ABSTRACT
It becomes apparent that content should be center to the model of information access in the current Internet, which results in research efforts been put together as content-oriented network architectures. In light of our observation that in practice it is not feasible to replace the current Internet architecture with a clean-slate one and that it is the existing transport protocols based on socket addresses that constrain the realization of content-oriented transport, in this article we present Content-Oriented Transport Protocol (COL4), that supports efficient content-oriented transport over the existing Internet infrastructure. We expect our protocol provides a new foundation for further research and development on content-oriented networking.

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, Protocol, Performance

Keywords
Content-oriented networking, transport protocol

1. INTRODUCTION

The Internet and its protocols are designed for the model of end-to-end data communication, such as client-server data communication. However, it becomes apparent that it is content itself that should be center to the model of information access [9, 11]. That is, information consumers do not care where the content is located, but which information they need. The trend of emerging content-oriented Internet access is responded by a variety of technologies through peer-to-peer (P2P) overlays, content delivery networks (CDN) [15], publish/subscription systems and cloud platforms. Unfortunately, these facilities are indirect and sometimes expensive for end users. Thus, we cannot expect all content in the Internet to be facilitated with these technologies.

Recently, research efforts have been put into adjusting and redesigning the Internet communication model in a clean-slate fashion, and their approaches are so-called content-oriented network architectures. In these architectures, user operations are centered on content rather than traditional host-based access [4]. Correspondingly, routers will forward packets according to interest of content instead of destination host address, and more importantly, routers will cache content using their buffer space to improve content access performance [2]. People argue the new architecture has advantages in three folds [17]. First, content-oriented trend is matched coherently and directly by the architecture. Second, end-to-end security, which is rarely met today, could be enforced in the architecture. Finally, the architecture could facilitate user choice and competition where possible.

Although clean-slate research is important to help people understand the problems and solutions, their solutions are not compatible with today’s Internet and could not be accepted without a transition process [10]. It is especially unaffordable to replace the current Internet Protocol (IP) with clean-slate ones, because this would result in redesigning addressing, routing and forwarding for the global Internet, which is desirable for neither end users nor network operators.

We argue that it is the current transport protocols that are based on socket addresses (a tuple of an IP address and a port), such as TCP, UDP and SCTP, that constrain the realization of content-oriented communication. In this paper, we propose Content-Oriented Transport Protocol (COL4), a transport layer protocol to support most features of content-oriented network
architectures over current Internet infrastructures by embracing following features:

1. COL4 is designed to be compatible with the current Internet Protocol (IP), with the ability to be incrementally implemented and deployed.

2. COL4 is a protocol abandoning socket addresses. All operations in this protocol are content-oriented and connection-less.

3. COL4 includes congestion control to be fair with existing transport protocols and enable traffic engineering.

As far as we know, COL4 is the first transport protocol operable today that supports a variety of content-oriented network concepts. Our evaluation experiments show good results in deploying content-oriented network architectures using COL4.

The remaining of this article is organized as follows. We describe design decisions we have made to realize content-oriented transportation over current Internet in Section 2. After that, we elaborate how the protocol works in Section 3. In Section 4, our implementation using network simulator is explained and evaluated. Finally, we discuss related work in Section 5 and concluding remarks are presented in Section 6.

2. DESIGN

In this section, we first discuss the problem of current transport protocol in supporting content-oriented networking architectures. We then describe major design decisions we have made in designing COL4 to realize content-oriented transfer over current Internet infrastructure.

2.1 Problem with the existing protocols

All transport protocols widely used nowadays, whether connection-oriented ones (such as TCP and SCTP) or connection-less ones (such as UDP), are operated based on Berkeley sockets [16]. Socket addresses, which are usually presented as a tuple of an IP address and a port, are used as the identifiers of data communication in these protocols. Thus, these communications are constrained to local and remote socket addresses, which obviously leads to the current host-based communication model.

Besides the constraint brought from using socket addresses, these protocols are not in favor of supporting content-oriented communication since they treat their payloads as arbitrary byte flows. This behavior in these protocols has two problems. First, the identity of content is excluded during transfer. Thus, content transfer is opaque to the whole network it traverses except negotiating endpoints and it is impossible for intelligent devices within network to advise a better location for content retrieval than the original destination. Second, the security of transfer relies on untrustworthy connection information such as is done in Transport Layer Security (TLS) [6], which ought to be placed in the transferred content itself.

In the light of this observation, we come up with the design decision that a transport protocol supporting content-oriented transport should abandon Berkeley sockets and identify content it is carrying.

2.2 Definition of content

Just as socket addresses are the bases of today’s host-based communication, content will be the foundation of content-oriented transport. A precise definition of content is required in designing the protocol.

In COL4, a piece of content is uniquely identified by its publisher, its name and version. A publisher identifier is a registered global-unique name such as (but not necessarily the same as) a domain name. Publisher identifiers are important as they correspond to certificates issued by registration organizations so that content could be signed and verified. Publishers could arbitrarily specify the name of some piece of content, which brings two advantages. First, although network devices will treat names as flat ones for faster processing, a publisher could still organize internal structure of content naming (for example, hierarchical one as adopted by [9]) according to the actual requirement of the publisher or its applications independent from other publishers. Second, different publishers could publish content under the same name, which enables content retrieval when the user has only the information about the name but does not know or care who the publisher of some piece of content is. Finally, a version number, an epoch timestamp, specifies the release time of some piece of content. Using the version number, a publisher could update its content with the same name, preventing both exhaust and complicity of names.

Sometimes, it is not necessary to acquire the whole piece of content, especially when it is very large, such as video clips. COL4 supports the partial transfer of content by defining partitions of a piece of content as the minimum unit in the content transfer. Except the last partition (or content with only one partition), all partitions are in size of 2^n bytes, which is assigned by the publisher ranging from 1KB to 1MB. In current implementation (see Section 3), there may be at most 32767 partitions in a piece of content, that is, the maximum length of any piece of content is 32GB, sufficient for use because it is even larger than a typical Blu-ray disk (with 25GB capacity).

2.3 In-network processing

One of the most essential things for supporting content-oriented networking architecture is that in-network processing should be enabled by the transport protocol.
That is to say, devices within network should be able to know which piece of content is being transferred by some packet. Moreover, such information should be able to be acquired without deep packet inspection (DPI) as it will bring devices high computation overhead and thus harm the scalability of the architecture.

COL4 fulfills this requirement by carrying necessary information to identify a piece of content in packets, as illustrated in Figure 1. As a transport protocol, it is intuitive that COL4 supports incremental deployment as it carries all information in layer-4 headers. Network devices that do not support COL4 (such as traditional IP routers) could simply forward packet according to IP headers and ignore COL4 headers as they typical do with today’s transport protocols. On the other hand, intelligent network devices supporting COL4 could do shallow packet inspection (SPI) to acquire content information, and this would not bring much overhead to those devices thanks to the information located in COL4 headers with constant structure.

Despite that COL4 is a transport protocol that expects in-network processing to assist content transport procedure, the protocol does not specify the forwarding strategy that should be adopted by network devices. Network operators, or network users themselves if they are using slice-based network facilities, could decide the forwarding strategy network by network as long as it would not tamper the reliability of the communication. That is, the forwarding strategies of COL4 must satisfy the following two rules. First, we enforce forwarding to be loop-less. Second, fall-back forwarding using IP header information ensures the reachability to a destination, not necessarily the optimal one.

2.4 Content security

Just like most content-oriented networking architectures, COL4 adopts content-based security. That is to say, the protection and trust of content is convinced by the transferred content itself, and not by the transferring method or its endpoints. In COL4, all content is mandatorily authenticated with digital signatures, and private content is optionally protected with encryption.

In contrast to Transport Layer Security (and some content-oriented networking architectures), COL4 does not perform per-packet signatures. COL4 requires all partitions to be publicly authenticatable instead. Each partition transferred in COL4 carries a small amount of auxiliary data authenticating the binding between the identifiers of the piece of content (that is, its publisher, name and version) and the actual data of the partition. For example, publisher could use standard public key signatures to generate signed checksum for every partition and append it to the partitioned data. Anyone that retrieved the partition with its signed checksum can verify the identifier-data binding is signed by a specific key of the publisher. We will not elaborate how to establish trust in keys as this issue is already discussed in the other literature, and there are existing models, for example, simple public key infrastructure (SPKI) [7] that can fulfill the content security requirements in COL4.

3. PROTOCOL

Content-oriented transport protocol is designed to operate on top of the current network layer protocols such as Internet Protocol (IP) and transfer data over the existing network infrastructures. This section describes the detailed processes and message formats of our proposed protocol by its four primitives: availability announcement, content request, content transmission, and transfer control.

3.1 Availability announcement

An end host sends availability announcement messages to declare that it holds a specific piece of content to neighboring hosts in edge networks in COL4. The destinations of these messages are all the hosts on the same network (that is, 224.0.0.1 in IPv4). Users of a host, or their agent applications, can configure which pieces of content are announced by the protocol.

An availability announcement message is composed of the availability announcement header without any payload, as illustrated in Figure 2. Since routers are
permits to forward availability announcement messages, weight of the message is used to denote a virtual distance of recipients of the message to the piece of content, which is calculated according to the following equation:

\[
\text{weight} = w_{in} + w_{local} + w_{out} \quad (w_{local} \geq 0, w_{out} \geq 1).
\]

\(w_{in}\) is the weight value of received message, which is 0 for original sender of the message; and \(w_{local}\) is a local weight value, reflecting the readiness for accepting content retrieve request by an end host or the willingness of transport content by a router; finally \(w_{out}\) is a link-specific weight value that shows the virtual distance. The always-positive \(w_{out}\) value prevents forwarding loop of availability announcement messages as routers are forbidden to forward messages for the same piece of content whose weight is larger than local cached ones from the same source. It also ensures that a sender of the message could refuse routers from forwarding it by setting the weight to 0xFF.

3.2 Content request

End hosts send content request messages in order to retrieve a specified piece of content. COL4 empowers two methods of content request: ambiguous request and precise request. Ambiguous content request is used when the requesting host does not have any information about the content, for example, in case that it only knows the name of desired information. In contrast to this, precise content request will be preferred when the host has complete information about the identifier of content, and is always chosen when the host wants to retrieve several partitions of a specific content.

Figure 3 illustrates these two sorts of content request messages, differentiated by the 8-bit method field. In ambiguous request, incomplete content identifier is allowed in the following way. First, publisher could be omitted in the request so that the request will match all content with the specified name published by anyone. Second, a version number in ambiguous content request is not used for exact match but for matching any version that is newer than the specified one. Such features could be utilized to inquire about later version of the same content that one already holds.

On the other hand, a complete content identifier is compulsory in precise content request and exact matches are performed to such requests. Moreover, there are two extra 16-bit fields in precise content request headers to indicate which partitions the sender of the request is interested in for partial retrieval.

3.3 Content transmission

Content transmission messages are sent in response to content request messages to provide a specified piece of content. Content transmission in COL4 is conducted in unit of partitions. Hence, the transmission procedure of a partition may span across several messages, as each message has to fit into a network layer packet. The first message in the transmission is distinguished from following messages. A content transmission message includes content transmission header followed by content data to be transmitted, as illustrated in Figure 4.

The header of the first message of content transmission, as shown in Figure 4(a), provides detailed infor-
Figure 5: Format of the transfer control header

3.4 Transfer control

The purpose of the transfer control in COL4 is two-fold, reliability assurance and congestion avoidance. The former guarantees that recipient could retrieve content efficiently while the latter prevents networks from congestion collapse. Both of the functions are carried out in COL4 by the sender of content transmission messages, by reacting to transfer control messages sent by recipients of content data. Unlike per-segment acknowledgment mechanism (with a maximum delay of 0.5 seconds allowed) [3] that is adopted in TCP, recipients in COL4 only provides minimum necessary feedback to the sender to reduce the overhead of transfer control message to the network and endpoints.

Recipients provide required information for transfer control with the message illustrated in Figure 5. The 8-bit TC_TAG field is the essential part for these messages, whose detail is described in Figure 5(b). The highest 3 bits of the field indicate finish, initial and jump control instructions respectively, while the lowest 4 bits indicates congestion levels which will be discussed later.

COL4 uses three kinds of control instructions to ensure efficient and reliable content transmission. When the first content transmission message reaches its recipient, it will immediately deliver the initial control message to confirm that the transmitted partition is requested. The initial control message is also used to calculate the round-trip time (RTT) by the sender. Similarly, when the recipient wants to terminate the transmission procedure as it has already held the whole partition or the partition is no longer required, a finish control message is sent to indicate the termination of current transmission. Furthermore, out-of-order transfer and packet loss happen in networks as COL4 operates on unreliable network layer protocols. Jump control messages signify the occurrence of such incidents and indicate the next expected offset of data in the partition by the 32-bit field carried in the messages. The sender will relocate its transmitting pointer to the specified offset and resume its transmission from there. Congestion avoidance will also be triggered by this control message.

Generally speaking, COL4 adopts additive increase / multiplicative decrease (AIMD) algorithm that is used in TCP congestion control [1] to retain the fairness of transmission between COL4 and other transport protocols. Thus, COL4 could operate efficiently even when there is no network device supporting this protocol. Moreover, borrowing ideas from explicit congestion notification (ECN) [8] and its latest research work [5], COL4 utilizes explicit congestion level indication to allow senders adaptively adjust sending rate according to different congestion levels on the bottleneck routers. If the congestion level of a router is higher than that indicated in the messages, the router will replace the congestion level value with that of local status. In this way, the recipient of a content transmission message will be aware of the status of most congested routers in the networks. Besides the three kinds of messages with transfer control instructions, a transfer control message without any instruction bit set will also be sent when the congestion level changes. The sender of content transmission messages can accurately estimate the congestion level with the existence of such indicator and will rapidly adjust to the optimal sending rate.

4. IMPLEMENTATION

In this section, we introduce our COL4 implementation in the ns-3 network simulator [12] where packet-level simulating environment and a complete TCP/IP protocol stack are provided. We first describe the overview of the implementation, and then present the evaluation the transfer control of the protocol using our implementation.

4.1 Overview

There are five principal modules in the COL4 implementation as explained in Figure 6.

The request dispatcher module provides application programming interface (API) for applications to regis-
Another important API for content registration is provided by the content manager module that accepts content data from applications, manages and signs each piece of content with the key the user holds. If a request matches local content, the request dispatcher will pass on this request to the content manager, who will subsequently invoke read and error callbacks provided by the application. In the case of no local match for the request, this request will be handed on to the retrieve logic module to issue content request message to the network via layer-3 protocol. Retrieved data will be verified and buffered by the protocol, and then submitted to applications via callbacks. If a complete piece of content is buffered by the module, it will be registered to the content manager if it is permitted by the user.

Another essential interface which is open to the layer-3 protocol is the receive callback function provided by the packet dispatcher module. The packet dispatcher distinguishes incoming packets by their types and then passes on them to different logic modules. Incoming content transmission messages will be dispatched to the retrieve logic that might respond with transfer control messages when necessary. On the other hand, incoming content request and transfer control messages will be handled by the transmission logic module. The transmission logic gets content data from the content manager and schedules content transmission whenever it is possible. The module also takes care of transfer control messages and adjusts its sending rates accordingly.

4.2 Evaluation

The transfer control mechanism of the protocol is evaluated using the implementation described above by running bulk transfer applications on top of both COL4 and TCP (with the existing protocol stack in ns-3). As illustrated in Figure 7, a pair of bulk transfer streams competes on a bottleneck link with 100Mbps bandwidth between routers R1 and R2. The experiment is carried out by changing the combination of the protocol the pair of bulk transfers used on the host pair SND1/RCV1 and SND2/RCV2.

Table 1 shows the evaluation experiment result with all three different protocol combinations. In the scenario where a COL4 stream competes bandwidth with a TCP one, the throughput of TCP stream regrades by about 1.4%, and that of COL4 is about 79.8%, compared to the scenario with two TCP streams. When
Table 1: Evaluation Experiment Result

<table>
<thead>
<tr>
<th>Protocols</th>
<th>Throughput 1</th>
<th>Throughput 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP and TCP</td>
<td>4.492 MB/s</td>
<td>4.491 MB/s</td>
</tr>
<tr>
<td>TCP and COL4</td>
<td>4.429 MB/s</td>
<td>3.585 MB/s</td>
</tr>
<tr>
<td>COL4 and COL4</td>
<td>5.076 MB/s</td>
<td>4.739 MB/s</td>
</tr>
</tbody>
</table>

Table 2: Evaluation Experiment Result

<table>
<thead>
<tr>
<th>Protocols</th>
<th>Throughput 1</th>
<th>Throughput 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP and TCP</td>
<td>4.492 MB/s</td>
<td>4.491 MB/s</td>
</tr>
<tr>
<td>TCP and COL4</td>
<td>4.200 MB/s</td>
<td>5.076 MB/s</td>
</tr>
<tr>
<td>COL4 and COL4</td>
<td>5.293 MB/s</td>
<td>5.238 MB/s</td>
</tr>
</tbody>
</table>

there are two COL4 streams, both transmissions achieve higher throughput compared to TCP ones. The average of the throughput increase is about 9.3% compared to TCP, and the difference between them is about 7.1%. The result shows that the transfer control mechanism in COL4 provides satisfactory reliability assurance and congestion avoidance. COL4 achieves great fairness between streams of the same protocol and between streams of COL4 and TCP (and intuitively, TCP-friendly ones).

As explained in Section 3, explicit congestion indication provided routers is usable for COL4. We install such congestion level indication module on router R1 to denote the congestion level of its outgoing queues on the COL4 message headers and conduct the same experiment again. The result is shown in Table 2.

The throughput of TCP stream, when competed by a COL4 stream with congestion indication, is reduced by about 6.5%. However, at the same time the COL4 stream performs a throughput with about 22.4% increase. Thus, although the throughput of TCP reduced by a small (but acceptable) amount, the overall link utilization is improved by about 7.9%. The reason for this is the COL4 stream with congestion indication effectively occupies the idle link. Similarly, the congestion indication also improves the performance and fairness between multiple COL4 streams. Compared with TCP streams, the average throughput increase of two COL4 streams with congestion indication is about 17.2%, and the difference of throughput between two streams is reduced to about 1.1%.

Concisely, the evaluation experiment result shows that COL4 is compatible with the current Internet infrastructure and the existing transport protocols, and its congestion indication feature could further improve the link utilization in the networks and achieve efficient content transmission for endpoints.

5. RELATED WORK

Even though content-oriented network architecture is an active research topic in recent years, transport protocols for them have received insufficient recognition. However, we are aware of several existing research work mentioning their solutions to transport content over current Internet infrastructure.

Data-Oriented (and Beyond) Network Architecture (DONA) [11] is implemented as an overlay between the transport and network layer, and the issues of content transmission is not handled in this work although evidences show that they use TCP as the transport protocol. We have shown in Section 2 that using the existing transport protocol constrain various content-oriented strategies from being applied.

Networking Named Content (NNC) [9] aims to provide a new protocol stack substituting current TCP/IP one, and they provide an implementation with XML-encoded datagrams encapsulated in UDP to forward their packet over existing networks. Their technique requires all network devices processing XML-encoded information in the packets, and therefore is neither feasible nor practical for those devices working at high line speed. Moreover, their technique is closely bound to their content distribution strategy, such as requiring content store in all the nodes.

Some people argue that HTTP is already a content-oriented protocol [13] and should be the central part of the future Internet. However, it is debatable whether HTTP is content-oriented as network devices are not able to be involved to assist content retrieval in the protocol.

A content-aware publish-subscribe protocol, which is called CAT, has also been proposed in [14]. The CAT protocol follows a proxy-based approach to connect a network using that protocol and legacy Internet as well as other content networks. TCP and HTTP are adopted as the carriers in this protocol. Besides the problems of TCP that we have already discussed, we have also shown that our protocol could support a variety of content-oriented network architectures without any translation. In this sense, our approach is a more general solution to the same problem discussed in CAT.

6. CONCLUSION

Researches on content-oriented network architecture are often in favor of clean-slate designs, which consequently result in difficulty in realizing the architecture with today’s Internet infrastructure. In this paper, we argue that it is the existing transport protocols that constrain the development of content-oriented networks with current practice. Accordingly, we propose the content-oriented transport protocol (COL4) to support most features of content-oriented network architectures over current Internet infrastructures. The protocol is designed to provide a “narrow-waist”, namely,
fundamental common transport mechanism for various kinds of content-oriented network architectures. In this paper, we have presented the design decision, protocol specification and an implementation in the network simulator of the protocol. We have shown that the protocol is compatible with the current Internet Protocol (IP) and can be incrementally implemented and deployed. In-network processing of content-oriented strategies could be benefited by the genuine connection-less feature of the protocol. Moreover, congestion control mechanism that achieves fairness with the existing transport protocols is included in the protocol so that content-oriented network architectures could focus on the strategy design.

We plan to extend our work in several directions. First, only preliminary analysis is done on the security mechanism for this protocol in current study. We expect an extensive study on it in the immediate future. Second, we will try to apply more content-oriented strategies on our proposed protocol, evaluate them and develop new strategies according to our findings. Finally, we will implement the protocol in open-source operating systems and provides applications (and patches for existing application) operating over the protocol implementation.

7. REFERENCES