DipZoom: an Open Ecosystem for Network Measurements

Sipat Triukose, Zhihua Wen, Adam Derewecki, and Michael Rabinovich
Electrical Engineering and Computer Science Department
Case Western Reserve University
Cleveland, OH 44106
{sipat.triukose,zhihua.wen,adam.derewecki,michael.rabinovich}@case.edu

This demonstration introduces an operational prototype of a novel platform for Internet measurements. Our system, called DipZoom (for "Deep Internet Performance Zoom-in"), attempts to make high-quality network measurements available and accessible to broad technical community, and to facilitate the inclusion of complex network measurement experiments into undergraduate curriculum. DipZoom is based on two key ideas. First, recognizing the difficulty any single provider would face in building a platform representative of the scale and diversity of the Internet, DipZoom implements a *matchmaking service* instead to bring together experimenters in need of measurements with external measurement providers in a peer-to-peer manner. Thus, DipZoom leverages experimenters themselves in deploying a large number and variety of measuring points. Second, DipZoom provides a coherent view over the entire network of measurement providers, which is accessible and controllable from a participant's local machine. Using DipZoom API, complex multi-step measurements involving potentially globally distributed measuring points, could be implemented and executed merely as a local Java application.

The DipZoom platform consists of measuring points (MPs), clients, and the core that implements matchmaking. In the peer-to-peer spirit, measurement requester software is bundled with measurement provider, so in order to request measurements from a computer, one must be willing to also provide measurements from this computer to others. Measurement points, however, can exist by themselves to allow their deployment on servers and other devices without direct access to end-users. The client part includes a DipZoom client library, which implements the DipZoom API, and a graphical front-end, which provides an exploratory access to the system and serves as an example application built on top of the API.

The DipZoom prototype has been operational since January and typically has between 100-150 online MPs. It currently supports ping, traceroute, wget, nslookup, dig, host, and curl measurements (although not every MP supports every measurement). Peer software is publicly available from [1], and our initial experiences are reported in [2].

Our demo will illustrate the DipZoom capabilities by performing the following tasks on a live system: we will demonstrate the ease of deploying new measuring points by downloading and installing DipZoom peer software on a laptop or other Java-capable device of volunteer delegates; we will demonstrate the exploratory access to the totality of the on-line MPs from the graphical front-end; and we will demonstrate how DipZoom lowers the barrier of entry for performing complex measurement experiments by executing a sample experiment live from our laptop.

The DipZoom graphical client (Figure 1) has an area where user can specify the filters for the MPs that suit his or her requirements, an area where he or she can describe the measurement request, and an area that displays the list of currently on-line MPs satisfying the filters. The typical user actions include specifying the filters in the top area, clicking the "Get MP List" button to retrieve the online MPs in the middle area, and then selecting the MPs to be used, specifying the measurement details in the bottom area and submitting the request by clicking the "Send Request" button. Upon retrieving and analyzing the results, the user typically iterates through the above sequence of action, zooming in specific MP sets and measurement types.

To illustrate the use of DipZoom API, we will present and execute live an application that investigates the quality of Akamai server selection [3] (Figure 2). The application represents a multi-step distributed experiment involving (1) discovering several Akamai edge servers using a DNS lookup from two MPs belonging to two different regions and (2) comparing the performance of the download of the same object from one of the MPs using all the discovered edge servers. The code shows how to complete this complex study as a locally executed Java application. The application accepts as the arguments the two regions to be investigated and returns the performance of the best and Akamai-selected edge server for an MP from the first region. While this code uses only two MPs for compactness, it is trivial to extend the code to a larger MP set.

References

[1] http://dipzoom.case.edu
[2] Zh. Wen, S.Triukose, and M. Rabinovich. Facilitating Focused Internet Measurements. To appear in Sigmetrics'2007. A preliminary draft is available at http://dipzoom.case.edu/files/documents/dipzoom_focus.pdf
[3] http://www.akamai.com/

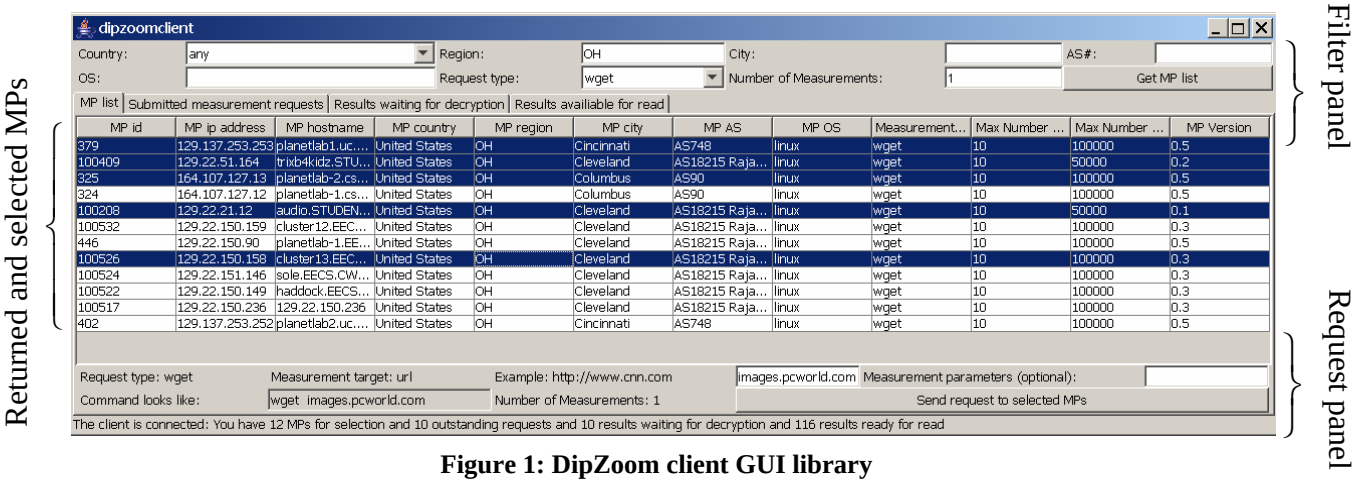

Figure 1: DipZoom client GUI library

```
package edu.cwru.netw.dipzoom;
import java.util.*;
public class Demo {
  public static void main(String[] args) {
   DipzoomClientLibrary dcl = new DipzoomClientLibrary();
   if (dcl.login("login.xml") != 1 ) return;
   // Take two regions parameter from command-line
   String region1 = args[0]; String region2 = args[1];
   Hashtable<String, String> mpParameters = new
                Hashtable<String,String>();

   // Get a list of measuring points from each region supports.
   // both nslookup and wget
   mpParameters.put("region", region1); // start with region1
   mpParameters.put("measurementType", Constants.NSLOOKUP );
   ArrayList<MeasuringPoint> mpRegion1 =
                dcl.getMeasuringPointList(mpParameters);
   mpParameters.put("measurementType", Constants.WGET);
   ArrayList<MeasuringPoint> mpRegion1Wget =
                dcl.getMeasuringPointList(mpParameters);
   mpRegion1.retainAll(mpRegion1Wget);  // intersect two lists of MP
   mpParameters.put("region", region2);  // change the region to region2
   ArrayList<MeasuringPoint> mpRegion2Wget =
                dcl.getMeasuringPointList(mpParameters);
   mpParameters.put("measurementType", Constants.NSLOOKUP);
   ArrayList<MeasuringPoint> mpRegion2 =
                dcl.getMeasuringPointList(mpParameters);
   mpRegion2.retainAll(mpRegion2Wget);  // Intersect two lists of MP
   // Consturct a list containing a measuring point from each region
   ArrayList<MeasuringPoint> measuringPoints = new ArrayList(
                Arrays.asList(new MeasuringPoint[]
                {mpRegion1.get(0),mpRegion2.get(0)}));

   // Obtain IP addresses of CDN-selected servers for both measuring points.
   MeasurementRequest nsLookupRequest = new MeasurementRequest();
   Hashtable<String,String> parameters = new Hashtable<String,String>();
   parameters.put("target", "firm-x.com");
   parameters.put("type", Constants.NSLOOKUP);
   nsLookupRequest.setParameters(parameters);
   ArrayList<MeasurementResult> pendingResults =
                dcl.sendRequest(nsLookupRequest, measuringPoints);
   ArrayList<NsLookupResult> finishedResults =  new
                ArrayList<NsLookupResult>();
while (pendingResults.size() > 0) // wait for the results
    for (int i = 0; i < pendingResults.size(); i++)
      if (pendingResults.get(i).getTransactionStatus() ==
                    Constants.RESULT_RECEIVED) {
       finishedResults.add((NsLookupResult)pendingResults.get(i));
       pendingResults.remove(i);
      }

   // Compare the performance provided by the discovered CDN
   // servers to   one of    the measuring points.
   // Select the measuring point
   MeasuringPoint measuringPoint = measuringPoints.get(0);
   String cdnSelectedServerIP = "";
  ArrayList<MeasurementRequest> wgetRequests = new
   ArrayList<MeasurementRequest>();
   for (NsLookupResult nsLookupResult : finishedResults) {
    parameters = new Hashtable<String,String>();
    parameters.put("type", Constants.WGET);
    parameters.put("parameter", "--header Host:images.pcworld.com");
    parameters.put("target",nslookupResult.getIPAddr()+
     "/images/header/logo_hd.jpg");
    MeasurementRequest wgetRequest = new MeasurementRequest();
    wgetRequest.setParameters(parameters);
    wgetRequests.add(wgetRequest);
    // Find the CDN-selected server for the measuring point
    if (nsLookupResult.getMeasuringPointID() == measuringPoint.getID())
     cdnSelectedServerIP = nsLookupResult.getIPAddr();
   }
   // Send wget requests for all CDN servers to the selected measuring point
   ArrayList<MeasurementResult> pendingWgetResults=
              dcl.sendRequest(wgetRequests,measuringPoint);
   ArrayList<WgetResult> wgetResults = new ArrayList<WgetResult>();
   while (pendingWgetResults.size() > 0)    // wait for all the results
    for (int i = 0; i < pendingWgetResults.size(); i++)
     if (pendingWgetResults.get(i).getTransactionStatus()==
      Constants.RESULT_RECEIVED) {
      finishedWgetResults.add((WgetResult)pendingWgetResults.get(i));
      pendingWgetResults.remove(i);
     }

   // findFastestServer and findCDNSelectedServer are user-defined functions,
   // They are not part of DipZoom API.
   WgetResult fastestServer = findFastestServer(wgetResults);
   System.out.printf("The fastest server to reach firm-x.com is %s with a time of
    %s.", fastestServer.getIp(), fastestServer.getCompletionTime());
   WgetResult cdnSelectedServer=
    findCDNSelectedServer(wgetResults,cdnSelectedServerIP);
   System.out.printf("The CDN server to reach firm-x.com is %s with a time of
    %s.",  cdnSelectedServerIP.getIp(),
  cdnSelectedServerIP.getCompletionTime());
  }
 }
```

Figure 2: Demo application using DipZoom API