```ruby
#!/usr/bin/ruby

require 'file/tail'
require 'mechanize'

if ARGV.count != 2 then abort('actuator.rb␣URL␣FILE') end

class MMechanize < Mechanize
        def initialize(url)
                @url = url
                @sync = Mutex.new
                super()
        end
        def get(params = {})
                @sync.synchronize {
                        super(@url, params)
                }
        end
end

def add_node(id, label, is_app = true)
        llabel = /([^\.\/]+)$/.match(label)[1]
        $agent.get({ 'q' => 'an', 'id' => id })
        $agent.get({ 'q' => 'ana', 'id' => id, 'key' => 'ui.label', 'value' => llabel })
        $agent.get({ 'q' => 'ana', 'id' => id, 'key' => 'ui.class', 'value' => is_app ? 'app' : 'service'
end

def add_edge(idf, idt)
        id = "#{idf}E#{idt}"
        idx = $edges.find_index(id)
        unless idx
                $edges.push WeightedElement.new(id)
                $agent.get({ 'q' => 'ae', 'id' => id, 'from' => idf, 'to' => idt, 'directed' => 1 })
        else
                $edges[idx].weight += 1
        end
        edge = idx ? $edges[idx] : $edges.last
        $agent.get({ 'q' => 'aea', 'id' => id, 'key' => 'ui.label', 'value' => "-#{edge.weight}-" })
        $agent.get({ 'q' => 'aea', 'id' => id, 'key' => 'weight', 'value' => edge.weight })
end

class Domain
        attr_reader :id
        @@count = 1
        def initialize(name)
                @name = name
                @services = Array.new
                @id =  @@count
                @@count += 1
                add_node(@id, @name)
        end
        def transaction_received(toserv)
                idx = @services.find_index(toserv)
                unless idx
                        @services.push(toserv)
                        sid = "#{@id}.#{@services.count}"
                        add_node(sid, toserv, false)
                else
                        sid = "#{@id}.#{idx␣+␣1}"
                end
                add_edge(@id, sid)
        end
        def ==(s)
                @name == s
        end
end

class WeightedElement
        attr_accessor :weight
        def initialize(id)
                @weight = 1
                @id = id
        end
        def ==(s)
```

```
                @id == s
        end
end

class DomainCollection
        attr_reader :domains
        def initialize
                @domains = Array.new
        end
        def push(dom)
                idx = @domains.find_index(dom)
                return @domains[idx] if idx
                @domains.push Domain.new(dom)
                @domains.last
        end
end

class AuditMonitor < File
        include File::Tail
        attr_reader :collection
        def initialize(filename)
                @reg = /^(ipc android_binder_transaction )([^ ]+) (.+)$/
                @collection = DomainCollection.new
                super
        end
        def push_line(line)
                return if line =~ /^#/
                if (line =~ /^</)
                        @domain = line
                        return
                end
                return if not line =~ @reg
                process(@domain, $3, $2)
        end
        def process(from, dest, service)
                dfrom = @collection.push(from)
                ddest = @collection.push(dest)
                add_edge(dfrom.id, ddest.id)
                ddest.transaction_received(service)
                $ipc_processed += 1
        end
end

def algo
        puts 'Centrality...'
        $stdin.gets
        $agent.get({ 'q' => 'centrality' })
        puts 'Weight␣Edge-Coloring...'
        $stdin.gets
        $agent.get({ 'q' => 'edgecoloring' })
        puts 'Spanning␣Tree...'
        $stdin.gets
        $agent.get({ 'q' => 'spantree' })
        puts 'DONE'
end

$agent = MMechanize.new(ARGV[0])
$edges = Array.new
io = AuditMonitor.new(ARGV[1])

$ipc_processed = 0
Signal.trap('SIGUSR1') {
        puts "IPC␣Processed:␣#{$ipc_processed}"
}

Thread.new { algo }
io.tail { |line| io.push_line(line.chomp!) }
```

```
package androidtomoyo;

import java.awt.event.MouseWheelEvent;
import java.awt.event.MouseWheelListener;
import java.io.BufferedReader;
import java.io.IOException;
```

```java
import java.io.InputStreamReader;

import org.graphstream.algorithm.BetweennessCentrality;
import org.graphstream.algorithm.Prim;
import org.graphstream.graph.Edge;
import org.graphstream.graph.Node;
import org.graphstream.graph.implementations.MultiGraph;
import org.graphstream.stream.file.FileSinkDGS;
import org.graphstream.stream.file.FileSinkImages;
import org.graphstream.stream.file.FileSinkImages.LayoutPolicy;
import org.graphstream.stream.file.FileSinkImages.OutputPolicy;
import org.graphstream.stream.file.FileSinkImages.OutputType;
import org.graphstream.stream.file.FileSinkImages.Quality;
import org.graphstream.stream.file.FileSinkImages.RendererType;
import org.graphstream.stream.file.FileSinkImages.Resolutions;
import org.graphstream.stream.file.FileSourceDGS;
import org.graphstream.ui.swingViewer.View;
import org.graphstream.ui.swingViewer.Viewer;
import org.graphstream.ui.swingViewer.util.Camera;

public class AndroidGraph implements MouseWheelListener {

        private static int base = 8083;
        private Camera cam;
        private MultiGraph graph;
        private FileSinkDGS dgs;

        public void start(String id) throws Exception {
                start(id, true);
        }

        public void start(String id, boolean rubyfeed) throws Exception {
                System.setProperty("org.graphstream.ui.renderer", "org.graphstream.ui.j2dviewer.J2DGraphRe

                graph = new MultiGraph(id);
                graph.addAttribute("ui.stylesheet", "url('file://" + System.getProperty("user.dir") + "/gr
                graph.addAttribute("ui.default.title", id);
                graph.addAttribute("ui.quality");
                graph.addAttribute("ui.antialiasing");

                Viewer viewer  = new Viewer(graph, Viewer.ThreadingModel.GRAPH_IN_ANOTHER_THREAD);
                viewer.enableAutoLayout();
                //viewer.disableAutoLayout();
                View view = viewer.addDefaultView(true);
                view.addMouseWheelListener(this);
                cam = view.getCamera();

                if (rubyfeed) {
                        dgs = new FileSinkDGS();
                        dgs.begin("android_binder.dgs");
                        HTTPSourceExtended hs = new HTTPSourceExtended(id, base++, this);
                        hs.addSink(graph);
                        hs.addSink(dgs);
                        hs.start();
                } else { /* DGS Feed */
                        FileSourceDGS sdgs = new FileSourceDGS();
                        FileSinkImages fsi = new FileSinkImages(OutputType.PNG, Resolutions.HD720);
                        fsi.setRenderer(RendererType.SCALA);
                        fsi.setStyleSheet("url('file://" + System.getProperty("user.dir") + "/graphstyle.c
                        // COMPUTED_FULLY_AT_NEW_IMAGE -- SCATTA TROPPO
                        // COMPUTED_IN_LAYOUT_RUNNER -- OK
                        fsi.setLayoutPolicy(LayoutPolicy.COMPUTED_ONCE_AT_NEW_IMAGE);
                        fsi.setQuality(Quality.HIGH);
                        fsi.setOutputPolicy(OutputPolicy.BY_EDGE_ADDED_REMOVED);
                        fsi.begin("/mnt/data/tmp/gs_");
                        sdgs.addSink(graph);
                        graph.addSink(fsi);
                        sdgs.readAll("android_binder.dgs");
                        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
                        fsi.setOutputPolicy(OutputPolicy.BY_EVENT);
                        graph.stepBegins(0);
                        System.out.println("\n***Centrality...");
                        //br.readLine();
                        Thread.sleep(10000);
                        doCentrality();
                        graph.stepBegins(1);
                        System.out.println("\n***Edge␣Coloring...");
                        //br.readLine();
                        doEdgeColoring();
                        graph.stepBegins(2);
                        System.out.println("\n***Spanning␣Tree...");
                        //br.readLine();
                        doSpanTree();
                        graph.stepBegins(3);
                        fsi.end();
                }
        }

        public void mouseWheelMoved(MouseWheelEvent e) {
                cam.setViewPercent(cam.getViewPercent() + e.getUnitsToScroll() * 0.01);
        }

        private float color_normalize(float n) {
                float nn = n * 15;
                if (nn > 1) return 1;
                return nn;
        }

        private float size_normalize(float n) {
                int nn = (int) (n * 12);
                if (nn == 0) return 1;
                return nn;
        }

        public void doCentrality() {
                BetweennessCentrality bcb = new BetweennessCentrality();
                //bcb.setWeightAttributeName("weight");
                bcb.init(graph);
                bcb.compute();
                float max = 0;
                for (Node n : graph) {
                        float cur = Float.parseFloat(n.getAttribute("Cb").toString());
                        if (cur > max) max = cur;
                }
                for (Node n : graph) {
                        float cur = Float.parseFloat(n.getAttribute("Cb").toString());
                        n.setAttribute("ui.color", color_normalize(cur/max));
                }
        }

        public void doSpanTree() {
                int count = 0;
                for (Edge e : graph.getEachEdge())
                        count++;
                System.out.println("Aristas␣antes:␣" + count);
                Prim prim = new Prim("ui.class", "intree", "notintree");
                prim.init(graph);
                prim.compute();
                count = 0;
                for (Edge e : graph.getEachEdge())
                        if (e.getAttribute("ui.class").toString().equals("intree"))
                                count++;
                System.out.println("Aristas␣despues:␣" + count);
        }

        public void doEdgeColoring() {
                float max = 0;
                for (Edge e : graph.getEachEdge()) {
                        float cur = Float.parseFloat(e.getAttribute("weight").toString());
                        if (cur > max) max = cur;
                }
                for (Edge e : graph.getEachEdge()) {
                        float cur = Float.parseFloat(e.getAttribute("weight").toString());
                        e.setAttribute("ui.color", color_normalize(cur/max));
                        e.setAttribute("ui.size", size_normalize(cur/max));
                }
        }
```

```java
        public void doDGSFlush() throws IOException {
                        dgs.flush();
                        dgs.end();
        }
}


package androidtomoyo;

import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.net.InetSocketAddress;
import java.net.URLDecoder;
import java.util.HashMap;
import java.util.LinkedList;

import org.graphstream.stream.SourceBase;

import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpHandler;
import com.sun.net.httpserver.HttpServer;

public class HTTPSourceExtended extends SourceBase {

        protected final HttpServer server;

        private AndroidGraph graph;

        public HTTPSourceExtended(String graphId, int port, AndroidGraph graph) throws IOException {
                super(String.format("http://%s", graphId));

                server = HttpServer.create(new InetSocketAddress(port), 4);
                server.createContext(String.format("/%s/edit", graphId),
                                new EditHandler());

                this.graph = graph;

        }

        public void start() {
                server.start();
        }

        public void stop() {
                server.stop(0);
        }

        private class EditHandler implements HttpHandler {

                public void handle(HttpExchange ex) throws IOException {
                        HashMap<String, Object> get = GET(ex);
                        Action a;

                        try {
                                a = Action.valueOf(get.get("q").toString().toUpperCase());
                        } catch (Exception e) {
                                error(ex, "invalid␣action");
                                return;
                        }

                        switch (a) {
                        case AN:
                                HTTPSourceExtended.this.sendNodeAdded(sourceId, get.get("id")
                                                .toString());
                                break;
                        case CN:
                                break;
                        case ANA:
                                HTTPSourceExtended.this.sendNodeAttributeAdded(sourceId, get.get("id")
                                                .toString(), get.get("key").toString(), get.get("value"));
                                break;
                        case AEA:
                                HTTPSourceExtended.this.sendEdgeAttributeAdded(sourceId, get.get("id")
                                                .toString(), get.get("key").toString(), get.get("value"));
                                break;
                case DN:
                        HTTPSourceExtended.this.sendNodeRemoved(sourceId, get.get("id")
                                        .toString());
                        break;
                case AE:
                        HTTPSourceExtended.this.sendEdgeAdded(sourceId, get.get("id")
                                        .toString(), get.get("from").toString(), get.get("to")
                                        .toString(), get.containsKey("directed"));
                        break;
                case CE:
                        break;
                case DE:
                        HTTPSourceExtended.this.sendEdgeRemoved(sourceId, get.get("id")
                                        .toString());
                        break;
                case CG:
                        break;
                case ST:
                        HTTPSourceExtended.this.sendStepBegins(sourceId, Double.valueOf(get
                                        .get("step").toString()));
                        break;
                case CENTRALITY:
                        graph.doCentrality();
                        break;
                case SPANTREE:
                        graph.doSpanTree();
                        break;
                case EDGECOLORING:
                        graph.doEdgeColoring();
                        break;
                case DGSFLUSH:
                        graph.doDGSFlush();
                        break;
                }

                ex.sendResponseHeaders(200, 0);
                ex.getResponseBody().close();
        }
}

protected static void error(HttpExchange ex, String message)
                throws IOException {
        byte[] data = message.getBytes();

        ex.sendResponseHeaders(400, data.length);
        ex.getResponseBody().write(data);
        ex.getResponseBody().close();
}

@SuppressWarnings("unchecked")
protected static HashMap<String, Object> GET(HttpExchange ex) {
        HashMap<String, Object> get = new HashMap<String, Object>();
        String[] args = ex.getRequestURI().getRawQuery().split("[&]");

        for (String arg : args) {
                String[] kv = arg.split("[=]");
                String k, v;

                k = null;
                v = null;

                try {
                        if (kv.length > 0)
                                k = URLDecoder.decode(kv[0], System
                                                .getProperty("file.encoding"));

                        if (kv.length > 1)
                                v = URLDecoder.decode(kv[1], System
                                                .getProperty("file.encoding"));

                        if (get.containsKey(k)) {
                                Object o = get.get(k);

                                if (o instanceof LinkedList<?>)
```

```
                                                        ((LinkedList<Object>) o).add(v);
                                                else {
                                                        LinkedList<Object> l = new LinkedList<Object>();
                                                        l.add(o);
                                                        l.add(v);
                                                        get.put(k, l);
                                                }
                                        } else {
                                                get.put(k, v);
                                        }
                                } catch (UnsupportedEncodingException e) {
                                        e.printStackTrace();
                                }
                        }

                        return get;
                }

                static enum Action {
                        AN, CN, DN, AE, CE, DE, CG, ST, ANA, AEA, CLEAR,
                        CENTRALITY, SPANTREE, EDGECOLORING, DGSFLUSH
                }
        }

        package androidtomoyo;

        public class Main {

                public static void main(String[] args) throws Exception {
                        new AndroidGraph().start("Android", false);
                        //new AndroidGraph().start("Binder_IPC");
                }

        }
```