwarding is executed by finding the best match between the name of the requested content and the entries of the routing tables.

A good example of a rendezvous-based architecture is the one proposed in [19]. The name of a content (and eventually other information, like the information-scope [8]), identifies a rendezvous-node (i.e., a location) by using a standardized function; all requests of that content will be forwarded towards that node by the network layer. The content provider stores in the rendezvous-node routing information, enabling the forwarding from the rendezvous-node to the node where content is effectively stored[1]. Therefore, the end-to-end path is composed of two parts: user-to-rendezvous and rendezvous-to-content. Routing protocols are inspired by overlay routing algorithms used in Distributed Hash Tables (DHTs) but they do not rely on an underlying network routing protocol [3, 15].

We argue that advertise-based architectures achieve optimal routing in terms of path length and are more effective in supporting content replication or caching, with respect to rendezvous-based ones; in fact, the same shortest-path anycast routing algorithm used for IP networks can be adopted for advertise-based architectures. In case of rendezvous-based architecture an end-to-end path is at the best stretched in two shortest-paths, and the absence of advertisements issued by nodes storing the same content (e.g., due to caching) makes more difficult to exploit content replication; i.e. the achievement of a perfect anycasting (see section 2.4).

Advertise-based architectures seem to better use transport resources; however we argue that advertise-based architectures may suffer of routing scalability and stability issues. Scalability concerns arise if we observe that the number of entries of a routing table may be proportional to the number of content-names, unless an

[1] Obviously, if the content is uploaded directly on the rendezvous-node, this routing information is not necessary

efficient and effective mechanism to compress name-based routing-tables will be devised. Nowadays IP routing is showing its scalability limits in terms of size of the routing tables [21, 22]; if we replace the actual IP prefix-based routing tables with name-based routing tables, surely the problem worsen as there are more contents than Autonomous Systems [2]. Moreover, we observe that the actual convergence delay of IP routing tables is short enough to cope with the actual dynamics of link creation and removal, thus ensuring the stability of the whole routing system. Conversely, the convergence delay of a name-based routing system has to cope with the dynamics of creation and removal of contents, and the occurrence of these events is more frequent than link creation and removal. Therefore, it is more difficult to ensure routing stability.

Finally, we observe that advertise-based architectures perfectly support a request-response interaction model but are less suitable for a publish-subscribe model [1]; indeed, it is difficult to handle the case of subscriptions issued before the corresponding publication, since routing tables are not configured in absence of actual content, i.e. of the publication. Conversely, in case of rendezvous-based architectures, a subscription can be routed and temporarily stored on the rendezvous-node, until the related publication gets available. However, rendezvous-based architectures can not easily control where in the network a rendezvous node handling a given content is located, e.g. in which geographical or administrative domain; a user publishing a content may not like that the related rendezvous node is located in the administrative domain of a competitor or in another country.

2.3 Content-delivery

While content-routing is concerned with the host-to-content routing of user requests, content-delivery regards the content-to-host data transfer; i.e. the transferring of a content from its storage device to the requesting host.

In a content-centric network, hosts are not addressed by routing tables; i.e., routing tables contain only information about contents and not about hosts. This implies that, differently from IP, host-to-content and content-to-host routing involve different protocol functionalities.

Currently, the literature [5, 16, 3] copes with content-to-host routing by defining a temporary ”response” channel, e.g. the sequence of downstream link-layer interfaces that content, or part of it, must follow from the storage device to the host. There are two proposed approaches to maintain the interface sequence: i) the sequence may be explicitly contained in the header of the data-units transporting the content [16] (similarly to a source-routing approach); ii) each downstream node may temporarily memorize information about the next-hop link-layer interface [5, 3] (similarly to the soft-state approach of the RSVP protocol).

In addition to the content-to-host routing issue, the delivery protocol may also include congestion control mechanisms. With this regard, the authors of [5] propose to download a content through a sequence of request-response interactions. At each interaction, a request conveys the interest for a number of ”chunks” of the content; chunks have a fixed size (e.g., 512 kB) and the number of required chunks follows the dynamic of the TCP congestion control algorithm; nevertheless, differently from TCP, these chunks have to be network layer data-units, to allow caching (see Section 2.4).

2.4 Content-distribution

Content-distribution concerns the possibility of caching (or replicating) a content or part of it in order to reduce the content-to-host path length. Reduction of path-length implies both a lower use of transport resource and, likely, a lower latency. In response to this challenge and considering the rapidly decreasing cost of memory, an approach that is attaining an increasing consensus is the ”in-network” caching; i.e., to supply network routers with memory, to cache traversing contents [5, 12, 17].

Practically, to enable in-network caching, content delivery should be structured in a sequence of network data-units, which are stored, recognized by content-name and by a sequence-number, eventually cached, and forwarded hop-by-hop by each router on the content-to-host path. Obviously, a greater size of the data-units reduces the computation load of the caching operation, since routers would have a lower number of content parts to handle. Practically, it seems reasonable to have data-unit of 256-512 kB; therefore network layer data-units of a content-centric network should support sizes similar to the chunks of P2P file transfer applications (e.g., BitTorrent), rather than the typical sizes of IP packets. Obviously, increasing the size of the network data-units increases the store-and-forward delay; nevertheless the increasing of the line-speed should make negligible this kind of delay, if compared to the queueing one.

In-network caching could be carried out by routers either in an autonomous way or in a coordinated way [12]. In the autonomous way, a router could choose to cache a chunk of the content (i.e. a network data-unit) on the basis of a local algorithm, for instance based on the analysis of the request frequencies. In this case, nevertheless, closer routers have an high probability to store the same data, thus using un-efficiently the whole system memory. Coordinated caching copes with this last issue by adopting caching algorithms based also on which are the data cached on closer routers.

Finally, in-network caching should inter-operate with content-routing in order to obtain an effective anycasting. This is simply achievable in case of advertise-based architecture, since every time a router has cached a content, the router advertises it. Conversely, in case of rendezvous-based architecture, the routing algorithm forwards user requests toward a specific rendezvous-node and then to a specific node storing the content; thus, if an intermediate router of the host-rendezvous-content path has a cached copy of the content, that router will serve directly the content request; nevertheless, it is more difficult to exploit also the caches of routers that are not on the path but that are close to the path (or elsewhere located).

3 Conclusion

A content-centric network is devised to directly provide what users value in the actual Internet, i.e. contents. This may imply a clean-slate re-design of the network layer, changing the actual communication paradigm (which has two hosts as end-points) in a novel paradigm (which has an host and a content as end-points). In this paper, we have classified and discussed pros and cons of some recent literature proposals coping with the main functionalities that a content-centric network layer should provide: content-naming, content-routing, content-delivery and content-distribution.

Although a content-centric architecture may better satisfy user needs, it is not easy to imagine a replacement of the actual TCP/IP infrastructure in the medium term. Thus, we argue that a content-centric network could more likely be deployed in a Future Internet consisting of software-routers running different network layer protocols, thus forming different virtual networks [20, 23].

Finally, we conclude the paper by stating some general requirements that, in our opinion, a content-centric network (CONET) should satisfy:

- A CONET should allow controlling where in the network contents or links to contents will be stored, e.g. in a given geographical or administrative domain; it is not acceptable that contents, or links to contents, are stored in ”random” nodes, as it happens for instance in some solutions based on DHTs.
- A CONET should allow advertising the publication of a content within a limited geographical, application or administrative scope, e.g. for instance limiting such advertising to a definite section of the network (a campus, a shopping mall, an airport).
- A CONET should support not only persistent naming of specific pieces of content (e.g. a song, a CV, a movie) but should also support naming of a source of information with a consistent purpose (e.g. a meteo service) and of a source of information made of possibly changing contents accessible with the same name (e.g. different edition of a news magazine).
- A CONET should support mechanisms to delete/update contents, supporting digital forgetting and garbage collection operations (e.g. giving to contents an expiry date and deleting all contents after the expiry date, or allowing users to delete/update contents that they generated and that are stored in the network and that they do not want anymore to be available to other users in their actual forms or at all).
- A CONET should allow selecting and retrieving the latest version (or earlier, or specific versions) of a series of contents identified by the same name (e.g. referenced by the same source of information).
- A CONET should allow supporting service sessions that require interactive exchange of data between two upper layers entities (e.g. a client server couple). This means that the network should provide functionality to deliver not only named content but also ”un-named” content made up of data that are necessary to upper layer entities. Un-named contents can be any kind of data that upper layers need

to exchange but that do not need to be named and do not need to be made accessible and identified in the network, per se. This is a key requirement necessary to natively support ”traditional” client/server service (e.g. HTTP, POP, SMTP). For example, in the case of a client-server service session, un-named-contents are upper layer data (e.g. HTTP, SMTP, POP, SQL data) that are exchanged between a local client application and a remote server one (e.g. an HTTP server). Thanks to this functionality, a CoNet could natively support most of the actual Internet services, and any service that requires a point-to-point bidirectional interaction. We point out that this functionality extends the capabilities of a ”plain” content-centric network and makes a CoNet suitable not only for content retrieval but for also for more traditional service deployment.

- A CONET should natively support a caching functionality without resorting to ”external” mechanisms such as Content Distribution Networks. Caching shall be supported in principle by each node and also by user terminals. Users should be able to get the desired content from whatever node/terminal (e.g. the closest) and not necessarily from the original source, and even when disconnected from the CONET, but connected with a single other node/terminal that has a copy of the desired content. This requirement is fundamental to support recently proposed schemes such as wireless caching, distributed storage, recommendation strategies; such caching functionality, and more in general, making the network aware of the contents that is handling allows moving away the traffic from congested regions and balancing the load, for instance offloading cellular networks, or implementing other distribution strategies beneficial to network operators. This functionality would also give more control to network operators to handle traffic generated by so-called over-the-top players (such as Facebook, Youtube, etc), which generate vast amounts of traffic, leaving very little margin of actions to network operators.

Acknowledgements This work has been performed in the framework of the European-funded project CONVERGENCE (www.ict-convergence.eu). The project has received research funding from the Community’s Seventh Framework programme.

References

1. P.T. Eugster, P.A. Felber, R. Guerraoui, A. Kermarrec, ”The Many Faces of Publish-Subscribe”, in ACM Computing Surveys (CSUR), Volume 35, Issue 2, 2003
2. D. Cheriton, M. Gritter, ”TRIAD: a Scalable Deployable NAT-based Internet Architecture”, Technical Report, 2000
3. M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, I. Stoica, and S. Shenker, ”ROFL: Routing on flat labels”, in procs. of ACM SIGCOMM06, 2006.
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,” in procs. of ACM SIGCOMM07, 2007
5. V. Jacobson, D. K. Smetters, et al. ”Networking named content”, Fifth ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT), 2009.

6. Smetters, D. K.; Jacobson, V. ”Securing network content” PARC technical report, 2009 October
7. ”4WARD”, EU FP7 project , home-page http://www.4ward-project.eu/
8. ”Publish-Subscribe Internet Routing Paradigm”, EU FP7 Project, home-page http://www.psirp.org/
9. ”CONVERGENCE”, EU FP7 Project, home-page http://www.ict-convergence.eu/
10. S. Saroiu, K.P. Gummadi, R.J. Dunn, S.D. Gribble, and H.M. Levy, ”An analysis of Internet content delivery systems,” SIGOPS Oper. Syst. Rev., vol. 36, 2002, pp. 315-327.
11. J. Roberts, ”The clean-slate approach to the future Internet design: a survey of research initiatives,” Springer, Ann. Telecommun. (2009), vol. 64, pp. 271-276
12. L. Dong, H. Liu, Y. Zhang, S. Paul, D. Raychudhuri, ”On the Cache-and-Forward Network Archiecture,” in procs. of IEEE International Conference on Communications 2009, ICC 2009
13. Mark Steigler, ”An Introduction to Petname Systems”, http://www.skyhunter.com/marcs/petnames/IntroPetNames.html
14. B. C. Popescu, M. v. Steen, B. Crispo, A. S. Tanenbaum, J. Sacha, I. Kuz, ”Securely replicated web documents”, in procs. of IPDPS 2005
15. M. Caesar, M. Castro, E. B. Nightingale, G. O’Shea, A. Rowstron, ”Virtual Ring Routing: Network Routing ispired by DHTs”, in procs. of ACM SIGCOMM’06, 2006
16. P. Jokela, A. Zahemszky, C. Esteve, S. Arianfar, and P. Nikander, ”LIPSIN: Line Speed Publish/Subscribe Inter-Networking”, in proc. of ACM SIGCOMM’09, 2009
17. Future Internet Assembly, ”FIA 2010 Conference Report” available at http://www.future-internet.eu/fileadmin/documents/valencia_documents/FIA_Valencia_Report_v3_0__out_final_0306.pdf
18. Antonio Carzaniga, Matthew J. Rutherford, Alexander L. Wolf, ”A Routing Scheme for Content-Based Networking”, in proc. of IEEE INFOCOM’04, 2004
19. I. Stoica, D. Adkins, S. Zhuang, S. Shenker, S. Surana, ”Internet Indirection Infrastructure”, ACM SIGCOMM’02, 2002
20. Jon Turner, David Taylor, ”Diversifying the internet”, IEEE GLOBECOM 2005
21. D. Krioukov, Kc Claffy, K. Fall, A. Brady, ”On compact routing for the internet”, in proc. of ACM SIGCOMM’07, 2007
22. D. Meyer, L. Zhang , K. Fall, ”Report from the IAB Workshop on Routing and Addressing”, IETF RFC 4984
23. OpenFlow Consortium, http://www.openflowswitch.org/