```
Orbis
=====

Orbis is a tool for generating analyzing, scaling and generating
realistic annotated topologies.

Dependencies
=================

Orbis was developed using Boost version 1.32.  It requires the Boost
Graph Library as well as the Boost "program_options" library.  The
latter can be downloaded at:

http://www.boost.org/doc/html/program_options.html


Components
====================

dkRewire: dkRewire takes as input a graph file and a k where k = {1,
2, 3}.  dkRewire performs a number of specified k-preserving rewirings
on the input graph, then outputs a randomly rewired graph whose dk
distribution is maintained.


dkDist: dkDist takes as input a graph file and a k, where k = {1, 2}
and gives out the 1k or 2k distribution of that graph.  A 1k
distribution has the following 2-column format:

   (degree1)  (# of nodes with degree1)
   (degree2)  (# of nodes with degree2)
      ...

A 2k distribution has the following 3-column format :

   (degree_i) (degree_j) (# of edges between nodes of degree_i and
              degree_j)


dkTopoGen1k/2k: dkTopoGen generates a random topology based given either a
1k or 2k distribution.



dkRescale1k/dkRescale2k: given a 1k or 2k distribution, dkRescale will
rescale that distribution up to a specified number of nodes.



dkMetrics: calculate various graph metrics for an input topology.
Currently the following scalar metrics are implemented :
  - # of nodes
  - # of edges
  - average degree
  - assortativity
  - average clustering
  - average distance

Also, the following distributions are available :
  - distance
  - betweenness



Source files summary
=====================

dkdist
  |
  |---------- dkDist.cc

dkRewire
  |
  |------- dkRewire.cc - main driver
  |------- randomizeGraph1k.h - algorithms for 1k random rewiring
  |------- randomizeGraph2k.h - algorithms for 2k random rewiring
  |------- randomizeGraph3k.h - algorithms for 3k random rewiring

dkMetrics
  |
  |------ dkMetrics.cc  - main driver
  |------ dkMetrics.h - algorithms for calculating various graph
          metrics

dkTopoGen
  |
  |------ dkTopoGen0k.cc - main driver for 0k topology generation
  |------ dkTopoGen1k.cc - main driver for 1k topology generation
  |------ dkTopoGen1k.h - algorithms for  1k topology generation
  |------ dkTopoGen2k.cc - main driver for 2k topology generation
  |------ dkTopoGen2k.h - algorithms for  12 topology generation

dkRescale
  |
  |------ dkRescale.cc - main driver for distribution rescaling.
          Algorithms for performing 1k and 2k rescaling topology are
          all here.

General utility
  |
  |----- dkUtils.h / dkUtils.cc - general utility methods. I/O,
  |      distribution conversions etc.
  |
  |----- scaleTopology - wrapper script that extracts and scales
     distributions, then generates a topology from the scaled
     distribution.





Example usage
====================

Suppose we have an input topology with 1000 nodes, and we want to
rescale using its 2k distribution to 2000 nodes.  The first step is to
```

```
extract this distribution from the input topology, which can be
achieved with the following:

Given: an input topology "input.topo" in INET format

./dkDist -k 2 -i input.topo > input.2k

Next we want to rescale the 1k distribution to 2000 nodes:

./dkRescale -k 2 -i input.2k -n 2000 > input.2k.rescaled

Finally we randomly generate a new topology from our scaled 2k
distribution:

./dkTopoGen2k -i input.2k.rescaled > input.rescaled.topo



The above steps are condensed into a single script
"scaleTopology.bash", which automates the above steps.
scaleTopology.bash has the following parameters:

./scaleTopology.bash <input graph> <{0,1,2}> <target # nodes> <r?>
f
The first parameter, <input graph>, is an input graph in INET format.
The second parameter defines which dk distribution to use (0k, 1k or
2k).  The third parameter specifies how many nodes the target topology
should have.  Finally the last parameter specifies whether or not the
input topology is a router topology.  For router topologies, a maximum
degree of 330 is enforced.

Using filenames from the example above, scaleTopology.bash can be used
as follows in order to generated a scaled topology:

./scaleTopology input.topo 2 2000 > input.2k.rescaled
```