VoCCN: Voice-over Content-Centric Networks

Van Jacobson Diana K. Smetters Nicholas H. Briggs Michael F. Plass
Paul Stewart James D. Thornton Rebecca L. Braynard

Palo Alto Research Center
Palo Alto, CA, USA
{van,smetters,briggs,plass,stewart,jthornton,rbraynar}@parc.com

ABSTRACT

A variety of proposals call for a new Internet architecture focused on retrieving content by name, but it has not been clear that any of these approaches are general enough to support Internet applications like real-time streaming or email. We present a detailed description of a prototype implementation of one such application – Voice over IP (VoIP) – in a content-based paradigm. This serves as a good example to show how content-based networking can offer advantages for the full range of Internet applications, if the architecture has certain key properties.

Categories and Subject Descriptors

C.2.1 [**Computer Systems Organization**]: Network Architecture and Design; C.2.2 [**Computer Systems Organization**]: Network Protocols

General Terms

Design, Experimentation, Performance, Security

1. INTRODUCTION

There is widespread agreement that *content* – what a user wants – should have a more central role in future network architectures than it does in the Internet's current host-to-host conversation model [9, 5, 23, 4, 2, 15, 6, 19, 20, 22, 7, 3, 8, 16, 17, 18]. But while it is clear that architectures based on Pub-Sub and similar data-oriented abstractions provide a good fit to the massive amounts of static content exchanged via the World Wide Web and various P2P overlay networks, it is less clear how well they fit more conversational traffic such as email, e-commerce transactions or VoIP.

ReArch'09, December 1, 2009, Rome, Italy.

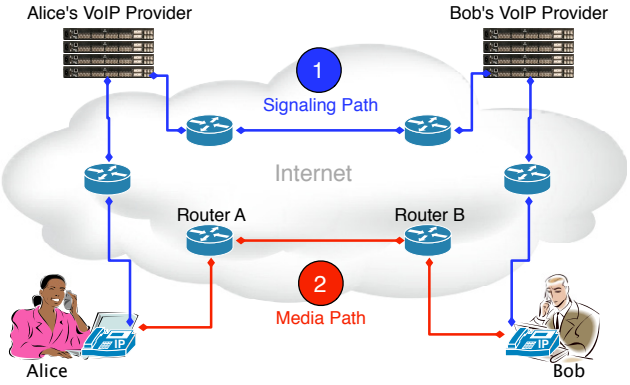

Figure 1: Voice-over-IP data flows

To investigate this question we have implemented VoCCN — a real-time, conversational, telephony application over Content-Centric Networking (CCN) [16, 17, 18] and found it to be simpler, more secure and more scalable than its VoIP (Voice-over-IP) equivalent. Our implementation uses standard SIP [11] and RTP [10] payloads which gives it complete *and secure* interoperability with standard-conforming VoIP implementations via a simple, stateless, IP-to-CCN gateway.

This paper describes how to map the existing VoIP architecture into CCN while preserving security, interoperability, and performance. The mapping techniques are not unique to VoIP, but are examples of general transformations that we believe can be applied to almost any conversational Internet protocol. Through the example of VoCCN we explore the properties that can enable networking models focused on content to be more general than traditional conversational models. These new architectures can therefore deliver benefits for both static content and the full range of conversational communication applications that are important in the Internet today.

2. VOIP BACKGROUND

Voice over Internet Protocol (VoIP) is the dominant open protocol for Internet telephony. Figure 1 depicts

the components of a standard VoIP exchange. When Alice and Bob wish to make a phone call, their VoIP phones set up an audio link using the Session Initiation Protocol (SIP) [11] via what is termed the *signaling path*. As VoIP endpoints are often mobile or located on dynamic IP addresses, signaling path exchanges are mediated by *proxies* – service providers or corporate VoIP signaling gateways that receive and forward messages on behalf of their client endpoints. To place a call to Bob, Alice's endpoint first contacts her SIP proxy who forwards the call invitation to Bob's SIP proxy, as only Bob's proxy knows the current IP address of Bob's endpoint. The body of the invitation contains both information about Alice and the RTP [10] address where she expects to receive audio (or other media streams) from Bob, should he accept the call. Bob's accept of the invite contains the RTP address of where he expects to receive audio from Alice which allows a direct, bi-directional *media path* between their endpoints.

VoIP media (voice, video, etc.) can be secured and authenticated using either an encrypted form of RTP (SRTP [12]), or by tunneling RTP inside another secure network protocol (*e.g.,* DTLS [14]). The encryption keys are either set up via the signaling path, which must then itself be encrypted, or in-band in the media path (ZRTP [24]). Signaling path authentication and encryption can be done via wrapping the signaling exchange in DTLS and relying on a Public Key Infrastructure (PKI) to authenticate the exchange, or using a key agreement protocol such as Multimedia Internet Keying (MIKEY [13]) embedded in the signaling messages. MIKEY has the advantage of providing end-to-end security and authenticating its own messages while minimizing signaling path roundtrips, but does not itself provide confidentiality of the signaling pathway. In practice, however, the perceived difficulty and cost of configuring cryptographic keys and establishing a PKI means that VoIP traffic is almost always unencrypted and unauthenticated.

3. ARCHITECTURE

The complex data paths of Figure 1 result from a mismatch between the user's goal and the network's means of achieving it: Alice simply wants to talk to Bob but the network requires that the communication be addressed to the IP address of Bob's phone. All the infrastructure in the Figure, together with the several services, devices and DNS name registrations that are not shown, exist solely to map from the user/application's world view into the network's world view. One strong driver for content-oriented networking is that this translation (typically referred to as *middleware*) is not needed. Data should instead ideally flow directly from producer to interested consumer, as shown in Figure 2. Our VoCCN prototype achieves this.

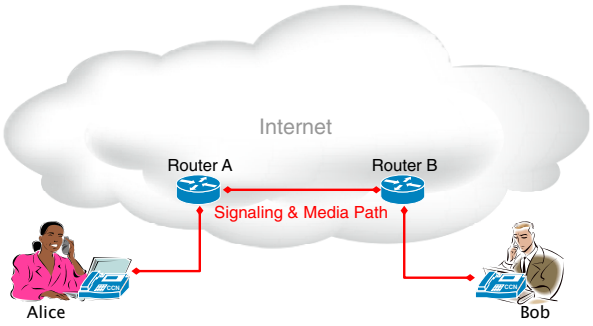

Figure 2: Voice-over-CCN data flows

There are a couple of problems that must be solved in order to map conversational protocols like SIP and RTP into a content-oriented model. First, we must support *service rendezvous*. To initiate a call, the caller's phone must be able to request a connection with the callee, and get a confirmation response. This requires the callee's phone to offer a service contact point. In standard IP, a port number serves as such a point where a process receives requests to which it dynamically generates responses. To translate this into a content-oriented model, we need *on-demand publishing*: the ability to request content that has not yet been published, route that request to potential publishers, and have them create, and then publish, the desired content in response.

Second, we must have a way to transition from this initial rendezvous to a bi-directional flow of conversational data. In standard IP, there are packets (either designated packets in the rendezvous sequence, or all TCP or UDP packets) that contain the information needed to name the destination to which replies should be sent. For example, a TCP packet header contains a source IP address and port plus a protocol identifier. In a SIP exchange, the SDP content of the SIP message (shown in Figure 3) identifies the address to use for the media conversation. To translate this into a content-oriented model, we need *constructable names*: it must be possible to construct the name of a desired piece of content *a priori*, without having been given the name up front or having previously seen the content – the service consumer must be able to figure out how to formulate a request that will reach the service provider. This can be done if:

- There is a deterministic algorithm by which the data provider and consumer arrive at the same (routable) name based on data available to both.
- Consumers can retrieve content based on partially-specified names.

The former requirement guarantees that consumer and producer will arrive at the same name, and that

```
INVITE sip:bob@parc.com SIP/2.0
Via:  SIP/2.0/CCN parc.com:5060
From:  Alice Briggs <sip:alice@ccnx.org>
To:  Bob Jacobs <sip:bob@parc.com>
Call-ID: 1911287229
CSeq:  20 INVITE
Content-Type:  application/sdp
Max-Forwards:  70
User-Agent:  Linphone/3.0.0 (eXosip2/3.1.0)
Subject:  Phone call
Expires:  120
Content-Length:  1477
[...
o=alice 123456 654321 IN IP4 13.2.117.34
c=IN IP4 13.2.117.34
a=key-mgmt:  mikey AQQFgE3dV+ACAA...
m=audio 7078 RTP/AVP 111 110 0 3 8 101
...]
```

Figure 3: Example of SIP INVITE message

names will not depend on data not available to both (such as the cryptographic digest of the content, impossible for data that does not yet exist and will be generated on demand). The latter deals with the fact that without significant prearrangement to allow for a source of shared randomness, such constructed names will not be unique. By allowing flexibility in the query mechanism, we can allow for uniquely named content, while matching it to deterministically generated queries. For example, allowing a query for a structured name that matches only the prefix of that name.

In CCN, each fragment of content that may be published in the network has a hierarchically structured name. Requests for content are expressed in *Interest* packets, which specify the prefix of the name of the desired content and a set of rules by which to determine what of the content under that prefix to return. CCN routing tables use prefix matching to directly route interests based on their name prefixes towards content sources that have registered availability of content by prefix. CCN does not require that data be published and registered with the infrastructure before it can be retrieved; a request merely needs to start with a prefix registered with intervening routers for delivery to an interested publisher. Such a publisher can then look at the request and create named content, or Data packets, dynamically in response. The network forwards matching Data packets back along the path taken by the Interests that requested them, so content reaches the requester and is never sent where it was not requested. For a detailed description of CCN and its current implementation, please see [18].

We map a SIP rendezvous onto CCN as shown in Figure 4. Each phone endpoint is configured at provisioning time with an identity (*e.g.,* `alice@ccnx.org`), and registers to offer data in a namespace derived from that identity and the name of the SIP service (`/ccnx.org/sip/alice/invite`). A caller maps a SIP INVITE (example shown in Figure 3) into an Interest packet asking for new content from the callee. The network routes the Interest to the callee, which generates a piece of Data with the requested name containing the SIP response, thus completing SIP signaling in a single round trip. The use of structured names in CCN allows a simple mapping of the callee identity and service name into the first part of the content name (used for hierarchical, prefix-based routing) while unique identifiers for the request are added to distinguish the desired content from any pre-existing content. Note that the entire SIP INVITE message is included in the requested content name (potentially in encrypted form). The callee receiving this Interest can unpack the name and generate a SIP response as the content of the Data packet that will satisfy the Interest.

To transition from the SIP rendezvous into the media conversation with RTP, each phone takes information exchanged in the rendezvous and uses it to construct a sequence of names for the individual packets of media data. This is also illustrated in Figure 4, where we see that the `call-id` from the SIP exchange, together with the identity of the other party and a sequence number is used to construct the name of each fragment of media. Sequence numbering provides a simple way for data provider and consumer to arrive at the same unique name for each piece of content. These names can be cryptographically anonymized to unlink them from the SIP exchange and provide user privacy.

In the content delivery architecture of CCN, Interests and Data flow in lock-step, each Interest retrieving a single data packet. In a dispersed or high-latency network, the round-trip latency can easily be large enough to delay reception times of media packets to the point where they become unplayable. To solve this problem, we employ pipelining by sending Interests for multiple future media packets. The CCN media receiver maintains some number of outstanding Interests in a media stream; when the stream is opened (or as network conditions change) it generates a number of Interests. Each time it receives content for that stream, it produces a new Interest, thereby maintaining the number of outstanding Interests in the pipeline.

3.1 Advantages

Our approach adds appealing properties beyond simply supporting an existing conversational protocol:

- Content networking infrastructures support multipoint routing – for example, automatically routing a call request to all likely places where it might be answered. This direct infrastructure support for

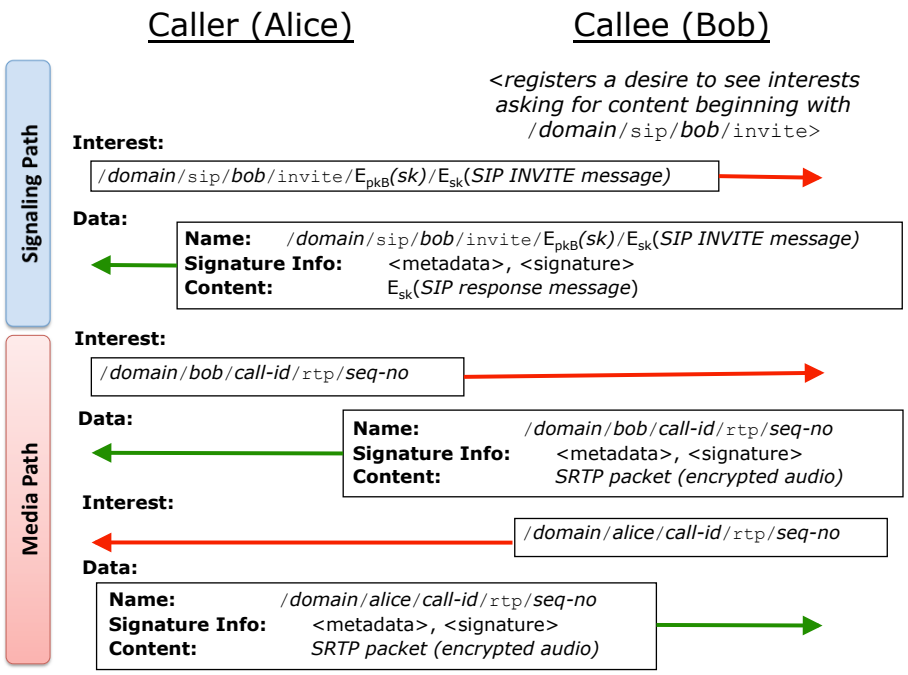

Figure 4: Protocol exchange.

multipoint calling removes the requirement that endpoints register their IP address every time they move, at least within the routing domain.

- The "identity" of a content infrastructure endpoint is represented by credentials located on that endpoint – *i.e.,* a signing key, identifying content (*e.g.,* voice packets) that it creates. Management in a voice content system is minimized, as provisioning a new endpoint consists only of giving that endpoint a credential. In contrast, VoIP configuration requires setting the mapping from callee identity to endpoint IP address at multiple points in the network, each time that IP address changes.

- Advanced services (voicemail, conference calling, call logging and recording) can be easily built on top of a content-based voice system as additional components utilizing multipoint routing to follow call requests or copy and process call contents.

3.2 VoCCN/VoIP Interoperability

We have chosen to implement standard VoCCN by encapsulating standard VoIP protocols (SIP, SRTP), to enable direct interoperability between VoCCN and unmodified legacy VoIP implementations. Such interoperability can be achieved using a stateless VoCCN-VoIP gateway.[1] Such a gateway allows an organization to incrementally deploy a VoCCN-based infrastructure, interoperating with VoIP calls coming in from or going to the outside world and internal legacy VoIP phones.

In this model, the VoCCN/VoIP proxy serves as the SIP proxy for external (and internal) inbound VoIP calls, and translates from VoIP packets (SIP and SRTP) to VoCCN packets (which again merely encapsulate SIP and SRTP for this application), and vice versa.

On receiving a SIP or SRTP packet, the proxy merely examines that packet and generates a corresponding CCN Data packet whose name is determined based on information in the original inbound packet header (see below). It then forwards it into the CCN routing fabric in response to an Interest from the other (CCN-aware) endpoint involved in the call. The proxy then performs the CCN-specific parts of the call on behalf of the legacy VoIP client – generating and sending an Interest in the next packet of the exchange, where the name used in the Interest is also computed as a function of information in the inbound VoIP packet.

The proxy retains no state about the call but responds only to received (CCN or VoIP) packets – the call "state" is contained at the endpoints and in the Interests noted in the forwarding tables along the path to the content source. The proxy also has limited participation in call security. Key exchange and media path encryption (if supported by the VoIP client) is end-to-end, provided by standard MIKEY and SRTP. SIP signaling security for the VoIP portion of the call, if

[1] We present the design of such a gateway here, but have not yet implemented one.

available, is provided by legacy mechanisms between the VoIP client and the VoCCN-VoIP gateway/SIP proxy. The proxy provides signaling security for the VoCCN portion of the call, which digitally signs all messages it translates before sending them into the CCN infrastructure, and additionally encrypts and authenticates the SIP messages between itself and the CCN endpoint.

4. IMPLEMENTATION

We implemented a proof-of-concept VoCCN client as an extension to an open source Linux VoIP phone, Linphone (version 3.0). We made Linphone exchange data over CCN by taking advantage of the ability to plug new transports into libeXoSIP and liboRTP, the libraries it uses for SIP and RTP.

Our CCN network layer is implemented as a content router, which runs on every CCN-aware node, together with an interface library that simplifies the process of writing content-based applications. Each VoCCN endpoint runs a CCN content router. These routers exchange CCN packets via an overlay consisting of UDP sent over preconfigured point-to-point or multicast links. The new libeXoSIP and liboRTP transports use the interface library to send and receive CCN packets through the local content router.

Our CCN library and network stack, together with an updated version of our Linphone client, is available as open source from [1].

4.1 Security

All VoCCN Data packets in both the signaling and the media paths are digitally signed, using per-user key pairs. CCN simplifies the problem of key distribution, easing deployment of end-to-end security. Public keys can be distributed via CCN itself, so obtaining the public key for a VoCCN user can be as simple as requesting a piece of content with a predetermined name – *e.g.,* `/ccnx.org/users/alice/KEY`. Such keys can be accepted on faith at first and remembered over time (*key continuity*), giving a historically-based notion of identity. They can be authenticated in-band by their users, giving equivalent security to ZRTP [24]. Or they can be published as CCN content signed by a publisher trusted by both endpoints, securely binding that key to a CCN name derived from the user's identity. This acts effectively as a digital certificate, allowing the construction of a CCN-based PKI.

To provide media path security, we ran all calls over SRTP,[2] using pre-existing hooks to integrate `libsrtp`, an open SRTP implementation, with Linphone. We implemented a MIKEY [13] library to perform key exchange in the signaling path. We selected MIKEY over DTLS for its ability to perform a complete SIP exchange and key setup in a single round trip.

[2]This provided both encryption and content authentication; though the latter was redundant given CCN's digital signature on each packet.

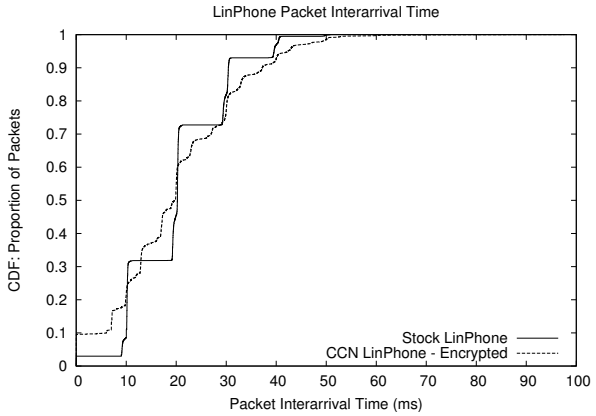

Figure 5: Cumulative distribution of inter-packet intervals, or jitter, for a 10-minute call.

To provide signaling path security, we implemented a simple inline message encryption and authentication scheme (shown in abbreviated form in Figure 4). The caller encrypts the SIP invite message using a randomly-generated symmetric key, sk, and authenticates that encrypted message using HMAC, a symmetric-key MAC. The caller then encrypts sk under the public key of the callee (pkB in Figure 4). To construct a call-initiating Interest message, the caller includes as components in the desired name not only information necessary to route the Interest to the callee, but also both the encrypted session key block ($E_{pkB}(sk)$) and the encrypted and authenticated SIP message ($E_{sk}(SIP\ INVITE\ message)$). The callee, on receiving the Interest, decrypts the encrypted key block with its private key, recovers sk, and uses it to verify and decrypt the SIP INVITE (which itself contains the first, separately authenticated, MIKEY key exchange message to establish messaging path security). The caller then uses sk to encrypt its SIP response message (contained in a signed Data packet).

This system offers much better security than most production VoIP deployments. These usually offer no security, or at best encrypt the signaling path hop-by-hop using tunnels between proxies. Our VoCCN implementation provides end-to-end security for both signaling and media traffic. It also supports end-to-end media path security when interoperating with VoIP clients, through its encapsulation of MIKEY and SRTP.

4.2 Performance

To evaluate CCN's ability to support timely delivery of realtime data we looked at the packet arrival times for our VoCCN implementation. We used two Linux 2.6.27 machines (a 3.4 GHz Intel P4 and a 2.66 GHz Intel Core2 Duo) on either 100Mbs or 1Gbs switched

networks for this experiment. Figure 5 shows the distribution of inter-packet intervals, effectively packet *jitter*, for a 10-minute voice call made using stock Linphone (solid line) over UDP RTP, and our Linphone-based VoCCN client (dashed line) (expected interval 20 msec). Steplike appearance is due to the Linux kernel scheduling quantum (for a single process in the case of stock Linphone, and 3 processes for VoCCN in this early prototype). The VoCCN call has slightly fewer packets at or below the expected inter-packet interval, and a small number of long-interval packets at the tail. No packets were lost by either stock Linphone or our VoCCN client, however a small number of VoCCN packets (less than 0.1%) were dropped by Linphone for late arrival. With almost equal delivery performance, VoCCN and VoIP have the same call quality.

Each packet in the VoCCN exchange was individually signed with a 1024-bit RSA key; interestingly this did not noticeably impact CPU load or performance on our test machines. For more constrained environments, signature aggregation techniques (see [18]) or fast signing algorithms such as ESIGN [21] can be used to increase throughput and lower computational burden.

5. CONCLUSIONS

Content-oriented network architectures not only move content scalably and efficiently, they can also implement IP-like conversational services like voice calls, email or transactions. To demonstrate this we have implemented and tested a Voice-over-CCN prototype. The result is functionally and performance equivalent to Voice-over-IP but substantially simpler in architecture, implementation and configuration. It is intrinsically more scalable because it does not require VoIP's inbound SIP proxy (with the associated signaling state concentration). Since CCN secures content rather than the connections it travels over, VoCCN does not require delegation of either trust or keys to proxies or other network intermediaries and thus is far more secure than VoIP. Finally, thanks to certain affordances offered by CCN's structured naming, VoCCN is completely interoperable with VoIP via simple, stateless gateways.

Acknowledgements

We thank Dirk Balfanz and Philippe Golle for useful input.

6. REFERENCES

[1] Project CCNx™. `http://www.ccnx.org`, Sep. 2009.
[2] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley. The Design and Implementation of an Intentional Naming System. *SIGOPS Oper. Syst. Rev.*, 33(5):186–201, 1999.
[3] B. Ahlgren, M. D'Ambrosio, M. Marchisio, I. Marsh, C. Dannewitz, B. Ohlman, K. Pentikousis, O. Strandberg, R. Rembarz, and V. Vercellone. Design considerations for a network of information. In *ReArch*, 2008.
[4] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish. A Layered Naming Architecture for the Internet. In *SIGCOMM*, 2004.
[5] H. Balakrishnan, S. Shenker, and M. Walfish. Semantic-Free Referencing in Linked Distributed Systems. In *IPTPS*, February 2003.
[6] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, I. Stoica, and S. Shenker. ROFL: Routing on Flat Labels. In *SIGCOMM*, 2006.
[7] C. Dannewitz, K. Pentikousis, R. Rembarz, E. Renault, O. Strandberg, and J. Ubillos. Scenarios and research issues for a network of information. In *Proc. 4th Int. Mobile Multimedia Communications Conf.*, July 2008.
[8] C. Esteve, F. L. Verdi, and M. F. Magalhães. Towards a new generation of information-oriented internetworking architectures. In *CONEXT*, 2008.
[9] M. Gritter and D. R. Cheriton. An architecture for content routing support in the internet. In *Usenix Symposium on Internet Technologies and Systems (USITS)*, 2001.
[10] IETF. RFC 1889 – RTP: A transport protocol for real-time applications.
[11] IETF. RFC 3261 – SIP: Session initiation protocol.
[12] IETF. RFC 3711 – The Secure Real-time Transport Protocol (SRTP).
[13] IETF. RFC 3830 – MIKEY: Multimedia Internet KEYing.
[14] IETF. RFC 4347 – Datagram Transport Layer Security.
[15] V. Jacobson. A New Way to Look at Networking, August 2006. `http://video.google.com/videoplay?docid=-6972678839686672840&ei=iUx3SajYAZPiqQLwjIS7BQ&q=tech+talks+van+jacobson`.
[16] V. Jacobson. Making the Case for Content-Centric Networking: An Interview with Van Jacobson. *ACM Queue*, January 2009.
[17] V. Jacobson. Special Plenary Invited Short Course: (CCN) Content-Centric Networking. In *Future Internet Summer School*, August 2009.
[18] V. Jacobson, D. K. Smetters, J. D. Thornton, M. Plass, N. Briggs, and R. Braynard. Networking Named Content. In *CoNext*, 2009.
[19] 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 *SIGCOMM*, 2007.
[20] B. Ohlman et al. First NetInf architecture description, April 2009. `http://www.4ward-project.eu/index.php?s=file_download&id=39`.
[21] T. Okamoto and J. Stern. Almost Uniform Density of Power Residues and the Provable Security of ESIGN. In *ASIACRYPT*, 2003.
[22] D. Trossen. Conceptual architecture of PSIRP including subcomponent descriptions (D2.2), June 2008. `http://psirp.org/publications`.
[23] M. Walfish, H. Balakrishnan, and S. Shenker. Untangling the Web from DNS. In *NSDI*, 2004.
[24] P. Zimmermann, A. Johnston, and J. Callas. ZRTP: Media Path Key Agreement for Secure RTP. `http://www.philzimmermann.com/zfone/draft-zimmermann-avt-zrtp-15.html`.