

MITATE : Mobile Internet Testbed for Application Traffic Experimentation

mitate@cs.montana.edu

<http://mitate.cs.montana.edu>

User Manual

v 1.0

Content

1. MITATE Overview
2. Configuring MITATE Experiments
 - 2.1 Traffic Definitions
 - 2.2 Defining a Transfer
 - 2.3 Defining an Explicit Content
 - 2.4 Defining a Criteria
 - 2.5 Configuring a MITATE Experiment to Send an HTTP GET Packet on Port 80 of a Web Server
 - 2.6 Configuring a MITATE Experiment to Perform DNS Lookup and Get a Response Back
 - 2.7 Configuring a MITATE Experiment to Send Packets of Different Message Sizes
 - 2.8 Configuring a MITATE Experiment to Send Packets with Transmission Rates (Transmission Intervals)
3. MITATE Command Line API
 - 3.1 Login to the API
 - 3.2 Download Script to Initialize Local MySQL Instance
 - 3.3 Validate XML Configuration File
 - 3.4 Get Experiment Cost
 - 3.5 Check User's Available Credits
 - 3.6 Upload an Experiment as XML
 - 3.7 Update Experiment Permissions
 - 3.8 Query Experiment Data
 - 3.9 Delete an Uploaded Experiment
 - 3.10 Logout from the API

1. MITATE Overview

The technical problem we address is a lack of a programmable testbed for mobile application prototyping in production cellular networks. We have identified two challenges to building such a testbed. First, the personal nature of mobile devices creates user concerns over privacy, accountability for actions of foreign code being prototyped, and abuse of limited data plan and battery resources. Striking a balance between a flexible application prototyping environment and the safe execution of foreign code has been a difficult problem even in the more permissive wired environment. Second, because mobile battery and data plan resources are limited, testbed participants need adequate incentives to share them. Difficulty in enlisting mobile users has limited measurement studies to small samples, high cost of testbeds based on dedicated hardware, and collection of only high level network performance metrics.

MITATE is a Mobile Internet Testbed for Application Traffic Experimentation made possible by novel solutions to the problems of security and mobile resource sharing. MITATE is unique in that it allows programmable application traffic experiments between mobile hosts and backend server infrastructure. MITATE provides strong client security by separating application code execution from traffic generation. MITATE also provides incentives and protections for mobile resource sharing through tit-for-tat mechanisms.

MITATE's specialized traffic experiments can help developers answer questions crucial to mobile application design such as: What is the largest game state update message that can be reliably delivered under 100 ms?," Does my application traffic need to contend with traffic shaping mechanisms?," or Which CDN provides fastest downloads through a particular mobile service provider's network peering points?"

2. Configuring MITATE Experiments

This section will elaborate the steps required in creating traffic definitions. These traffic definitions will serve as an input to the MITATE system. In order to submit these definitions to the MITATE backend servers and to start the experiments from your mobile device, you will be required to form a well defined XML file.

This section contains explanations for defining an experiment which consists of transaction, transfer, content and criteria definitions. Below are a few examples which could help to get started immediately, if you are new to MITATE. These examples contains traffic definitions for executing experiments such as sending a well formed HTTP GET, DNS lookup request.

2.1 Traffic Definitions

```
<transaction count="10">
  <criteria>
    <criteriaid>criteria1</criteriaid>
  </criteria>
  <transfers>
    <transfer repeat="1" delay="10">
      <transferid>transfer1</transferid>
    </transfer>
    <transfer repeat="2" delay="20">
      <transferid>transfer2</transferid>
    </transfer>
    <transfer repeat="1" delay="10">
      <transferid>transfer1</transferid>
    </transfer>
  </transfers>
</transaction>
```

where:

<u>transaction</u>	:	To define a transaction within an experiment.
<u>count</u> *	:	Attribute of transaction, refers to number of different clients on which this transaction is executed.
<u>repeat</u>	:	Attribute of transferid, refer to number of times transfer repeated.
<u>delay</u> *	:	Attribute of transferid, refers to delay after which this transfer is executed.

* refers to an optional field

2.2 Defining a Transfer

```
<transfer>
  <id>transfer1</id>
  <sourceip>client</sourceip>
  <destinationip>1.2.3.4</destinationip>
  <type>UDP</type>
  <portnumber>5060</portnumber>
  <response>0</response>
  <packetdelay>0</packetdelay>
  <bytes>
    <explicit>1</explicit>
    <contentid>content1</contentid>
    <noofbytes></noofbytes>
  </bytes>
  <noofpackets>1</noofpackets>
</transfer>
```

where:

<u>id</u>	:	Unique identifier for a transfer within an experiment.
<u>sourceip</u>	:	Source ip address of the flow. (If source is device then 'client' else a server ip address like xxx.xxx.xxx.xxx)
<u>destinationip</u>	:	Destination ip address of the flow. (If destination is device then 'client' else a server ip address like xxx.xxx.xxx.xxx)
<u>type</u>	:	Type of transport protocol. (UDP or TCP)
<u>portnumber</u>	:	Port number at source and destination.
<u>response</u>	:	Response if any expected from destination. (0 - No Response, 1 - Response expected)
<u>packetdelay</u>	:	Expected delay in milliseconds between two successive packets.
<u>bytes</u>	:	Data that needs to be transferred as described in contentid or noofbytes tag.
<u>explicit</u>	:	Content if explicitly defined or use default content.

(0 - Use default content, 1 - Use explicit content)

contentid : Id of the explicit content that needs to be transferred.

noofbytes : Number of bytes of random content to be transferred.

noofpackets : Number of packets to be sent.

Note:

Size of a packet = $\text{noofbytes} / \text{noofpackets}$

2.3 Defining Explicit Content

```
<content>
  <contentid>content1</contentid>
  <protocol>Skype</protocol>
  <data>
    <![CDATA[0x0100be07de55...]]>
  </data>
  <contenttype>HEX</contenttype>
</content>
```

where:

<u>content</u>	:	To define an explicit content.
<u>contentid</u>	:	Unique identifier for the defined explicit content.
<u>protocol</u>	:	Name of the content, as a user label only
<u>data</u>	:	Actual content either in ASCII, HEX.
<u>contenttype</u>	:	Type of the content defined. (ASCII or HEX)

2.4 Defining a Criteria

```
<criteria>
  <id>criteria1</id>
  <latitude>45.66962856799364</latitude>
  <longitude>-111.06049848720431</longitude>
  <radius>5000</radius>
  <networktype>cellular</networktype>
  <starttime>143001</starttime>
  <endtime>205959</endtime>
  <deviceid>client</deviceid>
  <minimumbatterypower>0</minimumbatterypower>
  <minimumsignalstrength>-1000</minimumsignalstrength>
  <networkcarrier>Verizon Wireless</networkcarrier>
  <devicemodelname>Galaxy Nexus</devicemodelname>
</criteria>
```

where:

<u>criteria</u>	:	To define criteria when transaction need to be executed.
<u>id</u>	:	Unique id for the defined criteria within an experiment.
<u>latitude</u>	:	Latitude of the reference location.
<u>longitude</u>	:	Longitude of the reference location.
<u>radius</u>	:	Maximum distance (in Km) from the reference location.
<u>networktype</u>	:	Type of the network (Wi-Fi or Cellular).
<u>starttime</u> *	:	Time after which transfer can be started. (HH24MMSS)
<u>endtime</u> *	:	Time before which transfer can be started. (HH24MMSS)
<u>deviceid</u> *	:	Unique identifier of the device on which transaction is executed. (client for any device matching criteria or a device id)
<u>minimumbatterypower</u> *	:	Minimum battery power required on the device.
<u>minimumsignalstregnth</u> *	:	Minimum signal strength required on the device.
<u>networkcarrier</u> *	:	Cellular Service Provider.
<u>devicemodelname</u> *	:	Device model name.

* refers to an optional field

2.5 Configuring a MITATE Experiment to Send an HTTP GET Packet on Port 80 of a Web Server

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<entry>
  <defines>
    <transferdefine>
      <transfers>
        <transfer>
          <id>transfer1</id>
          <sourceip>client</sourceip>
          <destinationip>mitate.cs.montana.edu</destinationip>
          <packetdelay>0</packetdelay>
          <type>TCP</type>
          <portnumber>80</portnumber>
          <response>0</response>
          <bytes>
            <explicit>1</explicit>
            <contentid>content1</contentid>
          </bytes>
          <noofpackets>1</noofpackets>
        </transfer>
      </transfers>
    </transferdefine>
    <contentdefine>
      <content>
        <contentid>content1</contentid>
        <protocol>HTTP</protocol>
        <data>
          <![CDATA[GET /images/cdn.jpg HTTP/1.1\r\nHost:
```

```

mitate.cs.montana.edu\r\nUser-Agent: Mozilla/5.0 (Windows NT
6.1; WOW64; rv:20.0) Gecko/20100101 Firefox/20.0\r\nAccept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\
\r\nAccept-Language: en-us,en;q=0.5\r\nAccept-Encoding:
gzip,deflate\r\nConnection: keep-alive\r\n]]>
    </data>
    <contenttype>ASCII</contenttype>
</content>
</contentdefine>
<criteriadefine>
    <criteria>
        <id>criteria1</id>
        <latitude>45.66962856799364</latitude>
        <longitude>-111.06049848720431</longitude>
        <radius>50000</radius>
        <networktype>cellular</networktype>
    </criteria>
</criteriadefine>
</defines>
<transactions>
    <transaction count="1">
        <transfers>
            <transfer repeat="1" delay="0">
                <transferid>transfer1</transferid>
            </transfer>
        </transfers>
        <criteria>
            <criteriaid>criteria1</criteriaid>
        </criteria>
    </transaction>
</transactions>

```

```
    </transaction>
  </transactions>
</entry>
```

2.6 Configuring a MITATE Experiment to Perform DNS Lookup and get a Response Back

For configuring a MITATE experiment as a DNS lookup, the XML file containing traffic definitions should look similar to

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<entry>
  <defines>
    <transferdefine>
      <transfers>
        <transfer>
          <id>transfer1</id>
          <sourceip>client</sourceip>
          <destinationip>8.8.8.8</destinationip>
          <packetdelay>0</packetdelay>
          <type>UDP</type>
          <portnumber>53</portnumber>
          <response>1</response>
          <bytes>
            <explicit>1</explicit>
            <contentid>content1</contentid>
          </bytes>
          <noofpackets>1</noofpackets>
        </transfer>
      </transfers>
    </transferdefine>
    <contentdefine>
      <content>
        <contentid>content1</contentid>
        <protocol>DNSLookup</protocol>
```

```

    <data>
      <![CDATA[1b3f0100000100000000000006736561726368096a6170616
e706f7374026a700000010001]]>
    </data>
    <contenttype>HEX</contenttype>
  </content>
</contentdefine>
<criteriadefine>
  <criteria>
    <id>criteria1</id>
    <latitude>45.66962856799364</latitude>
    <longitude>-111.06049848720431</longitude>
    <radius>50000</radius>
    <networktype>cellular</networktype>
  </criteria>
</criteriadefine>
</defines>
<transactions>
  <transaction count="1">
    <transfers>
      <transfer repeat="1" delay="0">
        <transferid>transfer1</transferid>
      </transfer>
    </transfers>
    <criteria>
      <criteriaid>criteria1</criteriaid>
    </criteria>
  </transaction>
</transactions>

```

</entry>

2.7 Configuring a MITATE Experiment to Send Packets of Different Message Size

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<entry>
  <defines>
    <transferdefine>
      <transfers>
        <transfer>
          <id>transfer1</id>
          <sourceip>client</sourceip>
          <destinationip>1.2.3.4</destinationip>
          <packetdelay>0</packetdelay>
          <type>UDP</type>
          <portnumber>1234</portnumber>
          <response>0</response>
          <bytes>
            <explicit>0</explicit>
            <noofbytes>256</noofbytes>
          </bytes>
          <noofpackets>1</noofpackets>
        </transfer>
        <transfer>
          <id>transfer2</id>
          <sourceip>client</sourceip>
          <destinationip>1.2.3.4</destinationip>
          <packetdelay>0</packetdelay>
          <type>UDP</type>
          <portnumber>1234</portnumber>
          <response>0</response>
```



```

    <bytes>
      <explicit>0</explicit>
      <noofbytes>512</noofbytes>
    </bytes>
    <noofpackets>1</noofpackets>
  </transfer>
</transfers>
</transferdefine>
<criteriadefine>
  <criteria>
    <id>criteria1</id>
    <latitude>45.66962856799364</latitude>
    <longitude>-111.06049848720431</longitude>
    <radius>50000</radius>
    <networktype>cellular</networktype>
  </criteria>
</criteriadefine>
</defines>
<transactions>
  <transaction count="1">
    <transfers>
      <transfer repeat="1" delay="0">
        <transferid>transfer1</transferid>
      </transfer>
      <transfer repeat="1" delay="0">
        <transferid>transfer2</transferid>
      </transfer>
    </transfers>
  </transaction>
</transactions>
</criteria>

```

```
        <criteriaid>criteria1</criteriaid>
    </criteria>
</transaction>
</transactions>
</entry>
```

2.8 Configuring a MITATE Experiment to Send Packets with Transmission Rates (Transmission Intervals)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<entry>
  <defines>
    <transferdefine>
      <transfers>
        <transfer>
          <id>transfer1</id>
          <sourceip>client</sourceip>
          <destinationip>1.2.3.4</destinationip>
          <packetdelay>100</packetdelay>
          <type>UDP</type>
          <portnumber>1234</portnumber>
          <response>0</response>
          <bytes>
            <explicit>0</explicit>
            <noofbytes>256</noofbytes>
          </bytes>
          <noofpackets>1</noofpackets>
        </transfer>
        <transfer>
          <id>transfer2</id>
          <sourceip>client</sourceip>
          <destinationip>1.2.3.4</destinationip>
          <packetdelay>200</packetdelay>
          <type>UDP</type>
          <portnumber>1234</portnumber>
          <response>0</response>
```

```

        <bytes>
            <explicit>0</explicit>
            <noofbytes>512</noofbytes>
        </bytes>
        <noofpackets>1</noofpackets>
    </transfer>
</transfers>
</transferdefine>
<criteriadefine>
    <criteria>
        <id>criteria1</id>
        <latitude>45.66962856799364</latitude>
        <longitude>-111.06049848720431</longitude>
        <radius>50000</radius>
        <networktype>cellular</networktype>
    </criteria>
</criteriadefine>
</defines>
<transactions>
    <transaction count="1">
        <transfers>
            <transfer repeat="1" delay="0">
                <transferid>transfer1</transferid>
            </transfer>
            <transfer repeat="1" delay="0">
                <transferid>transfer2</transferid>
            </transfer>
        </transfers>
    </transaction>
</transactions>
</criteria>

```

```
        <criteriaid>criteria1</criteriaid>
    </criteria>
</transaction>
</transactions>
</entry>
```

3. MITATE Command Line API

This section will elaborate various functionalities of the command line API required to interact with MITATE's database servers. This API will help you uploading, deleting, querying data for experiments from the database server. With respect to cellular and Wi-Fi credit, this API will help you retrieve the cost of experiments, your available credit.

An overview of the basic API functionality is as follows:

login	Logs in the user
init	Setup local MySQL instance
validate	Validate configuration file
getExpCost	Get experiment cost
checkcredits	Check available credit
upload	Uploads an XML file
update	Update experiment permissions
query	Retrieve data for an executed experiment
delete	Deletes the experiment with specified ID
logout	logout the user

3.1 Login to the API

If you are a first time user and would like to upload an experiment, the MITATE's Command line API will automatically ask you to enter your MITATE account credentials that you used to signup at http://mitate.cs.montana.edu/mitate_signup.php. Once you are prompted to enter your MITATE account credentials, you will be allowed to interact with MITATE's backed once the API has verified and authenticated the user.

For all commands that a user could initiate using MITATE Command Line API, there is a common outcome which is as follows

User Authentication Failed - The API will first authenticate the user before uploading each experiment. In case the account credentials do not match with the one stored in MITATE's database, the user will be prompted again to authenticate herself by entering her username and password.

3.2 Download Script to Initialize Local MySQL Instance - Since the output of any experiment is returned to the user in form of SQL Insert Statements, this initialization command is required to setup all the tables in the user's local MySQL instance. The following command will generate script required to setup the local database:

```
mitate.sh init <output_filename>
```

This command requires two command line parameters, initialization operation and output filename. After the above command is issued, the scripts required to setup the local MySQL instance will be stored in <output_filename> given by the user. The user is then required to copy the output, paste and run the commands in his MySQL Query Engine.

3.3 Validate XML Configuration File - This command will allow users to compare their Configuration file in the form of XML against the MITATE XML Schema Definition file as required before uploading experiments. Although the upload experiment command will validate the input XML file before uploading the experiment, the validate command will help users to validate their XML file without actually issuing an upload experiment request. The following is the syntax to validate an XML file:

```
mitate.sh validate <XML_Filename>
```

This command requires two command line parameters, validate operation and XML configuration filename.

3.4 Get Experiment Cost - This command will allow users to check the total amount of Cellular data credit and Wi-Fi data credit for an XML file configured as MITATE experiment.

```
mitate.sh getExpCost <XML_Filename>
```

This command requires two command line parameters, count operation and XML Configuration filename. The result of this command will be the total cellular and Wi-Fi data credits that the given XML file will require the user to have available in his account if this experiment were to be executed.

3.5 Check User's Available Credits - This command will allow users to check their available cellular and Wi-Fi data credit. The following command is required to be executed to check the available credits:

mitate.sh checkcredits

This command requires one command line parameter, a checkcredit operation.

3.6 Upload an Experiment as XML - To upload an experiment you will need to execute the following command from your terminal.

```
mitate.sh upload <XML_Filename>
```

This command requires two command line parameters, upload operation and input XML file name. There can be three outcomes of this query which are as follows:

1. *XML Validation Error* - Before uploading any experiment, the API first validates the input XML file against MITATE's XML Schema Definition File which can be found at http://mitate.cs.montana.edu/sample/Mitate_Sample_Configuration_File_XML_Form_at.xsd.
2. *Insufficient Available Measurement Credit* - If the total volume of data in an experiment, that needs to be exchanged between mobile devices and MITATE backend servers, exceed the available credit balance of the user, the API will generate an error message asking the user to either reduce the number of transfers from the XML file or increase the mobile and Wi-Fi credit for his account.
3. *Experiment Uploaded Successfully* - If none of the above three cases are true, the API will process and upload the XML file and finally output the success message with a unique experiment ID .

3.7 Update Experiment Permissions - Since by default all the experiments can be accessed by the user who uploaded them, this command will enable the owner of the experiment to set its permissions as public (with a confirmation) in which case its execution as well as the data associated with it can then be accessed by anyone. In general, an experiment whose permissions are set as “private” would allow the owner of the experiment to execute, delete and fetch any data associated with it. An experiment whose permissions are set as “public” would allow all the users to execute and fetch the data associated with it. To delete a public experiment, the user who uploaded the experiment needs to issue a delete experiment request as described above and cannot be deleted by any other user. The following command needs to be executed to update the permissions of an experiment:

```
mitate.sh update <experiment_ID>
```

This command requires two command line parameters, an update operation and experiment ID. After this command is issued, the API will check if the user is the owner of the experiment and will ask for a final confirmation to set the experiment as public. Once the experiment is set public, it can be made private.

3.8 Query Experiment Data - In order to retrieve all the data associated with a completed experiment, the user is required to issue the following command:

```
mitate.sh query user_experiment_list.txt <output_filename>
```

This command requires three command line parameters, query operation, an input file containing list of experiment IDs and an output file. The experiment IDs of all experiments that were uploaded successfully are stored in “user_experiment_list.txt” file where each experiment ID is stored in one separate line. When a user issues a request to query data associated with all the experiments in the “user_experiment_list.txt”, the API can generate two following outcomes:

1. *Error in Retrieving Data* - The API throws an error if it finds an experiment ID, in the user_experiment_list.txt file, that is not owned by the user requesting its data or the user does not have permissions to access its data or the experiment no longer exists in the database.
2. *Data Retrieved Successfully* - For all those experiments for which there were no errors, the data is appended in the output file in form of SQL Insert Statements that can be used to insert data into user’s local MySQL instance.

3.9 Delete an Uploaded Experiment - To delete an uploaded experiment you will need to execute the following command from your terminal.

```
mitate.sh delete <experiment_ID>
```

This command requires two command line parameters, delete operation and experiment ID. Once the above command is executed, the user will be asked to give a confirmation to issue a delete experiment request. If the user responded with “y” (as in yes), there can be two outcomes of issuing the command which are as follows:

1. *Invalid Experiment ID* - The API will throw an invalid experiment ID error if either the experiment ID does not belong to the user requesting its deletion or has already been deleted before by the user who uploaded that experiment.
2. *Experiment Deleted Successfully* - If the user requesting the deletion of an experiment ID is the owner of that experiment and the experiment still exists, the API will delete the experiment and all the data collected associated with that experiment.

3.10 Logout from the API - This command will log the user out of the API. The following command needs to be executed if the user wish to log out.

mitate.sh logout

This command requires one command line parameter, a logout operation. Once the logout command is issued, the user is asked to enter his username and password the next time the user wish to perform any operation using the API.