A Peer-To-Peer-Based Botnet Crawling Framework (Deployment and Running Manual)

A Peer-To-Peer-Based Botnet Crawling Framework (Deployment and Running Manual)
Master-Thesis of Isbel Isbel
1-Supervisor: Prof. Dr. Max Mühlhäuser
2-Supervisor: Ph.D. Shankar Karuppayah

TECHNISCHE
UNIVERSITÄT
DARMSTADT

1 Deployment

This chapter describes the deployment process. It describes the executable files, the operation systems and the deployment entities.

1.1 Deployment Operating Systems

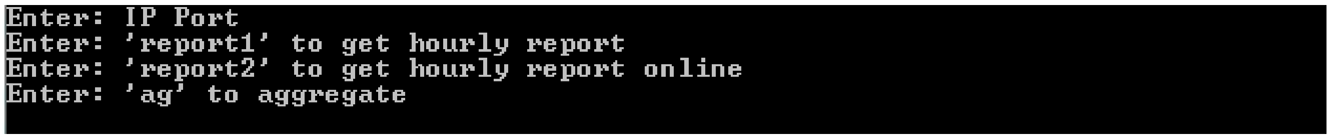
The Peer-To-Peer (P2P) based framework is developed using C# programing language. The generated files are **exe** executable files. These files could be executed in both **Windows** and **Linux** operating systems. These files could be executed in **Windows** without using any extra plug-ins or software. On **Linux**, a third party software should be used to execute these files. For example, **mono** project could be used to run these files on Linux.

1.2 Deployment Entities

Two types of entities should be deployed to deploy the P2P based framework.

Data Collector

All the crawling results and data are sent to this entity. The data collector aggregates the data and generate reports from the crawling data which are sent by the participants. Each P2P based framework has only one data collector. To deploy this entity, we need to execute the file **dataCollector.exe**. By executing this file, we get a console application as shown in Figure 1.1.



```
Enter: IP Port
Enter: 'report1' to get hourly report
Enter: 'report2' to get hourly report online
Enter: 'ag' to aggregate
```

Figure 1.1: Collector Initialization Phase

As shown in Figure 1.1, to deploy the collector we need to enter the IP Address and the Port Number splited by space. The IP Address should be the Address of the used machine. Then, the collector will be waiting for requests from the crawling participants. By running **dataCollector.exe**, three folders will be created:

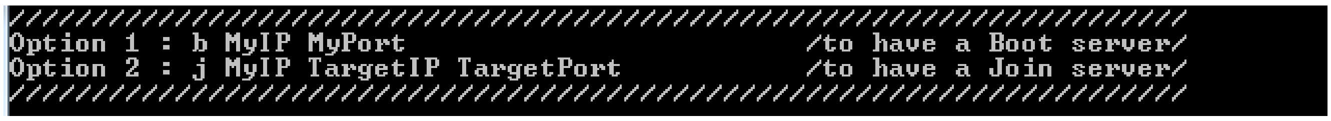
- **Collected Data**: The collector puts every received file into this folder. It creates a folder named with the address of the sender. It saves the received files into this sub-folder.
- **Aggregated Data**: The collector puts all the aggregated data into this folder.
- **Reports**: The collector puts all the generated reports in this folder.

While running the collector, the user could execute three different commands:

- **ag**: The collector aggregates all the files in the **Collected Data** folder and puts the aggregated data into the **Aggregated Data** folder.
- **report1**: The collector generates a report from the aggregated data and puts this report in the **Reports** folder. This report contains the number of discovered bots every hour.
- **report2**: The collector generates a report from the aggregated data and puts this report in the **Reports** folder. This report contains the number of online bots every hour.

Crawling Participant

Many crawling participants could join the crawling process in the P2P based framework. The participant could be deployed by executing the file **participant.exe**. By executing this file, we get a console application as shown in Figure 1.2.



```
////////////////////////////////////////////////////////////////////////////
Option 1 : b MyIP MyPort                           /to have a Boot server/
Option 2 : j MyIP TargetIP TargetPort              /to have a Join server/
////////////////////////////////////////////////////////////////////////////
```

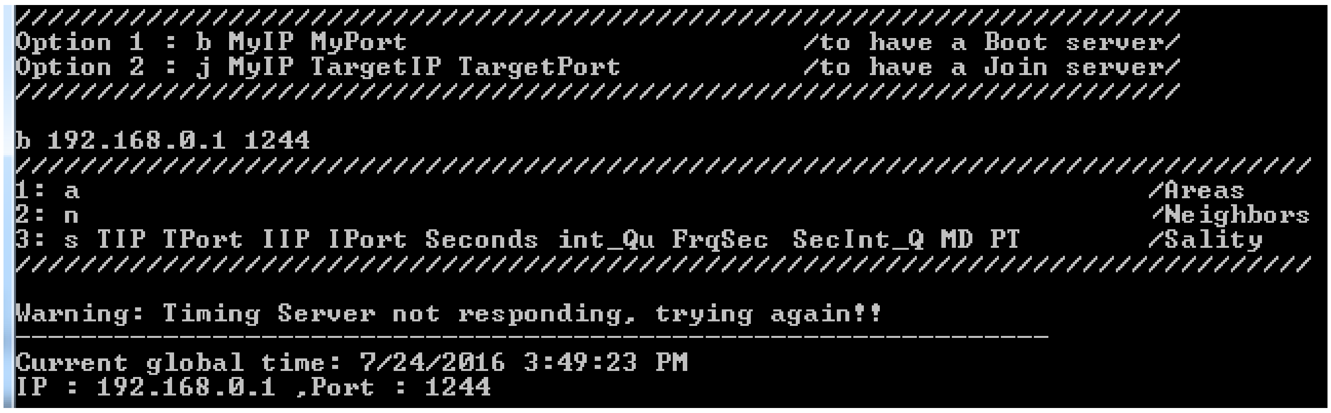
Figure 1.2: Participant Initialization Phase

The user can execute one of two commands to join the P2P based framework.

- **b MyIP MyPort**: This command should be used to deploy the first participant in the P2P based framework.
- **j MyIP TargetedIP TargetedPort**: This command should be used to join a P2P based framework.

After executing **participant.exe** and joining the P2P based framework (see Figure 1.3). The user can execute these different commands:

- **a**: Shows the coordinates of the responsible region.
- **n**: Lists all the neighbors.
- **s**: Starts a crawling process, this commend has the following parameters:
  - **TIP**: The IP Address of the targeted bot.
  - **TPort**: The Port Number of the targeted bot..
  - **IIP**: The IP Address of the collector.
  - **IPort**: The Port Number of the collector.
  - **Seconds**: The total length of the first crawling phase.
  - **int_Qu**: The number of queries that are sent during the first crawling phase.
  - **FrqSec**: How frequently (in seconds) should the crawler crawl.
  - **Sec_Int_Q**: The number of queries that are sent during the second crawling phase.
  - **MD**: The number of used ID-Spaces.
  - **PT**: This parameter has two options (true and false). It indicates whether the crawler pauses or not.



```
////////////////////////////////////////////////////////////////////////////
Option 1 : b MyIP MyPort                           /to have a Boot server/
Option 2 : j MyIP TargetIP TargetPort              /to have a Join server/
////////////////////////////////////////////////////////////////////////////

b 192.168.0.1 1244
///////////////////////////////////////////////////////////////////////////////////
1: a                                                                     /Areas
2: n                                                                     /Neighbors
3: s TIP TPort IIP IPort Seconds int_Qu FrqSec  SecInt_Q MD PT           /Sality
///////////////////////////////////////////////////////////////////////////////////

Warning: Timing Server not responding, trying again!!
---------------------------------------------------------------
Current global time: 7/24/2016 3:49:23 PM
IP : 192.168.0.1 ,Port : 1244
```

Figure 1.3: Participant Running Phase