Fast group management in IGMP

Luigi Rizzo
luigi@iet.unipi.it

These slides, and related papers and software, are available at:
http://www.iet.unipi.it/~luigi/mrouted/

Work partly supported by the Ministero dell' Università e della Ricerca Scientifica e Tecnologica (MURST) in the framework of the Project "Design Methodologies and Tools of High Performance Systems for Distributed Applications".

Summary

- Motivation for a Fast Leave mechanism.
- Our proposal.
- Implementation and Experimental results.

Motivation for a Fast Leave mechanism

- Recent proposal for multicast congestion control (RLM, RLC) are based on the ability of multicast routers to quickly start/stop group distribution.
- receivers subscrive to one or more groups depending on available bandwidth, routers simulate sender's adaptation to the different subtrees.
- We need to stop a group quickly in case of congestion, or the congestion control mechanism will not be effective.

Leave response in IGMP

IGMP (starting from v.2) permits blocking a group on demand, but leave response is still slow.

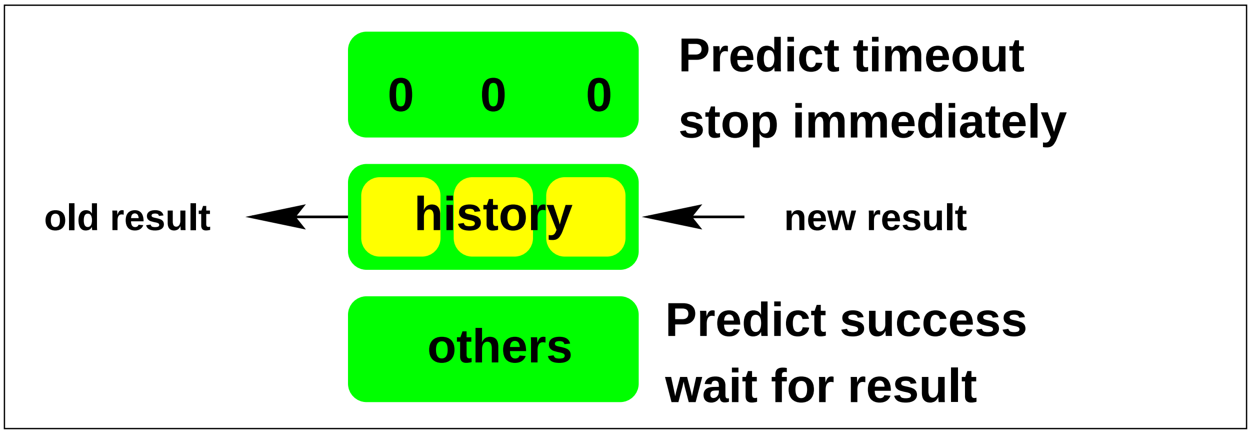
- For scalability reasons, IGMP does not hold exact membership information.
- On a leave request, a poll must be conducted to check the presence of active receivers.
- We must poll at least twice (possibly even more!) to compensate for lost messages.
- We cannot shorten timeouts too much. Can cause problems with slow receivers or large groups.

This results in a delay which is currently 3 seconds.

Our proposal

Use prediction to accelerate the response to leave requests:

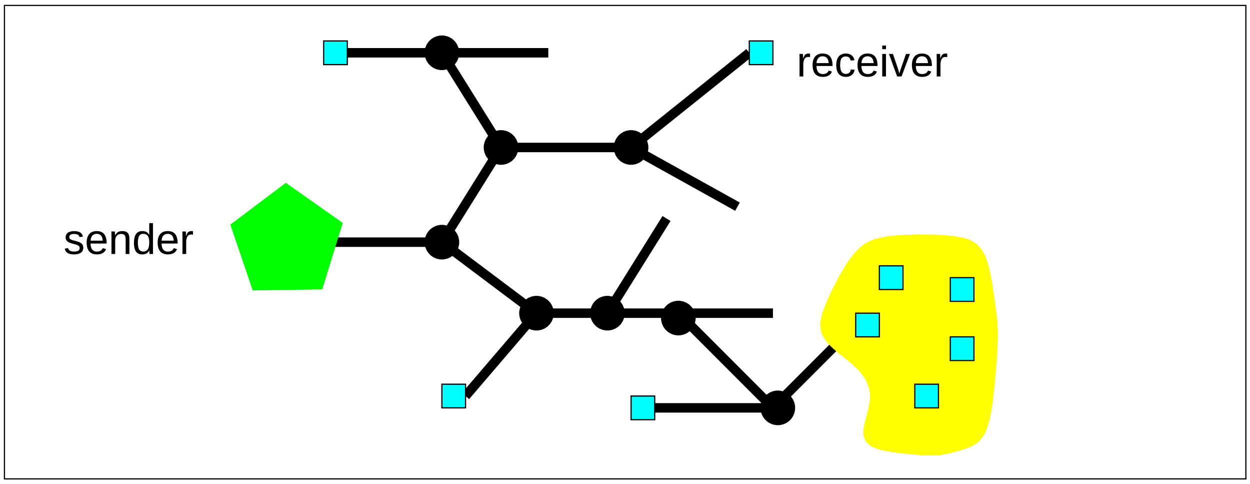
- keep a record of recent poll results;
- if recent polls always failed, predict a failure; otherwise, predict a success.
- always run a poll to verify the correctness of the prediction and recover from mistakes.



Does it work ?

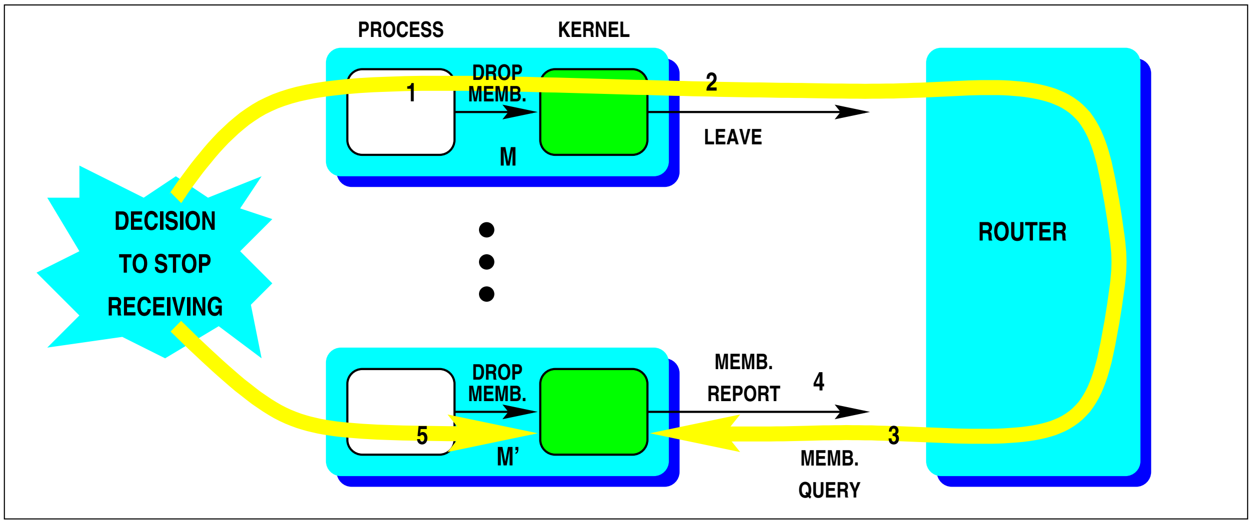
The predictor identifies well the following, very common, cases:

- only one receiver on the local net: we want quick leave response.
- multiple receivers for non-congestion controlled apps: receivers join/leave randomly, we don't want/need fast leave.



But there is a problem...

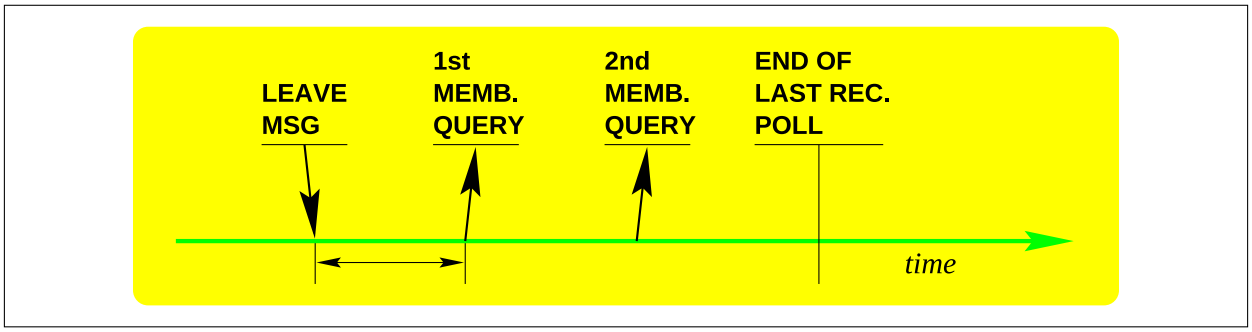
There is a race condition in presence of multiple receivers.



This can give rise to false membership reports which pollute the history vector and render the mechanism ineffective.

How to remove false membership reports

We delay the first membership query after the leave request, to give enough time to slow receivers.

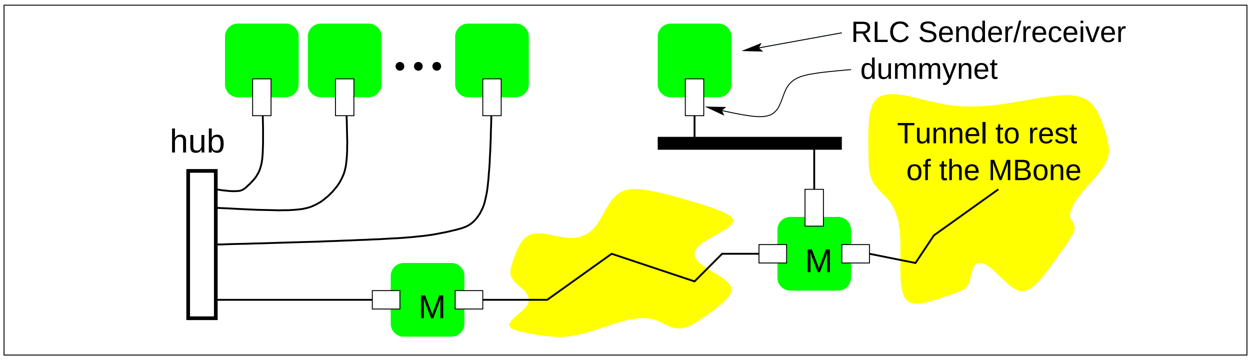


- There is only a minor penalty in case of a misprediction (but prediction is very effective);
- a good value for the delay (currently 1 second) is still under investigation.

Implementation status

- Requires minimal modifications to `mrouted3.9-beta3` code.
- no kernel or IGMP modifications required.
- code available at `http://www.iet.unipi.it/~luigi/mrouted/`
- we are running at our site.

Our Test setup



We have run some experiments using the above setup:

- modified mrouted3.9beta3 (Fenner)
- sample RLC sender/receiver (Vicisano);
- dummynet to simulate slow links/delays;

Tests have shown that the mechanism works as expected; but the network on the right is a black hole and never sends prunes.

Conclusions

Testing the effectiveness of the mechanism in a real network is very difficult because of

- IGMPv1 receivers;
- non-pruning routers;

and, mainly:

- no real applications using layered congestion control!

Simulation at this stage does not help much. We need real numbers to determine how to set delays, and how effective is the prediction mechanism.