Support for Notifications in CCN
(“draft-ravi-icnrg-ccn-notification-01.txt”)
IETF/ICN-RG -100, Singapore

Ravi Ravindran (ravi.ravindran@huawei.com)
Asit Chakraborti (asit.chakraborti@huawei.com)
Syed Obaid Amin (obaid.amin@huawei.com)
Jiachen Chen (jiachen@winlab.rutgers.com)
Marc Mosko (marc.mosko@parc.com)
Ignacio Solis (ignacio.solis@parc.com)

Draft History

• First presented in IETF 95
• IETF96 we added more discussion around flow and congestion control
  – Also a related ICN Sigcomm paper last year
    • Motivated by how simple AIMD and flow balance doesn’t prevent congestion with heterogeneous receivers.
      – They show, eventually slower ones fall behind and stop benefitting from the network cache.
• Feedback from chairs to include more discussions on why current Interest/Data Abstraction fails
• This revision attempts to do that.
Table of Contents

1. Introduction .............................................. 2
2. Notification Requirements in CCN ..................... 3
3. Using Interest/Data Abstraction for PUSH .......... 4
4. Proposed Notification Primitive in CCN ............ 9
5. Notification Message Encoding ....................... 10
6. Notification Processing ............................... 12
7. Security Considerations ............................... 12
8. Annex ..................................................... 13
   8.1. Flow and Congestion Control .................... 13
      8.1.1. Issues with Basic Notifications ............. 13
      8.1.2. Flow and Congestion Control Mechanisms ... 14
         8.1.2.1. End-to-End Approaches ................. 14
         8.1.2.2. Hybrid Approaches ....................... 15
      8.1.3. Receiver Reliability ......................... 17
8.2. Routing Notifications ............................... 18
8.3. Notification reliability ........................... 18
8.4. Use Case Scenarios ................................ 19
      8.4.1. Realizing PUB/SUB System .................. 19
9. Informative References ............................... 20
Authors' Addresses ...................................... 22
Motivation for PUSH in CCN

- PUSH is a norm in IoT system, many messaging systems e.g. MQTT
- From Fig 1., significant (>80%) number of M2M devices have traffic that is upstream heavy.
- From Fig. 2, the distribution between the transmission vary from mins to days.
- Some of these updates are mission critical [2], with latency and reliability requirements for URLLC class of applications in 5G 1-10ms, and no message loss.
- This is just one data point, pub/sub is standard in the industry e.g. Social Networks
- Other ICN protocols such as MobilityFirst, NetInf support both PUSH and PULL.

CCN PUSH Requirements

• **Supporting PUSH Intent**
  – This should match application’s intent to PUSH content similar to the PULL primitive.
  – Feature to be supported considering efficiency and scalability

• **Support Multicast**
  – Support network service where an application PUSH can be multicasted to all intended receivers (just like Interest Multicast)

• **Security**
  – Should be able to deliver secure (authenticated and encrypted) NDO

• **Routing and Forwarding Support**
  – Push prefixes (Multicast or Unicast) should be treated differently from prefixes for regular Interests from routing and forwarding perspective, to support the PUSH intent.

• **Minimizing Processing**
  – PUSH flows shouldn’t be subjected to PIT/CS processing, considering latency and application intention.
Using Interest/Data Abstraction for Push

• Discusses how Interest/Data Abstractions can be used to achieve PUSH.

• Four Basic Approaches
  – Long Lived Interests
  – Polling
  – Overloading Interests
  – Interest Trigger

• We offer design choice discussions for each of these cases with its pros and cons.

• The discussion assume multiple providers within the same GROUP_PREFIX generating content randomly and receivers seeking to sync with the producers.
Using Interest/Data Abstraction—Long Lived Interest v1

- Assume consumers know all the names [No message loss]
  - Content name: /GroupID/ContentID
    - ContentID: sequential across all providers
  - Query: /GroupID/ContentID (full name)

- Problem with solution:
  - Inefficiency in multi-provider case
    - All Interests have to be send to all providers
    - Redundant Interest delivery
    - Some PITs will never be consumed (e.g., pkt1 from P2 to R1)

- Problem with assumption:
  - How can the providers synchronize?
    E.g., 2 providers send at the same time, who uses which name?
  - Even if the providers can synchronize, what’s the cost?
  - Providers have to address another sync problem
Using Interest/Data Abstraction– Polling v1

• This is to prevent the issues with the previous approach, the providers can publish content using timestamps.
• Assume the consumer only know the group name
  – **Content name**: /GroupID/<Timestamp>
    • No need for synchronization across providers
  – **Query name**: /GroupID/<earliest after XXX>
    • XXX: the latest version (timestamp) I have

• Problem with solution:
  – Need to have a synchronized time over the service providers and consumers
  – Ambiguous content when two providers publish using the same timestamps or when the clocks drift apart
  – **Message loss**:
    • P1 has notification t=1234, P2 has notification t=1327
    • Consumer query with <earliest after t=1200> (he can’t query 1201, since he is not sure if there is such a content)
    • P2’s version might arrive before P1’s version
    • Consumer will query <earliest after t=1327> and miss P1’s content
Using Interest/Data Abstraction—Long Lived Interest v2

• Assume consumers know all the names
  – Content name: /GroupID/ProviderID/ContentID
    • ContentID: sequential per provider
  – Query: /GroupID/ProviderID/ContentID (full name)

• Problem with assumption:
  – Consumers have to know all the potential providers
  – The solution becomes more “host-centric” than “information-centric”

• Avoids the packet losses from the previous case

• Problem with solution:
  – Increases the PIT state in the network
  – If the Group_ID is shared among multiple devices (laptop, smartphone etc.), the issues are similar to the long lived interest- v1 case.
Using Interest/Data Abstraction– Polling v2

- To reduce the PIT states in the network, we could process Interests in the Application Layer
  - Useful in applications like Gaming
- Assume consumer knows all the providers
  - Content name: /GroupID/ProviderID/ContentName/<timestamp>
    - ContentID: sequential per provider
  - Query: /GroupID/ProviderID/<updates after t>/nonce
  - Response: all the contents during the period (in a single response), or “no update” response
  - Aggregates the responses & the providers do not have to follow the sequential version
- Problem with solution:
  - Inefficiency with polling
  - More load on the providers
  - Caching not useful here
  - Consumers have to know all the potential providers
Using Interest/Data Abstraction—Polling with a Server

Using Server for Aggregating Provider Notifications:

- Offloads Provider level data aggregation to a server
- The providers would publish data into the server and the consumers would poll for the updates from the server (similar to Twitter and Facebook in IP network).
- Server will offer aggregated response.

Problems with the Solution:

- Single point of failure, just as in case of IP services today
- Server has to use one of the previous mechanisms to sync their current content state with providers.
- Caches are not useful here just in the previous case
- This approach boils down to a host-centric approach by tying down to a server
Using Interest/Data Abstraction– Interest Overloading

Approach
• Notification Payload can be inserted into the Interest itself
  – Interests takes the form /GROUP_ID/NONCE/<Payload>

Problems with this Solution
• Routing and forwarding has to differentiate between Regular Interests from Interests with Notifications
• Storing PIT state has to be avoided for efficiency
• Consumer oriented FIB entry should reach all the providers
• Payloads beyond a certain size has to be avoided considering engineering assumptions on Interest sizes.
Using Interest/Data Abstraction– Interest Trigger

• **Solutions**
  – Takes care of avoiding inserting Payload into the Interest and routing and forwarding complexities of the previous scheme
  – Send a trigger with the content name, and the content will then be pulled

• **Problems**
  – Atleast a RTT delay, affects mission critical applications
  – Triggers still have to reach all the receiving points, so still has the routing and forwarding challenges.
  – Trigger name space should be defined carefully.
Other updates

• The remaining part of the draft hasn’t been changed.
• We provide discussions on protocol semantics, router operation
• Flow congestion control discussions are also provided
• Use case on using this for pub/sub is also provided.
Next Steps

• Comments from the chairs and the group to further this draft are welcome.