Pending Interest Table Behavior

Marc Mosko*, Nacho Solis, J.J. Garcia-Luna-Aceves

Abstract
In CCN, an request packet called an Interest follows a forwarding table to reach a producer or cache with desired named content. As the Interest follows the forwarding table, it leaves state at each hop so the response packet, called a Content Object, can return to the requester. This paper describes the prior methods of Interest forwarding with so-called "aggregation" and points out several shortcomings of this method. We propose three alternatives and describe a preferred method of Interest "accumulation."

Keywords
Content Centric Networks – Named Data Networks

Palo Alto Research Center
***Corresponding author**: marc.mosko@parc.com

Contents

Introduction

In CCN, an request packet called an Interest follows a forwarding table to reach a producer or cache with desired named content. As the Interest follows the forwarding table, it leaves state at each hop so the response packet, called a Content Object, can return to the requester. This paper describes the prior methods of Interest forwarding with so-called "aggregation" and points out several shortcomings of this method. We propose three alternatives and describe a preferred method of Interest "accumulation."

In this work, we study the behavior of an intermediate node servicing multiple requests for the same request, often called "similar" interests. Two interests are similar if they could both be satisfied by the same Content Object. For some implementations, that is a difficult predicate to analyze due to restrictions in the Interest beyond the name, so in practice one uses a more restrictive condition that two Interests ask the same exact question: all fields that affect Content Object matching are identical. In this work, when we speak of a PIT entry, we mean the entry that represents such similar Interests. PIT entries for non-similar Interests are independent.

In prior methods, described in Sec. **??**, each PIT entry is active for a time based on the Interest Lifetime, a parameter set by the origin of the Interest and possibly modified in-flight at each hop. The value of the Interest Lifetime is critical to ARQ retransmissions, because if there is an existing PIT entry, it will suppress retransmissions along the same path. Therefore, one must either set the Interest Lifetime to a value around the round-trip time or explore alternate paths for retransmissions. Otherwise, if an Interest with a long lifetime is lost, it will suppress other Interests when no response is coming. For example, in the NDN Path Splicing scheme [**?**], a consumer host times out after a round trip time and will retransmit an Interest using different tags. Using different tags heuristically tries to send an Interest long a different path than a previous Interest.

In the new methods described here, the PIT entry exists at an intermediate system for as long as the intermediate system can keep it or until it is satisfied. The Interest Lifetime no longer needs to be set to a value around the round-trip time, rather it represents the period over which the end system would like a response. An end system must still use a lively ARQ scheme to retransmit an Interest if it does not receive a response, there is no mandated in-network reliable links. An end system's ARQ scheme may operate at whatever time constants the system finds useful, such as milliseconds based on estimated round trip times or polling every few seconds. In the new methods, the expiry time – an absolute time – is always set as the local time plus the Interest Lifetime.

The Send Everything method keeps a reverse path entry per predecessor with an expiry time of the maximum seen from that predecessor. An intermediate node forwards all Interests without suppression. This method is based on the idea that aggregating multiple interests within the same round-trip time and without any on-path caching is rare, therefore the extra upstream traffic is minimum compared the the benefits.

The Predecessor Lifetime method keeps an Interest Lifetime per predecessor and a separate aggregation lifetime. The idea is that a specific predecessor will only be aggregated once during the aggregation lifetime, and subsequent Interests that would have been aggregated will be forwarded under the assumption they are ARQ attempts. This allows suppression of initial Interests without a round-trip timeout at the requestor, and then each requester can perform ARQ at their own rate. The fastest ARQ requester actually suffices for all requesters because the response Content Object would be sent back along all paths.

In the Predecessor Subscription method, an Interest will remain in the PIT for as long as necessary or until the router

needs to recycle the memory. The Interest Lifetime is how long, overall, that the consumer is interested in the content and could be a long period, such as seconds. An intermediate node keeps PIT entries until they are satisfied or the PIT table is full. When a data packet arrives, it will follow all reverse paths that have not past their expiry time. When the PIT capacity is reached, an intermediate node re-cycles entries based on its own eviction algorithm, such as LRU (where the "used" in this case is the time since last matching request). The consumer must still use an ARQ scheme and retransmit the Interest at whatever appropriate interval it decides, there is no guarantee of reliable links or custodial hand-off of Interests.

1. The need for aggregation

Interest aggregation claims to save network resources – in particular upstream Interest traffic – by grouping similar requests in the Pending Interest Table. Even in the best case, Interest aggregation will only save traffic if (a) the two or more requests happen within the round-trip time of the first Interest and (b) there is no on-path caching. If a first Interest arrives and then is satisfied before a second similar Interest arrives, there is no aggregation and the second Interest is forwarded as normal, if there is no caching. If there is caching, the second Interest is served from cache regardless of the aggregation scheme. One should then ask how often does Interest aggregation happen and what is the actual savings.

In some ICN approaches, such as CCNx 0.x and NDN, various forms of discovery probe for content that may not exist. Further, then may do this with long-lived Interests, as a kind of short-term subscription to the next piece of content in a stream of content. For example, a publisher creates version 7 of an image. After downloading version 7, a consumer may ask for version 8 with increasing lifetimes – say up to a second or two – so that if version 8 appears it will be downloaded immediately. Or, a consumer in CCNx 0.x or NDN may ask for name with exclusions, such as asking for "any version greater than 7."

In another scenario, which applies to CCNx 1.0 too, a consumer may be downloading a chunked object and the window of Interests may get ahead of of what the producer has created, such as due to signing overhead. In these cases, however, the Interest Lifetime should be around the round-trip delay as scaled up by the actual content creation time. The Interests are not being used to discover content. The chunking protocol should be providing feedback about the final chunk number so the consumer does not go beyond the actual end.

2. Prior Aggregation Method

In Named Data Networking [**?**] and CCNx 0.x [**?**], a notional table called the Pending Interest Table (PIT) records the reverse path for each Interest for a given name. When a first Interest for a name arrives at a system, it will first check to see if it is available from a local cache, and if so will respond with the data object. Otherwise, it will lookup the name in the PIT. If no record is found, it will add a record of the name and the previous hop, then forward the Interest according to the Forwarding Information Base (FIB). If a second similar Interest arrives while the first is pending, the second Interest will be aggregated in the same PIT entry by adding its previous hop to the existing record.

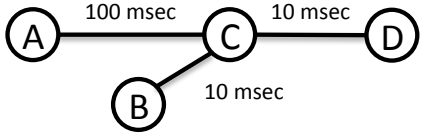

Figure 1. Example Topology

In practice, the NDN Interest handling rules are a bit more complex [**?**]. The PIT table key is the tuple of {name, selectors}, where the selectors include things like exclusions or other restrictions that affect data to interest matching. Each PIT entry has an Unsatisfy Timer and a Straggler Timer. The Unsatisfy Timer is set to the maximum of the existing Unsatisfy Timer or local time plus the Interest Lifetime (or 4 seconds if not included in the Interest). When it expires an Interest Unsatisfy Pipeline is executed. The Straggler Timer is updated to the local time plus 100 msec whenever the PIT entry is satisfied by returning data so it stays around longer than normal in an inactive state. The purpose of this is to retain the nonces associated with the PIT entries, as NDN uses those to detect loops. When the Straggler Timer expires, the PIT entry is deleted.

In NDN and CCNx 0.x, the effect of lost Interest packets is somewhat masked because each forwarder keeps its own round trip time estimate for each FIB entry and will timeout and retry an Interest on different next hops if it times out. This requires keeping a round trip time estimate per FIB entry and using a fast timer per PIT entry. If one is willing to pay the cost at each forwarder to keep a timer, complete Interest packet (for retry attempts) and retry transmissions, then some of the negative effects of Interest aggregation could be masked over, assuming there are multiple loop-free paths to try. This, however, appears to be a large cost for high-speed routers that might process 10's to 100's of millions of Interests per second.

Interest Aggregation has several shortcomings. Foremost is that packets with long Interest Lifetime will suppress retransmission requests from other nodes with short lifetimes. In Fig. **??**, for example, node A has a long latency to node C while node B has a short latency. Let us assume that each node knows its exact round-trip time and uses that as the Interest Lifetime. If a request for a given name from node A arrives at node C before the same request from node B, the request from node B will be aggregated at C. If the original request was lost going upstream to D, then the quickest ARQ could go is at the pace set by node A. In this example, even though node B re-transmits its request 40 msec later, that retransmission is

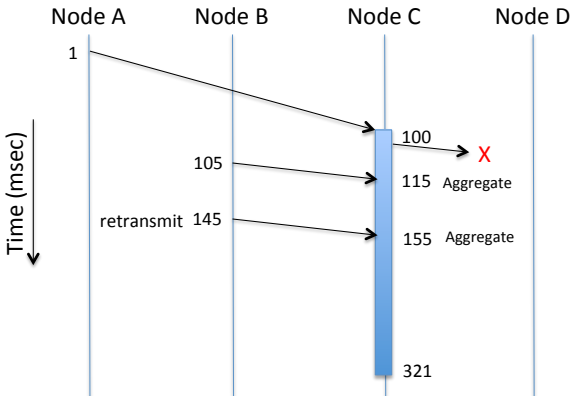

Figure 2. Prior Interest Lifetime Method

also aggregated.

The formal descriptions of the CCNx 0.8 and NDN protocols state that if a node – such as C – knows that a request arrived over a high-latency link it should appropriately decrement the Interest lifetime. This simplistic scheme is error prone. How does node C know that the A-C link is high-latency? Is it due to distance, in which case C could possibly measure it? Is it due to queuing at A, in which case A should have decremented it. It cold be that the A-C link is actually many links each with small delay, but they add up. If one has many links all averaging around the minimum granularity of the Interest Lifetime (e.g. 1 milli-second), then when does one decrement? Such a scheme could decrement the Interest Lifetime to 0 before it reaches the source. Also, node A is measuring the round-trip time at some time constant and likely averaging it via some means, like a moving average filter. Node C would measure the A-C delay through other means over other time constants and likely would never decrement the Interest Lifetime by the same amount node A used. Also, the A-C delay may be different than the C-A delay, in which case node C would need to know both those quantities and the expected response Content Object size to account for the serialization delay. Therefore, it is not simple for node C to know the link delays and subtract them from an Interest Lifetime. There are many corner cases.

Another issue with using the Interest Lifetime to represent the round-trip time is that the round-trip time is a statistical measure of the expected round-trip time, but it could have significant variation. How long should a specific PIT entry live? If the Interest Lifetime is set to the 95% value of round trip times – thus avoiding problems with Content not reaching the consumer because PIT entries expire, then the corresponding ARQ mechanism will operate in the bottom 5%, which is also not desirable.

Even if an intermediate node accounts for one-way link delays, as shown in Fig. **??**, Interest aggregation will still aggregate packets behind packet loss, causing all requesters to defer for at least one round trip time.

The prior method of Interest aggregation also uses a technique we call Delta-Transmit. When a node, such as C in

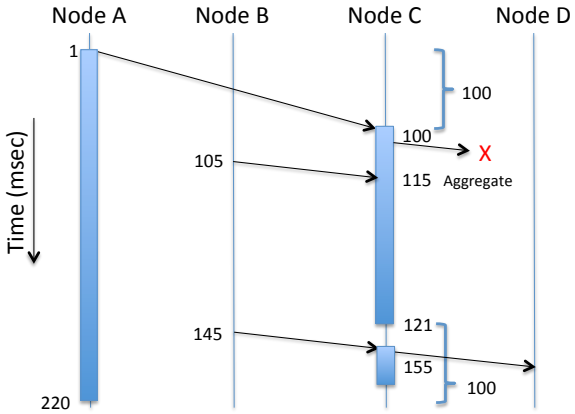

Figure 3. Aggregation with link delay adjustment

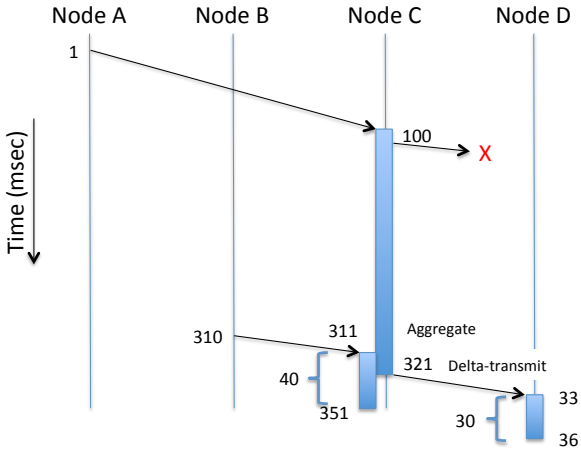

Figure 4. Delta-Transmit

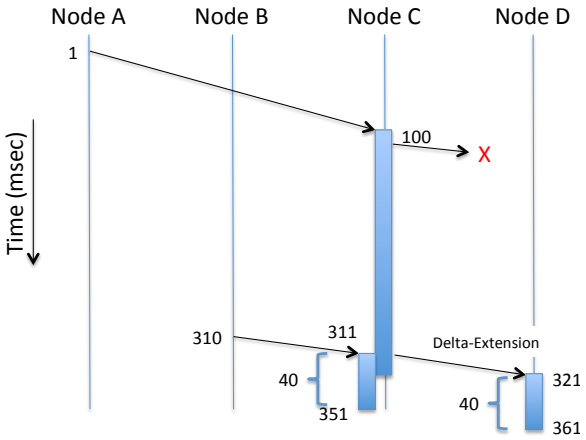

Figure 5. Delta-Extension

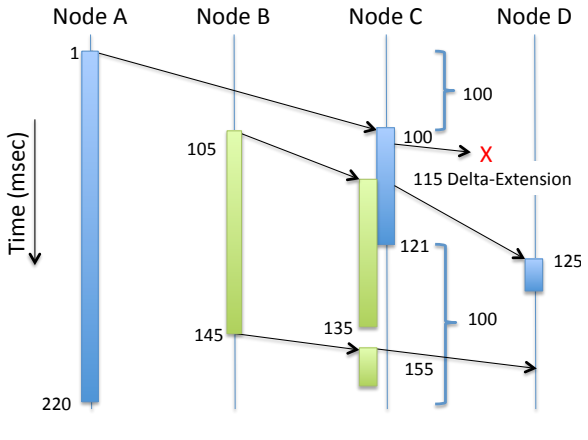

Figure 6. Delta methods With Delay Adjustment

Fig. **??** receives a second Interest whose lifetime would extend an existing Interest, it will aggregate that Interest and if the first Interest's lifetime expires before being satisfied, it will re-transmit the Interest with the delta lifetime. In the example in Fig. **??**, when the Interest from node B arrives, the one from node A still has 10 msec remaining. When that 10 msec expires, node C will re-transmit the Interest from node B, but with the 10 msec delta-time subtracted, resulting in a 30 msec Lifetime.

The method of Delta-Transmit thus requires that intermediate nodes, such as node C in the figure, cache the entire Interest message of any Interest that it aggregates which extends the lifetime of the prior Interest. It must also keep a timer for the prior Interest expiry so it can perform the Delta-Transmit, which is a much harder requirement than keeping just an Expiry time that could be lazily reclaimed.

A second variation of Interest lifetime extension is to use a method we call Delta-Extension. In Delta-Extension, shown in Fig. **??**, when an intermediate node receives an Interest that would extend the lifetime of a prior Interest that would otherwise be aggregated, the intermediate now immediately forwards the Interest with the delta lifetime. In the example in Fig. **??**, the intermediate node is not accounting for link delays, so the extension time is equal to the second Interest Lifetime. Note that in Delta-Extension, there is less savings of upstream traffic, because not all similar Interests are aggregated, only those that would not extend the lifetime. This means that Interest aggregation will mostly happen when an Interest with a longer lifetime is received first and during that round-trip time an Interest with a lifetime shorter than the original Interest minus its dwell time arrives. Delta-Extension does have the benefit that the intermediate node does not need to cache entire Interest packets for retransmission as in Delta-Transmission.

If Delta-Extension (or Delta-Transmission) is combined with link delay adjustment, there is still reduced savings on upstream traffic, as shown in Fig. **??**. In this example with perfect knowledge, all nodes would be contributing 20 msec of round trip time for the C-D link, therefore all arriving Interests at node C would be reduced to the same 20 msec lifetime. Because later packets arrive later each one would extend the PIT lifetime and result in either an immediate Delta-Extension or be aggregated behind the lost initial packet and suffer an extra round trip time due to the one loss.

The prior method of Interest aggregation, as described here, has the following undesirable features. A long-lifetime Interest can suppress ARQ retransmission requests from a node with shorter round-trip time if intermediate nodes do not account for link delays when forwarding Interests. If intermediate nodes account for link delays, they need to actually measuring delays in a timely manner, perform the math and packet re-write to decrement lifetimes per packet, and come up with heuristics to deal with link delays around the granularity of the Interest Lifetime field. If Delta-Transmission is used, then intermediate nodes need to cache some full Interest packets and keep actual timers – as opposed to lazy expiry times – for each PIT entry. If Delta-Extension is used, it reduces the potential benefit of Interest Aggregation because fewer Interests are suppressed.

3. Control Messages

The new methods of PIT behavior incorporate network control messages called NACKs. These packets are hop-by-hop control messages between peers to indicate an error condition. The trust context of these messages is only one-hop, so systems could use locally agreed keys or other methods.

If a system receives an Interest that it cannot process or that has an error condition that leads to the PIT entry being removed, that system should send a NACK control message. The NACK control message is a hop-by-hop message that indicates that a specific successor node will not answer an Interest from a specific predecessor. For example, in Fig. **??**, node C (successor of node A) might send a NACK to node A (predecessor of node C) to indicate that a specific Interest node A sent to node C will not be serviced. When a system sends a

NACK message, it should remove the predecessor from the list of predecessors for the PIT entry. If the predecessor sends another Interest for that name, that new Interest is considered a first interest and will create new state in the PIT entry, if it is accepted.

A system should send a NACK message whenever it will not save the predecessors PIT entry; that is whenever the successor will definitively not return data to the predecessor. For example, if node C does not have a route to the name requested by A, it would send a No Route NACK message back. If node C has reached a capacity limit and will not create a new PIT entry for A, it would send a No Resources NACK message back. If node C receives a NACK from node D and node C will not take any other corrective actions – i.e., node D is the only route or node C will not retry via a different path – then node C should send a NACK of similar type back to node A. Systems should not send NACKs for PIT timeout – those retransmissions are handled by the end systems.

A system should verify that it receives a NACK from an expected successor. For example, if node A sends an Interest to node C, which then forwards to node D, then node C should not accept a NACK from node B. This could be an off-path attack. Node C should verify that it actually sent the Interest referenced in the NACK to the node sending the NACK.

If a system will not retry an Interest along multiple paths if it receives a NACK, then it should send a No Data control message to its predecessor. For example, node A sends an Interest to node C, which then sends it to node D. Node D sends a No Route back to node C. Node C does have a route, but it will not return data, so rather than tell node A that it has no route, it will send a No Data NACK that indicates that node A should not expect any data for the given Interest from node C.

A system may send retransmissions along different paths than previous transmissions. In this case, the system should track which successors have outstanding Interests and process NACK messages per successor. Only if a system receives NACKs from all successors should a PIT entry be considered dead and a NACK send to that system's predecessors.

4. Send Everything Method

The Send Everything method forwards every Interest. It still records all Interests in the PIT, so a single Content Object could satisfy multiple Interests on the return path. This method is based on the idea that aggregating multiple interests within the same round-trip time and without any on-path caching is rare, therefore the extra upstream traffic is minimum compared the the benefits. The Send Everything method does not suffer the ARQ non-isolation problem, as each retransmitted may go at their own rate. It does not require caching Interests, as they are always forwarded immediately.

The Send Everything method uses the Interest Lifetime to determine when to lazily expire reverse paths, so a Content Object that arrives later will not be sent to a reverse path that

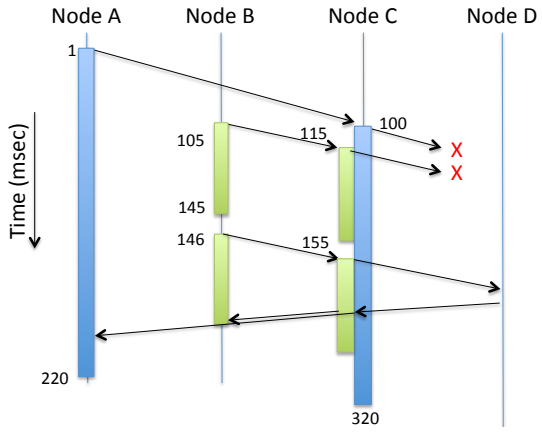

Figure 7. Send Everything Method

is no longer listening for it. It also allows lazy pruning the reverse path list.

A variation is the Send Everything method is to use a brief, node-dependent aggregation time. For example, if Interest A has a Lifetime of 220 msec and interest B has a lifetime of 40 msec, node C could apply a brief delay, such as a token bucket rate shaper, to avoid the case where many downstream nodes are requesting retransmission of the same thing within a short delay.

NACK messages have little affect on Send Everything. A NACK message will consume a PIT entry, which will clean up the PIT table, but otherwise the Send Everything method does not retry Interests so there is no other effect.

5. Predecessor Lifetime Method

The Predecessor Lifetime method keeps an Interest Lifetime per predecessor and a separate aggregation lifetime. The idea is that a specific predecessor will only be aggregated once during the aggregation lifetime, and subsequent Interests that would have been aggregated will be forwarded under the assumption they are ARQ attempts. The fastest ARQ requester actually suffices for all requesters because the response Content Object would be sent back along all paths.

If the first Interest for a name from a specific predecessor, such as node B in Fig. **??**, could be aggregated with a pending interest from a different predecessor, then that Interest is aggregated. The expiry time of the node B interest is set to its Interest Lifetime – that is about the round-trip time node B expects. The aggregation time of the predecessor B is set to the maximum of the aggregated PIT lifetimes at that moment, which could be substantially longer than the B Interest lifetime. The PIT entry for predecessor B will not be removed until it is satisfied or the aggregation lifetime expires. Predecessor B will only receive a satisfying Content Object if that object arrives during the active period of the reverse path entry (i.e. before the expiry time, not the aggregation time). Subsequent interests from the same predecessor, node B, that find that predecessor still in the PIT will be forwarded

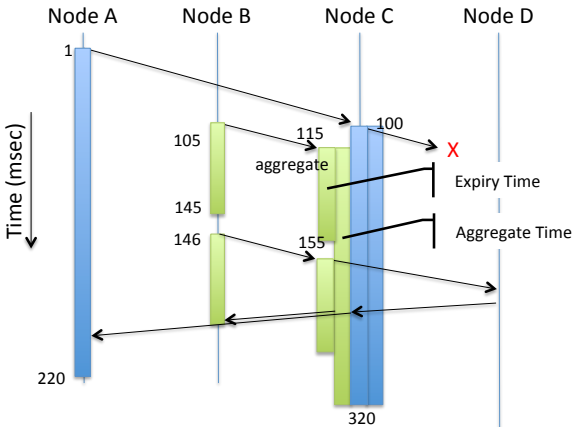

Figure 8. Predecessor Lifetime Method

immediately and their lifetime set before.

The Predecessor Lifetime method works well, even if nodes do not adjust Interest lifetimes for link delays. This means that the Interest Lifetime does not need to be very exact and substantial over-estimates to account for statistical variations in round trip delays do not harm ARQ mechanisms.

NACK messages will consume PIT entries for each affected predecessor. This should erase all PIT entries affected by that upstream node. This will reset the state of specific predecessors so the state machine will start again for that predecessor.

6. Predecessor Subscription Method

In the Predecessor Subscription method, the Interest Lifetime is not closely related to the round trip time. It is how long, overall, that the consumer is interested in the content and could be a long period, such as seconds. An intermediate node keeps PIT entries until they are satisfied or the PIT table is full. When the PIT capacity is reached, an intermediate node re-cycles entries based on its own eviction algorithm, such as LRU (where the "used" in this case is the time since last matching request).

On the first Interest from a predecessor that matches an existing PIT entry, an intermediate node may choose to aggregate them if they arrive within a short period of time. This choice and the time period are node-dependent and could be based on knowledge of upstream performance or could be a small implementation-dependent value. The second and later Interests from a predecessor already in the PIT table are forwarded immediately, as they are considered ARQ requests.

An intermediate node could apply a traffic shaper, such as a token bucket, to the "immediate" forwarding requests to avoid situations where many nodes attempt ARQ at approximately the same instance. This is a local decision and not related to the Interest Lifetime.

This method does not require link delay measurements or link delay adjustments. In fact, the Interest Lifetime could be part of signed information inside the Interest as it does not need to be adjusted per-hop. Because the Interest Lifetime is no longer associated with round-trip time, there is no longer the need to apply heuristics to measure the round trip time and pick a suitable value per packet. Existing PIT entries do not interfere with other predecessor's ARQ mechanisms, so there is isolation between long and short paths.

NACK messages will consume PIT entries for each affected predecessor. This should erase all PIT entries affected by that upstream node. This will reset the state of specific predecessors so the state machine will start again for that predecessor.

7. Conclusion