CCN: Estimating Optimum Interest Lifetime based on Network RTT

CSE 534 ADVANCED COMPUTER NETWORKS

Shankar Krishnamurthy, Sidharth Singh

Contents

1 Introduction

The usage pattern of internet has changed dramatically over the past twenty years but the core architecture has remained pretty much the same. The narrow waist of the Internet (IP) has served us very well while allowing innovations in layers above and below. It was the simplicity of the datagram model that led to such rapid expansion of Internet, but it is probably fair to say that this ubiquitousness has also changed the way people use Internet today.

Today the emphasis lies squarely on content and there has been a proliferation of that with mobile devices, tablets and social networking. Users don't exactly care, or are even aware of where they get data from so long as they get it in a timely manner. All of this points towards an impending paradigm shift somewhere down the line and the CCNX project is just one such manifestation that puts content in its rightful place in the modern day Internet.

2 Problem Statement

CCN relies on Interest packets to fetch content from a source. These Interest packets are stored inside a PIT Table on each router that an Interest visits along with the Interface on which it arrived. When content corresponding to the Interest packet arrives at the router, PIT lookup yields a list of Interface(s) on which to forward this content and the Interest entry is subsequently flushed from the PIT Table.

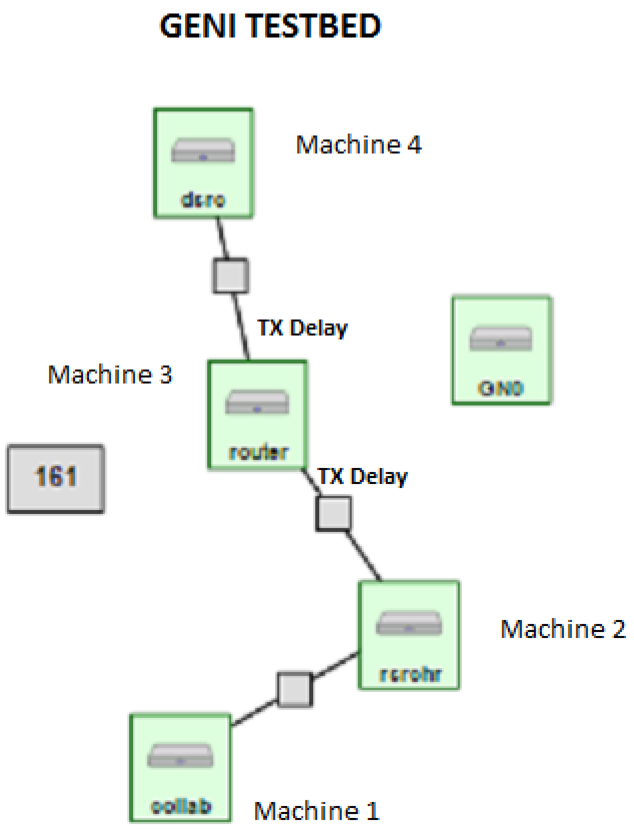
Every Interest packet has a lifetime value associated with so that the PIT Table doesn't get swamped with old and unserved Interest packets. The default value of the PIT timeout is set to 4s. In this experiment, we use a Linux utility (tc qdisc) to induce a latency of 100ms in the network. The CCN experiment is run using an open-source ‘ccnx-atmos’ utility that provides an interface created in python. The utility accepts a range of dates as an input and fetches weather reports as content for those dates.

The aim of the experiment is to find a suitable value of PIT timeout for a given network condition such that the combination of transaction latency and the number of Interest retries is minimum and the number of transactions that pass can be maximized. As a rule of thumb, CCN documents advocate that the Interest Lifetime can be set to a value that almost mirrors the Round Trip Time. In this experiment we try to evaluate this claim. Due to application level restrictions, we could set the value of Interest Lifetime to values that are powers of two. To account for that, we consider delta = Interest Lifetime – RTT as a metric for this experiment. Discovering an ideal value of delta is crucial because it keeps the PIT table free to take new Interest packets.

3 Requirements

1. GENI is a national scale sharable network testbed that allows experimentation with different architectures, fundamentally incompatible.
2. CCNX-0.8.2 is the open source CCNX release used for this project.
3. CCNX-Atmos is an open source CCNX Application used to send traffic in this project.
4. WIRESHARK is a packet analyser that we recompiled with a patch for CCN and used to analyse Interest and Content Objects in our experiment on a per packet basis.

3.1 Test Topology



4. Assumptions

1. Traffic pattern is uniform, ie unidirectional from Machine 1 (collab) to Machine 4 (dsrc).

2. Processing time for CCNX is negligible on all 4 Machines because there's no other traffic being pumped and hence CPU load is negligible.

5 Test Methodology

1. Make changes to INTEREST_LIFETIME_SEC on all the four machines and recompile CCNX code.
2. Check if 'ccnd' is running on all four machines, kill 'ccnd' process if already running.
3. Start CCND on all four machines: cd /tmp/ccnx-setup && sudo ./ccnx-setup router 4
4. Start Atmos on Machine 4 (dsrc). This script runs the CCN content repository on Machine 4 (dsrc).
5. Flush the Content Store on all 4 boxes using 'ccnrm' utility.
6. Induce a network delay on Machine 3 (router) using Linux utility (tc qdisc)
    a. sudo tc qdisc del dev <interface> root
    b. sudo tc qdisc add dev <interface> root netem delay 200ms
    c. In this experiment we induced a delay on both interfaces of Machine 3 (router) in the 'TX' direction.
7. Start a packet capture in .pcap format using Linux utility (tcpdump) on Machine 1 (collab)
8. Run the experiment from Machine 1 (collab): ./client.py <enter start date, enter end date>
9. Stop the packet capture once Content is received.
10. Remove the Content Repository using 'ccnr' utility to verify how many Content Objects were stored there.
11. Open the packet capture using Wireshark and verify the number of the Interest Packets sent, Content Objects received and Interest retries made. The number of Content Objects received as seen from the packet capture must be equal to the number of Content Objects removed on Machine 1 (collab).
12. Repeat every Test Case five times.

6 Test Cases

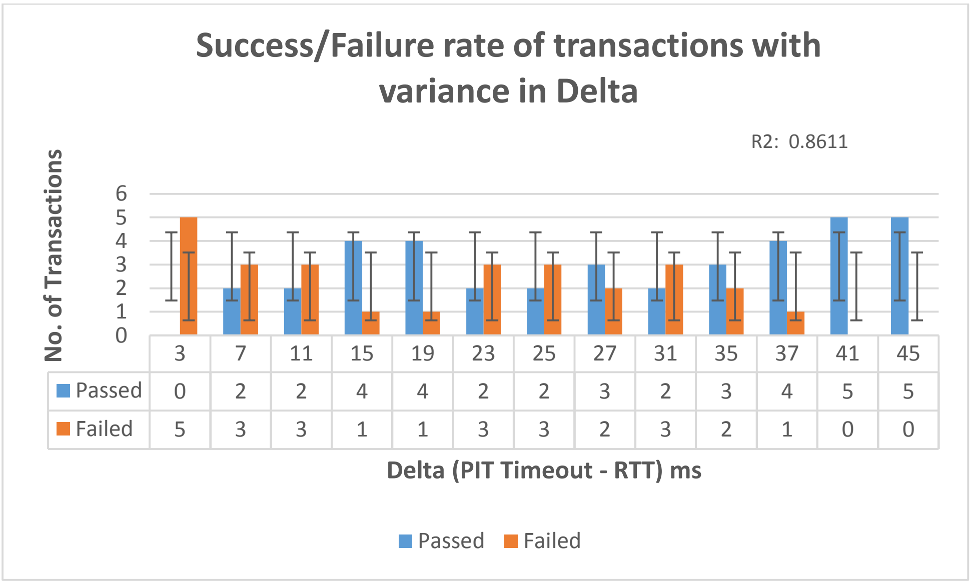
PIT Timeout (ms)	**Delay (ms)**	**Delta (Interest Lifetime – Delay) (ms)**	**PIT Size**	**Transaction Time (fully received content)**	**% Content Received**	**Trace Name (collab)**
125ms	122ms	3ms	10	(F), 12.012s	1/81	1_PIT_delta_3.pcap
125ms	122ms	3ms	10	(F), 12.012s	2/81	2_PIT_delta_3.pcap
125ms	122ms	3ms	10	(F), 12.012s	2/81	3_PIT_delta_3.pcap
125ms	122ms	3ms	10	(F), 12.012s	2/81	4_PIT_delta_3.pcap
125ms	122ms	3ms	10	(F), 12.012s	2/81	5_PIT_delta_3.pcap
125ms	118ms	7ms	10	(F), 9.66s	79/81	1_PIT_delta_7.pcap

125ms	118ms	7ms	10	(F), 20.322s	69/81	2_PIT_delta_7.pcap
125ms	118ms	7ms	10	(F), 1.34s	11/81	3_PIT_delta_7.pcap
125ms	118ms	7ms	10	(S), 17.900s	81/81	4_PIT_delta_7.pcap
125ms	118ms	7ms	10	(S), 17.899s	81/81	5_PIT_delta_7.pcap
125ms	114ms	11ms	10	(F), 18.749s	58/81	1_PIT_delta_11.pcap
125ms	114ms	11ms	10	(F), 2.604s	22/81	2_PIT_delta_11.pcap
125ms	114ms	11ms	10	(S), 17.584s	81/81	3_PIT_delta_11.pcap
125ms	114ms	11ms	10	(S), 17.580s	81/81	4_PIT_delta_11.pcap
125ms	114ms	11ms	10	(F), 12.127s	1/81	5_PIT_delta_11.pcap
125ms	110ms	15ms	10	(F), 14.067s	19/81	1_PIT_delta_15.pcap
125ms	110ms	15ms	10	(S), 17.259s	81/81	2_PIT_delta_15.pcap
125ms	110ms	15ms	10	(S), 17.259s	81/81	3_PIT_delta_15.pcap
125ms	110ms	15ms	10	(S), 17.269s	81/81	4_PIT_delta_15.pcap
125ms	110ms	15ms	10	(S), 17.255	81/81	5_PIT_delta_15.pcap
125ms	106ms	19ms	10	(F), 5.845s	53/81	1_PIT_delta_19.pcap
125ms	106ms	19ms	10	(S), 16.935s	81/81	2_PIT_delta_19.pcap
125ms	106ms	19ms	10	(S), 16.950s	81/81	3_PIT_delta_19.pcap
125ms	106ms	19ms	10	(S), 16.949s	81/81	4_PIT_delta_19.pcap
125ms	106ms	19ms	10	(S), 16.931s	81/81	5_PIT_delta_19.pcap
125ms	102ms	23ms	10	(S), 16.621s	81/81	1_PIT_delta_23.pcap
125ms	102ms	23ms	10	(F), 18.603s	63/81	2_PIT_delta_23.pcap
125ms	102ms	23ms	10	(F), 5.806s	17/81	3_PIT_delta_23.pcap
125ms	102ms	23ms	10	(S), 16.609s	81/81	4_PIT_delta_23.pcap
125ms	102ms	23ms	10	(F), 14.774s	27/81	5_PIT_delta_23.pcap
125ms	100ms	25ms	10	(S), 16.413s	81/81	1_PIT_delta_25.pcap
125ms	100ms	25ms	10	(S). 16.441s	81/81	2_PIT_delta_25.pcap
125ms	100ms	25ms	10	(F), 16.38s	43/81	3_PIT_delta_25.pcap

125ms	100ms	25ms	10	(F), 7.70s	37/81	4_PIT_delta_25.pcap
125ms	100ms	25ms	10	(S), 16.789s	81/81	5_PIT_delta_25.pcap
125ms	98ms	27ms	10	(F), 19.472s	74/81	1_PIT_delta_27.pcap
125ms	98ms	27ms	10	(F), 0.614s	6/81	2_PIT_delta_27.pcap
125ms	98ms	27ms	10	(S), 16.283s	81/81	3_PIT_delta_27.pcap
125ms	98ms	27ms	10	(S), 16.284s	81/81	4_PIT_delta_27.pcap
125ms	98ms	27ms	10	(S), 16.282s	81/81	5_PIT_delta_27.pcap
125ms	94ms	31ms	10	(F)4.61s	47/81	1_PIT_delta_31.pcap
125ms	94ms	31ms	10	(S),15.954s	81/81	2_PIT_delta_31.pcap
125ms	94ms	31ms	10	(F),14.172s	23/81	3_PIT_delta_31.pcap
125ms	94ms	31ms	10	(F),5.59s	57/81	4_PIT_delta_31.pcap
125ms	94ms	31ms	10	(S), 15.960s	81/81	5_PIT_delta_31.pcap
125ms	90ms	35ms	10	(S)15.585s,	81/81	1_PIT_delta_35.pcap
125ms	90ms	35ms	10	(S)15.584s,	81/81	2_PIT_delta_35.pcap
125ms	90ms	35ms	10	(S)15.587s,	81/81	3_PIT_delta_35.pcap
125ms	90ms	35ms	10	(F)19.26s	78/81	4_PIT_delta_35.pcap
125ms	90ms	35ms	10	(F).4.14s	2/81	5_PIT_delta_35.pcap
125ms	80ms	45ms	10	(S), 14.783s	81/81	1_PIT_delta_45.pcap
125ms	80ms	45ms	10	(S), 14.784s	81/81	2_PIT_delta_45.pcap
125ms	80ms	45ms	10	(S), 14.784s	81/81	3_PIT_delta_45.pcap
125ms	80ms	45ms	10	(S), 14.769s	81/81	4_PIT_delta_45.pcap
125ms	80ms	45ms	10	(S), 14.813s	81/81	5_PIT_delta_45.pcap
125ms	84ms	41ms	10	(S), 15.101s	81/81	1_PIT_delta_41.pcap
125ms	84ms	41ms	10	(S), 15.091s	81/81	2_PIT_delta_41.pcap
125ms	84ms	41ms	10	(S), 15.105s	81/81	3_PIT_delta_41.pcap
125ms	84ms	41ms	10	(S), 15.144s	81/81	4_PIT_delta_41.pcap
125ms	84ms	41ms	10	(S), 15.105s	81/81	5_PIT_delta_41.pcap

125ms	88ms	37ms	10	(S), 15.422s	81/81	1_PIT_delta_37.pcap
125ms	88ms	37ms	10	(S), 15.429s	81/81	2_PIT_delta_37.pcap
125ms	88ms	37ms	10	(S), 15.429s	81/81	3_PIT_delta_37.pcap
125ms	88ms	37ms	10	(S), 15.452s	81/81	4_PIT_delta_37.pcap
125ms	88ms	37ms	10	(F), 14.917s	33/81	5_PIT_delta_37.pcap

7 Analysis



	3	7	11	15	19	23	25	27	31	35	37	41	45
Passed	0	2	2	4	4	2	2	3	2	3	4	5	5
Failed	5	3	3	1	1	3	3	2	3	2	1	0	0

Figure 1: Five iterations were carried out for each value of Delta (PIT Timeout – RTT). The graph shows the number of passed/failed transactions for each value of delta. As the delta increases, we see an increase in the number of the passed transactions.

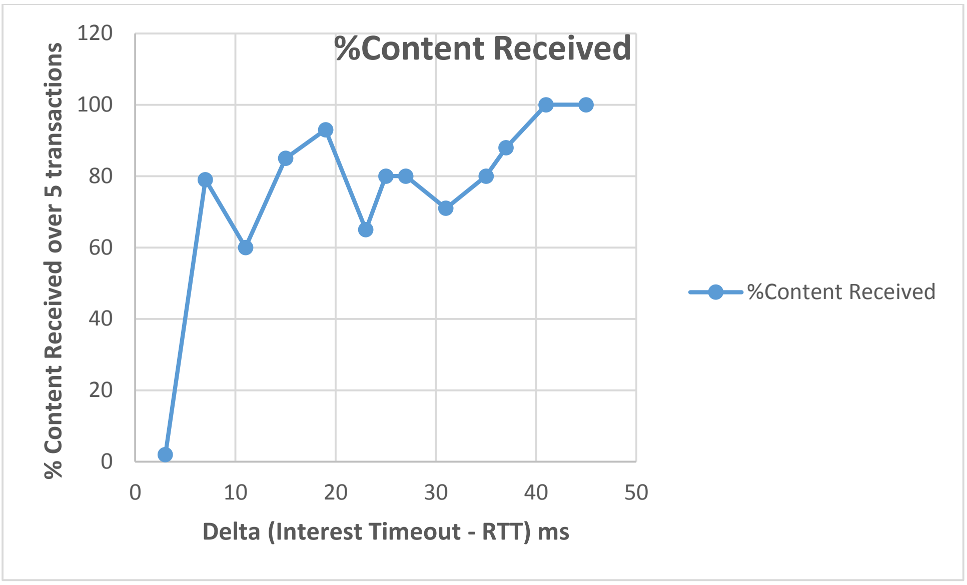


Figure 2: In CCNX-Atmos application used in this experiment, 81 chunks are needed to serve content for day's weather data (seen from packet capture). The above graph displays the percentage of chunk received across 5 transactions made by Machine 1 (collab) for different values of delta.

Note: Packet captures (.pcap) were taken for each iteration and they can be found at:

ssh syrotiuk@pc1.instageni.gpolab.bbn.com -p 36667:/opt/traces

7.1 ANOVA Analysis on Number of Passed Transactions

Figure 1 reports R^2 variance of 0.8611 for number of Passed transactions as the value of Delta (PIT Timeout – RTT) increases. With this value, we can conclude that roughly 86% of the time, the variance in number of passed transactions can be explained by variance in the value of Delta (PIT Timeout – RTT). The remaining 14% might be due other factors like processing time on each intermediate node.

8 Limitations

1. Serialization of Interest in the client application made changing the size of PIT Table irrelevant.

2. Number of Interest retries limited to 3 in the client application and changing the same caused errors in the client application.

3. The value of PIT Timeout could be expressed either in decimals or only in powers of base 2 (1/2,1/4,1/8).

9 Conclusion

Based on our experiment and the set of assumptions, we can conclude that the PIT Timeout value must be roughly 30% above the observed RTT in the network in order to completely eliminate Interest retries due to PIT Timeout. For this to work proactively, the application can add a probing mechanism that calculates RTT values during a transaction for every Content Object received and dynamically adjusts the PIT Timeout for that Interest based on RTT calculations.

10 Future work

1. Writing a client application for more control of experiments with respect to:
   a. Deserialization of Interest packets (a multithreaded client application can be used to achieve the same)
   b. Setting the upper ceiling on Interest retries to a higher value.
   c. More control of the application in general.

2. An full scale experiment that investigates the relationship between Interest Lifetime, PIT Size and the RTT in the network to arrive at an ideal value for Interest Lifetime and PIT Size in an adaptive manner depending upon the load in the network.

3. Automation of test methodology for greater repeatability and quicker turnaround time.

11 Changes made to CCNX Code:

In File: csrc/include/ccn/ccn.h
At Line: 65,66

```
#define CCN_INTEREST_LIFETIME_SEC <TIMEOUT VALUE>
#define CCN_INTEREST_LIFETIME_MICROSEC (CCN_INTEREST_LIFETIME_SEC *
1000000)
```

The client application code doesn’t specify Interest Lifetime value and hence in the underlying CCN code the default value of Interest Lifetime is called. So we manipulate this default Lifetime value from the code.

Code Flow:

The python client application calls an executable file named “client” in the ccnx-atmos directly. This is compiled and created using a C File called “client.c”.

In ccnx-atmos/client.c

<<Code from main() function>>

```
struct ccn_closure *incoming;
incoming = calloc(1, sizeof(*incoming));
incoming->p = incoming_interest;
res = ccn_express_interest(ccn, ccnb, incoming, NULL);
```

In ccnx/csrc/lib/ccn_client.c:

```
struct expressed_interest {
        int magic;                  /* for sanity checking */
        struct timeval lasttime;    /* time most recently expressed */
        struct ccn_closure *action; /* handler for incoming content */
        unsigned char *interest_msg; /* the interest message as sent */
        size_t size;                /* its size in bytes */
        int target;                 /* how many we want outstanding (0 or 1) */
        int outstanding;            /* number currently outstanding (0 or 1) */
        int lifetime_us;          /* interest lifetime in microseconds */
        struct ccn_charbuf *wanted_pub; /* waiting for this pub to arrive */
        struct expressed_interest *next; /* link to next in list */
};

int
ccn_express_interest(struct ccn *h,
    struct ccn_charbuf *namebuf,
    struct ccn_closure *action,
    struct ccn_charbuf *interest_template)
{
.
.
.
ccn_construct_interest(h, namebuf, interest_template, interest);
.
.
.
}

static void
ccn_construct_interest(struct ccn *h,
     struct ccn_charbuf *name_prefix,
     struct ccn_charbuf *interest_template,
     struct expressed_interest *dest)
{
        struct ccn_charbuf *c = h->interestbuf;
        size_t start;
        size_t size;
        int res;

        dest->lifetime_us = CCN_INTEREST_LIFETIME_MICROSEC; //This will change the
        change the value of PIT Timeout
        c->length = 0;
        ccnb_element_begin(c, CCN_DTAG_Interest);
        ccn_charbuf_append(c, name_prefix->buf, name_prefix->length);
    res = 0;
        if (interest_template != NULL)
        {
                //Not Applicable as client sends interest_template=NULL
        }
.
.
.
}
```