Hierarchical PIM

Mark Handley
Department of Computer Science
University College London

Overview

- Brief introduction to SM-PIM
  - Rendez-vous points
  - Building of shared trees
- The GARP Problem
- Hierarchical PIM
  - Rendez-vous Hierarchy
  - Building C-RP Lists
  - Loop Prevention
  - Optimising the Shared Tree
- Problems with HPIM

Background Assumption

- For high rate delay sensitive traffic, we would like a source-based shortest-path tree.
  - DVMRP does this directly
- For low rate delay-insensitive sources (particularly where there are a very large number of such sources), we would like a shared tree with little or no source-specific state.
  - CBT does this directly

SM-PIM observation

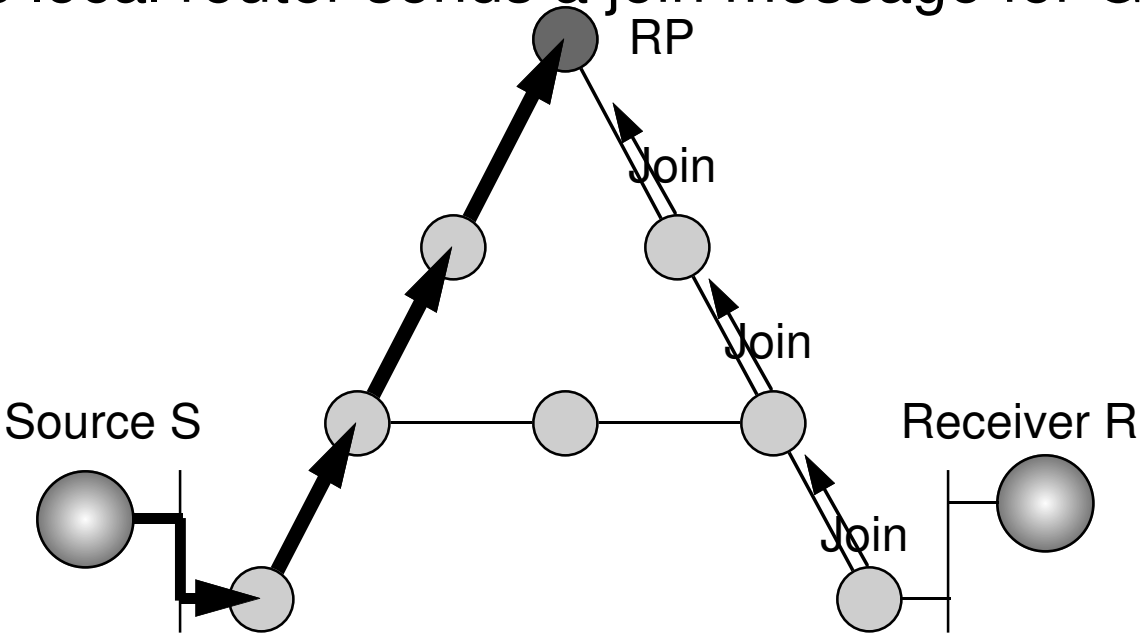
- Sparse-mode PIM makes the neat observation that finding the group can be separated from building an optimal tree.

SM-PIM

- Instead of flooding and pruning to directly build a shortest path tree, Sparse-mode PIM initially builds a shared tree.
- The shared tree is built by sending join messages towards a Rendezvous Point (RP)
- Once data is flowing, the shared tree can be converted to a shortest path tree if required.

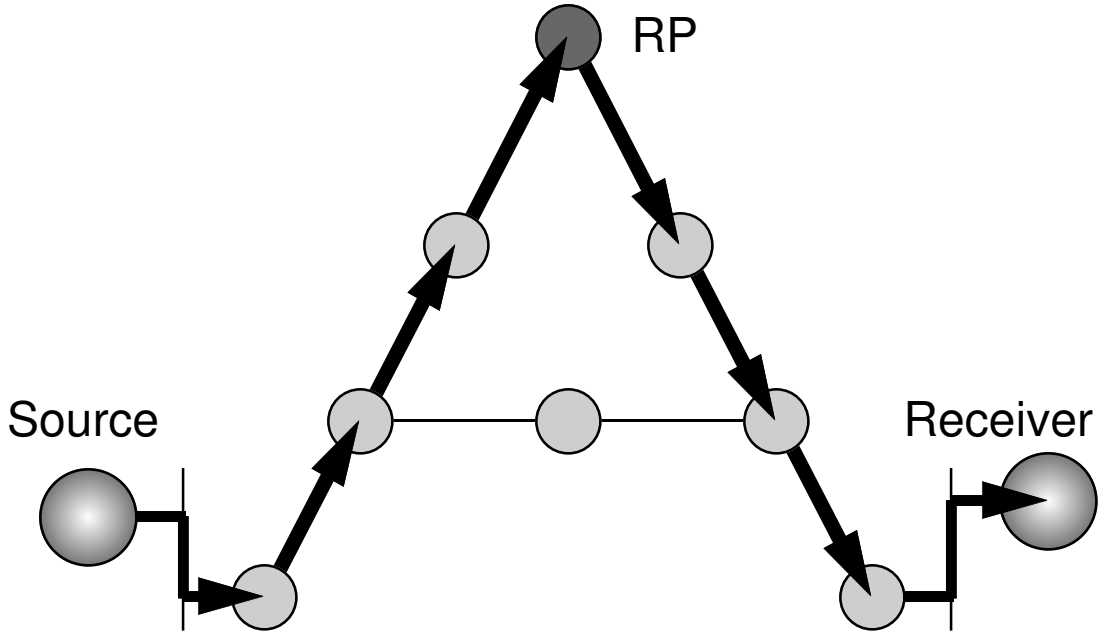
SM-PIM: Building a Shared Tree

- Source S sends traffic to group G
- S's local router encapsuletes group G traffic and sends it directly to the Rendezvous Point (RP) for group G
- R's local router sends a join message for G to RP



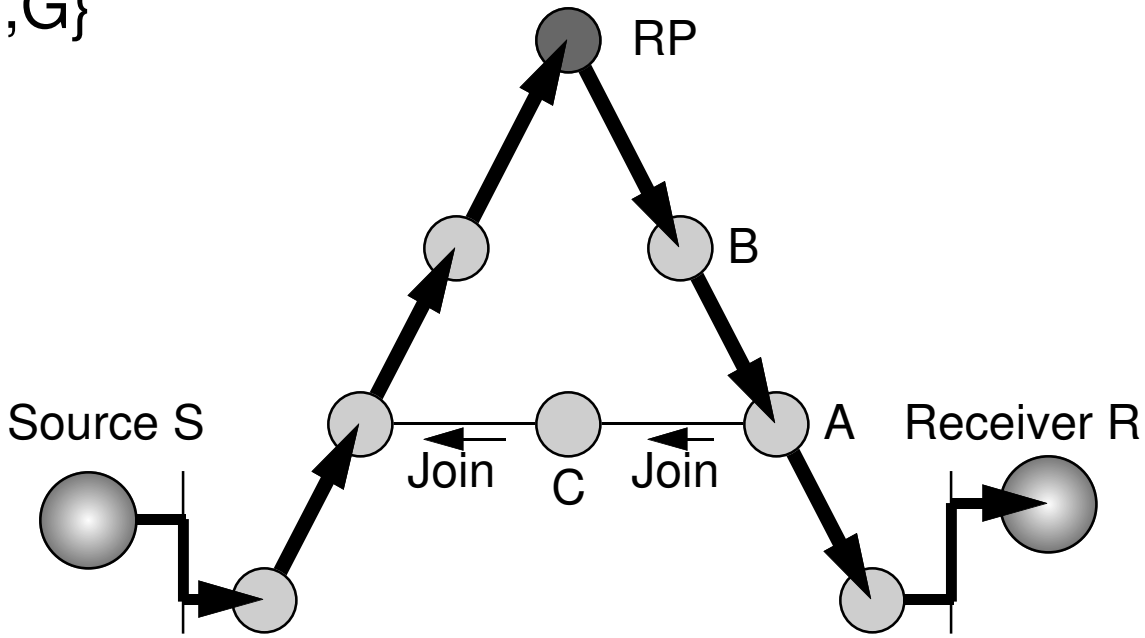
SM-PIM Shared Tree

- Initially data flows over RP tree



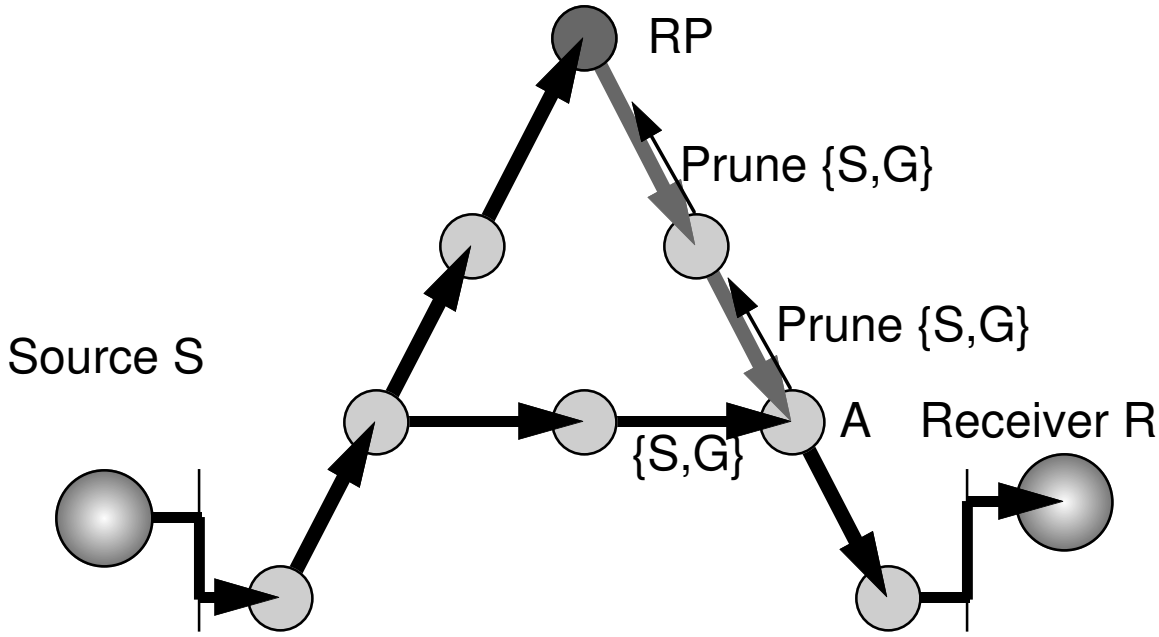
Transfer to SP Tree

- Router A sees traffic from S arriving over its B interface, when its unicast route to S is over its C interface.
- If traffic warrants it, A sends a join towards S for {S,G}



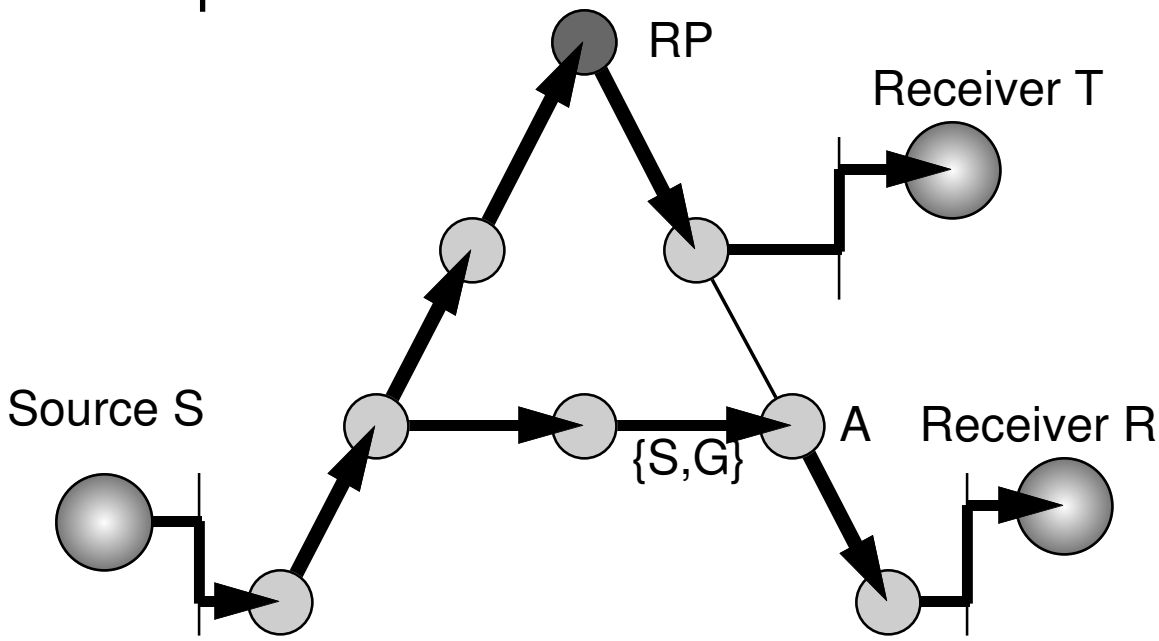
Transfer to SP Tree (cont)

- When the Join towards S is successful, group G traffic from S starts to flow along the shortest path tree. A now receives the traffic twice.
- A then sends a prune for {S,G} towards the RP for group G, and the transfer to SP tree is complete.



Sparse Mode PIM

- Receiver R receives {S,G} on the shortest path tree.
- Receiver T continues to receive {S,G} on the RP tree, although it too can switch to a shortest path tree if required.



IP Multicast: Service Model

- ***Receivers announce interest***
  - join multicast group
- ***Senders just send***
  - to multicast group address
- ***Routers "conspire" to deliver sender's data to all interested receivers***
  - using one or more multicast routing protocols

The GARP Problem

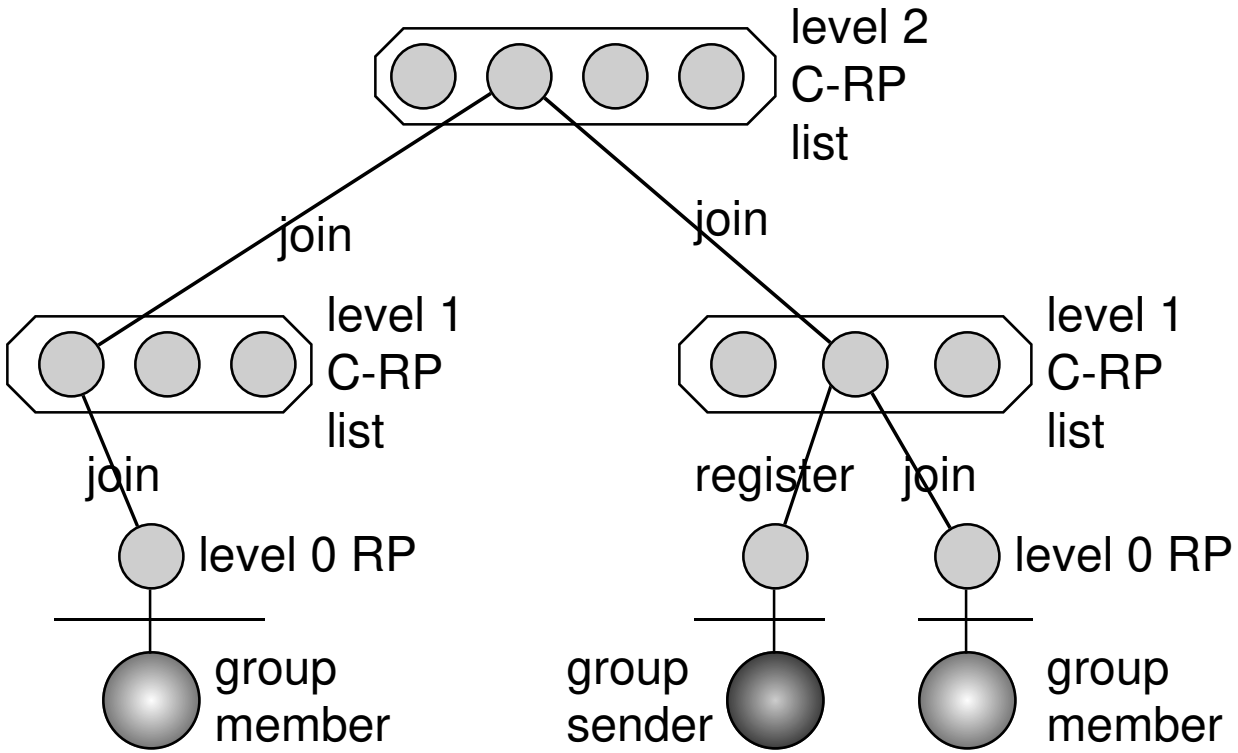
- GARP = Globally Advertised Rendez-vous Point
- Problem:
  - **Local routers near to senders and receivers need to be able to perform a multicast group to RP-list mapping.**

Some Possible Solutions

- distribute RP-lists along with multicast addresses using a session directory
  - undesirable host/application dependency
  - problem with unadvertised sessions
  - problem at interconnect points between different multicast routing protocols
- distribute RP-lists via a “routing protocol”
  - short lived groups may come and go before the RP-list arrives
  - too much state for all routers to maintain

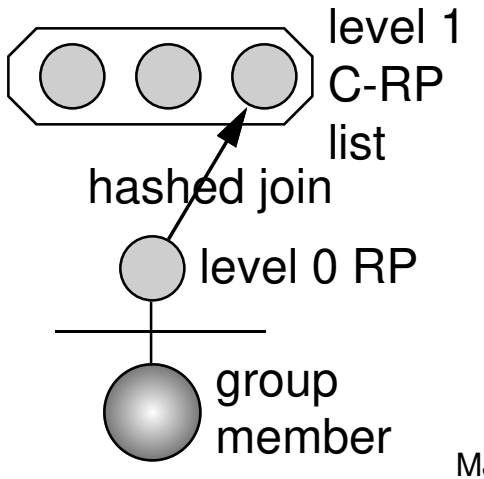
Hierarchical PIM

- Joins and Registers travel from RP to RP until the highest level for that group is reached.
- At each level, the RP makes a local decision based on information about the level above about which next level RP to join to.



Choosing the next RP

- Each RP has a level in the hierarchy.
- Each RP knows the Candidate-RP list for the level above it in the hierarchy.
- On receiving a join, an RP hashes the session's multicast address into the next level's C-RP list, and joins to that RP.
- Each multicast address range has a "maximum level", which determines the scope of the session



Join messages

- A join message travels from an RP at one level to an RP at the level above, establishing shared-tree forwarding state as it goes.
- Join state is bi-directional (can be used by packets travelling either direction).
- Join messages carry:
  - the group address
  - the previous and next RPs' addresses
  - the level of the previous RP.
- Joins are ACKed when the reach the next RP
- Failures are dealt with by retransmission with a timeout, then trying a secondary hash function to select a backup RP.

Register Messages (new senders)

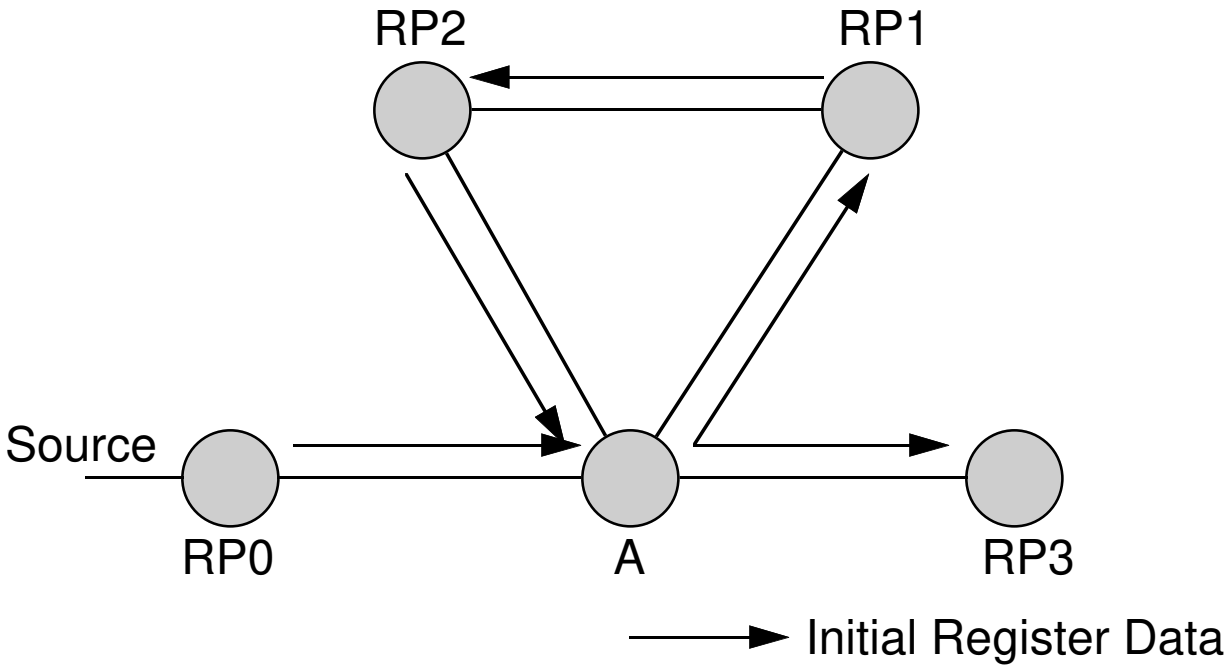
- If there is a local receiver, then its join state is used for the sender.
- If there is no local receiver, then we must perform a register (sender join).
  - It is important that the first packet(s) sent are not lost, and reach the tree.
  - Register messages must therefore encapsulate the data packets until register state is set up to the “join” tree.
- Register state is uni-directional (can only be used to forward traffic up the hierarchy).
  - Subsequent receiver joins can convert register state into join state, but registers always stop on reaching an active part of the “join” tree.

Register Encapsulation and Acknowledgement

- Three phase process:
  - RP sends an encapsulated register packet to the next hop router towards the next RP.
  - next hop router:
    - decapsulates the packet
    - sends a hop-register-ACK back one hop (the previous hop no longer needs to encapsulate - there is now “provisional” register state for the group)
    - encapsulates the packet and sends it to the next hop router.
  - on reaching either the next RP or a router with shared tree join state, a register ACK is send back to the previous RP.
  - previous RP then sends a register-confirm up the register state path to make the register state “active” .
    - if the state is “active” then later same-level same-group “provisional” registers wont destroy the existing state.

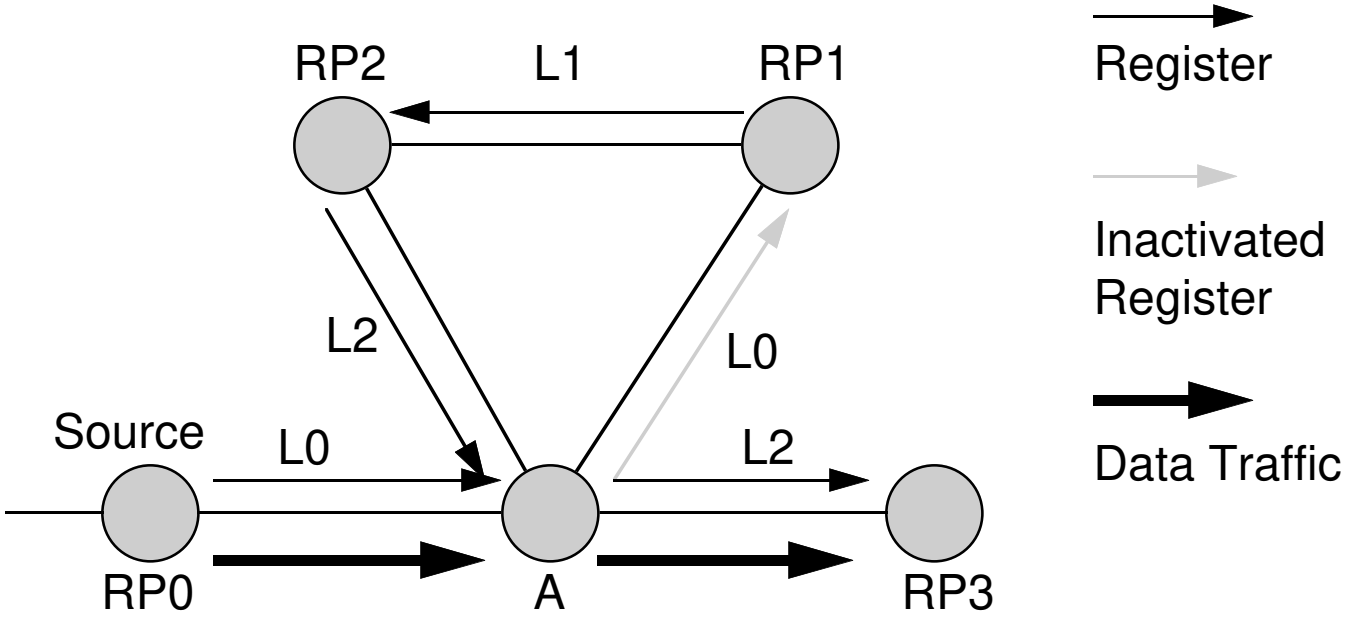
Loops

- Using a non-global “routing’ algorithm can inevitably lead to loops.
- Multicast loops are distasterous!



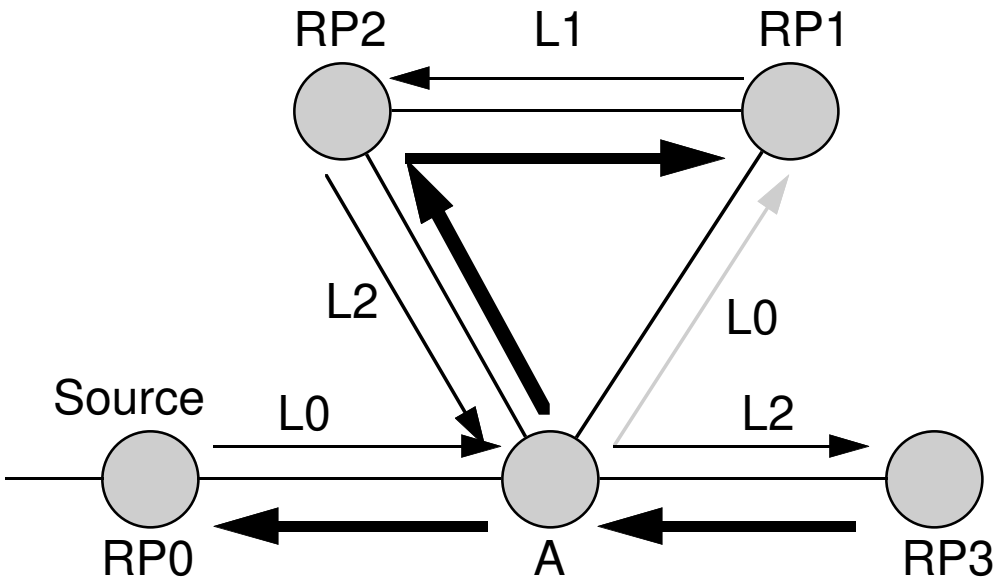
Loop prevention (registers)

- Rules:
  - no router can ***create*** active forwarding state on more than one interface simultaneously.
  - higher level registers/joins invalidate lower level ones



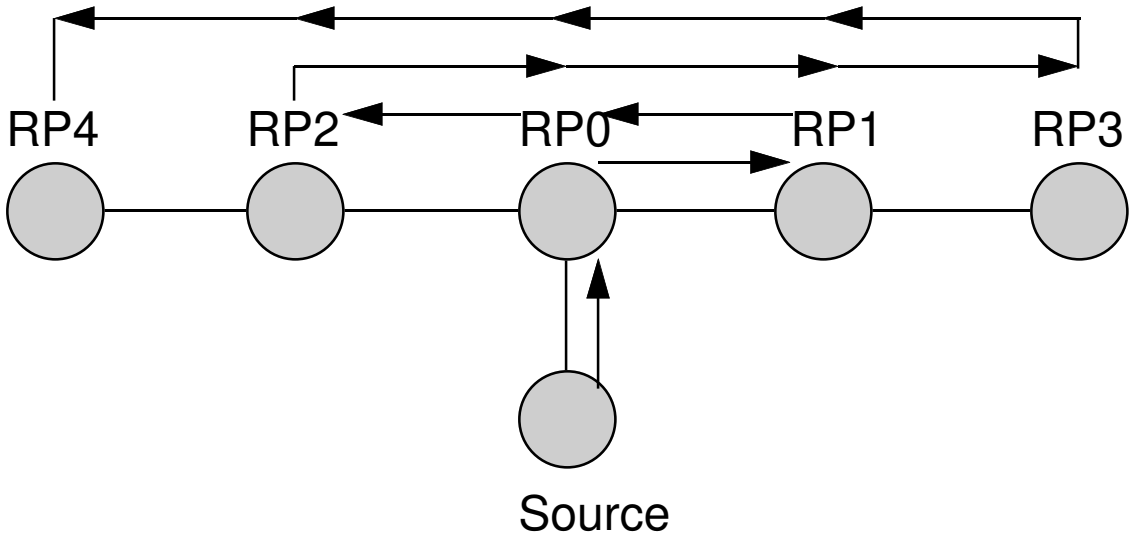
Loop Prevention (Joins)

- Same rules as for registers
  - note “inactive” bit only stays set for one hop.
  - may result in unnecessary traffic flow in some circumstances.



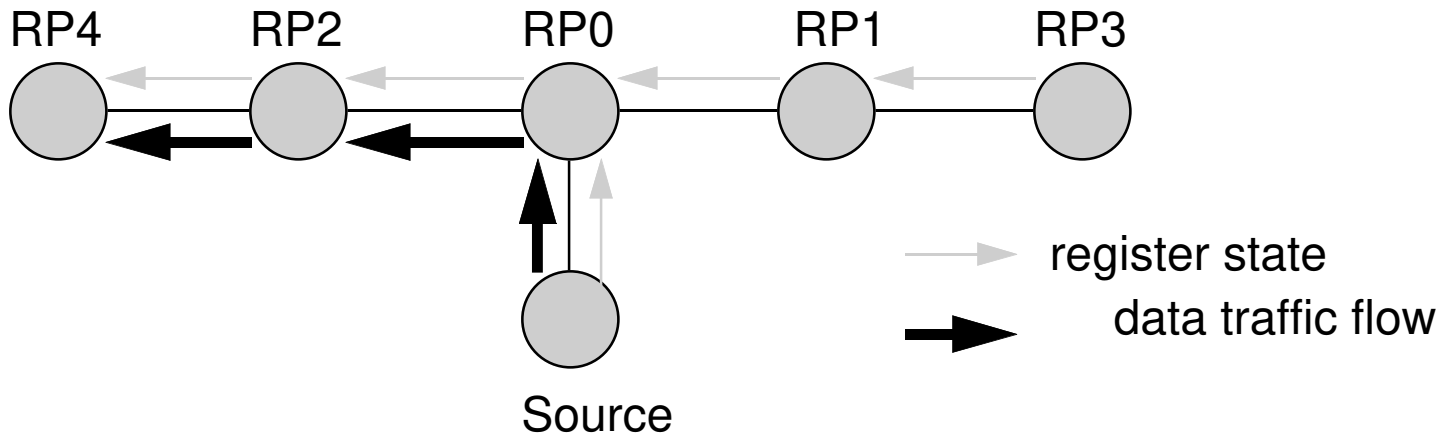
Short cut forwarding

- In some cases RP placement may cause very sub-optimal routing of register messages:



Short cut forwarding

- The loop prevention rules act so that only the path most directly to the top level RP remains in use.
  - there may be transient loss as this settles down unless register encapsulation continues for a little time.



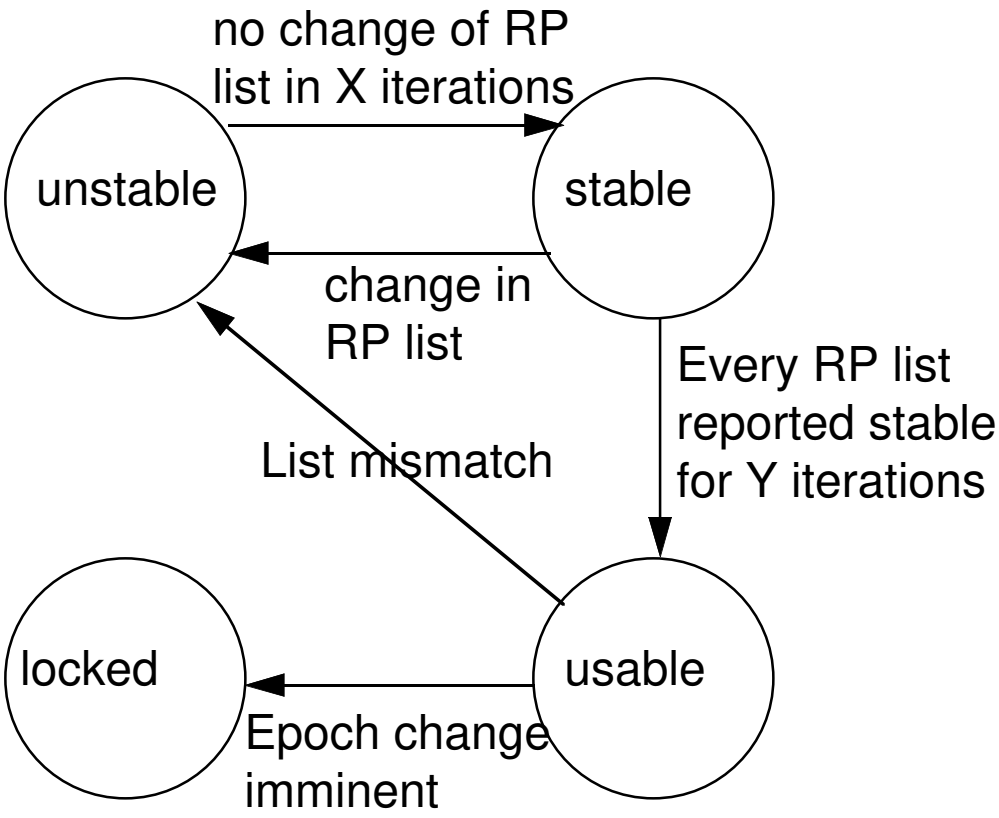
Candidate-RP lists

- Each C-RP at one level needs to know the C-RP list for the level above.
- At the top level for a group, it is important that the C-RP list received by the level below is consistent.
- At lower levels in the hierarchy, it's not so critical, but a level in the hierarchy is alwasy the maximum level for some group!
- There are many possible ways to do this, but desirable features are:
  - distributed algorithm (no single point of failure at any time)
  - robust in the face of network partitioning, router failures and loss.

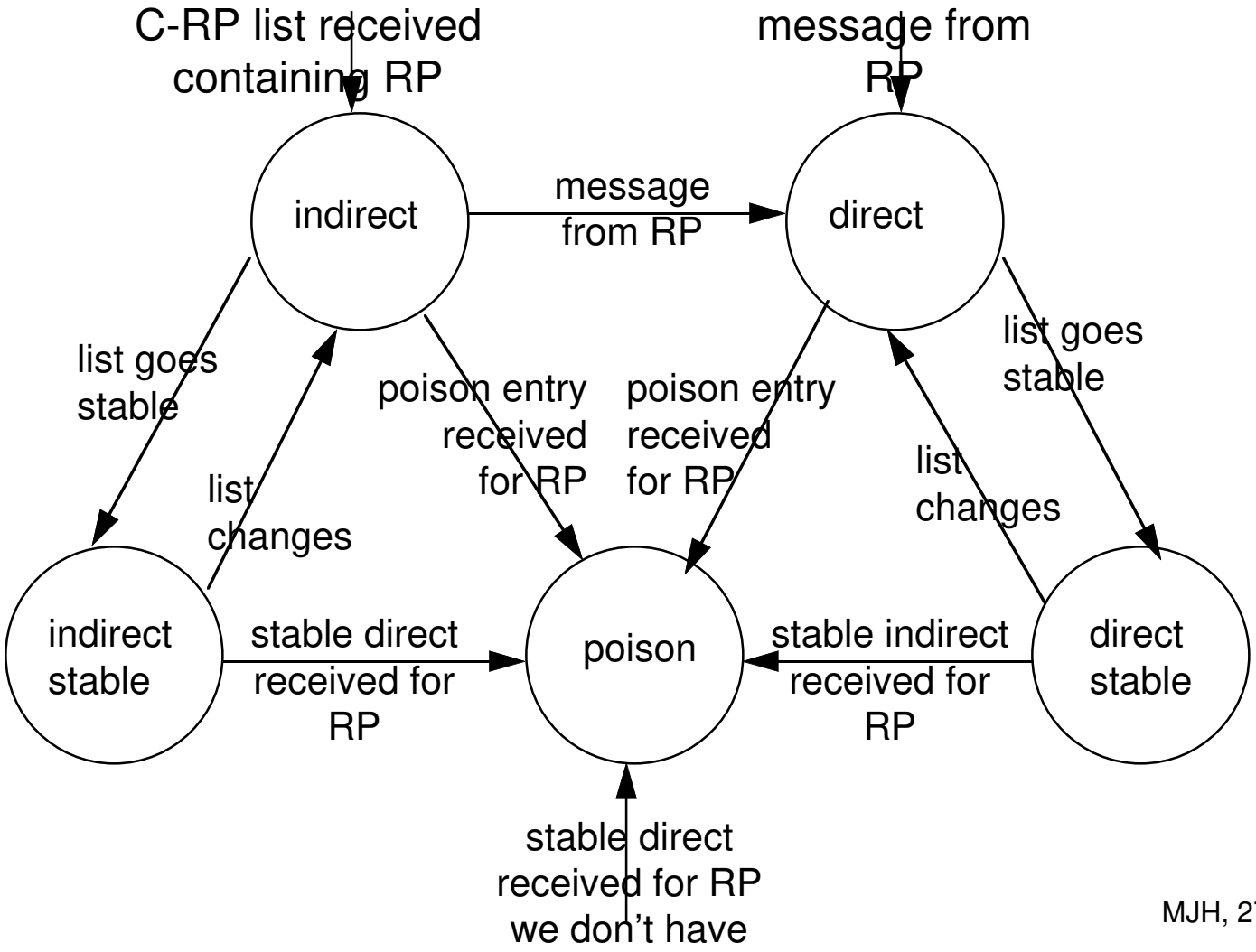
Building C-RP lists

- Each potential candidate-RP (PC-RP) in a scope region at a level dense-mode multicasts its availability to its peers.
- Each peer constructs a sorted list of PC-RPs and multicasts this list periodically. Each entry in the list can have 3 states:
  - **direct** - “I heard from it directly”
  - **indirect** - “Someone told me they heard it”
  - **poison** - “Someone detected a problem with this C-RP - poison it”
- Each list can also be “unstable”, “stable” or “usable”.

STD for a C-RP list



STD for a C-RP list entry



Building C-RP lists

- If everyone can hear someone, they're accepted.
- If only some people can hear someone, they're poisoned.
- Persistent symmetric partitions can cause failure.
- Assymetric partitions result in the poisoning of everyone in the partion that can still receive the other partition.
- Inconsistencies discovered after an epoch change are resolved by forming the union of the two lists.

RP Failure.

- RP failure is not a problem at any other level than the maximum level for the group
  - dealt with by marking the RP as down, and using a secondary hash function.
- At the maximum level, an RP receiving a join/ register can tell (from the C-RP list) whether it's the primary for that group.
  - If it's not the primary, it continually attempts to join to the primary.
  - this join must be an inactive join (to avoid destroying any existing state it crosses) until the join is ACKed.

Outstanding issues

- Dense-mode bootstrapping may be undesirable.
- Scope region boundaries may require too much configuration.
- Can lead to very sub-optimal placement of top level RP.
- May be possible to migrate top level RPs depending on traffic.
- Alternatives to Pseudo-Random hashs may result in better first-pass shared tree creation, but are not critical to it's success.