ICNRG
Internet-Draft
Intended status: Experimental
Expires: August 3, 2020

G. White
CableLabs
S. Shannigrahi
Tennessee Tech University
C. Fan
Colorado State University
January 31, 2020

Internet Protocol Tunneling over Content Centric Mobile Networks
draft-irtf-icnrg-ipoc-01

Abstract

This document describes a protocol that enables tunneling of Internet Protocol traffic over a Content Centric Network (CCNx) or a Named Data Network (NDN). The target use case for such a protocol is to provide an IP mobility plane for mobile networks that might otherwise use IP-over-IP tunneling, such as the GPRS Tunneling Protocol (GTP) used by the Evolved Packet Core in LTE networks (LTE-EPC). By leveraging the elegant, built-in support for mobility provided by CCNx or NDN, this protocol achieves performance on par with LTE-EPC, equivalent efficiency, and substantially lower implementation and protocol complexity [Shannigrahi]. Furthermore, the use of CCNx/NDN for this purpose paves the way for the deployment of ICN native applications on the mobile network.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 3, 2020.

Table of Contents

1. Introduction

Content Centric Networking (such as CCNx or NDN, though CCNx is used

for the rest of the document) provides some key advantages over IP networking that make it attractive as a replacement for IP for wireless networking. In particular, by employing stateful forwarding, CCNx elegantly supports information retrieval by mobile client devices without the need for tunneling or a location

registration protocol. Furthermore, CCNx supports a client device utilizing multiple network attachments (e.g. multiple radio links) simultaneously in order to provide greater reliability or greater performance. Finally, CCNx is optimized for content retrieval, where content can be easily retrieved from an on-path cache.

From an incremental deployment perspective, it may be attractive to consider supporting CCNx as an overlay, i.e. tunneled over an IP-based mobile core network. But doing so diminishes the value that the CCNx protocol could provide, for example by limiting the ability to utilize on-path caching, native mobility and multiple network attachments. Ultimately, a more powerful approach, one that retains these benefits, is to utilize CCNx as a replacement for IP and IP-over-IP tunneling as the mobility plane for the mobile network.

A significant hurdle that stands in the way of deploying a CCNx-only wireless network is that all of the applications in use today (both client and server) are built to use IP.

This hurdle could be addressed by requiring that all applications be rewritten to use CCNx natively, however, this is a tall order in a world with millions of smartphone apps. Another approach could be to deploy a hybrid network in which the routers support forwarding both IP and CCNx. However, this adds cost and complexity to the network, both in the equipment and in operations.

The protocol described in this document provides a way to eliminate this hurdle, by establishing an IP over CCNx tunneling protocol that is transparent to the IP applications on either end. In a sense, this protocol replaces the IP-over-GTP tunnels or IP-over-GRE tunnels that would exist in a traditional IP-based wireless network such as LTE or Community WiFi, but by using a networking plane (CCNx) that natively supports mobility, application developers have the option to update their applications to run directly over CCNx, gaining all of the advantages that come with this new protocol.

IPoC supports IP mobility within a domain in a manner similar to that supported by LTE-EPC, i.e. the mobile node utilizes an IP address associated with the mobile network to which it is connected, and a stationary gateway device (P-GW in the case of LTE-EPC, IPoC Gateway in the case of IPoC) takes care of forwarding IP packets to the mobile node via the mobile network. [Shannigrahi] compares IPoC to GTP from the perspective of complexity and performance. Other mobility solutions, such as MIPv6 [RFC3775] exist that aim for a broader definition of IP mobility, and support efficient routing even when the mobile device retains an IP address that is not associated with the network to which it is connected. However, all these solutions still inherit the shortcomings of IP networking for

mobility - for example, handover latency and packet loss are known problems with MIPv6 [RFC5268].

This protocol specification does not currently address support for IP multicast connectivity. Support can be achieved via unicast forwarding of IP multicast packets to group members. Other approaches that take CCNx features (such as multicast forwarding strategy and caching) into account could help improve efficiency for IP multicast connectivity.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. CCNx Overview

In the CCNx protocol, communication is achieved by an application sending an Interest packet that identifies, by name, a piece of content that it wishes to receive. The network routes Interest messages toward a producer of content corresponding to the name in the Interest, leaving a "breadcrumb" trail of state in the routers along that path. Once the Interest arrives at a node where the named piece of content is present, that node returns a Content Object message containing the named piece of content. The Content Object follows (and consumes) the breadcrumb trail back to the originating application. This process is commonly referred to as stateful forwarding. An application that only sends Interest messages is

referred to as a consumer, whereas an application that only sends Content Object messages (in response to Interests) is referred to as a producer.

Producers need to advertise the name prefixes for the content that they can provide, and this information needs to propagate to the routers of the network, much in the same way that IP prefixes need to propagate to routers in an IP network. However, consumers don't need to advertise their presence or location at all, they can simply send Interest messages from wherever they are in the network, and the resulting Content Objects will make it back to them via the stateful forwarding process. Furthermore, a consumer that is mobile can redirect data in flight to the its new location by resending Interest messages for those in-flight content objects using its new network attachment point. As a result, mobile consumer applications (which would be the majority of mobile applications) are handled very elegantly by the CCNx protocol.

In addition, if a mobile device has multiple network attachment points, e.g. both a WiFi and a 5G/LTE connection, it can choose to send Interests via both of those network paths. This capability can be used to enable higher capacity (by load balancing the Interests in an attempt to fully utilize multiple links simultaneously), higher reliability (by sending each Interest on multiple links), or seamless handover (by switching to a new link for all future Interest messages, while still waiting to receive Content Objects on an older link).

4. IPoC Overview

While consumer mobility and multipath connectivity is elegantly handled by the CCNx protocol, producer mobility (where a mobile device makes its resident content available to outside devices), is currently not. As a result, the IPoC protocol relies solely on consumer behavior on the client device.

This protocol defines two entities: an IPoC Client and an IPoC Gateway. The IPoC Client (henceforth referred to as the Client) would exist on the mobile device, and as mentioned above, only sends Interest messages. The IPoC Gateway (henceforth referred to as the

Gateway) exists at a fixed location in the network, and publishes a prefix that can be routed to via the CCNx network. In general, a network may have many Clients, and possibly several Gateways.

The switches and routers that exist in the path between the Client(s) and the Gateway(s) are assumed to provide CCNx forwarding, and are not required to support IP forwarding.

From the perspective of the IP applications running on the mobile device, the Client implementation functions as a tunnel endpoint, much in the same way that a VPN application does. All IP packets generated by applications on the mobile device are forwarded via this tunnel endpoint, which encapsulates them in CCNx Interest messages, and then sends them into the CCNx network. Similarly, the Gateway implementation also acts as a tunnel endpoint, in this case on an IP routing node. It receives Interest messages, unpacks the IP packets inside, and forwards them into an IP network. IP return traffic arriving at the Gateway is encapsulated into CCNx Content Object messages, and then launched into the CCNx network to follow the stateful forwarding path left by the associated Interest message.

4.1. Use of Interest Payloads

As described above, IPoC capitalizes on the consumer mobility features of CCNx, and as a result uses the optional interest payload mechanism described in the "Consumer Behavior" section of [RFC8569].

This behavior preserves the basic hop-by-hop flow balancing principle of ICN, in that intermediate routers can control traffic flow by delaying Interest messages as appropriate. Additionally, the interest payload allows transport of information in Interests outside of the name field, which can significantly reduce router complexity (memory and memory bandwidth), as the name field is stored in the router's Pending Interest Table.

5. Client Interest Table and Interest Deficit Report

In this communication model, the Client is able to send "upstream" packets at any time, by sending Interest messages. The Gateway on the other hand, can only send "downstream" packets when it has a pending Interest (i.e. it has received an Interest message and has not yet responded with an associated Content Object). As a result,

the Client and Gateway work together to ensure that the Gateway is receiving Interests sufficiently to support the downstream communication.

For each Client, the Gateway MUST maintain a FIFO queue of names for which it has received Interests from the Client. This queue is referred to as the Client Interest Table (CIT). As this is a FIFO queue, the order in which Interest names are received is the order in which the associated Content Object responses will be sent.

The typical behavior of a Client (described in more detail below) is to send an Interest message for every Content Object it receives, thus maintaining a constant number of CCNx packets "in flight". The Interest Deficit Report (IDR) is a message element sent in a Content Object from the Gateway to the Client in order to adjust the number of packets in flight and thus maintain an appropriate CIT size. The IDR can take the value +1, to request an increase (by one) of the in-flight count; 0 to indicate no change to the in-flight count; or -1 to request a decrease (by one) of the in-flight count. The IDR can be included in a Content Object that carries a packet payload, or in a Content Object that is otherwise empty.

The IDR is an unacknowledged message element, and as such is an inherently unreliable communication. Since the IDR values are small, the impact of a Content Object loss is minimal.

The Client MUST maintain an Interest Deficit Count (IDC) which it uses to maintain the in-flight count in response to sent Interests and received Content Objects. The Client MUST decrement by one the IDC upon transmission of a new Interest message. The Client MUST update the IDC by adding IDR+1 to its value upon receipt of a new Content Object.

The Gateway SHOULD NOT discard Interest names from the CIT, and thus SHOULD always respond to a received Interest with a Content Object in order to clear the associated PIT state in the intermediate routers. If a new Interest arrives and the CIT is full, the gateway MUST consume the name at the head of the CIT by sending an empty content object. In this case, the IDR value of the empty Content Object SHOULD be set to -1.

6. Handling PIT Entry Lifetimes

Intermediate routers between the Client and the gateway, as well as CCNx forwarder implementations within the two IPoC endpoints will store PIT entries for the Client's Interests for a finite lifetime, and will age-out (purge) Interests that exceed that lifetime. Since the CIT at the gateway stores Interest names for a time in anticipation of downstream packets, it would be possible, when there is a gap in the flow of downstream packets, that the name at the head of the CIT queue is associated with entries that have been aged-out of the PIT in one or more of the intermediate forwarders. If the gateway were to use this aged-out name in an attempt to deliver a downstream packet, the packet transmission would fail when the Content Object arrived at the PIT that no longer held an entry for this name.

To avoid this situation, the Gateway MUST record the arrival time of each CIT entry, and compare it against a CIT lifetime value. When the CIT entry at the head of the CIT "expires", the gateway MUST send a Content Object using that CIT entry, thereby cleaning up the PIT state in the intervening forwarders, and potentially triggering a new Interest to be sent by the Client (as discussed further below).

7. Managing the CIT, PIT lifetimes and the in-flight message count

At any instant in time, a certain number of Interest names can be considered "in-flight" from the Client's perspective (these in-flight Interests correspond to the entries in the Client's PIT). Some fraction of the in-flight Interest names will correspond to Interest messages (possibly containing IP packets) that are in transit to the gateway, some fraction will correspond to Content Object messages (also possibly containing IP packets) that are in transit to the Client, and the remainder correspond to the entries in the gateway's CIT or to messages that were lost in transit. The gateway controls the number of these in-flight messages via the IDR, which can either trigger or suppress the Client sending Interests.

Since the gateway cannot send a downstream packet to the Client unless it has a CIT entry, it would ideally like to ensure that it always has at least one CIT entry every time a downstream packet

arrives. However, due to the round trip time between the gateway and

the Client, and the fluctuation of downstream and upstream packet arrival rates, the number of in-transit messages (Interests or Content Objects) will fluctuate. If the only goal was that the CIT never becomes empty, the gateway could simply use the IDR to build a very high in-flight message count. This would ensure that the CIT never drains completely, even in the case where the upstream path and the downstream path are both saturated with in-transit messages. The problem with this approach is that when the connection becomes idle, ALL of the in-flight messages would then exist in the CIT, which could be a large memory burden on the gateway and on the PIT in each intervening router. Furthermore, since each of these CIT entries has a certain lifetime, driven by the PIT lifetime, they will shortly expire, triggering the gateway to transmit Content Objects that heavily utilize the downstream and upstream links for approximately one RTT. This pattern of unnecessary network traffic would then periodically repeat at a period equal to the CIT lifetime.

So, it is important that the gateway adjust the in-flight message count continuously, to minimize the times that the CIT is starved or flooded.

The gateway MUST establish a target minimum value for the number of CIT entries. This value "n" provides a bound on the number of downstream packets that can be sent in the first IPoC RTT (between gateway and client) after an idle period, and also establishes the quiescent IPoC message refresh rate during idle periods (this rate r = n/L, where L is the CIT lifetime). Selecting a low value of n minimizes the quiescent load on the network, but has the downside of reducing the size of packet burst that the IPoC connection can handle with low latency.

Whenever the gateway sends a Content Object and there are fewer than n CIT entries, it MUST include an IDR in the CO, with the value 1, triggering the Client to send two Interest messages in response to the CO.

The gateway also MUST establish a maximum CIT size "N". Whenever the gateway receives a new Interest while the CIT contains N entries, it MUST make room for the new CIT entry by using the head of line CIT entry to send an empty Content Object containing an IDR with the value -1, triggering the Client to suppress sending an Interest in response.

Further, whenever the CIT entry at the head of line expires (reaches its CIT lifetime), the Gateway MUST consume that CIT entry by sending an empty Content Object. The expiration of a CIT entry is a good indication that the CIT contains more entries than are needed to

support the current data rate. In this situation, the Gateway SHOULD use the IDR to reduce the in-flight count. One mechanism for doing this is described here:

If the number of CIT entries is less than n, the empty Content Object sent to consume the expiring CIT entry will contain an IDR with the value 1. If the number of CIT entries is greater than n, the CO will contain an IDR with value -1, and if it is equal to n, the value 0. The result of this process is that during idle periods, the CIT will drain down to the point of having n entries, and will refresh those entries as they expire.

8. Establishing Communication

Communication is established by the Client sending an Interest to a Gateway, where the name in the Interest message includes a Gateway prefix followed by /init/<random_string>. For example, if the established Gateway prefix is ccnx:/ipoc, the name might be ccnx:/ipoc/init/2Fhwte2452g5shH4. The Gateway has a process that will respond to the ccnx:/ipoc/init prefix by sending IP configuration information, similar to the information contained in a DHCP Offer, including an assigned IP address.

Upon configuring itself using the information in the init response, the Client can begin IP communication. The naming convention for subsequent Interest messages is described in the next section.

9. IPoC Naming Conventions

The IPoC protocol doesn't assign any relationship between the Interest / Content Object names and the contents of the encapsulated IP packets. Rather, the name only identifies the Client instance of the IPoC application, and provides a sequence number that disambiguates Interests and Content Objects and provides for in-order delivery of IP packets.

The Client and Gateway can use one of the following data naming conventions, the appropriate naming convention is chosen by the Gateway via configuration, and is communicated to the Client during the Establishing Communication protocol.

ccnx:/ipoc/<hex_ipaddr>/<b64_seq>

ccnx:/ipoc/<zone_id>/<hex_ipaddr>/<b64_seq>

The various components of an IPoC name are described in more detail

below:

- ccnx:/ipoc - The name prefix used in all IPoC messages
- zone_id - An optional zone identifier to allow for zone-based IP address re-use.
- hex_ipaddr - For IPv4 addresses, this field comprises 4 separate name segments, each representing a single octet of an IPv4 address encoded as a hexadecimal string. For example, a message from a Client with IPv4 address 192.0.2.100 would use: "c0/00/02/64" for this name component. For IPv6 addresses, the textual convention defined in Section 2.2 paragraph 1 of [RFC4291] is used, with each colon replaced by a CCNx name segment delimiter. For example a Client with the IPv6 address: 2001:DB8::fe21:67cf would use "2001/DB8/0/0/0/0/fe21/67cf" for this name component.
- b64_seq - This a base64-encoded value representing the Upstream Sequence Number for this upstream Interest message

An example Interest name is: ccnx:/ipoc/c0/00/02/64/AAAAGw==

10. Sequence Numbers

Upstream Sequence Numbers (USN) are monotonically increasing unsigned 32-bit integer values embedded in the Interest names to indicate the proper ordering for upstream data packets. Since Interest messages may arrive out-of-order due to the use of multiple network paths, the Gateway uses the USN to ensure that upstream IP packets are delivered in the proper order.

Content Objects that carry IP packet payloads include Downstream Sequence Numbers (DSN), which are monotonically increasing unsigned 32-bit integer values that indicate the proper ordering of downstream data packets. DSN are used by the Client to ensure that downstream IP packets are delivered in the proper order.

The USN and DSN are independent sequence numbers and thus have no relationship to one another.

11. Packet Sequencer

The Packet Sequencer (PS or Sequencer) is a FIFO queue that exists both at the Client and Gateway to ensure in-order delivery of IP packets contained in upstream Interests and downstream Content Objects. The order in which the packets are delivered is decided by the Packet Sequence Number (PSN) embedded in the Interest or Content Object names.

The client MUST implement a Packet Sequencer to ensure in-order delivery of IP packets. The gateway MUST implement a Packet Sequencer to ensure in-order delivery of IP packets.

11.1. Packet Sequencer Example Algorithm

The first PSN (FPSN) delivered to the Sequencer establishes a baseline to which all subsequent PSNs are evaluated based on an expected ascending incremental order. The Sequencer also notes the last PSN (LPSN) it forwarded, and for the first packet, FPSN is equal to LPSN. If an arriving packet has the expected sequence number (LPSN + 1), the sequencer does not queue the packet and simply forwards it. The Sequencer also tracks the highest sequence number that has arrived (MAXPSN).

Discontinuities in the sequence order result in a "gap" in the sequence. If the arriving packet has a sequence number LPSN + n, where n > 1, we declare this as a gap. For example, if the last forwarded PSN had a sequence number 6 (LPSN), and a new packet arrives with sequence number 10 (MAXPSN), a new gap is created which represents the sequence numbers 7, 8, and 9. A timer with a validity window is started providing a limited amount of time for the sequence numbers in the gap to arrive.

Each time a packet with a sequence number in the gap arrives, the Sequencer tries to do a partial release of the queue; this releases any consecutive packets between LPSN and MAXPSN. In our example, if sequence 8 arrives first, the Sequencer sees there are no consecutive packets to send and does nothing. If sequence 7 arrives after that, the Sequencer releases both 7 and 8 but waits for sequence 9. When sequence 9 arrives, it releases 9 and 10. If a packet does not arrive and the validity window expires, the Sequencer releases all

packets up to MAXPSN and reset the LPSN.

The sequencer removes data packets from the queue in sequence-order (lowest PSN first). If the queue exceeds capacity, the Sequencer discards the packet with the lowest PSN. Any IP packets in those Interests or content objects are discarded.

Ideally, the gap validity window should be set to the RTT between the Client and the Gateway. However, since packets can take multiple paths and the Sequencer may not know the RTT for each of these paths, it should dynamically adjust the validity window based on the inter-arrival time between consecutive packets.

12. Client Behavior

The three main functions of the Client are:

1. Send Interest messages containing upstream IP packets whenever they arrive

2. Send Interest messages to the gateway in order to keep the appropriate in-flight count

3. Receive downstream IP packet data in Content Object messages

Content Object messages containing downstream IP packet data are added to the Packet Sequencer and then forwarded to the IP stack on the device.

Once an IP address is acquired using the initialization process described above, the startup sequence for a particular Client looks like this:

- Initialize IDC to a startup value: INIT_IDC.

- Send Interest messages to the Gateway containing the initial upstream IP packets (e.g. TCP SYN packets or DNS queries), decrementing IDC for each Interest sent.

The client MUST decrement the IDC upon transmission of any Interest message, whether or not it contains an upstream packet.

Whenever the client receives a Content Object, it MUST increment the IDC by IDR+1 to ensure that the appropriate in-flight count is maintained.

The Client MUST maintain two internal timer intervals. A short timer (T0) is used to pace Interest messages when there are outstanding interests to be sent as per the Interest Deficit Counter. The long timer (T1) is used as a keep-alive when the Client has no outstanding Interests to be sent. Whenever the client sends an Interest message, it restarts the T0 and T1 timers. When the T0 timer expires, if the IDC is greater than zero, the Client MUST send an empty Interest message. When the T1 timer expires, the Client MUST send an empty Interest message (regardless of the IDC value).

13. Gateway Behavior

IPoC gateway behavior is slightly more complex since it must manage connections with multiple Clients simultaneously. The standard process for on-boarding a new Client looks something like this:

- An Interest is received with the /init/<random_string> name.
- The gateway establishes new CIT (and other Client-specific) structures for this Client and responds with a Content Object containing the IP parameters (yiaddr, giaddr, etc.) to configure the Client's IP stack.
- The gateway enters a normal processing loop in which it receives Interests from the Client and responds with Content Objects.

Interests received from the Client may contain IP packets that the gateway will add to its upstream Packet Sequencer using the PSN found in the Interest name. The Interest name will then be added the Client-specific CIT for later use in creating Content Objects. If the CIT is full, the gateway will immediately send an empty Content Object back to the Client, removing the first name from the CIT, and therefore making room for the new name to be added.

When downstream IP packets become available, the gateway will remove the first name from the CIT queue and use it to create a Content Object containing the IP packets. If the CIT is empty, IP packets are buffered by the gateway.

If IP packets are waiting in buffer when a new Interest (CIT entry) arrives, the gateway will immediately dequeue the waiting packets (up to a maximum CO size limit), form and transmit a Content Object using the newly arrived CIT name.

14. Security Considerations

This protocol is designed for use within a trusted domain (i.e. a mobile core network). This protocol definition does not address authentication between clients and gateway devices, nor does it address privacy of communications (beyond that already provided by the IP applications themselves). The CCNx protocol does provide for Interest message and Content Object message authentication (signing) [RFC8609], which can be utilized if desired.

15. IANA Considerations

This document has no actions for IANA.

16. References

16.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3775] Johnson, D., Perkins, C., and J. Arkko, "Mobility Support in IPv6", RFC 3775, DOI 10.17487/RFC3775, June 2004, <https://www.rfc-editor.org/info/rfc3775>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <https://www.rfc-editor.org/info/rfc4291>.

[RFC5268] Koodli, R., Ed., "Mobile IPv6 Fast Handovers", RFC 5268, DOI 10.17487/RFC5268, June 2008, <https://www.rfc-editor.org/info/rfc5268>.

[RFC8569] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Semantics", RFC 8569, DOI 10.17487/RFC8569, July 2019, <https://www.rfc-editor.org/info/rfc8569>.

[RFC8609] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Messages in TLV Format", RFC 8609, DOI 10.17487/RFC8609, July 2019, <https://www.rfc-editor.org/info/rfc8609>.

16.2. Informative References

[Shannigrahi] Shannigrahi, S., Fan, C., and G. White, "Bridging the ICN Deployment Gap with IPoC: An IP-over-ICN protocol for 5G Networks", SIGCOMM NEAT Workshop , August 2018, <https://dl.acm.org/citation.cfm?id=3229575>.

Authors' Addresses

Greg White
CableLabs
858 Coal Creek Circle
Louisville, CO 80027
US

Email: g.white@cablelabs.com

Susmit Shannigrahi
Tennessee Tech University
Computer Sc. Dept.
Cookeville, TN 38501

US

Email: sshannigrahi@tntech.edu

Chengyu Fan
Colorado State University
Computer Sc. Dept.
1100 Center Ave Mall
Ft. Collins, CO 80523
US

Email: chengyu.fan@colostate.edu