Feng Cheng, Thanh-Dien Tran, Sebastian Roschke, Christoph Meinel: "A Specialized Tool for Simulating Lock-Keeper Data Transfer", in Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications (AINA 2010), IEEE Press, Perth, Australia, pp. 182-189, 4, 2010. ISBN: 978-0-7695-4018-4.

A Specialized Tool for Simulating Lock-Keeper Data Transfer

Feng Cheng Thanh-Dien Tran Sebastian Roschke Christoph Meinel
Hasso Plattner Institute(HPI), University of Potsdam
P.O.Box 900460, 14440, Potsdam, Germany
{feng.cheng, thanh-dien.tran, sebastian.roschke, christoph.meinel}@hpi.uni-potsdam.de

***Abstract*—Simulation is an efficient way to model a real system by a computer program and is used to study and evaluate the characteristics or behaviors of the system. In this paper, we present an effective simulation tool that is designed for simulating the working procedure of the Lock-Keeper system, which is a high level security device for physically separating two networks. Due to the special mechanism of data exchange within the Lock-Keeper system, it is a challenging task to specify the Lock-Keeper's performance for a given application scenario. To get enough performance data, the intuitive way is to practically conduct numerous testing experiments and then analyze their results, which is extremely time consuming. Therefore, we are motivated to design and implement a simulator to predict the Lock-Keeper performance theoretically. Compare with most of available network simulation tools, the proposed simulator is capable of simulating the transfer of application layer data, i.e., file based data streams. The simulator is built based on a simple model of the Lock-Keeper data exchange procedure. Information on a target application scenario can be specified within an XML file, which is used as the input for the later calculation. Several kinds of reports are generated by this specialized simulator to indicate how the data is exchanged through Lock-Keeper. To verify the simulation results, we conduct several experiments, which practically test data transfer for the scenarios using the real Lock-Keeper system. The comparison between theoretical simulation and practical testing proves the effectiveness of our proposed simulation tool.**

Keywords - Network and Communication, Simulation, Performance Measurement, Data Exchange, Lock-Keeper

I. Introduction

Over the past years, simulation has proved its capabilities in many areas. Concerning the computer and communication networks, simulation is often used to gain insight into functionality, performance, optimization of the target or to conduct related work on safety engineering, training, education, and others [1]. Currently, there are lots of tools (either open source or commercial) available for simulating the computer and communication networks, e.g., NS-2 [2], NS-3 [3], OPNET Modeler [4], OMNeT++ [5], and JiST/SWANS [6]. As one of the most popular network simulators, NS-2 has been proposed for years to provide substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. NS-3 is a discrete event network simulator for Internet systems intended to replace the popular NS-2 simulator, as it is not backwards-compatible with NS-2. With these existing simulation tools, common scenarios of computer and communication networks can be simulated. However, most of them concentrate on simulating the information on traffic, performance, QoS, etc., of the target system at the network (packet) level, e.g., Ethernet, IP, TCP, etc. Although some tools, such as NS-2 and OPNET, are claimed to be capable of simulating certain application protocols, such as FTP, Telnet, http, etc., in most cases they are not flexible enough to simulate complex application scenarios. To design suitable simulation tools for modeling application layer data transmission within different practical user cases remains to be a challenge.

As a high level security solution, Lock-Keeper has been proposed as an efficient device for separating two networks on the physical layer [7], [8], [9]. It consists of three independent PCs, i.e., INNER, GATE, and OUTER. Besides, a patented switching PCB (Printed Circuit Board) is deployed to automatically change the connection inside the Lock-Keeper for guaranteeing that there is only one connection existing at a certain moment, either between INNER and GATE or between OUTER and GATE. The normal network traffic will be stopped on one side of Lock-Keeper, i.e., INNER or OUTER. Meanwhile, application layer data (i.e, files) will be created after analyzing the received traffic based on the used application layer protocols. The files are then pulled by GATE through the short live connection between GATE and source. When the connection is switched, GATE pushes those files to the other side. Afterwards, a normal network connection is built to the original termination in the network on the other side. By this way, online attacks, which mostly depend on direct network layer connections, can be prevented because there are always no physical connections between both networks.

Due to the complexity of the entire working procedure, it is not easy to represent and evaluate the Lock-Keeper data exchange mechanism. In particular, it is always expected but unfortunately challenging to formally specify the Lock-Keeper's performance. Due to various kinds of possible deployment scenarios, the performance of Lock-Keeper can not be simply specified using some traditional parameters, such as bandwidth, packet flow, and packet count, etc. On the other hand, to get the Lock-Keeper's performance for a certain application scenario, a complicated process is often required. It might consist of conducting a testing experimental environment, designing the testing schemes to cover all the possible data transmission cases required by this application, performing the practical testings, analyzing the output data, and deducing the results. Therefore, we expect to have a simulation tool to easily configure parameters of the scenario and predict its performance results. Obviously, the above mentioned simulation tools do not meet our needs because of the shortage on simulating application layer data transmissions.

The rest of this paper is organized as follows. In Section II, we shortly present the principle of Lock-Keeper and challenges to simulate its data exchange mechanism. The simple model for representing data transfer of Lock-Keeper, the architecture as well as the implementation of the proposed simulation tool are introduced in Section III. A set of experimental results on the simulation tool is presented in Section IV. Finally, we provide a short summary and an outlook on future works in Section V.

II. Overview of Lock-Keeper Data Transfer

In this section, we briefly introduce the Lock-Keeper system and discuss the challenges to specify its performance and simulate the working procedure.

A. Lock-Keeper Principle

Lock-Keeper, an implementation of the ”Physical Separation” concept, is a modern security system which can entirely prevent specific attacks by physically separating two networks[7], [8], [9]. A higher level of security can be guaranteed by a hardware-based implementation of the concept, which works as a sluice to prevent hackers and malign data to break into the internal network by any means of online attacks.

As shown in Figure 1, a SingleGate Lock-Keeper system consists of three active, autonomous, and PC-based components: INNER, OUTER, and GATE, which are connected using a patented switch unit that restricts their communication. Each Lock-Keeper component has its own system units (CPU, RAM, hard disk, network cards, etc). On each component, there is also an independent operating system and additional programs which help to transfer or verify data, e.g., for scanning viruses by the third-party anti-virus software.

The printed circuit board (PCB) of Lock-Keeper enables and disables connections on the physical level, i.e., it interrupts the network connections. The PCB has a fixed interval T which determines how long the connection will exist. At each interval, the switch on the PCB exclusively connects either the INNER with the GATE or the GATE with the OUTER. Therefore, the INNER and the OUTER are never directly connected. The function and timing of this unit is autonomous and cannot be changed or disengaged by anyone who has access to the rest of the system. On the INNER and the OUTER, there are application modules which only allow supported traffic to be transferred to the GATE, i.e., other traffic will be denied.

As indicated in Figure 1, the data is sent to and stored on one of the two external Lock-Keeper components (OUTER or INNER) firstly. OUTER or INNER then detects whether there currently is a connection to the GATE available. In case that there is no connection, it will wait until the switch

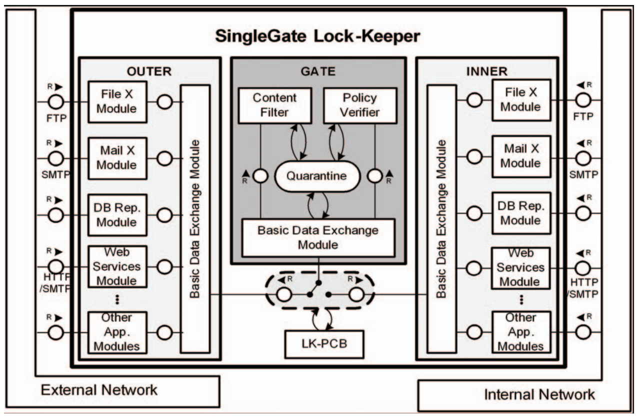


Fig. 1. Conceptual Architecture of Singlegate Lock-Keeper

establishes the connection. After this step, the data is transferred to the GATE where it is analyzed based on the security policy or other requirements. Once it has successfully passed the security check, for example virus scanning, the GATE detects whether a connection to the other external component (INNER/OUTER) is in place. If this is the case, the data is transferred and can now be forwarded to the other side of the Lock-Keeper system.

Because of the switching mechanism, the performance of the Lock-Keeper system is seriously affected. To improve its performance, the DualGate Lock-Keeper architecture is proposed [10]. In the DualGate Lock-Keeper system, there are two GATE components: GATE1 and GATE2. A new design of PCB is proposed to switch the connection states between GATE1-INNER/GATE2-OUTER and GATE1-OUTER/GATE2-INNER. With this new switching mechanism, files can be transferred as soon as they arrive the external computers without any other redundant waiting time [10]. The efficiency of DualGate Lock-keeper system is improved in most cases since there is no idle computer during the whole process as well as there is no need to wait for connections.

B. Simulation Challenges

Based on the above mentioned principle, the data exchange mechanism of Lock-Keeper has many special characteristics:

- **Keep the basic principle of *Physical Separation***: There is no direct physical connection between INNER and OUTER at any time. Data can not be continuously transferred inside Lock-Keeper.
- **Data is exchanged by a *Sluice* mechanism**: GATE works as an intermediate station for relaying data. There is only one door opened between GATE and INNER/OUTER, either INNER and GATE or OUTER and GATE. Meanwhile, the passing data should be checked for the purposes of access policy or other content-based security.
- **The physical connection inside Lock-Keeper is controlled by the hardware based PCB with a fixed interval T**: There is no possibilities to change the connection

Feng Cheng, Thanh-Dien Tran, Sebastian Roschke, Christoph Meinel: "A Specialized Tool for Simulating Lock-Keeper Data Transfer", in Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications (AINA 2010), IEEE Press, Perth, Australia, pp. 182-189, 4, 2010. ISBN: 978-0-7695-4018-4.

states. The transfer of a single file might be stopped by the switch and should be reliably resumed after the connection switches back.

- **Data is exchanged on the application layer**: Each component of the Lock-Keeper (INNER/ OUTER/ GATE) only transfers data in form of files. For the purpose of application layer scanning, the file should be relayed to the next stop only after all the content that belongs to the file has completely arrived and passed the scanning mechanism.
- **The bidirectional data transfer is supported**: Files can be transferred either from INNER to OUTER or from OUTER to INNER. During a single connection interval, data can be transferred bidirectionally too. At a certain point of connection interval, there could be maximal two file transfers exist.

Due to these features, the network secured by the Lock-Keeper significantly differs from traditional network. The popular network simulators are not capable of simulating the special working procedure of Lock-Keeper. On the other hand, the performance of Lock-Keeper data transfer highly depends on the practical application scenarios, i.e., average file size, arrival frequency, single or double directions, etc. To describe, explain, and measure the performance of Lock-Keeper for non-professional and normal Lock-Keeper customers, a new simulation tool, which can theoretically model the working mechanism of Lock-Keeper and thereafter predict the performance for a given scenario, is expected.

III. The Prototype of the proposed Simulation Tool

In this Section, we present a feasible architecture design for the expected simulation tool based on the working procedure of a simple but typical deployment of the SingleGate Lock-Keeper, as depicted in Figure 2. It includes a sending host and a receiving host that are respectively connected to OUTER and INNER of Lock-Keeper. When the sending host has a file to transfer, it will send that file to the OUTER and afterwards the file will be transmitted to the GATE when connection is available as shown in Figure 2(a). After the GATE receives the file from the OUTER, it will scan the file. If there is no problem with the content, the file will be sent to the INNER in the next available interval, as depicted in Figure 2(b). Similarly, files can also be transferred from the other direction.

A. Modeling Lock-Keeper Data Transfer

In order to simulate the data transfer of Lock-Keeper, we need a formal way to describe the data exchange mechanism of Lock-Keeper. A mathematical model to theoretically calculate the time required for transferring file(s) between components of Lock-Keeper is expected. Basically, a complete Lock-Keeper data transfer consists of two phases: from source to GATE (Phase I) and then from GATE to the target (Phase II). Both phases share the similar properties in spite that only an additional step for scanning the file before being put in

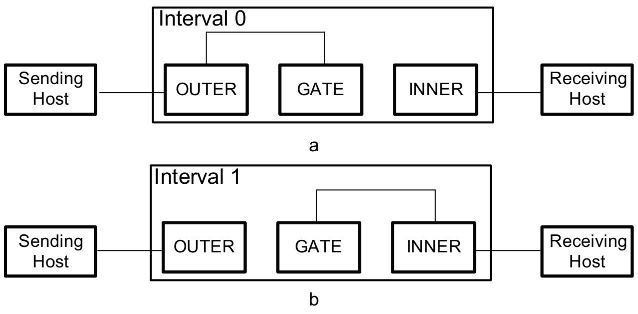


Fig. 2. Typical Deployment of a SingleGate Lock-Keeper

the queue is always required for Phase II. In this Section, we briefly describe a way to model how the data is processed in a phase of Lock-Keeper data transfer. The detail of a mathematical model of Lock-Keeper is proposed in [11].

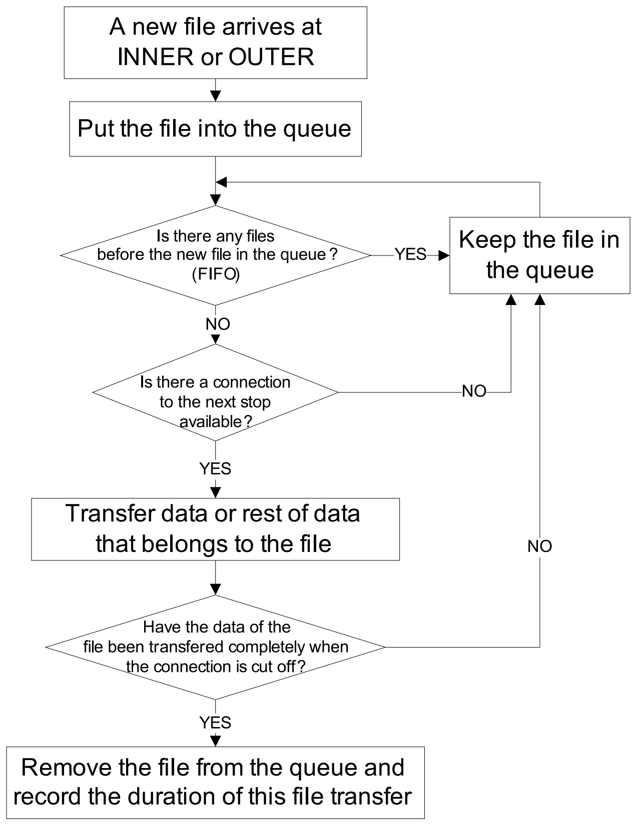

A new file arrives at INNER or OUTER
Put the file into the queue
Is there any files before the new file in the queue? (FIFO)
YES
Keep the file in the queue
NO
Is there a connection to the next stop available?
NO
YES
Transfer data or rest of data that belongs to the file
NO
Have the data of the file been transfered completely when the connection is cut off?
YES
Remove the file from the queue and record the duration of this file transfer

Fig. 3. Lock-Keeper File Transfer: Phase I or Phase II

The work flow of a data transfer phase, either Phase I or Phase II, can be depicted as Figure 3. When a Lock-Keeper component receives the file completely, it puts the file into the file queue for the next transfer. The queue can be processed according to different mechanisms, e.g., FIFO (First In First Out), FILO (First In Last Out), or other customized processing. Here we use FIFO as an example. The new file will wait in the queue until all the previous files in the queue completely finish their transfers. If currently there are no more files waiting before the new file in the queue, the process of the new file will be started immediately. First, the component detects the status of its connection with its neighbor component on the way to the destination of the file. If there is no connection available, the file has to be kept in the queue. If the connection is available, the data transmission for this new file can be started. The status of the network connection is monitored all the time. Whenever the connection is cut off, the file transfer will be stopped. The original file will be kept in the queue and the rest of its data will be transferred within the next interval while the physical connection comes back. If the file transfer is finished before the connection switches, the source file will be removed from the queue and some information about this data transfer (e.g., the starting time, the terminating time, and the duration of this file transfer, etc.) will be logged accordingly. Then, the transfer of the next file in the queue can be started.

Based on the above mentioned process, we propose a method to calculate the time to transfer a file among each component of Lock-Keeper as follows:

Algorithm for Calculating the Time Duration for a Phase of Lock-Keeper File Transfer

```
Input: a file
Do While <there is no connection available>
    wait
End Do
If <can transferred in the time left of the connection >
    Time = File Size / Throughput
Else
    Do While (File size > Interval * Throughput)
        File size = File size - Interval * Throughput
        Time = Time + Interval
    End Do
    Time = Time + File Size / Throughput
End IF
Output: Time to transfer file
```

B. The Architecture of the Simulation Tool

Based on the previous analysis, we present the architecture for our simulation tool as shown in Figure 4. To avoid the complexity, the tool proposed in this paper works only for the SingleGate Lock-Keeper. There are four major components: *Management*, *Scenario Editor*, *Simulator*, and *Visualization Interface*.

As described in Figure 4, the *Management* component controls the overall simulation process. To conduct a simulation, the user starts by specifying a scenario or loading an existing one. A new scenario can be created by using the *Scenario Editor*. Users can also create their scenario specifications by using other XML supported editors. After a scenario has been loaded, the *Simulator* module is able to be triggered. The *Simulator* will parse and analyze the scenario file as well as schedule the calculating tasks for file transfer (FT) agents (as depicted in Figure 5). After the calculation conducted by the respective agent, the results are stored in the simulation data file. This simulation data is used by end interface to produce

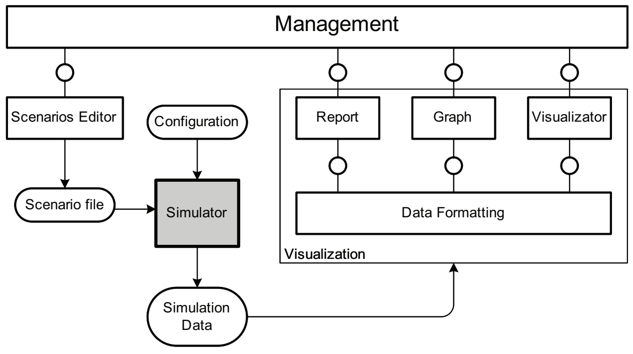


Fig. 4. The Architecture of the Lock-Keeper Simulation Tool

reports, graphs, or even to make a data flow visualization. Besides, some scenario-independent parameters of the *Simulator* can be specified in the *Configuration* file, e.g., the algorithms for creating random number, resolution of the graph, format of the output file, etc.

The *Simulator* is the core component of our simulation tool. As depicted in Figure 5, it comprises a number of FT agents and a scheduler to control FT agents. An FT agent is responsible for calculating the time to transfer a file. It handles the file queues on the each component of Lock-Keeper, and detects the status of the connection. Based on the information given in the scenario file, the scheduler knows when it has to send the request to which FT agent. When the FT agent receives a request from the scheduler, it calculates the time for transferring the file and returns the result to the scheduler. The FT agent is realized based on the Phase description in Figure 3. The algorithm described in Section III-A is implemented in the FT agent for calculating the time to transfer a file. When the scheduler receives the result from the FT Agent, it will instruct the *Result Processing* to postprocess and the store the result to the simulation data file.

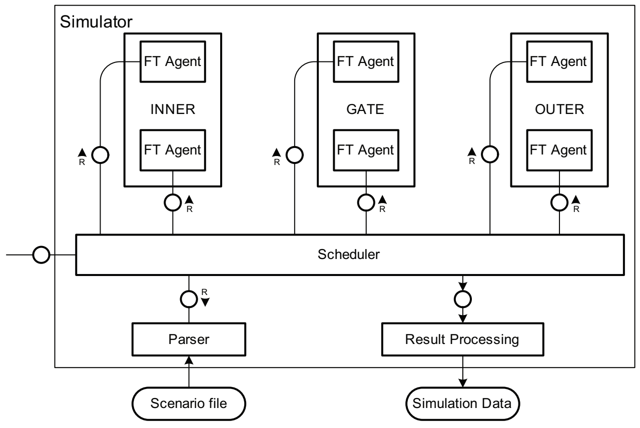


Fig. 5. The *Simulator* Component

C. The Structure of the Scenario File and the Simulation Data

As the input and output of the *Simulator* component, the scenario specification and the simulation result should be structured in the formal way.

Feng Cheng, Thanh-Dien Tran, Sebastian Roschke, Christoph Meinel: "A Specialized Tool for Simulating Lock-Keeper Data Transfer", in Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications (AINA 2010), IEEE Press, Perth, Australia, pp. 182-189, 4, 2010. ISBN: 978-0-7695-4018-4.

1) The Scenario Specification: The information required by the *Simulator* is specified in an XML file that contains Lock-Keeper parameters (e.g., the switching interval T, the network bandwidth between two possibly connected component, the version of Lock-Keeper, etc.) and descriptions on a given application scenario(the file size, the transfer direction, the number of the files, the arrival frequency, etc.). The scenario specification files can be simply created with the support of some easy-to-use interfaces provided by the *Scenario Editor*, stored in the scenario repository, and fed to the *Simulator*. An example of the scenario specification is shown below:

An Example of the Scenario Specification

```
<?xml version="1.0"?>
  <lk>
    <parameters>
      <interval>10</interval>
      <bandwidth>104875600</bandwidth>
    </parameters>
    <fileSets>
      <file>
        <fileName> File 1</fileName>
        <fileSize>12345</fileSize>
        <sendingHost>Internal</sendingHost>
        <receivingHost>External</receivingHost>
        <numberOfFile>1<numberOfFile>
        <putRate>0</putRate>
        <totalSize>1048576KB</totalSize>
        <roundTrip>0</ roundTrip>
      </file>
      ...
    </fileSets>
  </lk>
```

The scenario description consists of the information on all files to be transferred. Currently, we count only the following paraments:

- **fileSize**: the size of the file in Bytes.
- **sendingHost** and **receivingHost**: can be one of the following values INNER/ OUTER/ GATE/ Internal/ External
- **numberOfFile**: the number of files to be transferred. It is optional and only used for the scenario where multiple files need to be transferred.
- **putRate**: the rate that the files is sent from the source (in millisecond). This element is optional and only specified when the numberOfFile is specified.
- **totalSize**: the total size of all the files.
- **roundTrip**: is used for representing the transfer mode. The value could be set to either 0 or 1. If it is set to 1, the file will be transferred back as soon as it arrives at the destination.

2) The Simulation Data: The result produced by FT Agent is stored in the result file (also referred to as the simulation data). It is used to keep the detailed information about the journey of each file, such as when the file arrives at which component, when it is transferred to the next component, etc. It is also an XML file with the following structure:

An Example of the Simulation Data

```
<?xml version="1.0"?>
  <lk>
    <parameters>
      <interval>10</interval>
      <bandwidth>104875600</bandwidth>
      <startTime>10:20:30 08/12/2008 </startTime>
      <switchPosition>IN-GATE</swicthPosition>
      <putRate>0</putRate>
      <totalSize>1048576KB</totalSize>
    </parameters>
    <fileSets>
      <file>
        <fileName>...</fileName>
        <fileSize>...</fileSize>
        <sourceType>...</sourceType>
        <destinationType>...</destinationType>
        <t1>...</t1>
        <t2>...</t2>
        <t3>...</t3>
        <t4>...</t4>
        <t5>...</t5>
        <t6>...</t6>
        <t7>...</t7>
        <t8>...</t8>
      </file>
      ...
    </fileSets>
  </lk>
```

Where:

- **startTime**: the time when the simulation starts.
- **switchPosition**: the initial switching state when the Lock-Keeper is switched on. It is given randomly when a simulation instance is started, either IN-GATE or GATE-OUT.
- **t1**: the time when the sending host starts to send the file.
- **t2**: the time when the INNER/OUTER receives the entire file sent from the sending host.
- **t3**: the time when the INNER/OUTER starts to send the file to the GATE
- **t4**: the time when the file arrives at the GATE.
- **t5**: the time when the GATE starts to send the file to the OUTER/ INNER.
- **t6**: the time when the file, sent by the GATE, arrives at the OUTER/ INNER.
- **t7**: the time when the OUTER/INNER starts to send the file to the Receiving host.
- **t8**: the time when the entire file arrives at the receiving host.

D. Proof-of-Concept

Based on the above described architecture, a practical simulation tool is developed as a Proof-of-Concept(PoC). It can simulate most of possible scenarios of the SingleGate Lock-Keeper file transfer.

Using the tool, we test a scenario example for transferring *2096* files from the OUTER to the INNER. Each file has the random file size (between *2* and *1023* KB) and the total size of all files is *1* GB. The files are put one by one on OUTER every *250* ms. When all files reach its destination, the process will repeat in the reverse direction (from the INNER to the OUTER). We configure the *Simulator* for simulating the SingleGate Lock-Keeper with the switching interval *15* seconds and the bandwidth of the connection between the INNER/OUTER and GATE to be *100* Mbps. After processing this scenario file, our tool creates the report on the performance of the SingleGate Lock-Keeper, as depicted in Figure 6.

The detailed simulation results are visualized on the Figure 7. The two graphs on the left side describe the time duration while files are waiting on the OUTER/INNER and the GATE before the transmission (i.e., d2 and d4). The other two on the right side show the time duration of each file transfer (i.e., d3 and d5). As shown on the Figure 7, most of the files have to wait on the INNER and OUTER for a long time before being transferred (maximum is nearly *300*s). That is due to queue processing and connection switching. However, on the GATE the waiting time is much shorter (maximum is *30*s). The reason is that the queue length on GATE is usually very short, i.e., in most cases, the Phase II of the file transfer can be started immediately after the switch changes. In comparison with the waiting time, the transmission duration is much shorter. Only some peaks happen in case of the file transfer is interrupted by the connection switch while the rest of this file needs to wait a whole switching interval for being transferred. It can be recognized that the waiting time is one of the main factors which affects the performance of Lock-Keeper.

IV. EXPERIMENTS

In this Section, we run the simulations as well as conduct practical testings for several application scenarios of Lock-Keeper file transfer. The simulation data is compared with the real-world testing data. The parameters used by our simulation tool are listed as follows:

- **T**: *15* seconds
- **Link speed**: *100* Mbps

A. Transfer a Large File

In this scenario, we try to transfer one *1* GB file from OUTER to INNER. When the file arrives at its destination it will be sent back immediately. The results of the simulation and the practical experiment are shown in Table I.

TABLE I
THE DURATION FOR TRANSFERRING A 1GB LARGE FILE

Direction	Simulation (s)	Practice (s)
OUTER -> INNER	521.17	1124.86
INNER -> OUTER	505.55	1176.61

As shown on Table I, we separately list the overall duration for each "ONE-WAY TRANSFER", i.e., the sum of d2, d3, d4, and d5 on the way of OUTER to INNER or INNER to OUTER. It takes about *521* seconds in theory (but *1125* seconds in practice) to transfer the file from the OUTER to the INNER and *506* seconds in theory (but *1177* seconds in practice) to transfer the file from the INNER to the OUTER. The simulation results are faster, more than double times, than the practical experiment. One of the reasons could be the transmission rate (between INNER/OUTER and GATE) in practice is actually different with the transmission rate in theory although both are based on the same link speed of *100* Mbps. The transmission rate (i.e., how much the overall link speed can be actually reached, usually also called as bandwidth) used by our simulation follows the calculation method proposed in [12]. We use a fixed window size (*17* KB) in the simulation while the window size is changed all the time in practice by the Slow-start algorithm applied for TCP congestion control [13][14][15].

B. Transfer Multiple Files with the Same Size

In this experiment, we transfer *1000* files (each file has the same size *10* KB) with different putting rate (frequency). In the first case, we put all files on OUTER at the same time. As soon as all files arrive at the other side, the testing is repeated from the other direction. In the next case, we put one file every *250* ms on OUTER. The testing is repeated from the other direction after getting all files on the other side. In the following cases, the putting rate on OUTER are respectively *500*, *750* and *1000* ms/file. We count the time duration for transferring all the files from OUTER to INNER and then from INNER to OUTER as a "RETURN TRANSFER". The results are described in Table II.

TABLE II
THE OVERALL DURATION FOR A RETURN TRANSFER OF 1000 FILES WITH THE SAME FILE SIZE 10 KB

Put Rate (ms)	Simulation (s)	Practice (s)
0	190.28	522.60
250	514.07	519.74
500	1037.04	995.46
750	1537.94	1521.11
1000	2046.14	2028.54

C. Transfer Multiple Files with Random Size

In this experiment, we transfer *2096* files from OUTER to INNER and back. The files are randomly created with the size between *2* and *1024* KB and total size of *1* GB. The testings are repeated by cases of putting rate *250*, *500*, *750* and *1000* ms/file. Similarly, we just count the time required for finishing the transfer of all the files from both directions as a "RETURN TRANSFER". Table III shows the results.

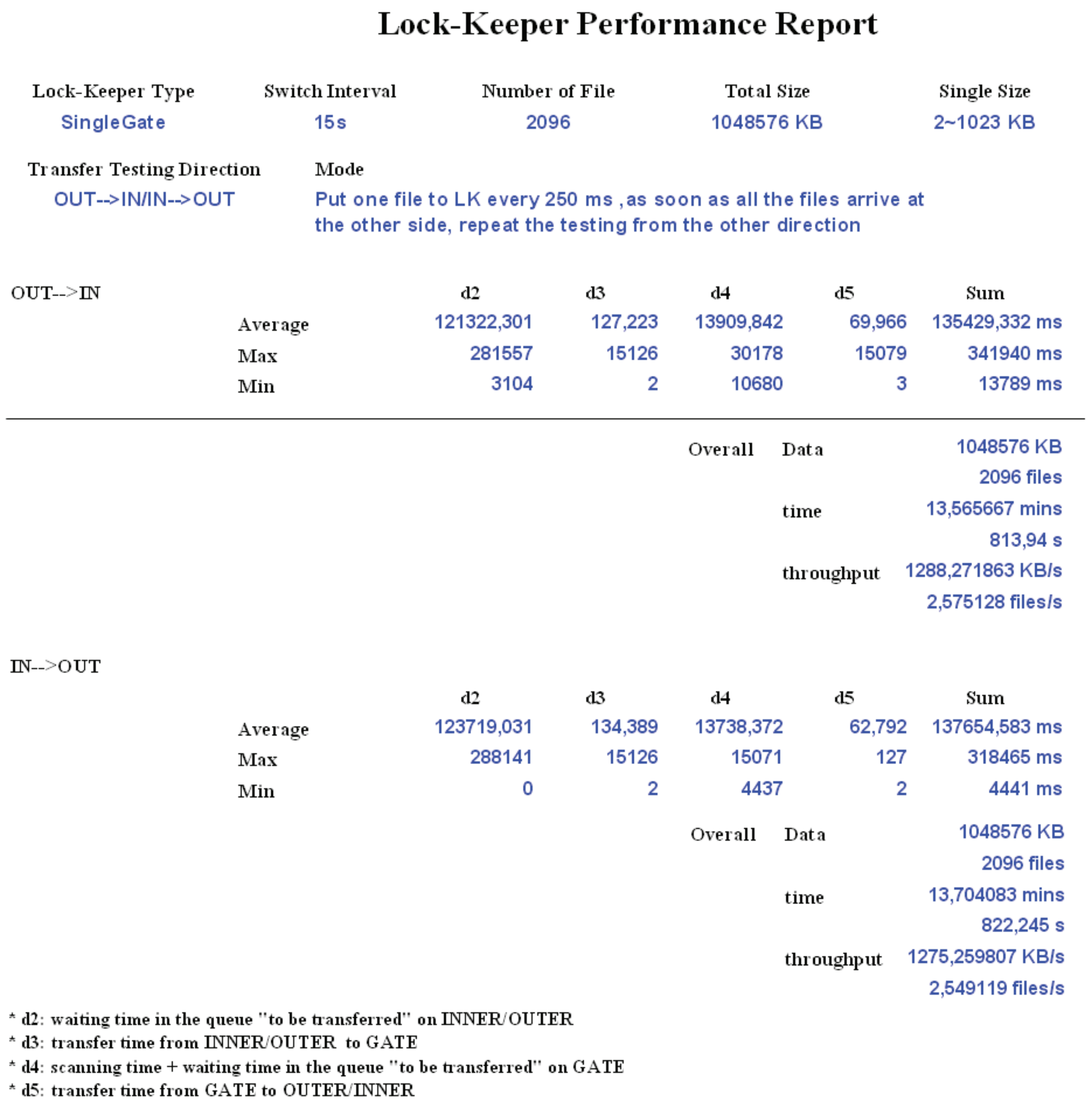
TABLE III
THE OVERALL DURATION FOR A RETURN TRANSFER OF 2096 FILES WITH RANDOM FILE SIZE

Put Rate (ms)	Simulation (s)	Practice (s)
0	1931.19	2175.48
250	1636.18	1971.79
500	2127.36	2131.28
750	3166.69	3151.43
1000	4210.82	4217.55

D. Preliminary Analysis of Results

As shown previously, our simulation tool works for most of the possible scenarios of file transfer through the Lock-Keeper

Feng Cheng, Thanh-Dien Tran, Sebastian Roschke, Christoph Meinel: "A Specialized Tool for Simulating Lock-Keeper Data Transfer", in Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications (AINA 2010), IEEE Press, Perth, Australia, pp. 182-189, 4, 2010. ISBN: 978-0-7695-4018-4.



Lock-Keeper Performance Report

Lock-Keeper Type	Switch Interval	Number of File	Total Size	Single Size
SingleGate	15s	2096	1048576 KB	2~1023 KB

Transfer Testing Direction	Mode
OUT-->IN/IN-->OUT	Put one file to LK every 250 ms ,as soon as all the files arrive at the other side, repeat the testing from the other direction

OUT-->IN

	d2	d3	d4	d5	Sum
Average	121322,301	127,223	13909,842	69,966	135429,332 ms
Max	281557	15126	30178	15079	341940 ms
Min	3104	2	10680	3	13789 ms

Overall		
	Data	1048576 KB
		2096 files
	time	13,565667 mins
		813,94 s
	throughput	1288,271863 KB/s
		2,575128 files/s

IN-->OUT

	d2	d3	d4	d5	Sum
Average	123719,031	134,389	13738,372	62,792	137654,583 ms
Max	288141	15126	15071	127	318465 ms
Min	0	2	4437	2	4441 ms

Overall		
	Data	1048576 KB
		2096 files
	time	13,704083 mins
		822,245 s
	throughput	1275,259807 KB/s
		2,549119 files/s

* d2: waiting time in the queue "to be transferred" on INNER/OUTER
* d3: transfer time from INNER/OUTER to GATE
* d4: scanning time + waiting time in the queue "to be transferred" on GATE
* d5: transfer time from GATE to OUTER/INNER

Fig. 6. The Simulation Report of Lock-Keeper File Transfer: An Example

system. The comparison results show that the simulation data is highly similar with the results we got from the practical testing experiments, which proofs the accuracy of our proposed simulation tool. For such a special scenario as transferring a very large file, the simulation tool needs to be optimized. More factors can be taken into account for improving the accuracy, e.g., size of congestion window, the time to process files in queue, the overhead of the switch, the time to scan files, etc. Furthermore, to simplify our simulation as well as the setup of practical experiments, we do not consider the communication between the *Sending Host* or *Receiving Host* and the Lock-Keeper. It makes sense to extend the simulation tool by integrating some possible influential factors, such as the transmission rate of receiving data from the external *Sending Host* and sending data to the external *Receiving Host*.

V. Conclusion

In this paper, we propose a specialized simulation tool for simulating the behaviors of data communication at the application layer, especially, for being applied to predict the performance of file transfer. The simulation tool is built based on a simple model which can formally represent the data exchange mechanism of Lock-Keeper. The tool can be used to simulate most of the Lock-Keeper application scenarios. A prototype implementation is presented in the paper. Besides, the simulation is verified by comparing the outputs from the

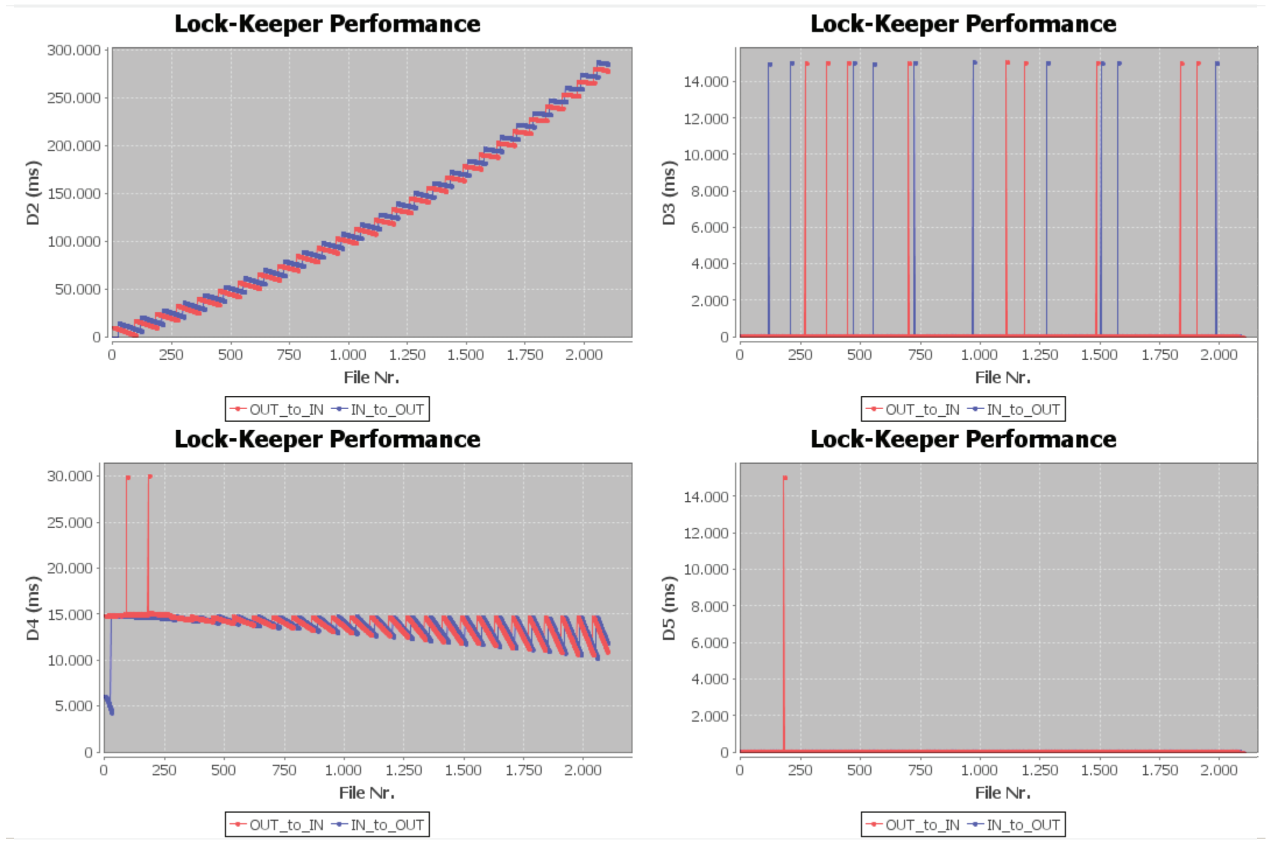


Fig. 7. The Visualization of the Simulation Results

simulation tool and results from the practical testing experiments. The tool can help to demonstrate the working procedure of Lock-Keeper as well as estimate its performance in case of customer-specified scenarios. Many future works are expected to optimize the tool, e.g., introducing new parameters for the the basic model, simulating and testing the DualGate Lock-Keeper, integrating the communication between Lock-Keeper (i.e., INNER or OUTER) with source and target hosts of the file, combining our application layer simulator with existing network simulation framework, such as NS-3 or OMNeT++ [16], visualizing the whole procedure on how the data is processed/transferred by the Lock-Keeper, etc.

References

[1] R. D. Smith, *Simulation Article*, ch. Encyclopedia of Computer Science. 4th edition ed., July 2000.

[2] K. Fall and K. Varadhan, *The ns Manual (formerly ns Notes and Documentation)*. UC Berkeley, LBL, USC/ISI, and Xerox PARC, January 2009.

[3] "NS-3 Project WebSite: http://www.nsnam.org/." (accessed January 2010).

[4] "OPNET Modeler WebSite: http://www.opnet.com/." (accessed January 2010).

[5] "OMNeT++ Project WebSite: http://www.omnetpp.org/." (accessed January 2010).

[6] "JiST/SWANS Project WebSite: http://jist.ece.cornell.edu/." (accessed January 2010).

[7] F. Cheng and C. Meinel, *Lock-Keeper: A New Implementation of Physical Separation Technology*, ch. Securing Electronic Business Processes - Highlights of the Information Security Solutions Europe, pp. 275–286. Vieweg-Verlag, October 2006.

[8] F. Cheng and C. Meinel, "Research on the Lock-Keeper technology: Architectures, applications and advancements," *International Journal of Computer & Information Science.*, vol. 5(3), pp. 236–245, 2004.

[9] "Lock-Keeper WebSite in Siemens Switzerland: http://www.siemens.ch/." (accessed January 2010).

[10] F. Cheng, C. Meinel, and et.al., "The dualgate Lock-Keeper: A highly efficient, flexible and applicable network security solution," in *Proceedings of the 4th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'03)*, (Luebeck, Germany), pp. 152–159, October 2003.

[11] S. Roschke, F. Cheng, T.-D. Tran, and C. Meinel, "A theoretical model of Lock-Keeper data exchange and its practical verification," in *Proceedings of the 6th IFIP International Conference on Network and Parallel Computing (NPC 2009)*, (Gold Coast, Australia), pp. 190–196, October 2009.

[12] R. Weiss, "File transfer protocol (FTP) throughput testing," white paper, JDS Uniphase Corporation, 2009.

[13] L. Brakmo and L. Perterson, "TCP vegas: End-to-end ongestion avoidance on a global internet," *IEEE Journal on Selected Areas in Communications*, vol. 13(8), pp. 1465 – 1480, 1995.

[14] J. C. Hoe, "Improving the start-up behavior of a congestion control scheme for TCP," in *Proceedings of the 1996 ACM Special Interest Group on Data Communication (SIGCOMM'96)*, (Stanford, USA), pp. 270–280, August 1996.

[15] R. Wang, K. Pau, G.;and Yamada, M. Sanadidi, and M. Gerla, "TCP, startup performance in large bandwidth delay networks," in *Proceedings of 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'04)*, (Hong Kong, China), pp. 796–805, March 2004.

[16] A. Varga, "The OMNeT++ discrete event simulation system," in *Proceedings of the 15th European Simulation Multiconference (ESM'01)*, (Prague, Czech Republic), pp. 319–324, June 2001.