International Journal of Technology (2017) 1: 177-185
ISSN 2086-9614

IMPROVING MOBILITY OF BASE TRANSCEIVER STATION LOCATING METHOD USING TELEGRAM'S APPLICATION

Tubagus Mohammad Akhriza[1*], Hizkia Yesarela Sahaduta[1,2], Antonius Duty Susilo[1]

[1] *Pradnya Paramita School of Informatics Management and Computer, Kampus STIMATA: Jl. LA Sucipto 249A – Malang 65125 – East Java Indonesia*
[2] *Telkomsel Core Network Operation Division 6th Region Malang – East Java Indonesia*

(Received: November 2016 / Revised: December 2016 / Accepted: January 2017)

ABSTRACT

One main problem that must be coped with by a telecommunication company (Telco) is the connection interference experienced by customers. When a cell phone number *Pn* in a region *R* suffers from a connection problem, the first step commonly taken by that Telco's technician is to find the base transceiver station (BTS) among existing BTSs in *R* that is currently covering *Pn*. However, the proprietary tools used to locate the covering BTS can usually be accessed only from the regional office's intranet with a specific IP address. Alternatively, a technician can use telnet to log in to a mobile switching center (MSC) server and search to determine whether *Pn* is being attached in a BTS that is registered in the related MSC server. However, this method is exhausting and inefficient because an MSC server usually registers hundreds to thousands of BTSs. This article proposes improving the efficiency, mobility, and interoperability of BTS location-finding by making use Telegram's bot and command-line interfaces. Mobility and interoperability are improved because the proposed method can run both on PCs and smartphones. The proposed method is investigated experimentally at Telkomsel Ltd., a known Telco in Indonesia. This method requires only 30 seconds to locate the covering BTS, which is 20 times and four to seven times faster than manual telnet and the proprietary tool, respectively.

Keywords: Base transceiver station; Cellular phone; Telecommunication; Telegram

1. INTRODUCTION

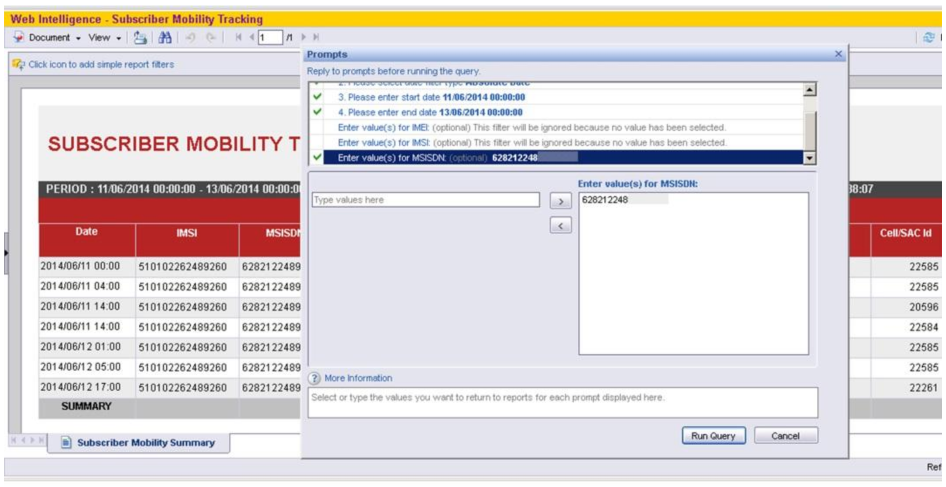
The human need for telecommunication services has increased significantly in the information age. Telecommunication companies (Telcos) continue to improve and develop their technologies and services as well. However, even given the fast development and invention of telecommunications technologies, the process of adopting such technologies in Telcos is not immune from emerging problems, particularly those related to the connection interference experienced by customers.

When a mobile subscriber integrated service digital number (MSISDN), i.e., the cell phone number *Pn* in a region *R*, suffers from a connection problem, the first step commonly taken by a Telco's technician is to locate the base transceiver station (BTS) that is currently covering *Pn* (called the covering BTS). Methods for finding the covering BTS are discussed in the literature. The most commonly used method in 2G/3G networks is called the Cell-ID (CI)-based location-finding method (Shad & Chen, 2012). Depending on the cell size and type, the accuracy of this

*Corresponding author's email: akhriza@stimata.ac.id, Tel. +62341-412699
Permalink/DOI: https://doi.org/10.14716/ijtech.v8i1.6012

method is tolerable particularly in high-population regions in which BTSs are located close one to another. Additionally, it is the most inexpensive method because no enhancements to network infrastructure or individual handsets must be made (Shad & Chen, 2012; Nnenna & Onyekachi, 2012; Smit et al., 2012). For comparison, the ability to use a highly accurate GSM location-finding method costs approximately USD 10,000 per base station (Kushwaha, 2014).

Using certain tools, such as subscriber mobility tracking (SMT, Figure 1), a technician connects to a mobile switching center (MSC) server maintained by a regional office and searches to determine the status of *Pn*’s attachment in every BTS listed in the related MSC server (MSS). The search targets are the CI and its location area code (LAC), i.e., the main identities of a BTS, and these are then converted to the geographical position of a BTS by using additional information owned by the Telco. An MSS usually registers hundreds to thousands of BTSs, and a regional office usually maintains several MSSs. However, one problem that must be solved is that SMT is accessible only from the office’s intranet via a restricted IP address. Because SMT is proprietarily owned by the Telco, it is obvious that such a tool is not freely available on the Internet. In addition, SMT’s subscription cost is considerably expensive. However, mobility limitation is clearly a disadvantage for a Telco because in certain circumstances, technicians must perform the locating process while they are completing field duties. Moreover, SMT does not search for the current position of *Pn* but its position within the last *h* hour(s), such as 1, 6, 12, or 24 hours. Thus, processing one *Pn* normally takes 3-10 minutes.


Web Intelligence - Subscriber Mobility Tracking
Document
View
Click icon to add simple report filters
Subscriber Mobility Summary
Prompts
More Information
Run Query
Cancel
Ref

SUBSCRIBER MOBILITY T

PERIOD : 11/06/2014 00:00:00 - 13/06/2014 00:00:0

Date	IMSI	MSISD	Cell/SAC Id
2014/06/11 00:00	510102262489260	6282122489	22585
2014/06/11 04:00	510102262489260	6282122489	22585
2014/06/11 14:00	510102262489260	6282122489	20596
2014/06/11 14:00	510102262489260	6282122489	22584
2014/06/12 01:00	510102262489260	6282122489	22585
2014/06/12 05:00	510102262489260	6282122489	22585
2014/06/12 17:00	510102262489260	6282122489	22261
SUMMARY			

Reply to prompts before running the query.

✓ 3. Please enter start date **11/06/2014 00:00:00**
✓ 4. Please enter end date **13/06/2014 00:00:00**
Enter value(s) for IMEI: (optional) This filter will be ignored because no value has been selected.
Enter value(s) for IMSI: (optional) This filter will be ignored because no value has been selected.
✓ Enter value(s) for MSISDN: (optional) **628212248**[redacted]

Type values here

Enter value(s) for MSISDN:
628212248[redacted]

Select or type the values you want to return to reports for each prompt displayed here.

Figure 1 Subscriber mobility tracking tool

A more traditional way of performing such a task is by using telnet to log into the MSS remotely and manually search for the status of Pn’s attachment to every single BTS listed in the MSS. However, such an approach is exhausting and inefficient considering the large number of BTSs that must be inspected. Processing one *Pn* takes at least 10 minutes and can take up to one hour because almost every process is performed manually, such as converting the hexadecimal number of the LAC and CI of each BTS into a decimal number.

Telkomsel Ltd. is a large well-known Telco in Indonesia, and the core network operation division in the 6th region of Telkomsel, located in Malang City, East Java Province, has been assigned the task of developing a mechanism for locating the covering BTS with high levels of mobility, efficiency, and interoperability. Thus, a BTS can be located at any place and any time in secure way. Of course, such a mechanism should only be available to trusted individuals. This article proposes a solution to such problems by making use Telegram’s bot (shorted from robot) and command-line interface (CLI). Both are third-party applications that are run inside

Telegram, which is a free multi-platform chat application. Thus, the bot and CLI can fulfill mobility and interoperability requirements. Some current projects related to Telegram's bot and CLI can be found in the literature. The bot can perform many operations – search, connect, integrate with other services, and even pass commands to the Internet of Things (Namiot, 2015). The bot and CLI have been utilized to control and monitor housing equipment (Imran, 2016). However, finding the location of a covering BTS using Telegram's applications is still very rare in the literature. Because the BTS-location process is the first step in solving connection problems, the faster the locating process can be completed, the more complaints a technician can handle. Thus, this work contributes to improving the handling of complaints by Telcos, particularly in Indonesia.

2. METHODOLOGY

For mobility data, location binding can be performed via various methods, such as global positioning service (GPS), service-provider-assisted faux GPS, and cell global identity (CGI) (Shad & Chen, 2012). However, among existing methods of location prediction, the CGI property of the global system for mobile communication (GSM) is most commonly implemented by companies because it is easy, readily available, and inexpensive. The CGI method, which mainly relies on LAC-CI, does not require that enhancements be made to network infrastructure or to individual handsets. The accuracy of this method depends on factors such as cell size and type, and its accuracy remains tolerable in areas in which BTS locations are close to one other, such as in urban areas (Shad & Chen, 2012; Nnenna & Onyekachi, 2012; Smit et al., 2012).

Dabrowski et al. (2014) explain that in general, a mobile network consists of base (transceiver) stations that use one or more radio interfaces to create geographically limited radio cells. A mobile station (MS), e.g., a phone and data modem, is registered in the operator network with its worldwide unique international mobile equipment identity (IMEI), international mobile subscriber identity (IMSI) number, and a secret key stored in the subscriber identity module (SIM). The network (GSM in this example) will assign a temporarily mobile subscriber identity (TMSI) number for addressing purposes. In GSM, a cell is uniquely identified by the mobile country code (MCC), mobile network code (MNC), LAC, and CI. According to the Third-Generation Partnership Project regarding location-management procedure, IMSI detachment from a cell indicates that an MS with such an IMSI is no longer being covered by a BTS (3GPP TSCGN, 2003). In other words, when locating the covering BTS, a technician should obtain the value of the IMSI detach flag attribute from the MSS. If its value is *no*, then the MS is currently covered by the related BTS. After the covering BTS is found, the next step is to find the LAC and CI of the related BTS and then transform them into the precise location of the BTS (Wang et al., 2012; Shad & Chen, 2012).

Referring to the above information, a mechanism for locating BTS using a bot and CLI is proposed, as depicted in Figure 2. Four main processes are developed. The bot is named STIMATA_Bot, and to use this Bot, a technician/user must first add the bot as his/her chat friend. In process (1), a user with a Chat-ID runs the LocateBTS program stored in the bot to locate the BTS covering a given *Pn* = 62xxx (62 is Indonesia's country code) by sending the command "/LocateBTS 62xxx" as a message to the bot. This message is then forwarded to a Linux machine (or *the Linux*) using Telnet via the Telco's IP proxy channel.

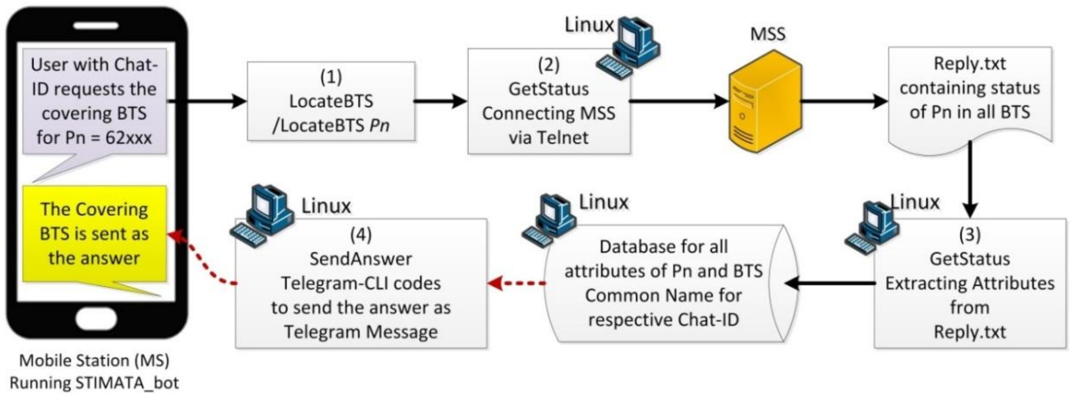

User with Chat-ID requests the covering BTS for Pn = 62xxx
The Covering BTS is sent as the answer
Mobile Station (MS) Running STIMATA_bot
(1) LocateBTS /LocateBTS Pn
Linux
(2) GetStatus Connecting MSS via Telnet
MSS
Reply.txt containing status of Pn in all BTS
Linux
(3) GetStatus Extracting Attributes from Reply.txt
Linux
Database for all attributes of Pn and BTS Common Name for respective Chat-ID
Linux
(4) SendAnswer Telegram-CLI codes to send the answer as Telegram Message

Figure 2 The proposed mechanism

In process (2), a program named GetStatus, which runs on the Linux, remotely logs into the MSS and performs all the procedures needed to obtain the status of Pn from all BTSs listed in the MSS. The statuses of Pn from all BTSs are sent to the Linux in a text file named Reply.txt. In process (3), GetStatus finds the covering BTS by extracting and interpreting the attributes in Reply.txt. The attributes of the covering BTS are stored in a database in process (2). In process (4), another program, SendAnswer, runs the Telegram CLI to obtain the attributes stored in database and send them as a Telegram message to the requesting user with a specific Chat-ID. Pseudo codes for LocateBTS, GetStatus and SendAnswer programs are provided in the next subsections.

In the experiment, Telegram and the bot are installed and run on an iPhone 6 with iOS 10.0.1, while the GetStatus and SendAnswer programs are run on Linux Ubuntu 229-4ubuntu7 64bit on a machine with a Six-Core AMD Opteron(tm) 2435 processor. The bots and programs are written in PHP and Linux bash scripting, while the database used to store the attributes is a MySQL database. The Linux is placed in the 6th Region office, which maintains nine MSSs covering more than 1,000 BTSs scattered throughout East Java Province. A part of the Reply.txt produced by Telkomsel’s MSS may look like Figure 3. The strings on the left side of the dots are attributes, while those on the right side are the values of the related attributes. An MSS owned by a different Telco may produce different content, but the attribute names should be similar. Some attributes have been explained previously, such as IMSI, TMSI, MCC, LAC, and CI. Among these, the IMSI detach flag attribute is the first attribute that must be checked. If its value is N, then *Pn* is being covered by the BTS. The other attributes needed in the experiment are listed in Table 1.

Table 1 Attributes’ meanings and usages

No	Attribute (Abbreviation)	Meaning and Usage
1	International Mobile Subscriber Identity (IMSI)	International identity of *Pn*
2	Mobile Station Center (MSC) Server	An MSS registering some BTS
3	IMSI Detach Flag	If value = N, then *Pn* is covered by MSS
4	Mobile Country Code (MCC)	The country code of IMSI
5	Mobile Network Code (MNC)	The network code of IMSI
6	Radio Access Info (RAI)	GSM for 2G and UMTS for 3G
7	Location Area Code (LAC)	Code representing area of IMSI
8	Cell Identity (CI)	Identity of BTS
9	Last Activate Date	Last date of *Pn* attachment in BTS

2.1. The LocateBTS Program

This program is actually an interface between the user and the GetStatus program, which handles all the procedures involved in obtaining the status of *Pn* from the MSS. The main task of this program is to validate the input given by the user; thus, the message that will be sent to the Linux is valid. For example, *Pn* must have at least ten digits. A pseudo code for the LocateBTS program is given in Pseudo code 1, which shows that *Pn* and Chat-ID parameters are passed to GetStatus.

PSEUDO CODE 1: LocateBTS

```
01: Input: Pn, Chat-ID
02: Output: allReply.txt
03: Repeat
04:   {isValid ← Validate Pn}
05: until isValid = true
06: Send request(Pn, Chat-ID) to GetStatus in the Linux
```

```
ZMVO:MSISDN=62xxx::;
MSCi XSBY7 2016-08-29 00:57:08
SUBSCRIBER INFORMATION:
 INTERNATIONAL MOBILE SUBSCRIBER IDENTITY .............          510101725710316
 TEMPORARY MOBILE SUBSCRIBER IDENTITY ..................... 692F9106H/1764724998D
 ACTIVATION STATUS  ............................................................. A
 MOBILE STATION CATEGORY.................................................. OR
 MOBILE COUNTRY CODE ..........................................................          0510D
 MOBILE NETWORK CODE ......................................................... 0010D
 LOCATION AREA CODE OF IMSI ..............................................          184CH/06220D
 RADIO ACCESS INFO  ..................................................................UMTS
 IMSI DETACH FLAG ........................................................................N
 LAST ACTIVATE DATE ................................................................08-29 00:53
 LAST USED CELL ID .................................................................... 2A0AH/10762D
```

Figure 3 Snapshot of reply.txt

2.2. The GetStatus Program

GetStatus performs the procedure to locate the covering BTS by connecting all MSSs maintained by a regional office. This program automatically and successively finds the status of Pn for all BTSs listed in all MSSs. As explained, each MSS replies regarding the status of Pn in the form of a text file named Reply.txt. The pseudo code of this program is given in Pseudo code 2. GetStatus extracts information from Reply.txt to determine a set of attributes for the BTS to which Pn is attached. It first needs to find a key string called “SUBSCRIBER INFORMATION,” which indicates that all strings below this key string are related to the needed attributes. However, as explained, only some attributes are needed by the technician, and these attributes are stored in a temporary array, attributArray, by GetStatus. The process of IMSI detachment checking is shown from line 12 to 13. After the covering BTS is found and the other needed attributes, such as LAC and CI, have been received, the attributes stored in the array with the Chat-ID of requesting user are flushed into database DB. Then, this process is stopped (line 16). In contrast, if the value is not N, the array is emptied, and the search process continues until the key string is found again. In addition, practically, a telnet connection will not always succeed. Thus, if the key string has not been found yet and unconnected MSSs still exist, GetStatus attempts to connect to the remaining unconnected MSSs until the covering BTS is found or is not found in any of the MSSs. If the latter state is reached, then it is concluded that Pn is covered by another region (line 19); thus, the Chat-ID and an empty attributArray are flushed into DB.

PSEUDO CODE 2: GetStatus

```
01: Input: Pn, Chat-ID, MSS_list, DB, Reply.txt
02: Output: DB
03: For each MSS in MSS_list
04:   {isConnected ← Telnet MSS.ip-address
05:    If isConnected
06:    { Reply.txt << get status from MSS
07:      line ← read a line in Reply.txt
08:      aBTSFound ← line.Find( "SUBSCRIBER INFORMATION")
09:      If aBTSFound
10:      { line ← read a line in Reply.txt
11:        attributArray.insert(line)
12:        If line contains "IMSI Detach Flag" BUT its value != N
13:        { attributArray.clear() // no need to store line into  attributArray
14:          continue to 06 }
15:        If all needed attributes are gotten
16:        { DB ← flush (Chat-ID, attributArray); Finished }}
17:      Else if !isConnected {save MSS to connect later}}
18: If all MSS is connected and "SUBSCRIBER INFORMATION" is not found
19:    {DB ← flush (Chat-ID, empty attributeArray}
```

GetStatus applies an information extraction approach called named entity recognition (NER), which involves identifying references to particular kinds of objects, such as names of people, companies, and locations (Mooney & Bunescu, 2005). NER is a fundamental task, and it is the core of natural language processing systems. NER involves two tasks: firstly, the identification of proper names in text and, secondly, the classification of these names into a set of predefined categories of interest, such as personal names, organizations (companies, government organizations, committees, etc.), locations (cities, countries, rivers, etc.), and date and time expressions (Mansouri et al., 2008). In our case, IMSI, LAC, and CI are the entities targeted by GetStatus.

2.3. Telegram CLI-based SendAnswer Program

A pseudo code called SendAnswer is developed using Telegram CLI methods that sends information about the covering BTS as a Telegram message to a user with a given Chat ID (see Pseudo code 3). This process is performed on the Linux.

PSEUDO CODE 3: SendAnswer

```
01: Input: Chat-ID, Glossary, DB
02: Output: Telegram message
03: Connect to Telegram https://api.telegram.org/.$Chat-ID
04: dbConnected ← connecting to DB
05: If dbConnected
06: { If attributArray is empty
07:     {Send Telegram for Chat-ID: "Pn is not found in this region"; Finish}
08:   Select all attributes for Chat-ID
09:   NetType = ""
10:   If RAI=="GSM" {NetType = "2G"}
11:   If RAI=="UMTS" {NetType = "3G"}
12:   BTS_common_name = GetCommonName(Glossary,LAC,CI)}
Send Telegram for Chat-ID about NetType, BTS_common_name and other attributes: MSS
name, IMSI, MCC, MCN, Last Active Date;Finish
```

The code first declares the use of Telegram CLI in line 03 and then connects to database DB in line 04. In lines 06-07, if attributArray is empty, then a message is sent to the Chat-ID, reporting that Pn is not found in this region. Otherwise, the program interprets certain attributes stored in DB. For example, in lines 09-11, RAI’s values are interpreted as indicating one of two types of network (2G or 3G), while BTS_common_name in line 12 contains the common name of the BTS, which is obtained from a BTS name glossary. An example of such a glossary is given in Table 2.

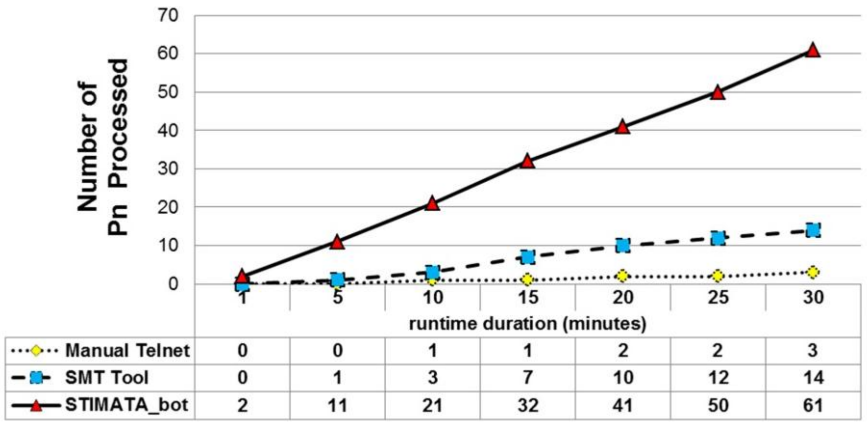
Table 2 Illustration of BTS Name Glossary

No	LAC	CI	BTS Common Name	Location	Latitude	Longitude
1	6223	13628	MLG362MW1_SULFATDMTMW	MALANG	-7.95905	112.648
2	6208	22189	PSN218MW1_PABEANPANDAAN	PANDAAN	-7.655305	112.706485
3	6204	30875	MLG087MW1_GAJAYANAIBS	MALANG	-7.94854	112.60887

3. RESULTS AND DISCUSSION

The efficiency of STIMATA_Bot in locating the covering BTS is compared to those of two other approaches, i.e., SMT and manual telnet. The locating process is said to be complete when the covering BTS’s common name and location are discovered. A technician is asked to successively process a number of *Pn* values for seven runtime durations, i.e., 1, 5, 10 15, 20, 25, and 30 minutes. The number of *Pn* values processed for each approach is recorded, and the results of this experiment are given in Figure 4. The bot is run on PCs and smartphones to evaluate its interoperability; but because the runtime needed by both devices to accomplish the given task is the same, Figure 4 contains only one line to represent results of the bot experiment.

As shown in Figure 4, STIMATA_Bot completed two requests in one minute, while the two other approaches did not finish processing any *Pn* values at all. In five minutes, eleven *Pn* values were processed by STIMATA_Bot, while SMT only processed one *Pn*. In contrast, manual telnet still could not find the covering BTS within five minutes. For the remaining runtimes, it is concluded that STIMATA_Bot processes two *Pn*s per minute on average. This means that the proposed Bot improves BTS location-finding efficiency, reaching speeds up to 20 times faster than a manual telnet approach and about four to seven times faster than SMT in the 30-minute runtime duration.



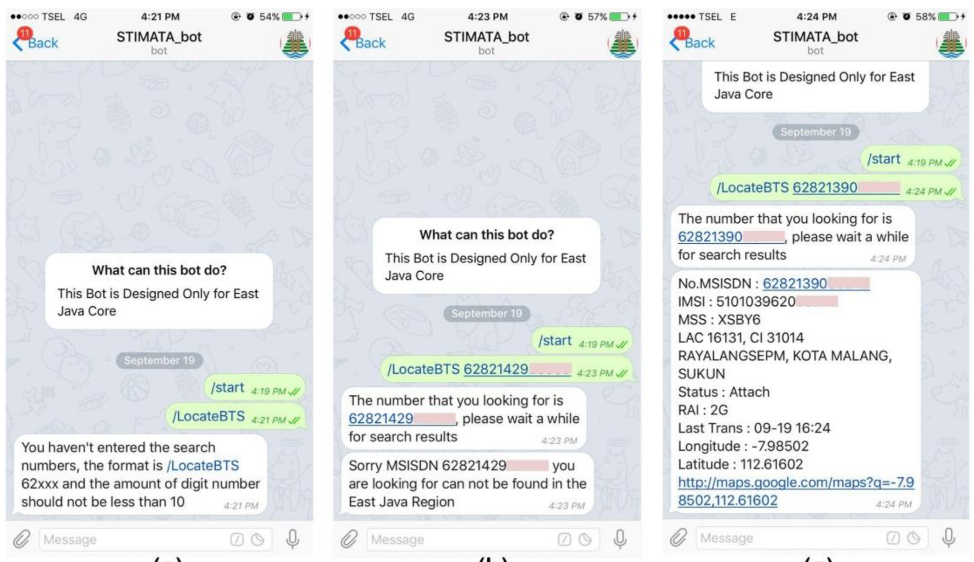
	1	5	10	15	20	25	30
Manual Telnet	0	0	1	1	2	2	3
SMT Tool	0	1	3	7	10	12	14
STIMATA_bot	2	11	21	32	41	50	61

Figure 4 Number of *Pn*s processed in a given runtime duration

Using the SMT tool is not as fast as using the proposed bot because such tool is used to find the mobility of *Pn* over the last *h* hours, such as the last 1‒6 hours, while the bot is designed to search for only the location of *Pn* at the present moment. Also, the usage of the bot avoids human error, which improves process efficiency, because most of procedures are performed automatically. Other methods still involve several human-based operations, such as BTS common name conversion. Thus, human errors are possible. To provide a clearer explanation of how the bot works, some snapshots of this bot are provided in Figure 5.

Figure 5a is the initial screen of the bot, and it shows that this Bot is designed only for East Java Core. This means that the usage privileges of this bot are restricted for certain trusted individuals. This figure also shows the validation process for *Pn*. All figures demonstrate that requesting the location of a BTS is as easy as chatting with a friend. The technician must only send one command as a message to the bot in order to obtain detailed information about the covering BTS. Figure 5b records the moment when Pn is not covered by any BTS in the 6th region, while Figure 5c shows the moment when the covering BTS is found. From Figure 5c, it is known that *Pn* is being covered by a BTS in MSS XSBY7, with the LAC and CI for the related MSS also being obtained. However, here, LAC and CI are also transformed into BTS common names, i.e., RAYALANGSEPM, Kota Malang, Sukun District. Such auto-transformation obviously helps technicians to more quickly understand the precise position of the covering BTS as compared to manual transformation. Below that information, the *Attached* status explains that *Pn* is being attached to the covering BTS. Other information is also provided, such as the latitude and longitude of the BTS, as well as a Google Maps link to these coordinates. These additional features improve the readability of the finding results as compared to other methods.

The time stamps for sending, processing, and receiving the answers are also recorded, such as in Figure 5c, i.e., 4:24 PM for all three processes. This is the evidence that locating a BTS on a smartphone requires no more than one minute. Figure 5b shows that it takes no more than one minute to check for the presence of the covering BTS in all nine MSSs before concluding that *Pn* is not covered Region 6, which is very fast. The interoperability of the proposed applications has also been tested in various places using PCs and smartphones, and the results show that performance of the method on PCs is as fast as on smartphones as long as the Internet connection is acceptable. The results regarding the bot's runtime performance on PCs are depicted in the chart in Figure 4. On the other hand, even if SMT and manual telnet methods can be performed remotely from smartphones, resource limitations, such as data and screen size limitations, make these processes take more time to complete.



What can this bot do?

This Bot is Designed Only for East Java Core

September 19

/start 4:19 PM

/LocateBTS 4:21 PM

You haven't entered the search numbers, the format is /LocateBTS 62xxx and the amount of digit number should not be less than 10 4:21 PM

What can this bot do?

This Bot is Designed Only for East Java Core

September 19

/start 4:19 PM

/LocateBTS 62821429[redacted] 4:23 PM

The number that you looking for is 62821429[redacted], please wait a while for search results 4:23 PM

Sorry MSISDN 62821429[redacted] you are looking for can not be found in the East Java Region 4:23 PM

This Bot is Designed Only for East Java Core

September 19

/start 4:19 PM

/LocateBTS 62821390[redacted] 4:24 PM

The number that you looking for is 62821390[redacted], please wait a while for search results 4:24 PM

No.MSISDN : 62821390[redacted]
IMSI : 5101039620[redacted]
MSS : XSBY6
LAC 16131, CI 31014
RAYALANGSEPM, KOTA MALANG, SUKUN
Status : Attach
RAI : 2G
Last Trans : 09-19 16:24
Longitude : -7.98502
Latitude : 112.61602
http://maps.google.com/maps?q=-7.98502,112.61602 4:24 PM

(a) (b) (c)

Figure 5 (a) Validation process; (b) BTS is not found; (c) BTS is found

These results show that the efficiency, mobility, and interoperability of the proposed solution in finding the covering BTS outstrip those of the commercial SMT application and manual telnet approaches. Additionally, investing in one Linux machine does not significantly affect the cost of a Telco's operational, especially considering its usefulness in significantly improving the speed of the locating process. Ubuntu Linux, MySQL database, Telegram, and its third-party programs are free software. Thus, the implementation of the proposed solution may even reduce annual SMT application subscription costs drastically. Furthermore, this machine can also perform the locating process from inside of the Telco's office; thus, the cost of the needed investment is worth it.

4. CONCLUSION

The proposed solution is able to outperform a commercial SMT application and a manual telnet approach to finding a covering BTS in terms of of time, cost-efficiency, mobility, and interoperability. It improves efficiency in locating the covering BTS, reaching speeds up to 20 times faster and about four to seven times faster than the SMT and telnet approaches, respectively, over a 30 minute runtime duration. Future improvements should focus on handling situations in which multiple requests are received by system at the same time. The development of a more interactive user interface should also be included in future work.

5. REFERENCES

3GPP TSGCN, 2003. *Location Management Procedures (Release 5)*. 3GPP Technical Specification Group Core Network, Valbonne, France

Dabrowski, A., Pianta, N., Klepp, T., Mulazzani, M., Weippl, E., 2014. IMSI-Catch Me If You Can: IMSI-Catcher-Catchers. *Proc. 30th Annual Computer Security Applications Conference*, pp. 246–255

Imran, M.I., 2016. Intelligent Home Control and Monitoring System via Internet. *International Journal of Scientific Development and Research (IJSDR),* Volume 1(4), pp. 82–87

Kushwaha, J., 2014. Location Estimation of GSM user using Enhanced Triangularisation Method with Implementation of RF Figure Printing. *International Journal of Engineering Science and Innovative Technology*, Volume 3(6), pp.347–351

Mansouri, A., Affendey, L.S., Mamat, A., 2008. Named Entity Recognition Approaches. *International Journal of Computer Science and Network Security*, Volume 8(2), pp. 339–344

Mooney, R.J., Bunescu, R., 2005. Mining Knowledge from Text using Information Extraction. *ACM SIGKDD Explorations Newsletter - Natural language processing and text mining*, Volume 7(1), pp. 3–10

Namiot, D., 2015. Twitter as Transport Layer Problem. *Artificial Intelligence and Natural Language and Information Extraction, Social Media and Web Search FRUCT Conference (AINL-ISMW FRUCT)*

Nnenna, E.J., Onyekachi, O.H., 2012. Mobile Positioning Techniques in GSM Cellular Networks: A Comparative Performance Analysis. *International Journal of Computer Technology and Electronics Engineering*, Volume 2(6), pp. 21–29

Shad, S.A., Chen, E.H., 2012. Precise Location Acquisition of Mobility Data using Cell-Id. *International Journal of Computer Science*, Volume 9(3), pp. 222–231

Smit, L., Stander, A., Ophoff, J., 2012. An Analysis of Base Station Location Accuracy within Mobile-Cellular Networks. *International Journal of Cyber Security and Digital Forensic*, Volume 1(4), pp. 272–279

Wang, X.L., Wong, A.K., Kong, Y.P., 2012. Mobility Tracking using GPS, Wi-Fi and Cell-ID. *In*: Proc. Of IEEE ICOIN, pp.171–176