DOI: 10.1051/itmconf/20171201023

Ros Based Remote Robot Task Monitoring and Control System

Yao-Yao CHENG, Dian-Xi SHI

National Laboratory for Parallel and Distributed Processing National University of Defense Technology,Changsha,China
905199669@qq.com, dxshi@nudt.edu.cn

Abstract—This paper defines a novel robot task based on ROS. Meanwhile, a ROS based remote robot task monitoring and control system is implemented， through which operators can remotely operate robots to achieve tasks as defined above. In addition, operators can use the system to monitor each task's running status and each task's running node process information so that the work load of remote robots can be known.

1 Introduction

George Devol of America designed the first electronic procedure programmable industrial robot in 1954. And, Unimation's first robot Unimate was put into use in General Motors in 1962, marking the birth of the first generation of robots[1]. Since then, the robot began to slowly penetrate into human life. With the strengthening of technology and communication capabilities, the infiltration speed faster and faster. In decades of years, the human has created the robot, has realized from static ones to movable ones, and make them learn to crawl and walk. Today, robots even can also use artificial intelligence to challenge human, such as the chess competition between Lee Se-dol and AlphaGo. At present, robots mostly serve the human as the service robot helping to do cleaning and repeat boring industrial work. At the same time, they can be also used to explore the deep sea and outer space, which makes an indelible contribution to mankind and in turn promotes the robot technology. It has developed into a cross discipline of many fields--robotics, including mechanics, electronics, biology, cybernetics, computer science, bionics, artificial intelligence, systems engineering and so on[2].

With the rapid development of robotics and more kinds of sensor data is acquired[3], more and more robots are being applied to our life, how to effectively control the robot has become a problem which is worth being studied. Moreover, the extensive use of robots makes the work environment of robots more extreme, more complex and more high-risky[4], such as disaster relief environment, high voltage and high radiation area. Considering mankind's both safety and capability of operating robots, researchers turn to how to remotely control a robot, which is called tele-robotics.

In terms of recent research on tele-robotics, there are a lot of related work. In the research and implementation of robots' virtual simulation and remote control system, a virtual reality environment[5] was constructed by using OpenGL, which was used for safety testing before sending remote control commands. The C/S mode remote communication method, based on WinSock mechanism, was used for robots' communication. The modular design concept was applied into the establishment of the system. Wang et al. put forward a novel control framework for internet based tele-robotics[6], which can guarantee the non-distortion-transfer of control information and can reduce the difference of action time between the local simulated virtual robot and the remote real robot, solving the inherent network time delay problem in the remote control. By the way, the remote tracking control of a mobile robot subject to a bilateral time-delay was addressed and a proposed delay compensation strategy was demonstrated by means of experiments[7]. The event-based remote control strategy was adopted to overcome the problem of network time delay and the algorithm to control robots' formation was proposed using "leader-follower" approach to realize the simulation result of robots' formation, which enables robots to have high self-organizing ability[8].Based on the robot operating system, a human-computer interaction system for space robot was designed to control the remote mechanical arm using human body posture and voice of the operator, and receive the state of the arm and its surrounding virtual reality environment using sensors[9].The work[10] focused on providing the control and software architecture that bridges the tele operator, the sensorial data from the task circumstance and the robot's locomotion system as well as the end-effectors. And two key components was added to promote the system: a video display of predicted operator intentions and a haptic-based controller for automated

© The Authors, published by EDP Sciences. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (http://creativecommons.org/licenses/by/4.0/).

grasping. The work refers that mostly remote control only controls robots to pick and place, in other words, some simple discrete action commands. What the remote control system proposed in this paper was very different from previous research. A Multi-level Fusion Architecture (MUFA) for controlling the navigation of a tele-commanded Autonomous Guided Vehicle was designed[11].By combining the client station's instructions and the ability to move autonomously according to information around, an intelligent navigation system was achieved[11].And the paper[12] introduces a formal framework of muti-agent system, so do I will design a control framework for robots' running tasks.

It's found that the tele-robotics research often combines local simulation and remote control to strengthen the control performance of the remote robot. On the other hand, there are much work on network latency when remote control information is transferred. Numerous surveys have indicated that that most tele-robotics systems select socket mechanism for communication. One reason is that socket can be used to build long-term links, not like HTTP turned off once the request has received[13]. The other is that command messages can be easily handled as short byte streams, which is suitable for socket to transfer, rather than large files, which call for HTTP.

When operating robots carrying out tasks remotely, the simulation system can only show the surface layer of the robots' working state, but the workload is not visible to operators. So a system able to monitor task execution status and how much a task occupies system resources remotely is anticipated. This paper, first of all, defines a novel task. Next, the design framework of the remote robot task monitoring and control system is presented. The forth section, three kinds important technology, such as multi-thread socket communication mechanism, the algorithm of parsing tasks' nodes and the access to information of running nodes, is illustrated one by one. The fifth section, the functions of the realized system are displayed. Conduction lies in the last section.

2 Task definition

The task, in daily life, is the work which to be taken, duty and responsibility need to pay attention to. In the field of computer, it refers to a process or a series of transaction, completed by the computer, as well as the professional term of the basic work unit. In the paper, we define the novel task by instantiation. First of all, a few professional terms are introduced as follows.

ROS node: A node is an executable file, which can run in the ROS environment. It can communicate with other nodes by ROS Master. It is also a separate process. Nodes mentioned later in this paper refers to the ROS node.

Launch file: ROS uses the tool ---"roslaunch" to start the file. A full launch file, written in the form of XML, includes basic node list. The launch file executes, which can start the tool---"roscore" and one-time start the batch multiple nodes defined in the launch file.

Package: Package, is the most basic unit organization of ROS code. Each Package includes library files, executable files, scripts, and other files. When a node is to be started, the node is searched by the package where it lies.

In the ROS environment, we used to define the executable file of a node in a package, where there can exist multiple executable files. Then, batch multiple nodes run after starting the launch file where the node list is defined through the tool---"roslaunch". Here, the task remotely operated is defined as the transaction the multiple nodes completed. So, a launch file defines a task for the robot to complete. Starting a launch file means starting a task. Each package, of course, there can be multiple launch files so that different tasks, made up of different nodes, can execute. As a result of the batch starting of nodes, operations of the robot is simplified. So it is reasonable that writing the node list into launch files and defining a launch file as a task.

3 System design framework

3.1 Function structure

In general, the function structure of the remote robot task monitoring and control system includes the server-side console, intermediate socket communication mechanism and the remote robot client, as shown in Figure 1. The sever-side console is consist of the task control and monitoring module, the command encapsulation and sending module and the task status update module. On the other hand, the remote robot client's functions are split into the robot registration module, the command reception, parse and execution module, the task node parsing module, and running information acquisition module. The detailed descriptions of each module function and the connections between them will be discussed in subsequent sections.

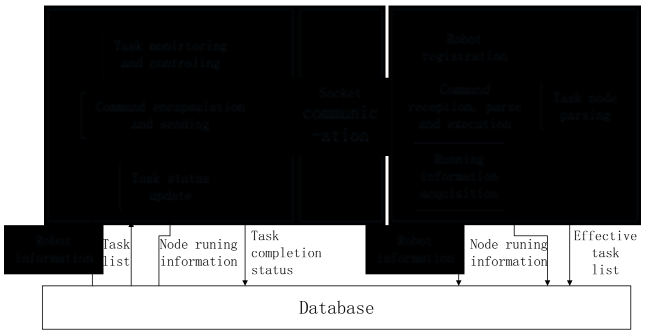


Figure 1. Function structure diagram

3.1.1 The robot client

Robot registration module: A robot will store information, including robot IP and name, into the database by registering when the client on the robot starts, so that the server-side console can directionally manage and control the robot identified by the unique IP, in other word, directionally send commands and obtain the robot's task running information. Moreover, effective task list of the robot will be also registered to the database, in order that the console visually starts effective tasks on the robot.

Command reception, parse and execution module: The robot client receives commands to operate tasks from the console through the socket communication mechanism. Because commands is transferred in the form of byte streams, they need to be parsed into the executable command after reception, then execute in the ROS environment to operate the robot.

Task node parsing module: Each launch file defines a task indirectly, and each task contains at least one node, so extracting the node list from the launch file is achievable. Using DOM parse method the task list is gained easily.

Running information acquisition module: In the ROS environment, each node is a process, so the access to the node information is the process information acquisition. The name of a node is known and unique, and we have access to the node's progress ID from the ROS Master by submitting the node name to the ROS Master, which is realized by a tool rewritten from the ROS tool---"rosnode". Finally, the process ID is used to acquire the node process running information.

3.1.2 The Server-client console

Task monitoring and control module:It includes the control of tasks and the monitoring of task statuses. Control of the task refers to a task start, pause and end. Task control, must get a robot list and effective task list from the database to send commands directionally. Task status monitoring shows us each robot's running task information and task completion progress and real-timely updates.

Command encapsulation and sending module: A command includes command type and related parameters, such as the name of the task which the command operates. Multiple fields are encapsulated into a byte stream and then are sent to the robot client through the socket communication mechanism.

Task status update module: Task statuses --- non-started, running, finished, among which the task status changes. These changes can be stored into status tables in the database rea-timely.

3.2 Running flow

The remote control and monitoring system flow chart introduces how the whole system work and is operated, as shown in Figure 2.

First, the robot client registers IP and local effective task list to the database, followed by the console directionally sending task operation commands to the robot. The command to start a task or finish a task sent to the robot client will be parsed to commands working on the nodes of the task. Running task's node list and the running information of all nodes of the task obtained in the local will register into the database real-timely. Sequentially, the console will query all the task's nodes information from the database and display on the console. The console changes the task status by operating the task, and the change will update to the database, so that the console can real-timely query the task completion progress. What aforementioned is the control process of the remote control and monitoring system, including real-time remote operations on the task , the console's remote monitoring the task completion progress and running information. The running information reflects how much the system resource the task has occupied. The system function complete and effective, will not cause the heavy load to the local robot.

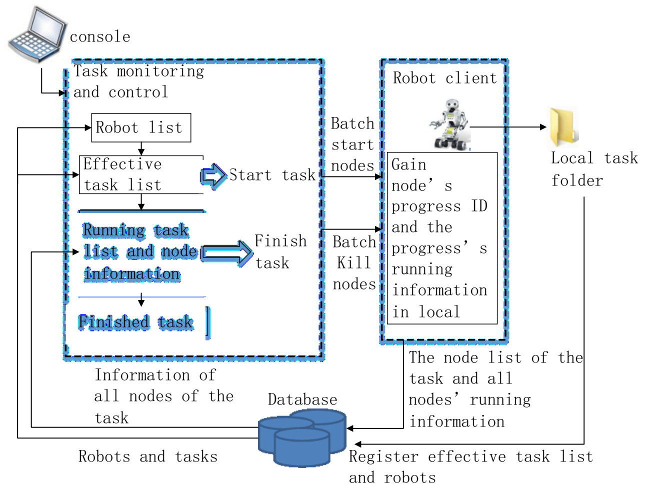

console
Task monitoring and control
Robot list
Effective task list
Start task
Running task list and node information
Finish task
Finished task
Batch start nodes
Batch Kill nodes
Robot client
Gain node' s progress ID and the progress' s running information in local
Local task folder
The node list of the task and all nodes' running information
Information of all nodes of the task
Database
Robots and tasks
Register effective task list and robots

Figure 2. Function structure diagram

4 Core technology

4.1 Socket muti-thread communication mechanism

The robot client as the socket communication server, uses multiple threads to monitor and handle the command sent from the console. The console as the socket communication client, can continuously send commands to the robot client. Two kinds of commands work. One is to operate the task, the other is the request to refresh the running task information.

There is the socket server running a permanently monitoring procedure so as to receive requests from the console continuously and provide appropriate services. In order to realize the function that the server can respond to the continuous request, the multi-threaded parallel processing mechanism[14] needs to be adopted. That is, the server listeners on the specified port to monitor whether the client request comes, and once the request heard, the server will start a special service thread to respond to the client request but keep listening in aim to wait for the arrival of the next request.

What the socket communication has to deal with is command encapsulation and command sending.

First of all, command encapsulation is to pack information to be transferred into a byte stream. The content of the command message contains the machine number, the command type, the name of the task to execute. The three fields of the command are spliced into a byte stream in a fixed format.

Next, command messages transfer from the client to the server through the socket. Each time a command request is sent, the socket connection is reopened, so each message transfer is independent. The procedure stated above, is called command sending.

4.2 Task node parsing

XML(Extensible markup language) is a standard language W3C (World Wide Web Con2 association) defined for data transfer and exchange[15]. It is independent of any language and architecture, providing a loose tree structure to represent a semi-structured data suitably. With the advantages of strict definition, clear structure and flexibility, it can be used to describe all kinds of complex information, becoming one of the most widely used formats for data exchange and storage in computer system[16].

Launch files are written in XML described above, and the node list is extracted by retrieval of elements whose tag is <node>. Because of the fact that each element has the name attribute, through which we can obtain the names of all the nodes. The list of all nodes of each task will register into the database and under the task as node list. For understanding more easily, the following example is given. TABLE I is an example of a launch file.

TABLE I. LAUNCH FILE EXAMPLE

```
<launch>
  <node name="listener-1" pkg="rospy_tutorials"
              type="listener" launch-prefix="xterm -e"/>
  <node name="listener-2" pkg="rospy_tutorials"
          type="listener" launch-prefix="xterm -e"/>
  <node name="listener-3" pkg="rospy_tutorials"
          type="listener" launch-prefix="xterm -e"/>
  <node name="talker" pkg="rospy_tutorials
         type="talker"  respawn="true" />
  <node pkg="gmapping " type="slam_gmapping"
name=" slam_gmapping ">
       <remap from="scan" to="base_scan"/>
  </node>
   <node pkg="rviz" type="rviz" name="rviz"
respawn="true" />
</launch>
```

After parsing the launch file through the method above, then the parsed node list is: listener-1, listener-2, listener-3, talker, slam_gmapping and rviz and will be stored into the database. When viewing the task's detail running information, the nodes of the node list will be visited one by one and be displayed to operators.

4.3 Node information acquisition

When the task starts, the task information registers into the running status table, including the node list. And then retrieve the node list gaining the name of each node. There exists no same node name in the same ROS network, that is, the name of a node in the same robot is unique. In view of the fact that each node is a process, the node's progress ID is gotten from the ROS Master by submitting the node name to the ROS Master, which is realized by a tool rewritten from the ROS tool---"rosnode". Performed by the shell process, the running information is obtained through the process ID and is stored into the node running information table. The console visit the database to get the running information of all nodes of the running task. In addition, the console makes the robot client update the latest node information into the database through the refresh request, so the console will refresh the running information of the nodes from the database. The node running information acquisition flow chart is as shown in Figure 3.

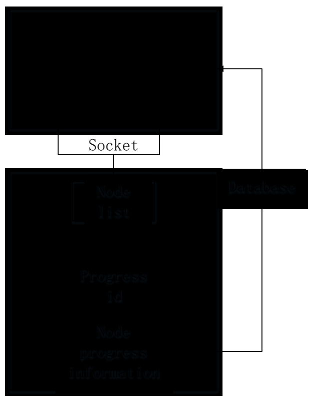


Figure 3. Node information acquisition chart

5 Experimental result

5.1 Experimental Condition Description

Two computers is needed to test the system performance. One acts as the host machine of the console, the other provides ROS to simulate executing robot task. In this section, RVIZ is used to detect the system's performance by the robot's dynamically mapping through path exploration, which is considered as a task. The gmaping package provided by ROS is used to generate the map, and it is a implementation of the famous open source OpenSlam package in the ROS framework. This package provides the Slam for the laser device, and according to the input and attitude data of the laser device builds a grid based 2D map. Due to hardware limitation, we will replay the data collected already to simulate the robot input, and the data will be exported to the SLAM for mapping, finally the map will be displayed in the RVIZ. The following TABLE II is the remote launch file launched.

DOI: 10.1051/itmconf/20171201023

TABLE II. LAUNCH FILE OF THE TASK FOR TEST

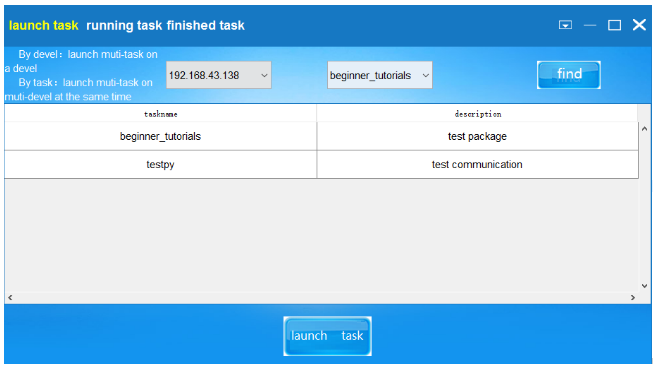
```
<launch>
<node name="listener" pkg="rospy_tutorials"
type="listener" launch-prefix="xterm -e"/>
<node name="talker" pkg="rospy_tutorials"
type="talker" launch-prefix="xterm -e"/>
<node pkg="gmapping " type="slam_gmapping"
name=" slam_gmapping ">
<remap from="scan" to="base_scan"/>
</node>
   <node pkg="rviz" type="rviz" name="rviz"
```

5.2 Experimental Demonstration

5.2.1 Console

Install the console program on the master control computer, and run the program. The main function window is shown as Figure 4. The functions includes starting tasks, stopping tasks, viewing task running information and task completion status.

Viewing the task running information is as shown in Figure 5. The running information of all the nodes of a special task is viewed, namely monitoring the occupation of system resources. It can be seen from Figure 5 that the list of the nodes running under the task called "testpy" we launched remotely contains talker, listener, slam_gmapping and rviz. Each node's process ID, program name, execution file directory, process status, running duration, memory occupation rate, etc. are presented.



By devel：launch muti-task on a devel
By task：launch muti-task on muti-devel at the same time

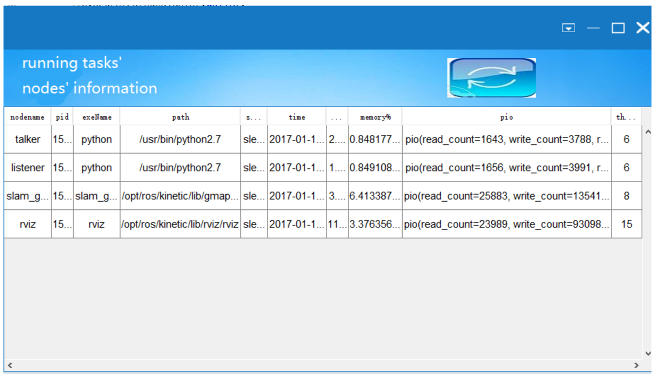
192.168.43.138
beginner_tutorials

taskname	description
beginner_tutorials	test package
testpy	test communication

Figure 4. Main function window figure caption

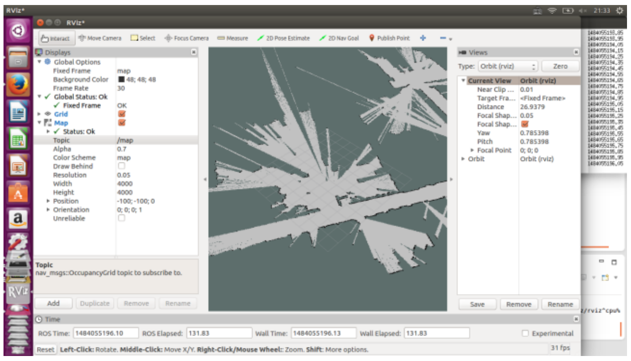
5.2.2 Robot client

After the task is remotely started, the robot client will start listener, talker, slam_gmapping and rviz. These nodes are defined in the same task, so they will start at the same time. Figure 5.3 is the process of building a 2D map.



nodename	pid	exeName	path	s...	time	...	memory%	pio	th...
talker	15...	python	/usr/bin/python2.7	sle...	2017-01-1...	2....	0.848177...	pio(read_count=1643, write_count=3788, r...	6
listener	15...	python	/usr/bin/python2.7	sle...	2017-01-1...	1....	0.849108...	pio(read_count=1656, write_count=3991, r...	6
slam_g...	15...	slam_g...	/opt/ros/kinetic/lib/gmap...	sle...	2017-01-1...	3....	6.413387...	pio(read_count=25883, write_count=13541...	8
rviz	15...	rviz	/opt/ros/kinetic/lib/rviz/rviz	sle...	2017-01-1...	11...	3.376356...	pio(read_count=23989, write_count=93098...	15

Figure 5. Running node information



Global Options	
Fixed Frame	map
Background Color	48; 48; 48
Frame Rate	30
Global Status: Ok	
Fixed Frame	OK
Grid	✓
Map	✓
Status: Ok	
Topic	/map
Alpha	0.7
Color Scheme	map
Draw Behind	☐
Resolution	0.05
Width	4000
Height	4000
Position	-100; -100; 0
Orientation	0; 0; 0; 1
Unreliable	☐

Topic
nav_msgs::OccupancyGrid topic to subscribe to.

ROS Time: 1484055196.10 ROS Elapsed: 131.83 Wall Time: 1484055196.13 Wall Elapsed: 131.83

Figure 6. the process of building a 2D map

5.3 System Performance

As is stated above, the experiment has launched two tasks, which are proved to work very well.

The system performance analysis is carried out in terms of the timeliness of the communication and the work load the system has brought to the host computer.

It has been tested and concluded that the socket communication has guaranteed adequate reliability and timeliness. Meanwhile, the map exploring task, which typically occupies much system resources, tells us that the system causes relative work load, but it is within the range of the computer.

Discussion

In this paper, we propose a new definition of the task, which is the combination of the transactions done by multiple nodes in a launch file. We also implement an artificial interaction system for managing tasks and monitoring tasks. The previous study focuses on simple commands controlling remotely the surface behaviors of mobile robots. When controlled and monitored, tasks are materialized to specific nodes, which is of basic need to remotely operating robots to perform complex tasks. These two contributions are of great significance for tele-robotics and robots' better serving for human.

The running information of each task, such as occupied memory, CPU and other system resources, has not been strictly counted, so the next step work can go around monitoring the running information of a task but not a task's nodes. As a result, something about the remote robot's resource load is clearer.

References

[1] Cai Zixing. Robotics[M]. Beijing: Tsinghua University Press, 2000.
[2] Jing, Zhu. Robotics and control technology [M]. Zhejiang University press, 1991 (in Chinese)
[3] Peng S L, Li S S, Chen L, et al. Scalable base-station model-based multicast in wireless sensor networks[J]. Journal of computer science and technology, 2008, 23(5): 780-791.
[4] Tang Yusong, Liu Jingtai, Lu Guizhang. Analysis of [J]. robot based on robot remote operating system remote network technology, 2000, 22(1): 67-72, 80.
[5] LI Lianzhong, ZHAI Jingmei, HE Haiyang. Research and implementation of robots' virtual simulation and remote control system 2015 [J].
[6] Yong-ming W, Nan-feng X, Hong-li Y I N, et al. A novel control framework for internet based tele-robotics[J]. Journal of Harbin Institute of Technology: English edition, 2008, 15 (5): 602-607
[7] Alvarezaguirre A, Nijmeijer H, Oguchi T, et al. REMOTE CONTROL OF A MOBILE ROBOT SUBJECT TO A COMMUNICATION DELAY[C]. international conference on informatics in control, automation and robotics, 2016: 55-62.
[8] LONG Xiao-Lin, JIANG Jing-Ping. Web- based remote control of multi-robot [J]. Engineering Journal of Wuhan University, 2005, 38(3): 122-125.
[9] Zuo XuanChen, Han Liangliang, Zhuang Jie, et al. Design of space robot human machine interaction system based on ROS.Computer engineering and design, 2015, 36 (12): 3370-3374.
[10] Belzunce A, Li M, Handroos H. Control system design of a teleoperated omnidirectional mobile robot using ROS[C]//Industrial Electronics and Applications (ICIEA), 2016 IEEE 11th Conference on. IEEE, 2016: 1283-1287.
[11] Aude E P L, Carneiro G, Serdeira H, et al. CONTROLAB MUFA: A multi-level fusion architecture for intelligent navigation of a telerobot[C]//Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on. IEEE, 1999, 1: 465-472.
[12] Wang H, Wu Q. A formal framework of multi-agent systems with requirement/service cooperative style[J]. Journal of Computer Science and Technology, 2000, 15(2): 106-115.
[13] Chen Hao, Zhang Wei. Based on java socket network TCP/IP programming [J]. CD-ROM software and applications, 2013 (2): 35-37.
[14] Sun Xiaomeng, Wang Zhibin. Multi thread Socket communication based on [J]. TCP Journal of Eastern Liaoning University: Natural Science Edition, 2013, 20 (): 178-182.
[15] Extensible Markup Language(XML)1. 1(Second Edition)[EB/ OL] . [2007207226] .http://www. w3. org/TR/ 2006/REC2xml11220060816/.
[16] Beeri C , Milo T. Schemas for Integration and Translation ofSt ructured and Semi2St ructured Data [M] // Lecture Notes in Computer Science,1999.