Contrace: A Tool for Measuring and Tracing Content-Centric Networks

Hitoshi Asaeda, Kazuhisa Matsuzono, Thierry Turletti

Contrace: A Tool for Measuring and Tracing Content-Centric Networks

Author 1: Hitoshi Asaeda (Email: asaeda@nict.go.jp), National Institute of Information and Communications Technology (NICT), Tokyo, Japan

Author 2: Kazuhisa Matsuzono, National Institute of Information and Communications Technology (NICT), Tokyo, Japan

Author 3: Thierry Turletti, INRIA, Sophia Antipolis, France

Abstract

Content-Centric Networks (CCNs) are fundamental evolutionary technologies that promise to form the cornerstone of the future Internet. The information flow in these networks is based on named data requesting, in-network caching, and forwarding -- which are unique and can be independent of IP routing. As a result, common IP-based network tools such as *ping* and *traceroute* can neither trace a forwarding path in CCNs nor feasibly evaluate CCN performance. We propose "*contrace*," a network tool for CCNs (particularly, CCNx implementation running on top of IP) that can be used to investigate 1) the Round-Trip Time (RTT) between content forwarder and consumer, 2) the states of in-network cache per name prefix, and 3) the forwarding path information per name prefix. We report a series of experiments conducted using *contrace* on a CCN topology created on a local testbed and the GEANT network topology emulated by the Mini-CCNx emulator. The results confirm that *contrace* is not only a useful tool for monitoring and operating a network, but also a helpful analysis tool for enhancing the design of CCNs. Further, *contrace* can report the number of received interests per cache or per chunk on the forwarding routers. This enables us to estimate the content popularity and design more effective cache control mechanisms in experimental networks.

Keywords: CCN, NDN, ICN, Contrace, In-network cache, Measurements

I. INTRODUCTION

Content-Centric Networks (CCNs) [1] or *Named-Data Networks* (NDNs) [2] are promising approaches towards the future Internet. In a CCN, publishers provide content through the network, and receivers retrieve content by name. In this network architecture, routers forward content requests by means of their Forwarding Information Bases (FIBs), which are populated by "name-based routing protocols." A CCN enables receivers to retrieve content from an "in-network cache": if a router receiving a content request has the requested named data in its cache, it forwards the data toward the receiver without forwarding the content request to the

publisher. As such, implementing this new communication technology requires innovative ideas and research in terms of "naming," "routing," and "caching." To evaluate the architectures and protocols, researchers validate their proposals by running actual implementations, such as the CCN prototype CCNx [3], on testbeds such as PlanetLab [4]. However, because requesting and transmitting named data is unique and can be independent of IP routing, common IP-based network tools such as *ping* [5] and *traceroute* [6] are unable to evaluate CCN protocols and architectures. Furthermore, these tools do not allow the state of the in-network cache to be discovered. Hence, we need a network tool that can assist troubleshooting activities in CCNs while helping the design of in-network caching and name-based routing mechanisms.

In this paper, we discuss the design and implementation of *contrace*, an active network measurement tool for investigating the path and caching condition in CCNs. We assume that such a CCN is formed by a CCNx implementation running on top of IP. *Contrace* consists of a user program and a forwarding daemon implementation. *Contrace* "Query" messages are invoked by the *contrace* user and forwarded by CCN routers toward a specified source or caching routers. *Contrace* "Response" messages are initiated by a router holding the requested content (i.e., cache) or the publisher, and forwarded toward the *contrace* user. Thus, *contrace* facilitates the tracing of a routing path and provides additional information such as the Round-Trip Time (RTT) between routers along the path. In addition, *contrace* identifies the state of the cache, such as the access count and lifetime of cached content, on a router. Obtaining such information allows us to estimate content popularity as a means of optimizing in-network caching. Furthermore, *contrace* implements policy-based information provisioning that enables administrators to "hide" secure or private information, but does not disrupt the forwarding of messages. This policy-based information provisioning reduces the deployment barrier faced by operators in installing and running *contrace* on their routers. We developed and implemented *contrace* and conducted experiments in a CCN topology created on a local testbed and the GEANT network topology emulated by the Mini-CCNx [7]. The results show that *contrace* provides the forwarding path information per name prefix, and facilitates investigation of the content popularity and the potential cache capacity in the network.

II. MEASURING PERFORMANCE IN CCNS

A. *Overview*

CCNs enable consumers (i.e., receivers) to obtain information or content without continuous end-to-end communication channels between hosts. CCNs focus on information dissemination in the network -- how consumers retrieve content by name using "interest" packets, and how publishers (i.e., content owners) provide content through the network using "content object" packets. CCNs advocate receiver-driven communication using name prefixes for routing. In this paradigm, consumers initiate communication by sending a request

(interest) for specific content, and CCN routers forward the interest toward the content publisher responsible for the requested content. A FIB is a lookup table that is used to determine the incoming interfaces (called incoming Faces) for all content. The FIB entries comprise a name prefix and incoming Face pairs. Each CCN router also maintains a Pending Interest Table (PIT), a lookup table containing name prefixes, and outgoing Faces used to forward received content.

When a CCN router receives an interest, the router examines the content to determine whether it is cached in its local Content Store (CS). If the named content is in the router's CS, the router forwards the cached content out via the Face on which it arrived. If the named content is not in the router's CS, it searches its PIT to determine whether an interest for the same named content has already been processed. If the same named content is in the router's PIT but the Face receiving the interest has not been set, the router updates the PIT entry by adding the Face as the outgoing Face and discards the interest. If the named content is not in the router's PIT, the router creates a new PIT entry, and forwards the interest via the incoming Faces specified in its FIB.

If any router along the path toward the publisher does not have the requested content in its cache, the interest packet will finally reach the publisher. On receiving an interest, the publisher forwards the content object to its downstream router. Each CCN router on the path that receives the interest forwards the data via the outgoing Faces, stores the forwarded content in its CS if needed, and deletes the corresponding PIT entry.

B. Motivation

CCN technology changes the data retrieval mechanism from a host-centric model to a content-centric model. Name-based routing paths for forwarding interests and data are organized based on the name prefixes of the content and do not depend on IP routing. The content forwarder is not always the publisher (i.e., content owner) -- any router or node along the path may become the content forwarder if it has the requested content in its cache. Conceptually, CCN also has high affinity with multipath routing, as it can perform packet-by-packet forwarding, although there are optimization difficulties [8]. The forwarding paths can be determined by the unique CCN strategy and/or routing layer [9]. Although there are standard methods for measuring the performance of IP networks [10] and common IP-based network tools such as *ping* [5] and *traceroute* [6], they cannot be directly used to measure or trace the forwarding path in CCNs. Moreover, in-network caching is an innovative core function of CCNs, whereas there are currently no mechanisms to investigative the cache state in routers and the content distribution in the networks.

A network tool for CCNs could be used as an analytical tool to investigate fine-grained cache information such as the number of interests received per cache, and to design forwarding and caching strategies as well as new routing protocols in CCNs.

C. *Performance Metrics and Forwarding Path Information*

First, it is necessary to identify which performance metrics should be measured in CCNs. According to Pentikousis et al. [11], CCN performance metrics can be subdivided into "traffic" and "system" metrics. The most important traffic metrics for consumers are the throughput and the RTT between content forwarder (i.e., forwarding router or publisher) and consumer. Although throughput depends on the transport protocol, RTT is the natural metric for ascertaining the end-to-end network conditions, because it has no application or transport protocol dependency.

In general, although traffic metrics are important for consumers, while publishers and researchers are more interested in system metrics such as memory usage and signaling overhead, as these allow the efficiency of each networking protocol or architecture to be evaluated. In CCNs, however, system metrics that are firmly linked to in-network caching are important to all parties. This is because in-network caching affects the overall content retrieval performance, and the use of a large amount of cache may affect the memory usage of the caching router, which would reduce the forwarding performance.

Furthermore, unlike IP-based routing, both the publisher and the routers are capable of forwarding content in CCN. While the consumer does not generally need to know which content forwarder is transmitting the content to the consumer, operators need to identify the content forwarder and observe the forwarding path information per name prefix for troubleshooting and monitoring networks.

D. *Related Work*

Current CCNx components [3] incorporate a command called *ccndstatus*. *Ccndstatus* shows information about Faces and Content Store (CS) information, such as neighbor routers addresses, cache list, lifetime, and expiration time per cache in a CCN router. However, *ccndstatus* does not provide any information about transmission delay or other traffic metrics. *Ccndstatus* also requires the user to access all routers along the path to examine the in-network cache. Examining all potential forwarding routers for some specific content by accessing each router is not a practical solution for troubleshooting or monitoring networks.

In addition, network tools should allow researchers to investigate the performance of content retrieval or design algorithms/mechanisms for in-network cache or name-based routing in CCNs. Unfortunately, in many cases, researchers are forced to use simulators for their experiments because there are no tools to satisfy their needs. For instance, Rossi and Rossini [12] were forced to use simulations to analyze the caching performances of: 1) homogeneous cache sizing (in which the CS spaces of all routers are the same) and 2) heterogeneous cache sizing (in which a given amount of memory resources are allocated across routers by considering graph-related centrality metrics). It is preferable to investigate such strategies using the actual running codes on a real (or emulated) network.

III. DESIGN AND IMPLEMENTATION

A. Design

Traceroute is a useful tool for analyzing the routing conditions in IP networks as it provides intermediate router addresses along the path between source and destination and the RTT for the path. However, as outlined in Section II.B, this IP-based network tool cannot trace the name prefix paths used in CCNs. Furthermore, given a source-rooted forwarding path per name prefix, tracing from a source (i.e., forwarding router or publisher) to a consumer is difficult, because we do not know which branch of the source tree the consumer is on. Additionally, it is not feasible to flood the entire source-rooted tree to find the path from a source to a consumer. Conversely, walking up the tree from a consumer toward a source is feasible because CCN routing protocols know the upstream router for a given source. Tracing from a receiver to a source can involve only the routers on the direct path. This idea is inspired from the design of the IP multicast traceroute facility [13].

To determine the state of in-network caching under CCN, the efficiency and performance of in-network caching can be measured by the following metrics for CS in the router: 1) size of the cached content, 2) number of chunks for the content, 3) number of accesses (i.e., received interests) per cache or chunk, and 4) lifetime and expiration time per cache or chunk. A fifth metric of the forwarding router or publisher's IP address is included among the states of in-network caching.

We propose "*contrace*," a network tool for active measurement in CCNx-based CCNs. *Contrace* provides: 1) the RTT between content forwarder (i.e., forwarding router or publisher) and consumer, 2) the states of in-network cache per name prefix, and 3) the forwarding path information per name prefix. *Contrace* is also a useful analysis tool to aid in the future design of CCNs. It can report the number of received interests per cache or chunk on the forwarding routers, indicating the popularity of the content. By comparing this number across all forwarding routers with *contrace*, we can illustrate the content popularity and evaluate potential cache controls in experimental networks.

The *contrace* implementation consists of the *contrace* command and the *contraced* daemon implementation running on a CCNx router (*ccnd*). Figure 1 shows the socket communications between the *contrace* command, *contraced*, and *ccnd*. *Ccnd*, which is a component of CCNx, is a daemon that forwards and caches content objects. The *contrace* command is invoked by specifying the name prefix of the content and/or the publisher or forwarding router. It initiates the request (i.e., creates the Query message described in Section III.D) to obtain forwarding path and cache information. *Contraced* running on the *contrace* user receives the request and communicates with local *ccnd*. *Contraced* retrieves cache and other information from local *ccnd* via a UNIX domain socket, and forwards the request to the neighbor *contraced* running on the upstream neighbor router (if it is not the content forwarder or the specified end point). When the request reaches the content forwarder,

contraced on the forwarder transmits the reply (by sending the response message described in Section III.D) to *contraced* running on the downstream neighbor router, and the reply is forwarded back to the *contrace* user.

B. *Contrace Command*

The *contrace* command enables the *contrace* user to investigate the forwarding path based on the name prefix of the content (e.g., ccnx:/news/today/). The usage of *contrace* is as follows (the descriptions of all the options are shown in Table 1):

Usage: contrace [-R] *name_prefix* [--nocache] [-h *content_forwarder*] [-g *gateway* [-g *gateway* ...]] [-s *skip_node* [-s *skip_node* ...]] [-p *port*] [-w *wait_time*]

The name prefix is a mandatory option for the *contrace* command. If the *contrace* user does not know the name prefix of the content should be specified, s/he can specify "ccnx:/," the default name prefix. If this default name prefix is given, the user will receive all cache information from the upstream neighbor (adjacent) router(s).

It is also possible to specify both the name prefix of the content and use the "-h" option with the content forwarder, which could be the publisher or forwarding router for the content (e.g., ccnx:/news/today/ –h pub.example.com). If both name prefix and content forwarder are specified, the query message is forwarded in accordance with the IP routes, which could be independent of the CCN FIBs. The *contrace* user would then obtain the forwarding path information and cache information for the specified name prefix from the specified content forwarder. If the default name prefix and content forwarder are specified, the *contrace* user obtains all cache information from the specified content forwarder.

In addition to the RTT between content forwarder and consumer, when the *contrace* command specifies the "-h" option and the IP address of the content forwarder (or "-g" option and the IP address of gateway), the RTTs of other alternative paths for the specified name prefix are measured. This is useful for troubleshooting and in the design and evaluation of routing protocols, because the output potentially indicates a better path for the named content.

When the *contrace* user only wants to check the forwarding path information (i.e., without cache information) and the RTT between content forwarder and consumer, s/he can utilize the "--nocache" option.

After the *contrace* command is invoked, it waits for the reply from the upstream router, extracts the returned response messages, and shows the information on the standard output. To cancel the request when the message packets are lost during transmission, or if there is no route for the specified name prefix or publisher/router, *contrace* forces the session to close after the timeout. Following the timeout, even if the *contrace* user receives a response, it is discarded. The "-w" option specifies the timeout value (in seconds) that the *contrace* user will wait for the response. The default timeout value is 4 s.

C. *Contraced Forwarding Daemon*

Contraced is a daemon that transmits query and response messages. As seen in Fig. 1, when *contraced* receives a query message from a *contrace* program, it communicates with local *ccnd* via a UNIX domain socket to retrieve information about the upstream router(s) from its FIB. For *ccnd* to communicate with *contraced*, a small patch must be applied to *ccnd*. If the local *ccnd* is not the content forwarder for the request, *contraced* forwards the query message to the neighbor *contraced* running on its upstream neighbor *ccnd* over TCP (currently using 9696 port). When the request reaches the content forwarder, *contraced* on the router or publisher retrieves the cache information from its CS. The response message is then transmitted back to the neighbor *contraced* that had forwarded the query message. The response message is forwarded toward the *contrace* user in a hop-by-hop manner.

Contraced can transmit messages to any node running *contraced* without referring to the CCN FIB. When *contraced* receives a query message including the "-h" option record (mentioned in the previous section), it forwards the query message to the specified address in accordance with the IP routes. This implementation enables the *contrace* user to investigate the alternative route or cache information for the specified name prefix.

As introduced in Section II.B, if it is difficult to optimize the transmission timing or select the best route, the CCN strategy or routing layer can use different Faces defined in multipath routes to transmit a request for specific content. To compare the performance of content fetch from the different paths, *contraced* immediately and simultaneously transmits the query message to these Faces, whenever multipath routes are configured for the name prefix.

D. *Messages and Records*

Contrace uses two message types: Query and Response. Both messages are encoded in Type, Length, and Value (TLV) format. The Type field is eight bits long, the Length field is 16 bits long, and the length of the Value field is variable. The Query and Response Type values are 0x01 and 0x02, respectively. The Query message is forwarded in a hop-by-hop manner. When it reaches the content forwarder, the content forwarder turns the Query message into a Response message by changing the Type field value from 0x01 to 0x02.

When Query or Response messages are transmitted, *contraced* appends one or more "records" to the Response message. The following eight records are defined and transmitted with the same TLV format: 1) *contrace* user address, 2) name prefix, 3) -h option, 4) -g option, 5) -s option, 6) timestamp, 7) responder address, and 8) cache status. Records 2) -- 5) are appended by the *contrace* command only if specified, the timestamp record is appended by every *contraced* along the path upon receiving the Query, and the responder address (and cache status records if requested) are appended by the content forwarder. The *contrace* user address and responder address records are appended to the Query and Response, respectively. A timestamp

record that specifies the forwarding router or consumer's IP address and UNIX time on the local machine is mandatory for the content forwarder.

E. Response Policy and Configuration

Although *contrace* gives excellent troubleshooting cues, some network administrators or operators may not want to disclose everything about their network to the public, or may wish to securely transmit private information to specific members of their networks. *Contrace* provides policy-based information provisioning allowing network administrators to specify their response policy in a configuration file, *contrace.conf*, for each router.

We define the following three options that can be specified in the *contrace.conf* file for when the router receives a Query message: 1) whether "-h" option is allowed or rejected, 2) whether "-g" option is allowed or rejected, and 3) whether cache information is disclosed, partially disclosed (i.e., except the request specifying the default name prefix (i.e., ccnx:/)), or not disclosed at all.

On the other hand, we entail that each router runs *contraced* and does not disrupt forwarding *contrace* Query and Response messages. When a Query message is received, the router appends the timestamp record and forwards the request to the upstream router toward the content forwarder, but can hide other information according to the policy configuration.

Cases will arise in which a router along the path does not support *contraced*. In such cases, a downstream router that supports *contrace* will reply to the *contrace* client with a message indicating that the upstream router does not support contrace. Further, when an upstream router rejects a Query messages, its downstream router will reply to the *contrace* client with a message indicating that *contrace* Query is prohibited.

IV. *CONTRACE* IN ACTION

First, we deployed seven VMs on two physical PCs. Both the VMs and the PCs ran the Linux operating system (Ubuntu Desktop 12.04 LTS 64-bit version). The VMs communicated with each other via a fast ethernet switch, as shown in Fig. 2(a). CCNx 0.8.2 and *contrace* components (*contrace* command and *contraced*) were installed on each VM. The *ccnds* on the VMs formed the logical CCN topology shown in Fig. 2(b).

Figure 3 shows the output of *contrace* commands input by a consumer (descriptions of all outputs are listed in Table 2). We verified that *contrace* messages were forwarded in a hop-by-hop manner, and that each forwarding router appended its own timestamp record. Further, when the *contrace* Response message returned to the *contrace* user, the *contrace* program calculated the One-Way Delays (OWDs) between each router and the RTT from the *contrace* user to the content forwarder initiating the Response. To ensure the OWDs shown

in the output were correct, the Network Time Protocol (NTP) [14] was installed on each router to synchronize their internal clocks. Note that even if the routers' internal clocks are not synchronized, the RTT measured between the *contrace* user and the content forwarder (yellow oval in Fig. 3) is always correct because both start and end times are given by the *contrace* user.

In this experiment, router B was configured as a multipath for receiving the content of ccnx:/news/today/. As shown in Fig. 3(a), the Query message was transmitted from router B in two directions (to routers C and E) simultaneously. Two Response messages (transmitted from router C and the publisher) were then received by the *contrace* user. Another observation from Fig. 3(a) is that router C obtained the cache of ccnx:/news/today/ from router D, not the publisher. From Fig. 3(b), we can observe that the publisher (with IP address 172.16.6.196) cached four contents, while the output of Fig 3(c) indicates only ccnx:/news/today/ as its cache information, because the *contrace* command explicitly specified the prefix.

Next, we investigated the caching performance of CCNx using *contrace* with a realistic topology on an emulator. Rossi and Rossini [12] assessed the cache hit probability and path stretch $d / |P| \in [0, 1]$, where d is the hop count of the data and $|P|$ is the hop count from the consumer to the publisher (i.e., without caching). We built the GEANT network topology, which consists of 27 nodes, using Rocketfuel [15] on top of the Mini-CCNx emulator [7]. The publisher was assumed to have *100* contents. These were characterized by a Zipf probability distribution with the value of the Zipf exponent $\alpha = 1.25$. Each consumer sends *2500* interests in total, based on the Zipf popularity setting. Under the homogeneous cache sizing approach, each router has equal CS space to satisfy received interests with a probability of about *50%*. In the heterogeneous approach, the Degree Centrality (*DC*) metric was adopted (*DC(n)* is defined as the number of links incident upon a router *n*) for its simplicity and good allocation criterion (i.e., the greater the *DC* of a router becomes, the more CS space the router should have) [12]. We allocated a CS space that can satisfy the received interests with a probability of about *80%* to each router with $DC \geq 4$.

By running *contrace* at the consumer, we assessed path stretch, because *contrace* traces the path from the consumer to the router/publisher having the requested data. Figure 4(a) shows the measured path stretch, and compares the homogeneous and heterogeneous approaches at four consumers randomly selected. These results illustrate the improvement in path stretch in the heterogeneous case. Furthermore, because *contrace* can show the number of received interests per cache or chunk on a router, the content popularity (i.e., the access number of accesses for each content) can be measured (as shown in Fig. 4(b) for the publisher). It is also clear that, under the heterogeneous approach, the access number of popular content (e.g., content-index 1, 2, and 3) decreases with the improvement in path stretch. We conclude that *contrace* facilitates investigation of the performance of the CCNx protocol, routing/caching strategy in testbeds and/or emulated networks, and the design of future CCN architectures.

V. Conclusion and Outlook

The novel features introduced by the content-centric communication model necessitate specialized network tools that do not currently exist. In this article, we proposed "*contrace*," a network tool for CCNs that can be used to investigate: 1) the RTT between content forwarder (i.e., forwarding router or publisher) and consumer, 2) the state of an in-network cache per name prefix, and 3) forwarding path information per name prefix. *Contrace* facilitates troubleshooting of CCNs and can help in the design and the optimization of routing protocols and routing/caching strategies.

We implemented *contrace* on a CCNx router, and showcased this tool on a testbed and on a large emulated network. Our experiments demonstrated that *contrace* provides forwarding path information per name prefix, and enables the measurement of CCN traffic and system performance metrics. We also investigated cache information for each name prefix, the content popularity, and the potential cache capacity in experimental networks.

In future work, we plan to add new functionalities to *contrace*, to obtain the cache hit probability at routers along a path, and to visualize characteristics of the dispersal of target content (i.e., where/when requests for target content are concentrated). This will promote the extensive analysis of caching/routing strategies, and may engender a fundamentally new design approach.

VI. Acknowledgement

This research is partially supported by the Japan Society for the Promotion of Science (JSPS) through the Bilateral Joint Research Program and INRIA through the AYAME Program in the context of the Designet/Simulbed associated team.

References

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking Named Content," *Proc. ACM CoNEXT 2009*, December 2009.

[2] Named Data Networking, available at: http://named-data.net/.

[3] CCNx, available at: http://www.ccnx.org/.

[4] PlanetLab, available at: http://www.planet-lab.org/.

[5] J. Postel, "Internet Control Message Protocol," *IETF RFC 792*, September 1981.

[6] G. Malkin, "Traceroute Using an IP Option," *IETF RFC 1393*, January 1993.

[7] C. Cabral, C. E. Rothenberg, and M. Magalhaes, "Mini-CCNx Fast Prototyping for Named Data Networking," *Proc. ACM SIGCOMM ICN'13 Workshop*, August 2013.

[8] G. Carofiglio, M. Gallo, L. Muscariello, M. Papalini, and S. Wang, "Optimal Multipath Congestion Control and Request Forwarding in Information-Centric Networks," *Proc. IEEE ICNP 2013*, October 2013.

[9] S. Braun, M. Monti, M. Sifalakis, and C. Tschudin, "An Empirical Study of Receiver-based AIMD Flow-Control Strategies for CCN," *Proc. ICCCN 2013*, August 2013.

[10] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis, "Framework for IP Performance Metrics," *IETF RFC 2330*, May 1998.

[11] K. Pentikousis, B. Ohlman, E. Davies, S. Spirou, G. Boggia, and P. Mahadevan, “Information-centric Networking: Evaluation Methodology,” *IRTF ICNRG draft (work-in-progress)*, July 2014.

[12] D. Rossi and G. Rossini, “On Sizing CCN Content Stores by Exploiting Topological Information,” *Proc. IEEE INFOCOM NOMEN Workshop*, March 2012.

[13] H. Asaeda and W. Lee, “Mtrace Version 2: Traceroute Facility for IP Multicast,” *IETF Internet-Draft (work-in-progress)*, October 2014.

[14] D. L. Mills, “Network Time Protocol (Version 3) Specification, Implementation and Analysis,” *IETF RFC 1305*, March 1992.

[15] N. Spring, R. Mahajan, and D. Wetherall, “Measuring ISP Topologies with Rocketfuel,” *Proc. ACM SIGCOMM*, August 2002.

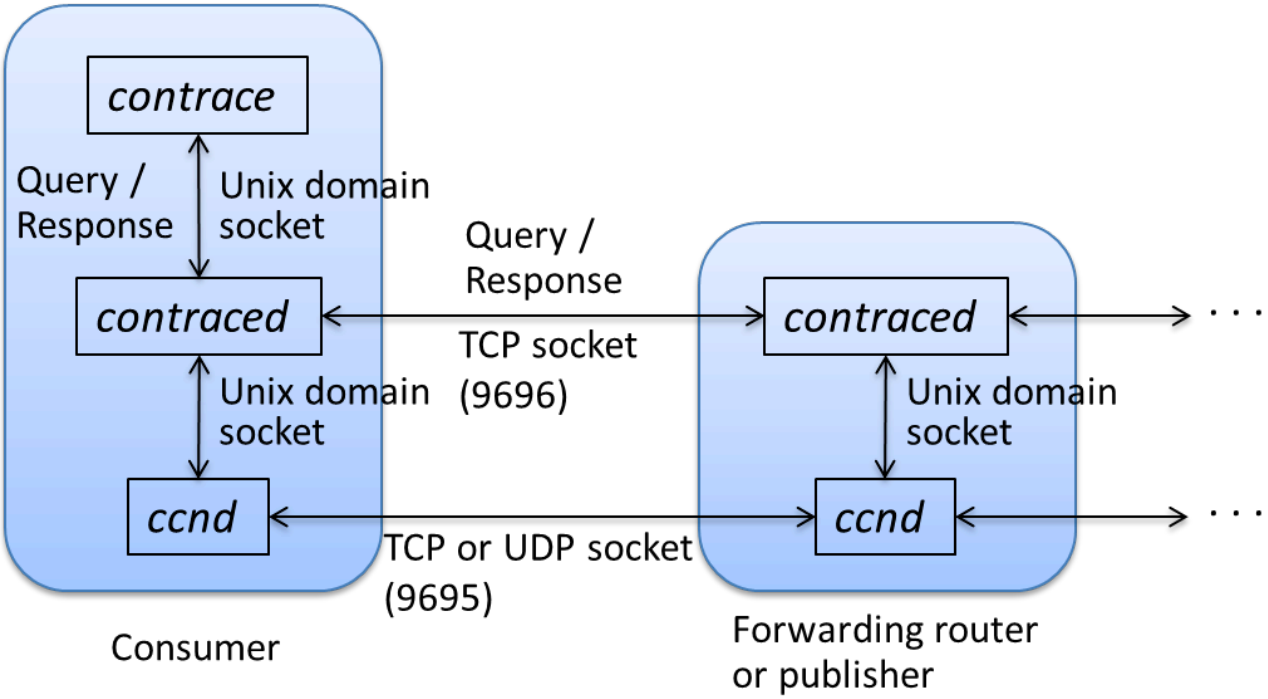


Fig. 1. Socket communications among *contrace* command, *contraced*, and *ccnd*.

Table 1. Description of *contrace* command options.

Option	Description
[-R] *name_prefix*	Mandatory option. Name prefix of the content (e.g., ccnx:/news/today/) the *contrace* user wants to trace. "ccnx:/" is the default name prefix. The -R option requires an exact match for the name prefix; otherwise, a partial match is allowed.
--nocache	No cache information is requested. Only the router names and RTT along the path are returned.
-h *content_forwarder*	Node name or IP address of the publisher or forwarding router (e.g., -h pub.example.com). If this is specified, Query is forwarded in accordance with IP routing path (without looking up CCN FIB).
-g *gateway* [-g *gateway*] …	Node name or IP address of gateway router(s) to be passed through. Similar to source routing. If this is specified, Query is forwarded in accordance with IP routing path (without looking up CCN FIB). This option must be used with the -h option (an error is returned if this is not specified).
-s *skip_node* [-s *skip_node*] …	Node name or IP address of the forwarding router to be skipped. If the specified router receives a Query, it forwards the Query to the upstream router regardless of its cache contents. This option must not be used with the -h option (an error is returned if both are specified).
-p *port*	TCP port number for Query/Response. Default: 9697.
-w *wait_time*	Timeout value (in seconds) that the *contrace* user waits for a Response. After the timeout, the *contrace* user silently discards the Response. Default: 4 s.

Table 2. Description of *contrace* command outputs.

Category	Description	
Requested prefix (starting from “Contrace to prefix name”)	Name prefix of content asked for by *contrace* user	
Routing information (starting from “Path from the content forwarder (publisher or forwarding router) to *contrace* user”)	Col. 1	Sequential number of one-way hops on the path
	Col. 2	Node name or IP address on the path
	Col. 3	One-way delay from previous node to this node
	Col. 4	Sum of one-way delays from *contrace* user to this node. If this node is *contrace* user, this value indicates the RTT between *contrace* user and the content forwarder.
Content Store (CS) information (starting from “Content store on the content forwarder”)	Col. 1	Sequential number
	Col. 2	Name prefix of the cached content
	Col. 3	Size of the cached content
	Col. 4	Number of content objects for the cached content
	Col. 5	Node name or IP address from which the cached content is provided
	Col. 6	Number of received interests
	Col. 7	Lifetime of the cached content
	Col. 8	Expiration time of the cached content

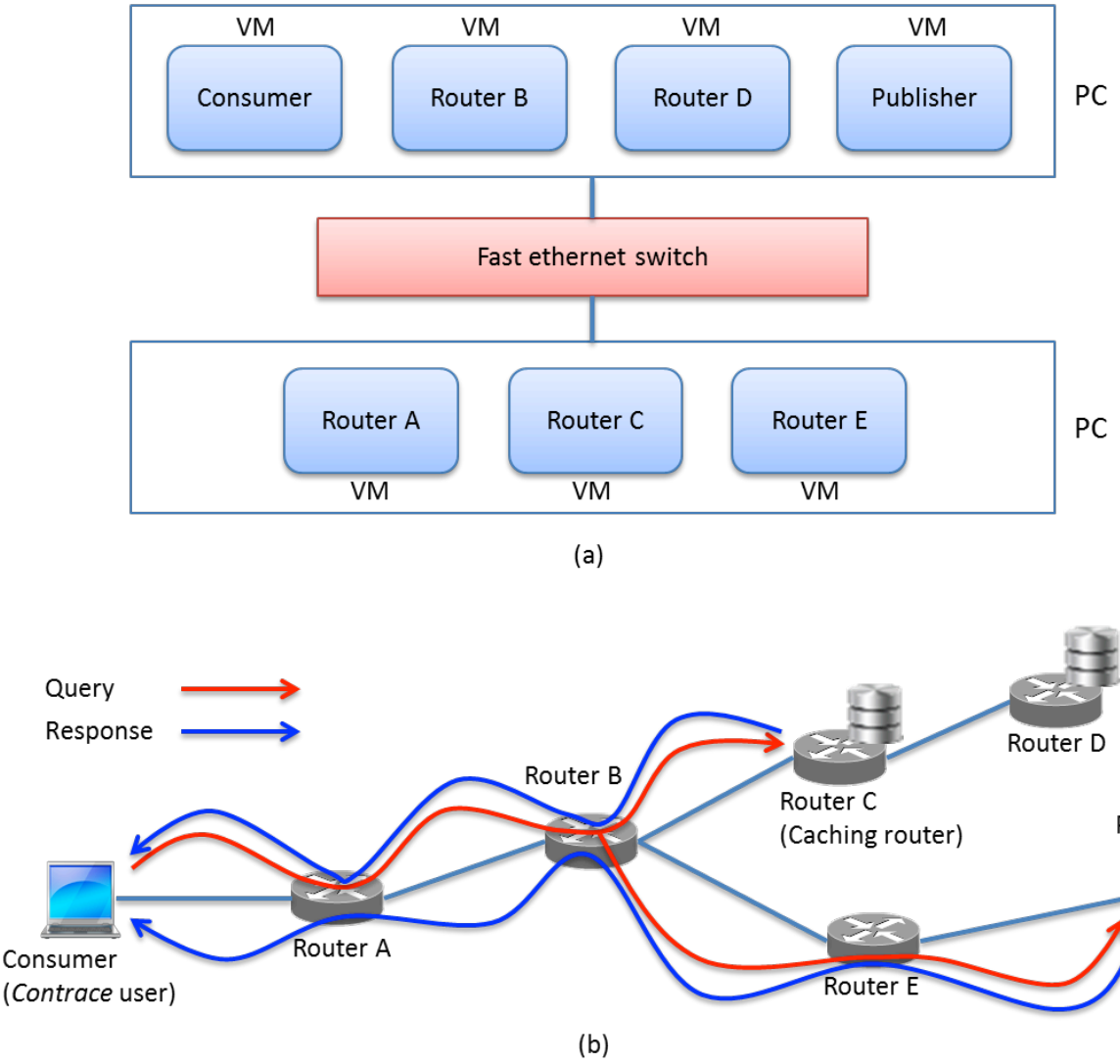
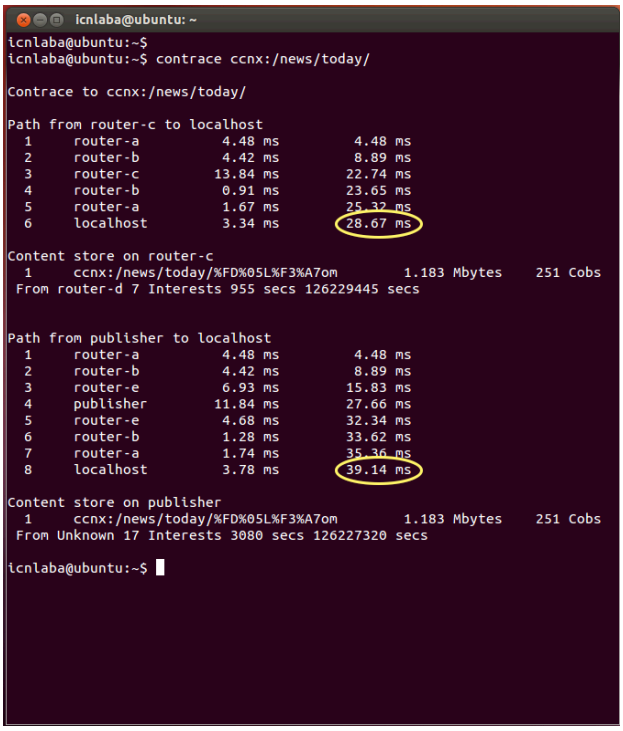
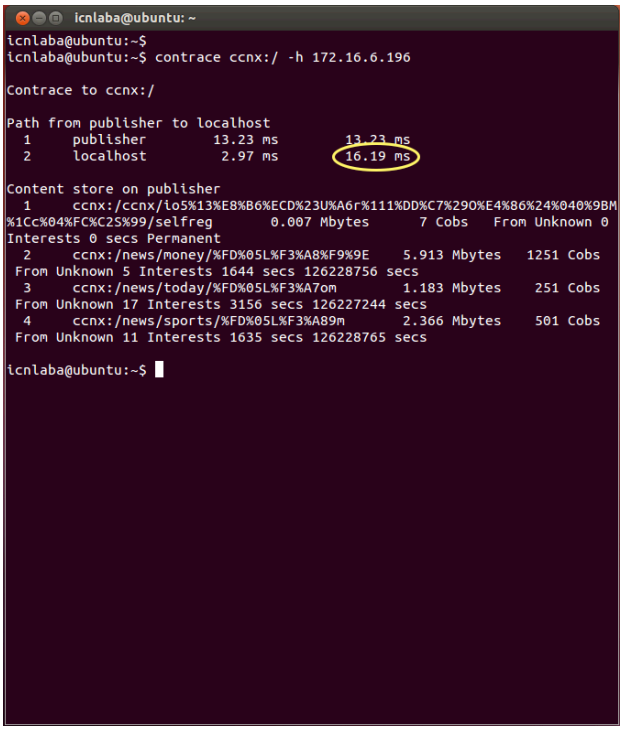


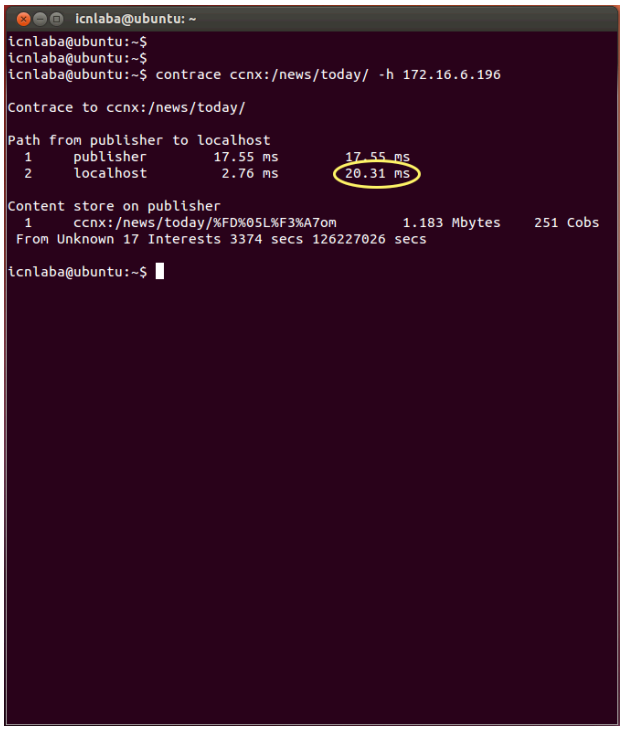
Fig. 2. Experimental configuration of the local testbed: (a) configuration of seven VMs on two PCs, (b) CCN topology formed by the VMs.



(a)



(b)



(c)

Fig. 3. Output of *contrace* command with different options: (a) name prefix, (b) default name prefix and content forwarder, and (c) name prefix and content forwarder.

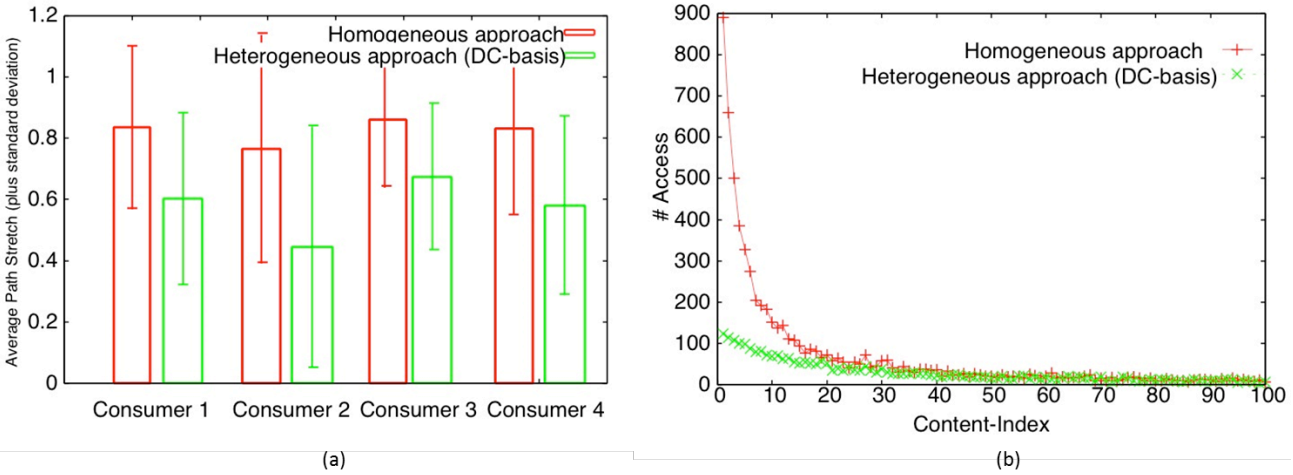


Fig. 4. (a) Average path stretch measured at the four consumers and (b) content popularity measured at the publisher in the homogeneous approach and the heterogeneous approach measured with *contrace* running on the GEANT topology emulated by Mini-CCNx.