```
package io.scalecube.cluster;

/**
 * Utility class which contains math computation on cluster properties.
 *
 * <p>NOTE: used for test purposes.
 */
public final class ClusterMath {

  private ClusterMath() {
    // Do not instantiate
  }

  /**
   * Returns gossipConvergencePercent.
   *
   * @param fanout fan out
   * @param repeatMult repeat multiplier
   * @param clusterSize cluster size
   * @param lossPercent precentage of lost msgs
   * @return gossipConvergencePercent
   */
  public static double gossipConvergencePercent(
      int fanout, int repeatMult, int clusterSize, double lossPercent) {
    double msgLossProb = lossPercent / 100.0;
    return gossipConvergenceProbability(fanout, repeatMult, clusterSize,
msgLossProb) * 100;
  }

  /**
   * Returns gossipConvergenceProbability.
   *
   * @param fanout fan out
   * @param repeatMult repeat multiplier
   * @param clusterSize cluster size
   * @param loss percentage of lost msgs
   * @return gossipConvergenceProbability
   */
  public static double gossipConvergenceProbability(
      int fanout, int repeatMult, int clusterSize, double loss) {
    double fanoutWithLoss = (1.0 - loss) * fanout;
    double spreadSize = clusterSize - Math.pow(clusterSize, -(fanoutWithLoss *
repeatMult - 2));
    return spreadSize / clusterSize;
  }

  /**
   * Returns maxMessagesPerGossipTotal.
   *
   * @param fanout fan out
   * @param repeatMult repeat multiplier
   * @param clusterSize cluster size
   * @return maxMessagesPerGossipTotal
   */
  public static int maxMessagesPerGossipTotal(int fanout, int repeatMult, int
clusterSize) {
    return clusterSize * maxMessagesPerGossipPerNode(fanout, repeatMult,
clusterSize);
  }
```

```
  /**
   * Returns maxMessagesPerGossipPerNode.
   *
   * @param fanout fan out
   * @param repeatMult repeat multiplier
   * @param clusterSize cluster size
   * @return maxMessagesPerGossipPerNode
   */
  public static int maxMessagesPerGossipPerNode(int fanout, int repeatMult, int
clusterSize) {
    return fanout * repeatMult * ceilLog2(clusterSize);
  }

  /**
   * Returns gossipDisseminationTime.
   *
   * @param repeatMult repeat multiplier
   * @param clusterSize cluster size
   * @param gossipInterval gossip interval
   * @return gossipDisseminationTime
   */
  public static long gossipDisseminationTime(int repeatMult, int clusterSize, long
gossipInterval) {
    return gossipPeriodsToSpread(repeatMult, clusterSize) * gossipInterval;
  }

  /**
   * Returns gossipTimeoutToSweep.
   *
   * @param repeatMult repeat multiplier
   * @param clusterSize cluster size
   * @return gossipTimeoutToSweep
   */
  public static long gossipTimeoutToSweep(int repeatMult, int clusterSize, long
gossipInterval) {
    return gossipPeriodsToSweep(repeatMult, clusterSize) * gossipInterval;
  }

  /**
   * Returns gossipPeriodsToSweep.
   *
   * @param repeatMult repeat multiplier
   * @param clusterSize cluster size
   * @return gossipPeriodsToSweep
   */
  public static int gossipPeriodsToSweep(int repeatMult, int clusterSize) {
    int periodsToSpread = gossipPeriodsToSpread(repeatMult, clusterSize);
    return 2 * (periodsToSpread + 1);
  }

  /**
   * Returns gossipPeriodsToSpread.
   *
   * @param repeatMult repeat multiplier
   * @param clusterSize clsuter size
   * @return gossipPeriodsToSpread
   */
  public static int gossipPeriodsToSpread(int repeatMult, int clusterSize) {
```

```
    return repeatMult * ceilLog2(clusterSize);
  }

  /**
   * Returns suspicionTimeout.
   *
   * @param suspicionMult suspect multiplier
   * @param clusterSize cluster size
   * @param pingInterval ping interval
   * @return suspicionTimeout
   */
  public static long suspicionTimeout(int suspicionMult, int clusterSize, long
pingInterval) {
    return suspicionMult * ceilLog2(clusterSize) * pingInterval;
  }

  /**
   * Returns ceilLog2.
   *
   * @param num num
   * @return ceil(log2(n + 1))
   */
  public static int ceilLog2(int num) {
    return 32 - Integer.numberOfLeadingZeros(num);
  }
}
```