Mobility in RIFT

Pascal Thubert

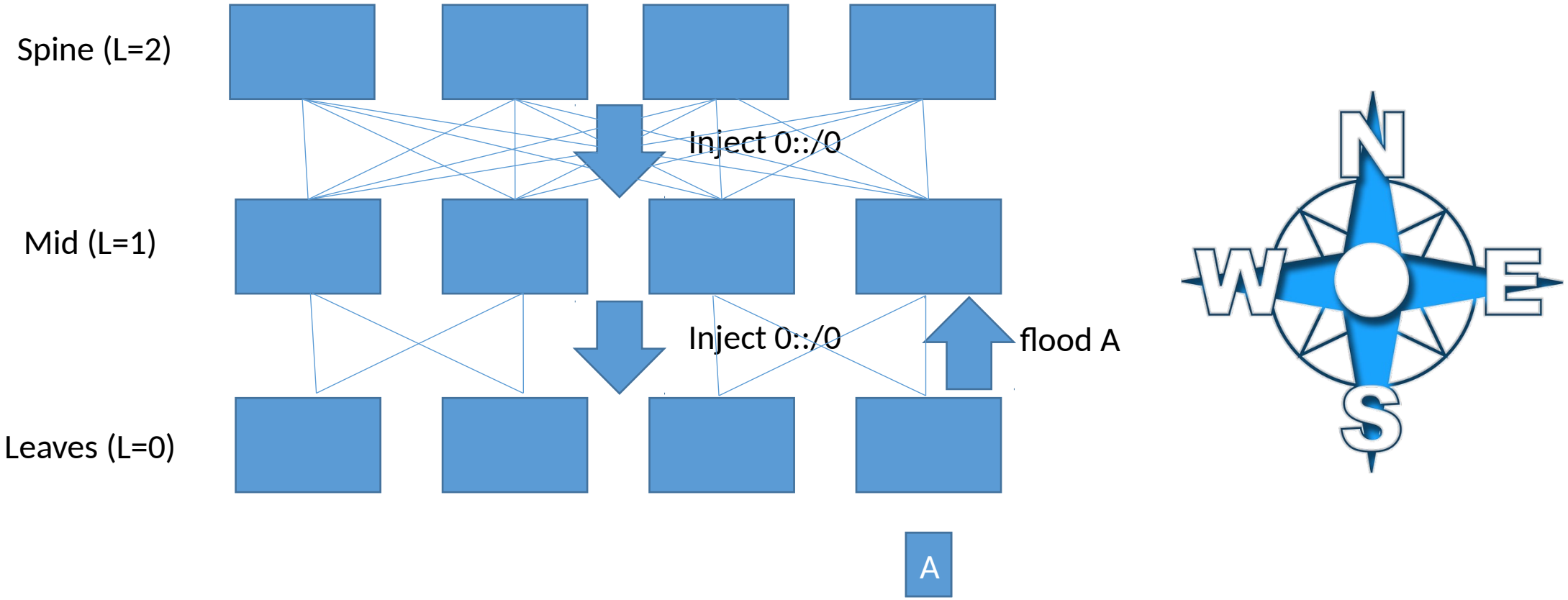
New requirement from London (added in draft-ietf-rift-01)

REQ17:

The control plane should be able to unambiguously determine the current point of attachment (which port on which leaf node) of a prefix, even in a context of fast mobility, e.g., when the prefix is a host address on a wireless node that 1) may associate to any of multiple access points (APs) that are attached to different ports on a same leaf node or to different leaf nodes, and 2) may move and reassociate several times to a different AP within a sub-second period.

The basics of RIFT is illustrated below; take a fat tree that is more or less fully meshed between the spine and the level below, and then partitioned in pods:

- Goal is to speed up convergence in control plane; instead of sync'ing all to all nodes like in a classical link state:
- RIFT nodes flood down South (from the spine) the advertisement of a default route so by default packets are forwarded northwards
- RIFT nodes flood up North (towards the spine) the advertisement of more specific routes reachable via this node so packet for match more specific routes are forwarded southwards

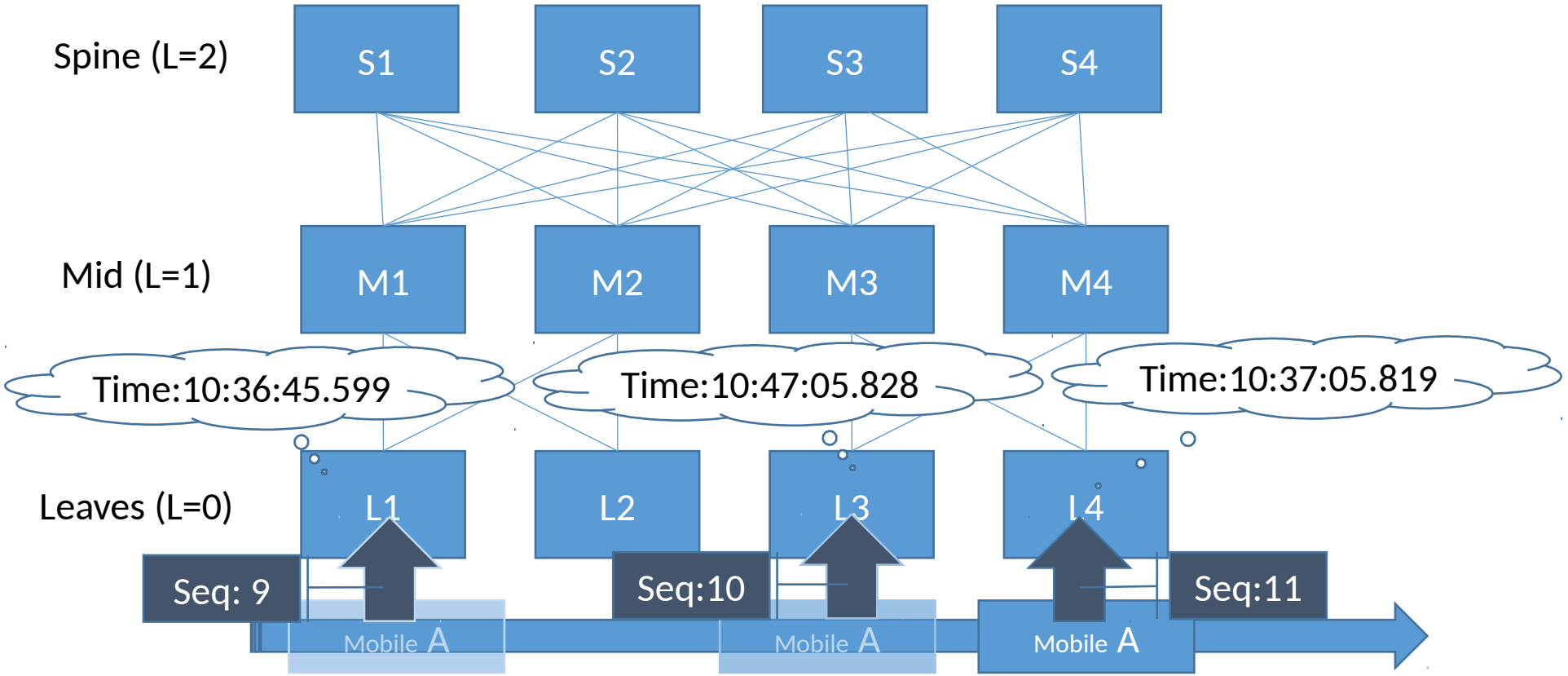


In case of a quick movement, there are 2 classical ways of discriminating the real (most recent) location:

- A sequence counter from the mobile device (requires a protocol, e.g., IPv6 ND draft-ietf-6lo-rfc6775-update)
- A time stamp by the access point / switch - the leaf- (requires precise time, e.g.; IEEE Std 802.1AS / 1588 PTP)

RIFT uses both and defines a strict order in a bi-dimensional arithmetic:

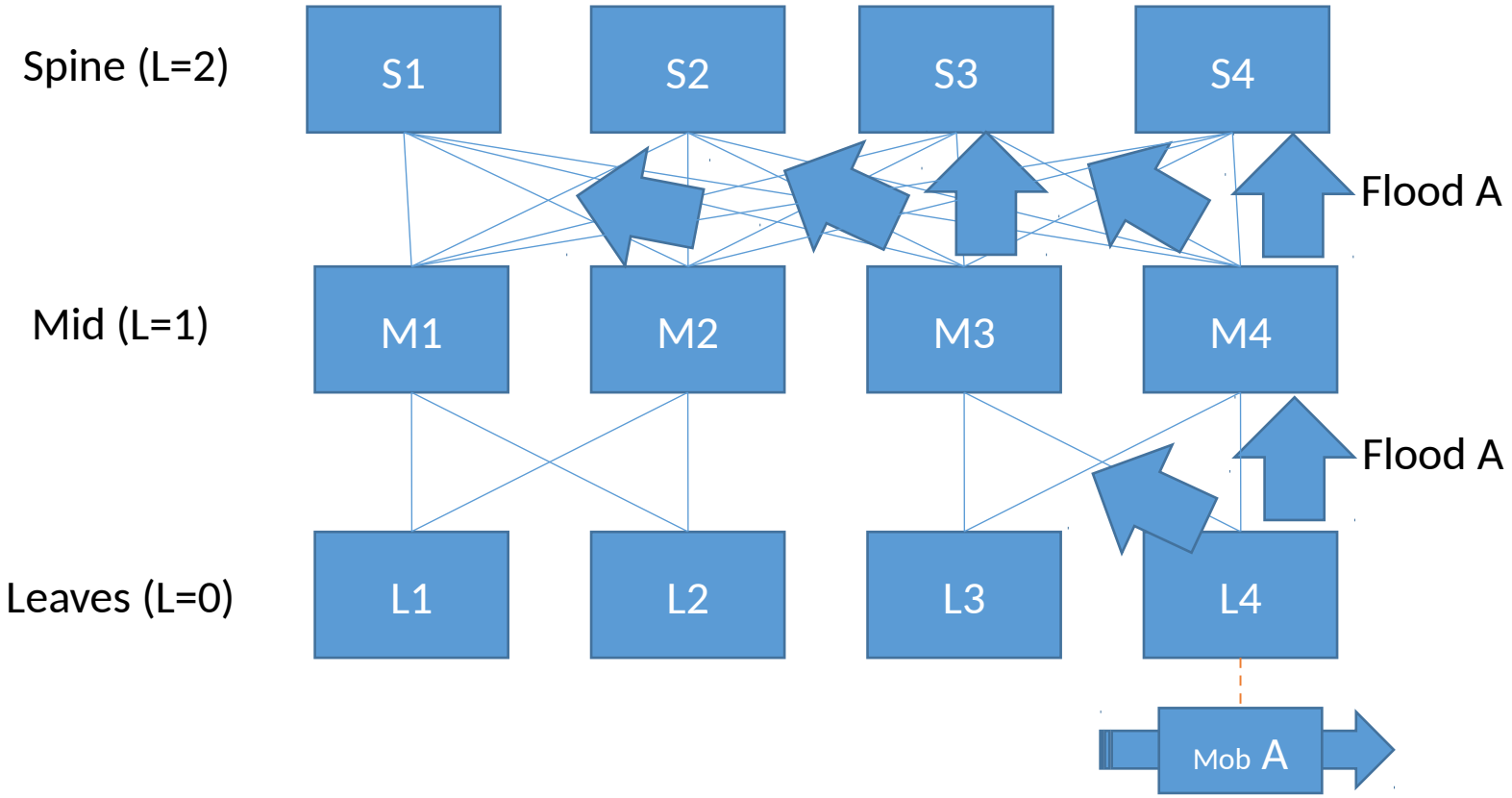
- Use time when the delta(time) > max_timing_error; this requires only rough sense of time (e.g., +/-100ms)
- Use sequence counter if available for events closer in time (must not wrap within max_timing_error)



The Northbound advertisements are flooded north throughout the fabric. This yields a large amount of traffic for any prefix, in particular for mobile prefixes. This can be alleviated by using individual Ies per mobile prefix

When the node moves from a leaf to another, old routes must be invalidated, and new routes installed when the node shows up elsewhere (attached to a different leaf). This process includes potential disaggregation etc…

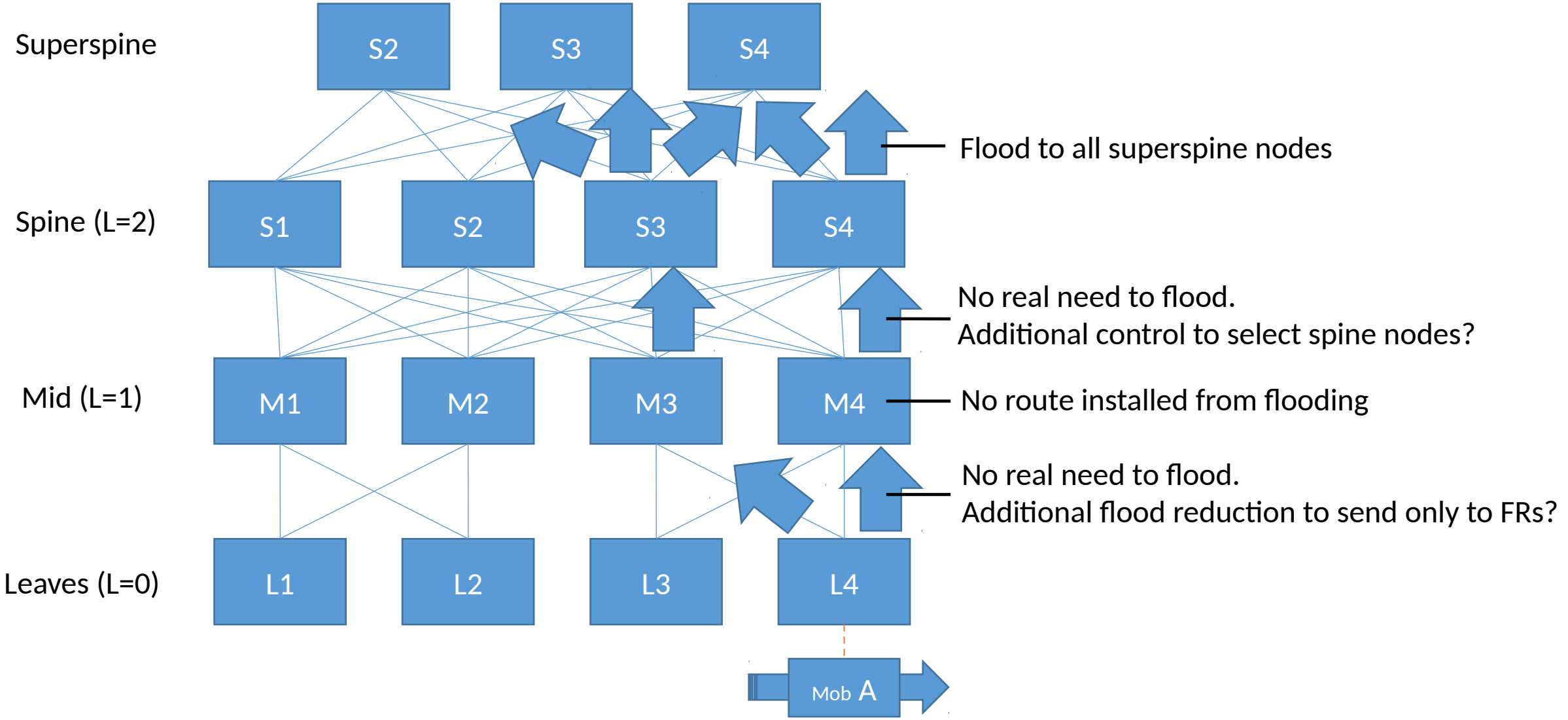
Note that this can be seen as an overlay problem. If the mobility can be fed into the overlay quickly enough then RIFT does not need to advertise the mobile prefix, and traffic to the mobile prefix can be tunneled to the attachment leaf using the overlay, e.g., LISP+VxLAN.



An alternate is to hybrid RIFT with a classical mobility approach based on dynamic tunnels to the attachment leaf.

In that approach “mobile” N-TIEs are flooded to the superspine, no need to learn from them in intermediate levels

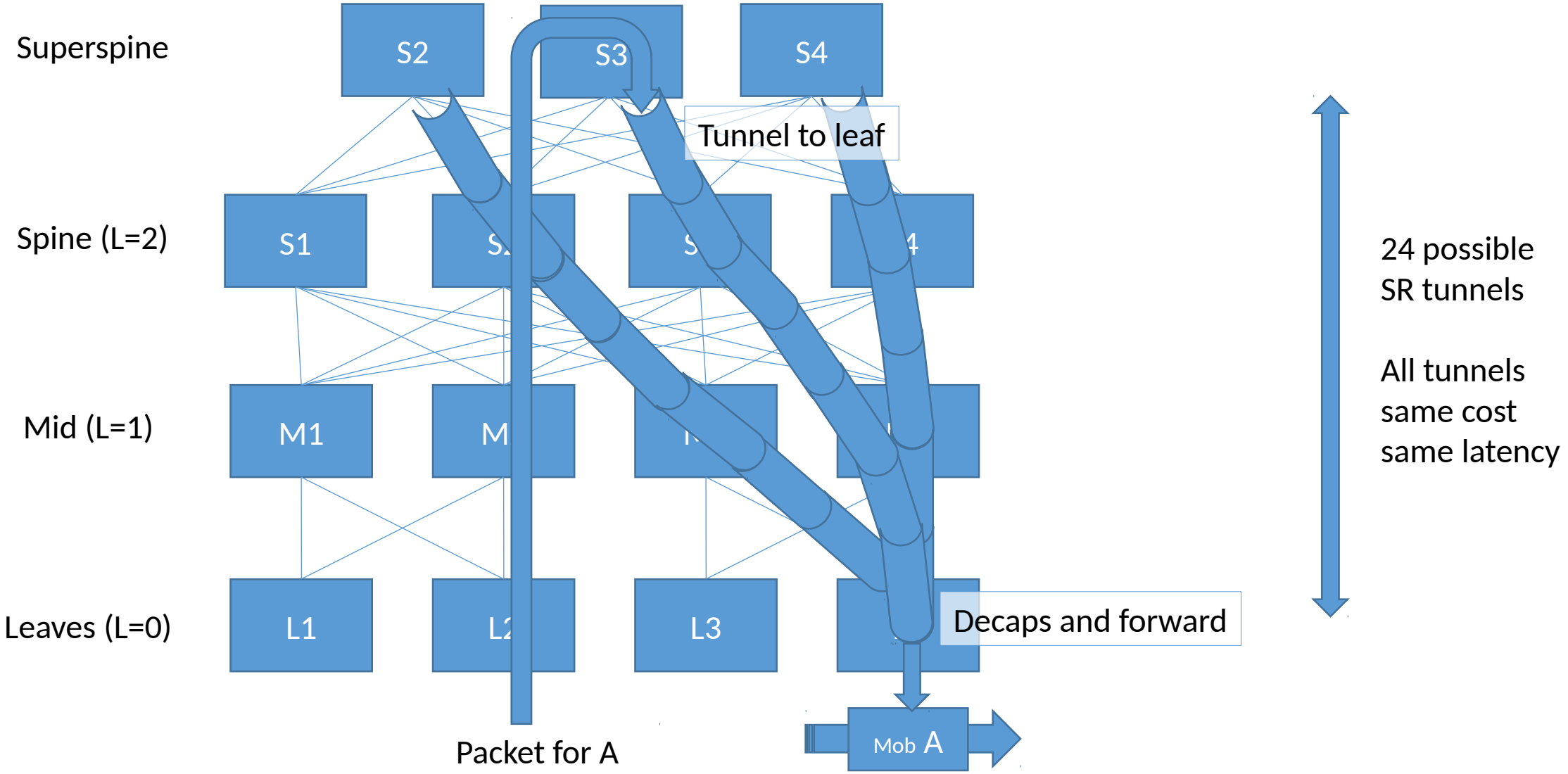
As a result, any superspine node can tunnel back to the attachement leaf



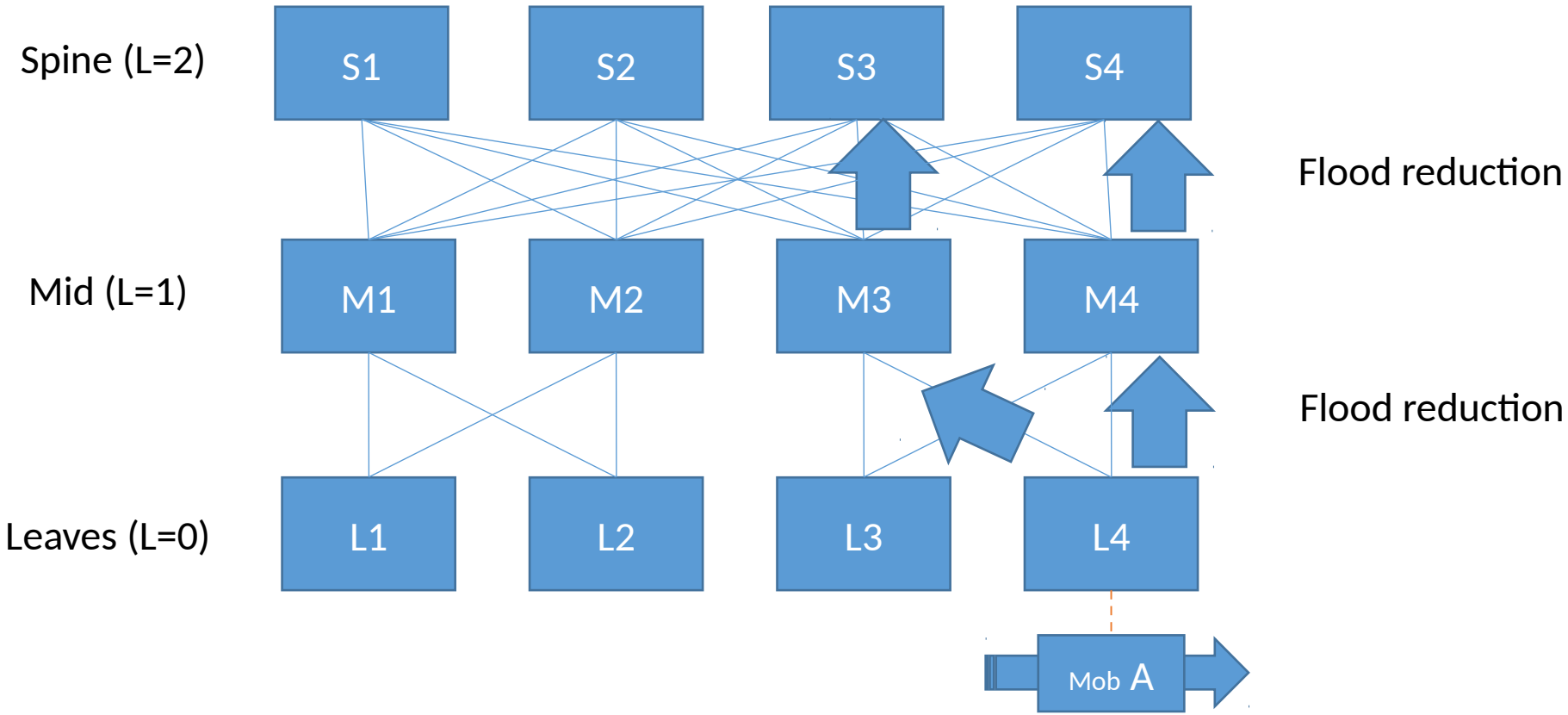
Since there is no knowledge of A south of it, packets eventually reach the superspine on any superspine node

Any given superspine node can tunnel back; e.g., using Segment Routing (SR); all tunnels have a same cost/latency

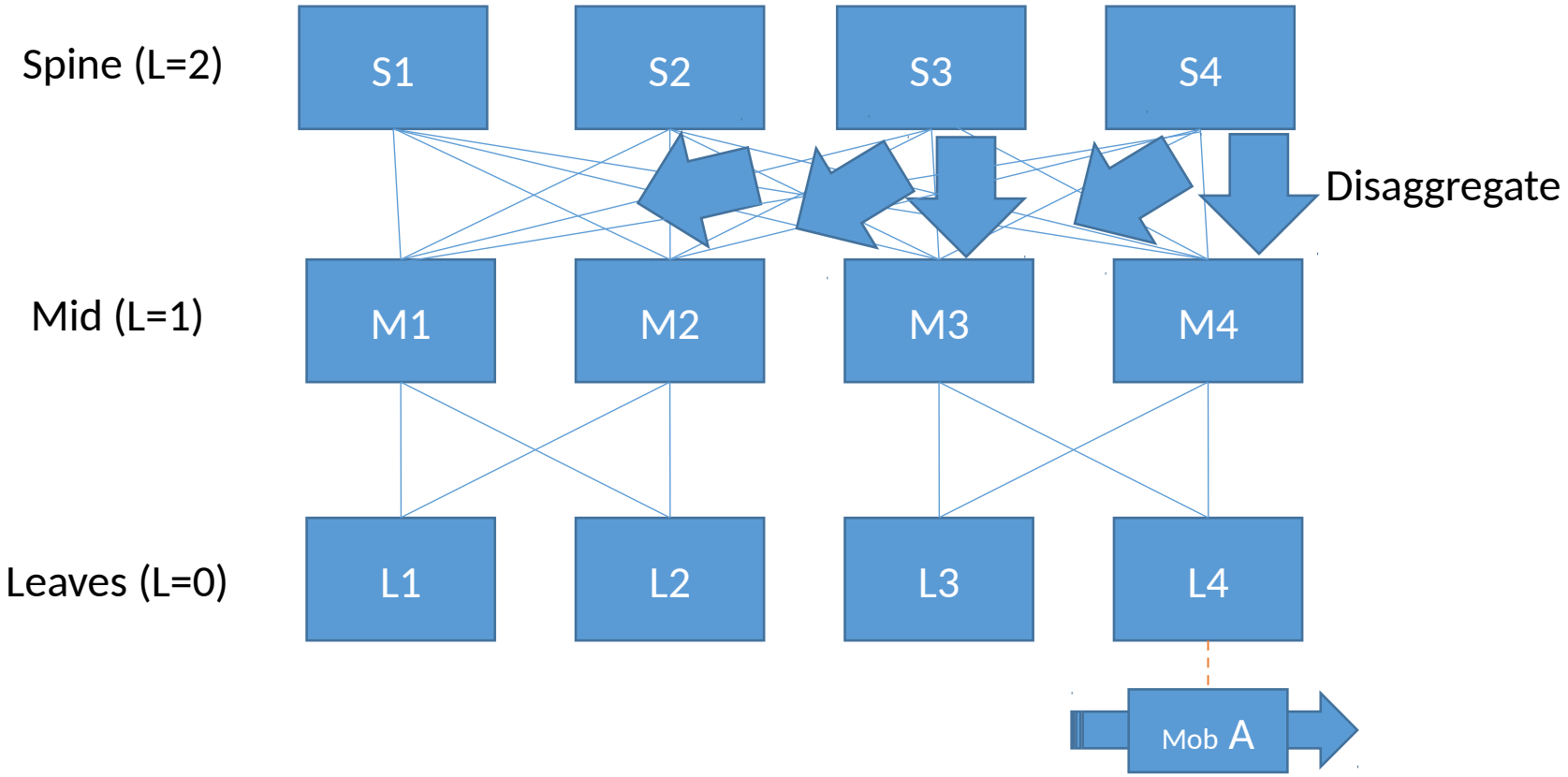
pros: fast, no stale route to clean up in the fabric; **cons:** longer path all the way through the fabric



Also considered but not retained;



the selected spine nodes need to disaggregate to be the ones getting the traffic



Since there is no knowledge of A but at the spine, packets to A get routed the spine along the more specific disaggregated route and then tunneled down by the spine to the attachment leaf