SC4MEC: Automated Implementation of A Secure Hierarchical Calculus for Mobile Edge Computing

Jiaqi Yin* Huibiao Zhu* Yuan Fei†
*Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, China
†School of Information, Mechanical and Electrical Engineering, Shanghai Normal University, Shanghai, China

***Abstract*—Mobile Edge Computing (MEC), as an emerging technology, is proposed to solve the time delay problem in 5G era, especially in the field of autonomous driving. The core idea of MEC is to offload the task to the nearest device/server for computation, i.e., sinking the computation, so as to reduce the delay and congestion. Actually, there are lots of researches on MEC offloading strategy, but there is little research on the computation of its offloading characteristics. Therefore, in this paper, we first propose a secure hierarchical calculus SC4MEC to describe the features of MEC. We also give the syntax and operational semantics of this calculus from the process and network levels, and simulate the calculus in Maude. Meanwhile, local ecology is applied to the communication channel to further reduce the authentication delay of the device with the same identity and ensure the security of the transmission data. We also propose to extend the communication radius of MEC server or cloud server by rule *Enlarge*, in order to ensure the mobile devices's connectivity while the consumption of resources is minimized. Finally, we employ SC4MEC calculus to a small example about device to device communication with automated implementation.**

***Index Terms*—MEC, Secure Hierarchical Calculus, Semantics, Maude**

I. Introduction

With the rapid development of mobile Internet, smart phones have become an indispensable tool in people's daily life, and the functions of smart phone applications are becoming more and more powerful to meet the needs of people in social circumstances–shopping, travel, entertainment, etc. However, for the limited computing and storage resources, and battery capacity of mobile terminals, some applications may not be able to meet the requirements of processing capacity and battery life. How to solve the contradiction between the limited resources of mobile terminals and the large amount of resources consumed by application process has become one of the main problems to be solved in mobile communication network.

Computational offloading technology is considered to be one of the key technologies to solve the above problems. In the network framework of Mobile Edge Computing (MEC) [1], the mobile terminal can unload the task to the nearby edge computing server to run. Mobile terminals only need to send computing tasks and receive computation results, without occupying local computing and storage resources, so it can effectively solve the problem of limited mobile terminal resources. In addition, the computing offloading technology can enhance the service life of the mobile terminal and avoid the battery power consumption of the terminal during the local computing.

✉Huibiao Zhu: hbzhu@sei.ecnu.edu.cn

Currently, MEC has attracted much attention. Most of the work [2], [3] focused on the allocation of computing resources based on MEC, including single node and multi-node. Their goals are to minimize delay or energy consumption or to achieve a balance between the above two aims. In addition, there is also much work [4]–[6] on the offloading strategy, mainly including full offloading and partial offloading. Their goals and final evaluation standards are to see how much energy is saved. However, to the best of our knowledge, there is nearly no work to describe the computational offloading characteristics of MEC from the perspective of process calculus, and also there is little work to analyze the security of MEC from the perspective of formal methods.

Hence, in this paper, we describe the above work from a new perspective. The main contributions are as follows:

- A secure hierarchical calculus SC4MEC is proposed with syntax, semantics and transformation rules, which can be successfully applied to simulate the task offloading example.
- Local ecology is applied to the communication channel $friendly$ to further reduce the authentication delay of the device with the same identity and ensure the security of the transmission data.
- The SC4MEC calculus is implemented in Maude to simulate the realistic small example automatically. Additionally, through enlarging the communication radius of MEC server or cloud server, the calculus can ensure the connectivity of the mobile devices while the power consumption is minimized.

Structure of the paper. The remainder of this paper is organized as follows: Section II gives the syntax and semantics of the SC4MEC calculus from the perspective of process and network levels. In Section III, a small example on the basis of the calculus is automatically implemented in the rewrite engine Maude. Section IV shows the conclusions and future work.

II. The SC4MEC Calculus

A. Syntax

The SC4MEC calculus given in Table I contains three parts–processes, networks and functions, which is detailedly explained as follows:

Processes Level. (1) 0 indicates the termination.
(2) $P \parallel Q$ means processes P and Q run in parallel.
(3) $P + Q$ shows the whole process could be P or Q.
(4) $A(\widetilde{x})$ represents the recursion in the process.
(5) $[w1 = w2]P, Q$ means the conditional choice. If w_1 is equal to w_2, process P executes; otherwise, process Q.

TABLE I
THE SC4MEC SYNTAX

Processes	$P, Q ::= 0$	(termination)
	$\mid P \parallel Q$	(parallel)
	$\mid P + Q$	(nondeterministic choice)
	$\mid A(\widetilde{x})$	(recursion)
	$\mid [w1 = w2]P, Q$	(conditional)
	$\mid a!\langle k, v\rangle .P$	(send)
	$\mid a?(x, y).P$	(receive)
	$\mid \lfloor a!\langle k, v\rangle .a?(x, y)\rfloor .P$	(send and wait to receive)
	$\mid \lfloor a?(x, y).Calculate.a!\langle x', y'\rangle\rfloor .P$	(receive, calculate and then send)
	$Calculate ::= x := e$	
Networks	$N ::= 0$	(termination)
	$\mid N_1 \mid\mid N_2$	(parallel combination)
	$\mid n^{\eta}[P]^{r}_{l,\delta}$	(network node)
	$\eta ::= D \mid M \mid C$	(device type)
	$\delta ::= public \mid protected \mid friendly$	(channel type)
Functions	$F ::= Dis(l_1, l_2)$	(distance function)
	$\mid Type(k)$	(data type)

(6) $a!\langle k, v\rangle .P$ indicates that after sending message with type k and value v through channel a, process P works.

(7) $a?(x, y).P$ means that after receiving message from channel a, process P works.

(8) $\lfloor a!\langle k, v\rangle .a?(x, y)\rfloor .P$ shows the data interaction about the sender. The sender first transmits the message from channel a to other nodes for task offloading and then receives the new calculated data from the same channel.

(9) $\lfloor a?(x, y).Calculate.a!\langle x', y'\rangle\rfloor .P$ represents that the data interaction about the receiver. The receiver gets the message from the task offloading sender, calculates the data and finally sends the new message out.

(10) $Calculate$ process can contain four different notations: assignment, sequential composition, conditional and iteration. We take $x := e$ as an example to show the computation action.

Networks Level. (1) 0 means the network terminates.

(2) $N_1||N_2$ shows the networks N_1 and N_2 run in parallel.

(3) $n^{\eta}[P]^{r}_{l,\delta}$ represents the network node in the system. n is the name of the node; η is the type of the node, including D (Device), M (MEC Server), C (Cloud Server); l is the location of the node representing by three-dimensional coordinates; r is the radius of the node; δ is the type of the channel, including $public$, $protected$ and $friendly$.

Here, $public$ means the channel is without any protection and can be used for broadcasting service according to different scene requirements; $protected$ shows the channel is protected and can be selected for communication with different ID; $friendly$ indicates that if the devices with the same ID belong to the same ecology, the $friendly$ channel can be directly used for non encryption and decryption communication to reduce the delay and energy consumption by the unique user.

Functions Level. (1) $Dis(l_1, l_2)$ calculates the distance between two nodes based on three dimensional space coordinates.

(2) $Type(k)$ is employed to identify the four types of the message. Here, we first briefly introduce the four kinds of data types in the MEC task offloading communication, named as K_1, K_2, K_3 and K_4, respectively:

• Data type 1 (K_1): small data input and small data output. As a case in point, the mobile terminal temporarily processes some message information and makes quick feedback. The task offloads in the device level, i.e. Device to Device (D2D).

• Data type 2 (K_2): large data input and large data output. By way of illustration, users convert the format of some files or videos with large amount of data. The task offloads in the MEC server level, i.e. Device to MEC (D2M).

• Data type 3 (K_3): small data input and large data output. For instance, users request audio and video files. The task offloads in the MEC server level, i.e. Device to MEC (D2M).

• Data type 4 (K_4): large data input and small data output. For example, a variety of information is integrated in the scene where the mobile terminal is located. The task offloads in the cloud server level, i.e. Device to MEC then to Cloud (D2C).

B. Operational Semantics

Here, we give the process and network levels of label transition semantics in Table II and Table III, respectively.

Process Level of Transition Semantics. The transition rule can be described as $P \xrightarrow{\alpha} P'$, where P means the original process, and after α action, it changes into a primed process P'. Here, the detailed explanations of Table II are as follows:

• Rule (Parallel) describes the transition of parallel processes. Since the two processes are parallel, any change in either process will not affect the results of the other.

• Rules (Nondeterministic and Recursion) show the nondeterministic choice and the recursion of the program in that order.

• Rules (Then and Else) explain the conditional choice. If the condition is sastified, the process is P; otherwise, Q.

• Rules (Send and Receive) represent the sending and receiving messages respectively. No matter sending or receiving messages, once the action is completed, it becomes P.

• Rules (Send-Rec and Rec-Cal-Send) mean the interaction communication from the perspective of sender and receiver process. *Send-Rec* shows that the process first sends the message needing to be calculated and then waits for the new data information back. *Rec-Cal-Send* means that after receiving the message needing to be calculated, the process performs the computation and then sends the new message back.

Network Level of Transition Semantics. For the space limit, Table III takes the parallel nodes as an example to depict the network level of transition semantics as below:

• Rule (Parallel) shows the parallism of the different networks. It shows that there is no interference between them.

• Rule (Move) is applied to device level, and it represents node mobility, i.e., once a node moves, only the position l' changes, the rest parameters remain unchanged.

• Rule (Enlarge) is employed to the MEC server and cloud server level, and it indicates that once a device node is out of the communication range of the server node, the server node can expand the radius temporarily to extend coverage. The rule

TABLE II
PROCESS LEVEL OF LABEL TRANSITION SEMANTICS

(Parallel)	$\frac{P \xrightarrow{\alpha} P'}{P \parallel Q \xrightarrow{\alpha} P' \parallel Q}$	**(Nondeterministic)**	$\frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P' + Q}$	**(Recursion)**	$\frac{A(\widetilde{x}) \overset{def}{=\!=} P}{A(\widetilde{y}) \rightarrow P\{\widetilde{y}/\widetilde{x}\}}$
(Then)	$\frac{w_1 = w_2}{\lfloor w_1 = w_2 \rfloor P, Q \rightarrow P}$	**(Else)**	$\frac{w_1 \neq w_2}{\lfloor w_1 = w_2 \rfloor P, Q \rightarrow Q}$	**(Send)**	$\frac{-}{a!\langle k, v\rangle .P \xrightarrow{a!\langle k,v\rangle} P}$
(Receive)	$\frac{-}{a?(x,y).P \xrightarrow{a?(x,y)} P}$	**(Send-Rec)**	$\frac{-}{\lfloor a!\langle k, v\rangle .a?(x,y)\rfloor .P \xrightarrow{a!<k,v>} a?(x,y).P}$	**(Rec-Cal-Send)**	$\frac{Calculate ::= \ x := e}{\lfloor a?(x,y).Calculate.a!\langle x', y'\rangle\rfloor .P \xrightarrow{a?(x,y)} a!\langle x'/x, y'/e\rangle .P}$

TABLE III
NETWORK LEVEL OF LABEL TRANSITION SEMANTICS

(Parallel)	$\frac{N_1 \rightarrow N_1'}{N_1 \parallel N_2 \rightarrow N_1' \parallel N_2}$
(Move)	$\frac{P \xrightarrow{\alpha} P'}{n_1^{\eta}[P]^{r_1}_{l_1,\delta_1} \xrightarrow{\alpha} n_1^{\eta}[P']^{r_1}_{l_1',\delta_1}, \eta = D}$
(Enlarge)	$\frac{P \xrightarrow{\alpha} P'}{n_1^{\eta}[P]^{r_1}_{l_1,\delta_1} \xrightarrow{\alpha} n_1^{\eta}[P']^{r_1'}_{l_1,\delta_1}, \eta = M,\ C}$

Note: For the space limit, the following six rules are given as an example from the sender's perspective, where $P_1 = a!\langle k, v\rangle .P_1'$ and $P_i = a?(x,y).P_i, i = 2, 3, 4$.

(D2D-I) $$\frac{Type(k) = K_1 \wedge Dis(l_1, l_2) \leq r_1 + r_2 \wedge \delta_1 = \delta_2 \wedge P_1 \xrightarrow{\alpha} P_1'}{n_1^D[P_1]^{r_1}_{l_1,\delta_1} \parallel n_2^D[P_2]^{r_2}_{l_2,\delta_2} \xrightarrow{\alpha} n_1^D[P_1']^{r_1}_{l_1,\delta_1} \parallel n_2^D[P_2]^{r_2}_{l_2,\delta_2}}$$

(D2D-II) $$\frac{Type(k) = K_1 \wedge Dis(l_1, l_2) > r_1 + r_2 \wedge \delta_1 = \delta_2 \wedge P_1 \xrightarrow{\alpha} P_1'}{n_1^D[P_1]^{r_1}_{l_1,\delta_1} \parallel n_2^D[P_2]^{r_2}_{l_2,\delta_2} \xrightarrow{\alpha} n_1^D[P_1]^{r_1}_{l_1,\delta_1} \parallel n_2^D[P_2]^{r_2}_{l_2,\delta_2}}$$

(D2M) $$\frac{(Type(k) = K_2 \vee Type(k) = K_3) \wedge Dis(l_1, l_2) \leq r_1 + r_2 \wedge \delta_1 = \delta_2 \wedge P_1 \xrightarrow{\alpha} P_1'}{n_1^D[P_1]^{r_1}_{l_1,\delta_1} \parallel n_2^M[P_2]^{r_2}_{l_2,\delta_2} \xrightarrow{\alpha} n_1^D[P_1']^{r_1}_{l_1,\delta_1} \parallel n_2^M[P_2]^{r_2}_{l_2,\delta_2}}$$

(M2C) $$\frac{Type(k) = K_4 \wedge Dis(l_1, l_2) \leq r_1 + r_2 \wedge \delta_1 = \delta_2 \wedge P_1 \xrightarrow{\alpha} P_1'}{n_1^M[P_1]^{r_1}_{l_1,\delta_1} \parallel n_2^C[P_2]^{r_2}_{l_2,\delta_2} \xrightarrow{\alpha} n_1^M[P_1']^{r_1}_{l_1,\delta_1} \parallel n_2^C[P_2]^{r_2}_{l_2,\delta_2}}$$

(D2C-I) $$\frac{Type(k) = K_4 \wedge Dis(l_1, l_2) \leq r_1 + r_2 \wedge Dis(l_2, l_3) \leq r_1 + r_3 \wedge \delta_1 = \delta_2 = \delta_3 \wedge P_1 \xrightarrow{\alpha} P_1'}{n_1^D[P_1]^{r_1}_{l_1,\delta_1} \parallel n_2^M[P_2]^{r_2}_{l_2,\delta_2} \parallel n_3^C[P_3]^{r_3}_{l_3,\delta_3} \xrightarrow{\alpha} n_1^D[P_1']^{r_1}_{l_1,\delta_1} \parallel n_2^M[P_2]^{r_2}_{l_2,\delta_2} \parallel n_3^C[P_3]^{r_3}_{l_3,\delta_3}}$$

(D2C-II) $$\frac{Type(k) = K_i \wedge Dis(l_1, l_2) > r_1 + r_2 \wedge Dis(l_1, l_3) > r_1 + r_3 \wedge Dis(l_1, l_4) \leq r_1 + r_4 \wedge \delta_1 = \delta_4 \wedge P_1 \xrightarrow{\alpha} P_1'}{n_1^D[P_1]^{r_1}_{l_1,\delta_1} \parallel n_2^D[P_2]^{r_2}_{l_2,\delta_2} \parallel n_3^M[P_3]^{r_3}_{l_3,\delta_3} \parallel n_4^C[P_4]^{r_4}_{l_4,\delta_4} \xrightarrow{\alpha} n_1^D[P_1']^{r_1}_{l_1,\delta_1} \parallel n_2^D[P_2]^{r_2}_{l_2,\delta_2} \parallel n_3^M[P_3]^{r_3}_{l_3,\delta_3} \parallel n_4^C[P_4]^{r_4}_{l_4,\delta_4} \ (i = 1, 2, 3, 4)}$$

can ensure the continuity of mobile device node communication and also reduce the cost of one more server node.

• Rule (D2D-I) means that the message type is K_1. Both nodes, n_1 and n_2, are mobile devices and in their own communication range. In addition, they have the same channel type. Thus, after the action α, n_1 and n_2 can communicate successfully and then n_1 can make changes.

• Rule (D2D-II) shows as an counterexample of D2D-I to indicate that if one of the conditions is not satisfied, such as the distance of the two nodes is out of the max range, n_1 and n_2 cannot communicate smoothly.

• Rule (D2M) represents that the message type is K_2 or K_3. n_1 and n_2 are mobile devices and MEC server, respectively. Meanwhile, they are in the scope of the communication, have the same channel type and can interact with each other.

• Rule (M2C) depicts the message type is K_4. n_1 and n_2 are MEC server and cloud server, respectively. They are in the coverage of the communication and have the same channel type as well. Thus, they can communicate with each other.

• Rule (D2C-I) illustrates the message type is K_4. n_1, n_2 and n_3 are mobile devices, MEC server and cloud server, respectively. In addition, they own the same channel type and are within the scope of communication. Hence, the interaction is executed smoothly.

• Rule (D2C-II) presents the scene that the mobile device is only in the range of the cloud server. Thus, no matter what the message type is, the mobile device just communicates with the cloud server, which is a way to sacrifice time for continuous reading and writing.

Overall, the last six rules illustrates features of the MEC task offloading and the communcation conditions of each data type.

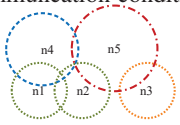

Fig. 1. Example of D2D Communication.

C. Example

Here, we give an example in the device level communication in Fig. 1: Assuming that nodes n_1, n_2 and n_3 are all mobile divices, where the first two nodes belong to the same ID user, i.e., the devices can calculate and communicate with the same database, and n_3 is the node which is out of the commnucation domain of the other two devices. Node n_4 belongs to the MEC server and n_5 belongs to the cloud server. From the figure, we can find that n_3 is out of the range of the nodes n_1 and n_2, but can communicate with the node n_5 directly. Fig. 1 illustrates that five different devices make up the whole network. Their information and migration is shown as below.

$$\begin{aligned} N =_{df} \ & n_1{}^D[\lfloor a!\langle k, v\rangle .a?(x,y)\rfloor .P_1]^{rl}_{ll,\ friendly} \\ \parallel \ & n_2{}^D[\lfloor a?(x,y).Calculate.a!(x',y')\rfloor .P_2]^{r2}_{l2,\ friendly} \\ \parallel \ & n_3{}^D[a!\langle k, v\rangle .P_3]^{r3}_{l3,\ public} \\ \parallel \ & n_4{}^M[\lfloor a?(x,y).Calculate.a!(x',y')\rfloor .P_4]^{r4}_{l4,\ public} \\ \parallel \ & n_5{}^C[\lfloor a?(x,y).Calculate.a!(x',y')\rfloor .P_5]^{r5}_{l5,\ public} \end{aligned}$$

Then the migration paths are described as below.

$$\begin{aligned}
Step\ 1:\ N \xrightarrow{a!<k,v>} N_1 =_{df}\ & {n_1}^D[a?(x,y).P_1]^{r1}_{l1,\ friendly} \\
\parallel\ & {n_2}^D[Calculate.a!(x',e)\rfloor.P_2]^{r2}_{l2,\ friendly} \\
\parallel\ & {n_3}^D[a!\ \langle k,v\rangle.P_3]^{r3}_{l3,\ public} \\
\parallel\ & {n_4}^M[\lfloor a?(x,y).Calculate.a!(x',y')\rfloor.P_4]^{r4}_{l4,\ public} \\
\parallel\ & {n_5}^C[\lfloor a?(x,y).Calculate.a!(x',y')\rfloor.P_5]^{r5}_{l5,\ public}
\end{aligned}$$

$$\begin{aligned}
Step\ 2:\ N_1 \xrightarrow{\tau} N_2 =_{df}\ & {n_1}^D[a?(x,y).P_1]^{r1}_{l1,\ friendly} \\
\parallel\ & {n_2}^D[a!(x'/x,y'/e)\rfloor.P_2]^{r2}_{l2,\ friendly} \\
\parallel\ & {n_3}^D[a!\ \langle k,v\rangle.P_3]^{r3}_{l3,\ public} \\
\parallel\ & {n_4}^M[\lfloor a?(x,y).Calculate.a!(x',y')\rfloor.P_4]^{r4}_{l4,\ public} \\
\parallel\ & {n_5}^C[\lfloor a?(x,y).Calculate.a!(x',y')\rfloor.P_5]^{r5}_{l5,\ public}
\end{aligned}$$

$$\begin{aligned}
Step\ 3:\ N_2 \xrightarrow{a!<x',y'>} N_3 =_{df}\ & {n_1}^D[P_1\{(x/x',y/e)\}]^{r1}_{l1,\ friendly} \\
\parallel\ & {n_2}^D[P_2]^{r2}_{l2,\ friendly} \\
\parallel\ & {n_3}^D[a!\ \langle k,v\rangle.P_3]^{r3}_{l3,\ public} \\
\parallel\ & {n_4}^M[\lfloor a?(x,y).Calculate.a!(x',y')\rfloor.P_4]^{r4}_{l4,\ public} \\
\parallel\ & {n_5}^C[\lfloor a?(x,y).Calculate.a!(x',y')\rfloor.P_5]^{r5}_{l5,\ public}
\end{aligned}$$

Through the application of our operational semantics, we can see that in the presence of the above network topology and the same ecological equipment, the user can directly carry out information exchange and processing between their own devices, so as to complete D2D communication and successfully describe its communication process transformation.

III. Automatic Implementation of SC4MEC

In this section, SC4MEC calculus is implemented in the rewriting engine Maude [7]–[9]. For the space limit, we only briefly give automatic simulation of Device-to-Device Communication shown in Fig. 1. In this case, nodes n_1 and n_2 belongs to the same user, thus we set the channel to $friendly$. Other channels' type are all set to $public$.

Here, we first give the equation of *simulation* from the example as below. It can be easily found that the first two devices have the same channel type *friendly*, different space positions and communication radii. Besides, the action is to send messages (k_1, v_1) by the node n_1.

```
eq simulation =
(mesfrom n1 c1 d 5 Coordinate(1, 3, 5) friendly to n3 trans
      k1 v1)
< n1 : Node | nodeType : d, pr : p, r : 5, l : Coordinate
    (1, 3, 5), ch : friendly >
< n2 : Node | nodeType : d, pr : c1(x, y) . q, r : 4, l :
    Coordinate(2, 4, 6), ch : friendly >
< n3 : Node | nodeType : d, pr : c1(x, y) . q, r : 4, l :
    Coordinate(3, 5, 7), ch : public >
< n4 : Node | nodeType : m, pr : c1(x, y) . q, r : 5, l :
    Coordinate(4, 6, 8), ch : public >
< n5 : Node | nodeType : c, pr : c1(x, y) . q, r : 8, l :
    Coordinate(5, 7, 9), ch : public >
```

The simulation result in the rewrite engine Maude is shown in Fig. 2, from which we can find that the result of $simulation$ is in line with that simulated by manual in the Subsection II.C.

IV. Conclusion

In this paper, we first proposed a secure hierarchical process calculus SC4MEC to describe the task offloading features of mobile edge computing. Meanwhile, the syntax and operational

```
Maude> (rew simulate .)
rewrite in SC4MEC :
 simulate
result [Configuration] :
  < n1 : Node | ch : friendly, l : Coordinate(1,3,5),nodeType : d,pr : p,r : 5 >
  < n2 : Node | ch : friendly,l : Coordinate(2,4,6),nodeType : d,pr :(q{k1 / x,v1 / y}),r : 4 >
  < n3 : Node | ch : public,l :Coordinate(3,5,7),nodeType : d,pr :(c1(x,y). q),r : 4 >
  < n4 : Node | ch : public,l :Coordinate(4,6,8),nodeType : m,pr :(c1(x,y). q),r : 5 >
  < n5 : Node | ch : public,l :Coordinate(5,7,9),nodeType : c,pr :(c1(x,y). q),r : 8 >
```

Fig. 2. Simulation of the D2D Communication

semantics of the calculus is provided from the process and network levels. Security feature is implemented using three different channel types, where $friendly$ channel is employed to reduce the delay and power consumption produced by the reduction of encryption, authentication and other protection measures. The nearest server can utilize the rule $Enlarge$ to expand its radius temporarily to ensure the communication stability of the mobile device, reduce the deployment of adding one more server, and finally achieve the purpose of reducing energy consumption. Additionally, we presented the automatic implementation of the SC4MEC calculus in Maude. Finally, the calculus is applied to a small example of the D2D communication and this indicates that the SC4MEC calculus is extensible to the realistic scenes.

In the future, we will further improve the process calculus by introducing more operators to describe delay-aware and energy-aware secure hierarchical calculus.

Acknowledgement. This work was partly supported by National Key Research and Development Program of China (Grant No. 2018YFB2101300) and National Natural Science Foundation of China (Grant Nos. 61872145, 62032024).

References

[1] P. Mach and Z. Becvar, "Mobile edge computing: a survey on architecture and computation offloading," *IEEE Commun. Surv. Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[2] V. D. Valerio and F. L. Presti, "Optimal virtual machines allocation in mobile femto-cloud computing: An MDP approach," in *Proc. IEEE Wireless Communications and Networking Conference Workshops, April*, 2014, pp. 7–11.

[3] J. Oueis, E. C. Strinati, S. Sardellitti, and S. Barbarossa, "Small cell clustering for efficient distributed fog computing: a multi-user case," in *Proc. the 82nd IEEE Vehicular Technology Conference, September*, 2015, pp. 1–5.

[4] W. Labidi, M. Sarkiss, and M. Kamoun, "Energy-optimal resource scheduling and computation offloading in small cell networks," in *Proc. the 22nd Int. Conference on Telecommunications, April*, 2015, pp. 313–318.

[5] A. Bozorgchenani, "Energy and delay efficient computation offloading solutions for edge computing," Ph.D. dissertation, University of Bologna, Italy, 2020.

[6] P. Zhao, H. Huang, X. Zhao, and D. Huang, "P^3: Privacy-preserving scheme against poisoning attacks in mobile-edge computing," *IEEE Trans. Comput. Soc. Syst.*, vol. 7, no. 3, pp. 818–826, 2020.

[7] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. L. Talcott, "Introduction," in *All About Maude - A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic*, 2007, pp. 1–28.

[8] A. Rodríguez, F. Durán, A. Rutle, and L. M. Kristensen, "Executing multilevel domain-specific models in maude," *J. Object Technol.*, vol. 18, no. 2, pp. 4:1–21, 2019.

[9] U. Berger, P. James, A. Lawrence, M. Roggenbach, and M. Seisenberger, "Verification of the european rail traffic management system in real-time maude," *Sci. Comput. Program.*, vol. 154, pp. 61–88, 2018.