This project implements a complete link-state routing protocol for TinyOS networks, providing efficient multi-hop communication through shortest-path routing rather than flooding. The design incorporates LSA generation and propagation, Dijkstra's shortest path algorithm, and topology validation to ensure reliable packet delivery across complex network topologies. The implementation builds upon Project 1's neighbor discovery and flooding components, with routing table computation and next-hop forwarding.

Link-state advertisement generation operates through periodic and event-driven LSA creation using dedicated packets with protocol=2. Each node maintains its own sequence number and generates LSAs containing its current neighbor list obtained from the neighbor discovery module. LSAs are created immediately when neighbor relationships change and periodically every 3 seconds with randomness to prevent synchronization. The LSA packet structure includes the source node ID, sequence number, neighbor count, and up to 16 neighbor addresses. These advertisements are flooded throughout the network using the existing flooding infrastructure, ensuring all nodes receive topology updates. Sequence numbers provide ordering and duplicate detection, while timestamps enable aging to remove stale topology information.

The routing table computation uses Dijkstra's shortest path algorithm on the complete network topology stored in each node's link-state database. When LSAs are received, they update the local topology view, triggering routing table recalculation after a brief delay to allow multiple updates to arrive. The algorithm constructs a complete graph of all known nodes and their advertised neighbors, then computes shortest paths from the local node to all destinations. The resulting routing table contains destination addresses, next-hop neighbors, and path costs measured in hop counts.

Packet forwarding utilizes the computed routing tables to make next-hop decisions rather than flooding to all neighbors. When a ping command is issued, the source node first checks if it has a route to the destination in its routing table. If a route exists, the packet is forwarded directly to the computed next-hop neighbor using SimpleSend. Each intermediate node receives the packet, looks up the destination in its own routing table, and forwards to the appropriate next-hop, creating efficient point-to-point paths across the network. If no route exists, the system falls back to flooding behavior to ensure connectivity during topology convergence.

LSA aging uses differentiated timeouts to balance responsiveness with network stability. Direct neighbors are aged aggressively (15-second timeout) for rapid failure detection, while non-neighbors use 90-second timeouts to preserve topology information during network fragmentation. Maximum LSA age is set to 120 seconds, ensuring distant topology knowledge persists long enough for alternate path calculation during failures.

Several critical challenges emerged during development that required systematic solutions. The most significant issue was widespread topology corruption where nodes computed direct routes to destinations multiple hops away, essentially believing they could reach distant nodes in a single hop. Initially, individual nodes like Node 19 were found routing to Node 17 instead of the correct next-hop Node 18. The root cause was inadequate validation of computed next-hops against actual neighbors. The solution was universal topology validation that verifies every computed next-hop in the node's actual neighbor list, with distance-based fallback selection when the computed path was invalid. Another challenge was when failed nodes' LSAs would be removed locally but immediately re-flooded by distant nodes unaware of the failure. This issue was fixed through a more comprehensive node failure detection system, which detects failed nodes and immediately removes all of their LSAs.