```java
//******************************************************************************
//
// File:    Router.java
// Package: ---
// Unit:    Class Router
//
//******************************************************************************

import edu.rit.sim.Simulation;
import edu.rit.util.AList;
import edu.rit.util.Random;

/**
 * Class models a router's behavior where packets are transmitted on a
 * preferred link if that link is available, otherwise a secondary link is
 * chosen at random until an available link is found. If no secondary links
 * are available, the packet is dropped.
 *
 * @author Jimi Ford (jhf3617)
 * @version 5-2-2015
 */
public class Router extends Routable {

    // private data members

    private final Random prng;
    private Link primary;
    private int dropCount;
    private int receiveCount;
    private int reRouteCount;
    private final AList<Link> secondary;

    /**
     * Construct a router object
     *
     * @param prng the pseudorandom number generator to use for choosing what
     * secondary routables to use
     * @param sim the simulation object this router should be associated with
     */
    public Router(Random prng, Simulation sim) {
        super(sim);
        this.prng = prng;
        this.dropCount = 0;
        this.receiveCount = 0;
        this.reRouteCount = 0;
        this.secondary = new AList<Link>();
    }

    /**
     * Set the primary link this router should prefer to send its received
     * packets on
     *
     * @param link the link to prioritize
     */
    public void setPrimary(Link link) {
        this.primary = link;
    }

    /**
     * add a secondary link to the list of secondary links
     *
     * @param link the link to add
     */
    public void addSecondary(Link link) {
        this.secondary.addLast(link);
    }

    /**
     * Called when this routable object finished receiving a packet on a certain
     * link
     * @param packet the packet this object received
     * @param link the link that the packet was received on
     */
    public void receivePacket(final Packet packet, final Link l) {
        l.open();
        Link link = null;
        ++receiveCount;
        boolean goodToGo = false;
        if(primary.ready()) {
            goodToGo = true;
            link = primary;
        } else if(secondary.size() > 0) {

            int[] indices = ShuffleHelper.shuffledArray(prng, secondary.size());
            for(int i = 0; i < indices.length && !goodToGo; i++) {
                link = secondary.get(indices[i]);
                if(link.ready()) {
                    goodToGo = true;
                    ++reRouteCount;
                }
            }
        }
        if(goodToGo) {
            startSending(packet, link);
        } else {
            // drop packet
            ++dropCount;
        }
    }

    /**
     * Get the fraction of packets that this router dropped
     *
     * @param totalPacketCount the total number of packets generated in the
     * simulation
     * @return a number between 0 and 1
     */
    public double dropFraction(int totalPacketCount) {
        return ((double)this.dropCount)/(double)totalPacketCount;
    }

    /**
     * Get the fraction of the packets that the router had to re-route along
     * a secondary route
     */
    public double reRouteFraction() {
        return receiveCount == 0 ? 0 :
            ((double)reRouteCount)/(double)receiveCount;
    }
}

```