LISP ITR Graceful Restart
draft-saucez-lisp-itr-graceful-00

D. Saucez, O. Bonaventure, L. Iannone, C. Filsfils
Motivation

- Traffic shifts upon ITR restarts
Motivation

• Traffic shifts upon ITR restarts
Motivation

- Traffic shifts upon ITR restarts
Motivation

- Traffic shifts upon ITR restarts
Motivation

- Traffic shifts upon ITR restarts
Motivation

- Traffic shifts upon ITR restarts
Motivation

- Traffic shifts cause cache-miss storms
- upon ITR stop
Motivation

- Traffic shifts cause *cache-miss storms*
- upon ITR (re)start
Motivation

- Traffic shifts cause cache-miss storms
- upon ITR (re)start
How to avoid or at least mitigate cache miss storms?
How to avoid or at least mitigate cache miss storms?

- 3 techniques
  - non-volatile mapping storage
  - ITR deflection
  - ITR cache synchronization
Non-volatile mapping storage

• Before restarting, the ITR stores its cache on a persistent storage unit
  + transparent for the network
  + limited overhead
  - only suited for short ITR un-availability periods
  - not applicable to ITR addition
ITR deflection

- During warmup, upon miss, deflect the packet to another ITR, in parallel send a Map-Request to populate the local cache
  - deployable today without changing code (only configuration)
- how long is the warmup phase?
- how to determine ITR to which deflect packets?
- risk of loops between ITR
ITR cache synchronization

- ITRs synchronize their cache with the other ITRs
  + efficient (next slide)
  + no time constraints
  + no black voodoo!
  - possible network overhead
  - possible memory overhead
No miss storm with synchronized caches*

- Every cache has the same entries

<table>
<thead>
<tr>
<th>Time</th>
<th>Number of cache misses</th>
</tr>
</thead>
<tbody>
<tr>
<td>30 Min</td>
<td>0</td>
</tr>
<tr>
<td>20 Min</td>
<td>1K</td>
</tr>
<tr>
<td>10 Min</td>
<td>2K</td>
</tr>
<tr>
<td>Stop</td>
<td>3K</td>
</tr>
<tr>
<td>+10 Min</td>
<td>4K</td>
</tr>
<tr>
<td>+20 Min</td>
<td>5K</td>
</tr>
<tr>
<td>+30 Min</td>
<td>6K</td>
</tr>
<tr>
<td>MAX</td>
<td>7K</td>
</tr>
<tr>
<td>ITR2 (ITR1 stop)</td>
<td>8K</td>
</tr>
<tr>
<td>ITR1 (ITR2 stop)</td>
<td>9K</td>
</tr>
</tbody>
</table>

Campus (peak hour)

ISP (peak hour)
Two cache synchronization patterns

- Push
  - mappings are \textit{pushed} to the ITRs cache
- Notification
  - ITRs are \textit{notified} of the presence of a new mapping they might cache
  - upon notification reception, ITR decides to retrieve the mapping or not
Questions to the WG

• Is it an interesting subject for the WG?

• Which technique (persistence, deflection, synchronization)

• At what level?
  • LISP, EGP, IGP, configuration, home made...
Conclusion

• The restart of an ITR can cause traffic shifts to the other ITRs
  • traffic shifts lead to cache miss storm

• For predictable ITR operations (e.g., maintenance, decommission, addition...), countermeasures can be taken to gracefully handle the event
Backup
How to synchronize ITR caches?
Two possible patterns

- Push
  - mappings are pushed to the ITRs cache
- Notification
  - ITRs are notified about the presence of a new mapping they should install in their cache
  - upon notification reception, Map-Request is sent for the notified EID
Many technologies

- Configuration
- New protocol
  - add new control-plane messages types to LISP
- Extend current protocols
  - BGP (communities, AFI/SAFI)
  - IGP (new TLVs)
  - ...

Push
Push methods

- MR Included Map-Reply
- ITR Included Map-Reply
- IGP Push
- MR Multi-Request
- MR Multicast
- Dynamic Push
MR included Map-Reply

MR (proxy mode)

1. itr1.Map-Request(e)
2. mr.Map-Request(e)
3. Map-Reply(e/E, m)->mr
4. Map-Reply(e/E, m)->itr1
5. included-Reply(e/E, m)
ITR included Map-Reply

1. itr1.Map-Request(e)

2. Map-Reply(e/E, m)->itr1

3. included-Reply(e/E, m)
IGP Push

IT1.cache:
<e1/E1, m1> ...
<en/En, mn>

IT2.cache:
<e1/E1’, m1’> ...
<en/En’, mn’>

IT3.cache:
<e1/E1”, m1”> ...
<en/En”, mn”>
MR Multi-Request

To avoid the burden of Map-Requests, MR can operate in proxy mode and generate the Map- Replies
MR Multicast

ITRs belong to the same multicast group

MR (proxy mode)

1. mcast.Map-Request(e)
2. mr.Map-Request(e)
3. Map-Reply(e/E, m) -> mr
4. Map-Reply(e/E, m) -> mcast
Dynamic Push

- network is monitored to detect “important” EID prefixes
- mappings for “important” EID prefixes are pushed to the ITRs
Notify
Notify methods

- MR Notify
- ITR Notify
- IGP Notify
- Dynamic Notify
MR Notify

MR (proxy mode)

1. itr1.Map-Request(e)
2. mr.Map-Request(e)
3. Map-Reply(e/E, m)->mr
4. Map-Reply(e/E, m)->itr1
5. Notification(e/E)
6. itr2.Map-Request(e/E)
5. Notification(e/E)
6. itr3.Map-Request(e/E)

ITR1
ITR2
ITR3
1. itr1.Map-Request(e)
2. Map-Reply(e/E, m)->itr1
3. Notification(e/E)
4. itr2.Map-Request(e/E)
4. itr2.Map-Request(e/E)

ITR Notify
at ITRx, upon IGP change, send Map-Request for the new redistributed EID prefix
Dynamic Notification

- Network is monitored to detect “important” EID prefixes
- Mappings for “important” EID prefixes are notified to the ITRs
BGP based IGP notification (1/2)

- Full iBGP mesh between ITRs
- Redistribute EID prefixes from the cache to iBGP sessions
- Tag each prefix with a non-transitive extended community (+ no-export)
  - the community value is the synchronization set identifier
- can be repeated several times
BGP based IGP notification (2/2)

- ITRs should Map-Request the EID prefix when a new advertisement is received
- Withdrawals MUST have no effect on the cache
How to determine the ITRs that must be synchronized?

Synchronization Set
Not necessary to synchronize all the ITRs

• Group ITRs potentially serving the same hosts in synchronization sets

• Synchronize ITRs cache only with the other ITRs of the synchronization set