| # | Invariant Name | Description |
|---|---|---|
| 1 | init_con | Ensures that all message inbox lengths, history lengths, and vote counts are non-negative, and each node has voted for at least one candidate. |
| 2 | common_msg_pool_prop | Ensures that all messages in the common message pool have valid sources and destinations, excluding dud (invalid) and net (network). |
| 3 | very_basic | Ensures that if a node is not a follower, it has voted for itself and has set its quorum correctly. |
| 4 | very_basic_1 | Ensures that none of the nodes have given their vote to the network (net). |
| 5 | very_basic_2 | Ensures that all received messages in each node's inbox come from valid sources (i.e., not net or dud). |
| 6 | very_basic_3 | Ensures that each node's history contains at least one record, and the first recorded vote is dud (indicating no prior vote). |
| 7 | very_basic_4 | Ensures that if a node changes its vote within the same term, its previous vote was dud (i.e., it must have been uninitialized before switching). |
| 8 | very_basic_5_0 | Ensures that Node0's last recorded history matches its current quorum, consensus state, and vote. |
| 9 | very_basic_5_1 | Ensures that Node1's last recorded history matches its current quorum, consensus state, and vote. |
| 10 | very_basic_5_2 | Ensures that Node2's last recorded history matches its current quorum, consensus state, and vote. |
| 11 | very_basic_6 | Ensures that all messages in each node's message history have valid sources (src  net, dud) and are correctly addressed to the respective node. |
| 12 | leader_criterion_0 | Ensures that if Node0 is a leader, it must have received votes from at least one other node (vote_count ¿ 1). |
| 13 | basic_1_0 | Ensures that Node0's term is at least 1 (no zero or negative terms). |
| 14 | basic_2_0 | Ensures that if Node0 has a history, its latest historical term matches its current term. |
| 15 | basic_3_0 | Ensures that if Node0 has a history, its latest recorded vote matches its current vote. |
| 16 | basic_4_0 | Ensures that if Node0 is a candidate, it must have received at least one vote and must have its own quorum entry set. |
| 17 | constant_id_0 | Ensures that all historical records of Node0 maintain the correct node ID (id0) and have a valid term (term ¿ 0). |
| 18 | monotone_term_0 | Ensures that Node0's term is non-decreasing over time (i.e., terms only increase or stay the same). |
| 19 | consistent_vote_given_0 | Ensures that within a given term, Node0 can only vote for one node (i.e., it doesn't change its vote within the same term). |
| 20 | another_consistent_vote_given_0 (commented out) | Ensures that if a node's term increases, its vote resets to dud. |
| 21 | another_consistent_vote_given_0 (commented out) | Ensures that if Node0 receives a vote response from Node2, all conditions for it to be valid are satisfied. |
| 22 | another_consistent_vote_given_0 (commented out) | Ensures that if Node2 is a follower and Node0 is a candidate in the same term, and Node2's quorum is set, then Node2 must have voted for Node0. |
| 23 | another_consistent_vote_given_0_con_1 | Ensures that if Node0's quorum status changes in history, it must have received a valid vote response message (vote_req_resp). |
| 24 | another_consistent_vote_given_0_con_2 | Ensures that if Node2's vote_given changes, it must have processed a vote request (vote_req) message from the candidate it voted for. |

Table 1: Description of Invariants

| # | Invariant Name | Description |
|---|---|---|
| 25 | leader_criterion_0 | If Node0 is a leader, it must have received more than one vote. |
| 26 | basic_1_0 | Node0's term is always at least 1. |
| 27 | basic_2_0 | If Node0 has a history, the last recorded term must match Node0's current term. |
| 28 | basic_3_0 | If Node0 has a history, the last recorded vote given must match Node0's current vote given. |
| 29 | basic_4_0 | If Node0 is a candidate, it must have received at least one vote and be part of a quorum. |
| 30 | constant_id_0 | Node0's ID remains constant in all its recorded history. |
| 31 | monotone_term_0 | Node0's term never decreases over time. |
| 32 | consistent_vote_given_0 | Within a given term, Node0 can vote for at most one candidate. |
| 33 | leader_criterion_1 | If Node1 is a leader, it must have received more than one vote. |
| 34 | basic_1_1 | Node1's term is always at least 1. |
| 35 | basic_2_1 | If Node1 has a history, the last recorded term must match Node1's current term. |
| 36 | basic_3_1 | If Node1 has a history, the last recorded vote given must match Node1's current vote given. |
| 37 | basic_4_1 | If Node1 is a candidate, it must have received at least one vote and be part of a quorum. |
| 38 | constant_id_1 | Node1's ID remains constant in all its recorded history. |
| 39 | monotone_term_1 | Node1's term never decreases over time. |
| 40 | consistent_vote_given_1 | Within a given term, Node1 can vote for at most one candidate. |
| 41 | leader_criterion_2 | If Node2 is a leader, it must have received more than one vote. |
| 42 | basic_1_2 | Node2's term is always at least 1. |
| 43 | basic_2_2 | If Node2 has a history, the last recorded term must match Node2's current term. |
| 44 | basic_3_2 | If Node2 has a history, the last recorded vote given must match Node2's current vote given. |
| 45 | basic_4_2 | If Node2 is a candidate, it must have received at least one vote and be part of a quorum. |
| 46 | constant_id_2 | Node2's ID remains constant in all its recorded history. |
| 47 | monotone_term_2 | Node2's term never decreases over time. |
| 48 | consistent_vote_given_2 | Within a given term, Node2 can vote for at most one candidate. |
| 49 | another_consistent_vote_given_0 | If Node0 receives a valid vote response, certain conditions must hold for the voter's state and term. |
| 50 | another_consistent_vote_given_0_con_1 | If a node has received a vote from another node, it must have processed a vote_req_resp message from that node. |
| 51 | another_consistent_vote_given_0_con_2 | If a node's vote_given changes, it must have processed a vote_req message. |
| 52 | another_consistent_vote_given_0_con_3 | If Node0 receives a vote_req_resp message from Node2, then Node2's vote_given must be id0 under certain conditions. |
| 53 | another_consistent_vote_given_final_0 | Node2's vote_given remains valid across certain term transitions. |

Table 2: Additional Invariants for Nodes

| # | Invariant Name | Description |
|---|---|---|
| 54 | another_consistent_vote_given_0_con_3 | If Node0 receives a vote_req_resp message from Node2 and meets certain conditions (matching terms, follower state, quorum agreement), then Node2 must have voted for Node0. |
| 55 | another_consistent_vote_given_final_0 | If Node2 has a message history and is a follower in the same term as Node0, then it must have given a valid vote. |
| 56 | leader_criterion_1 | If Node1 is a leader, it must have received more than one vote. |
| 57 | basic_1_1 | Node1's term is always at least 1. |
| 58 | basic_2_1 | If Node1 has a history, the last recorded term must match Node1's current term. |
| 59 | basic_3_1 | If Node1 has a history, the last recorded vote given must match Node1's current vote given. |
| 60 | basic_4_1 | If Node1 is a candidate, it must have received at least one vote and be part of a quorum. |
| 61 | constant_id_1 | Node1's ID remains constant in all its recorded history. |
| 62 | monotone_term_1 | Node1's term never decreases over time. |
| 63 | consistent_vote_given_1 | Within a given term, Node1 can vote for at most one candidate. |
| 64 | leader_criterion_2 | If Node2 is a leader, it must have received more than one vote. |
| 65 | basic_1_2 | Node2's term is always at least 1. |
| 66 | basic_2_2 | If Node2 has a history, the last recorded term must match Node2's current term. |
| 67 | basic_3_2 | If Node2 has a history, the last recorded vote given must match Node2's current vote given. |
| 68 | basic_4_2 | If Node2 is a candidate, it must have received at least one vote and be part of a quorum. |
| 69 | constant_id_2 | Node2's ID remains constant in all its recorded history. |
| 70 | monotone_term_2 | Node2's term never decreases over time. |
| 71 | consistent_vote_given_2 | Within a given term, Node2 can vote for at most one candidate. |
| 72 | bounded_vote_count | Each node's vote count is always between 1 and the total number of nodes + 1, and the latest recorded vote count must match the current vote count. |
| 73 | previous_records_vote_count | Every recorded vote count for Node1 must be between 1 and the total number of nodes + 1. |
| 74 | init_previous_records_con_1 | The initial record for Node1 must be a follower with term 1, a vote count of 1, and all quorum values set to false. |
| 75 | quorum_relation_con_1_1 | Node1's quorum relation holds if it has a quorum with at least two nodes (any two out of Node0, Node1, and Node2). |
| 76 | quorum_relation_con_1_2 | In every history entry for Node1, the quorum relation condition must hold (same as above, but for historical records). |
| 77 | quorum_relation_con_1_3 | The latest recorded quorum state for Node1 must match its current quorum state, and the quorum relation condition must hold. |
| 78 | quorum_relation_con_1_4 | If Node1 is a candidate in history, it must have quorum_1 set to true. |
| 79 | quorum_relation_con_0_1 | Node0's quorum relation must hold, meaning it must have a quorum with at least two nodes. |

Table 3: Additional Invariants for Nodes (continued)

| # | Invariant Name | Description |
|---|---|---|
| 80 | quorum_relation_con_0_2 | The quorum relation condition must hold for all historical records of Node0. |
| 81 | quorum_relation_con_0_3 | The latest recorded quorum state of Node0 must match its current state, and the quorum relation condition must hold. |
| 82 | quorum_relation_con_0_4 | If Node0 is a candidate in history, it must have quorum_0 set to true. |
| 83 | quorum_relation_con_2_1 | Node2's quorum relation must hold, meaning it must have a quorum with at least two nodes. |
| 84 | quorum_relation_con_2_2 | The quorum relation condition must hold for all historical records of Node2. |
| 85 | quorum_relation_con_2_3 | The latest recorded quorum state of Node2 must match its current state, and the quorum relation condition must hold. |
| 86 | quorum_relation_con_2_4 | If Node2 is a candidate in history, it must have quorum_2 set to true. |
| 87 | connect_vote_count_and_quorum_1 | If Node1 achieves quorum in a given term, then it must have received a vote from a node that had not voted before. |
| 88 | connect_vote_count_and_quorum_2 | If Node2 achieves quorum in a given term, then it must have received a vote from a node that had not voted before. |

Table 4: Additional Invariants for Nodes (continued)

| # | Invariant Name | Description |
|---|---|---|
| 89 | connect_vote_count_and_quorum_0 | Ensures that if a quorum relation changes in consecutive records with the same term (and nodes are not followers), at least one of quorum_1 or quorum_2 changes. |
| 90 | history_property_1 | Ensures that if a node is not a follower, its history length is greater than zero. |
| 91 | all_msg_history_property | Ensures that the length of the message history for each node is non-negative. |
| 92 | msg_property_2_0, msg_property_2_1, msg_property_2_2 | Ensures that all received messages in each node's inbox are addressed to the respective node. |
| 93 | msg_property_3_0, msg_property_3_1, msg_property_3_2 | Ensures that messages (except heartbeat messages) in each node's inbox are not sent to the same node that originated them. |
| 94 | msg_property_4 | Ensures that messages in the common message pool (except heartbeat messages) are not sent by a node to itself. |
| 95 | history_len_my_inbox | Ensures that the length of received messages in each node's inbox is within the range of its message history length. |
| 96 | vote_req_msg_vote_given_0 | Ensures that if a node has given a valid vote, it has received at least one vote request message from that node. |
| 97 | msg_property_1_0, msg_property_1_1, msg_property_1_2 | Ensures that all messages in a node's history (except the first one) are addressed to the respective node. |
| 98 | my_inbox_all_msg_history_1, my_inbox_all_msg_history_2 | Ensures that if a message exists in a node's inbox, a corresponding message with the same attributes exists in the node's full message history. |

Table 5: Additional Invariants for Nodes (continued)