RT-THREAD Network Toolset **(NETUTILS)**

Application Notes

RT-THREAD Document Center

WWW.RT-THREAD.ORG

Saturday 13th October, 2018

Table of contents

This application note introduces how to use RT-Thread NetUtils, helping developers to better use RT-Thread NetUtils components to solve problems encountered during network development.

1 Purpose and background of this paper

When developing and debugging network-related products, some useful tools can often achieve twice the result with half the effort.

Based on this application scenario, the NetUtils component develops and encapsulates a series of concise and easy-to-use network tool sets to provide convenience for developers.

In order to facilitate users to develop network applications, RT-Thread makes the commonly used network tools into NetUtils component package, which is dynamically configured through env.

It can be used immediately after installation, effectively reducing resource usage.

2 Structure of this paper

• Introduction to NetUtils components

• Configuration and use of the Ping tool •

Configuration and use of the NTP time synchronization tool • Use

of the TFTP file transfer tool • Configuration and use of

the Iperf network bandwidth test tool • Configuration and use of other

network debugging tools

3. Problem Statement

This application note will introduce the RT-Thread NetUtils component around the following issues.

• What are the main features of RT-Thread NetUtils? What are their functions? • How do I use the Ping tool

to diagnose network stability? • How do I test network stability and

bandwidth? • How do I transfer files over the network?

4. Problem Solving

4.1 Introduction to NetUtils Components

RT-Thread NetUtils is a collection of network tools, including the Ping command for testing and debugging, the NTP tool for time synchronization, Iperf and NetIO for performance and bandwidth testing, and TFTP, a lightweight file transfer tool widely used in embedded systems, for conveniently transferring files between two devices over the network. RT-Thread also provides advanced auxiliary tools to address practical development challenges, such as Telnet for remote login to the RT-Thread Finsh/MSH Shell, and tcpdump, a network packet capture tool based on lwIP.

The following is the classification and introduction of RT-Thread NetUtils:

name	Classification	Function Introduction
Ping	Debugging and testing	Use the "ping" command to check the network Whether the network is connected can help Help us analyze and determine network failures
NTP	Time synchronization	Network Time Protocol
TFTP	File Transfer	TFTP is a simple way to transfer files Single protocol, lighter than FTP
Iperf	Performance Testing	Test the maximum TCP and UDP bandwidth Broadband performance, can report bandwidth, delay Jitter and packet loss
NetIO	Performance Testing	Tools for testing network throughput
Telnet	Remote Access	Can remotely log in to RT-Thread Finsh/MSH Shell
tcpdump	Network debugging	tcpdump is RT-Thread based A network packet capture tool based on lwIP

Each gadget can be enabled/disabled independently using menuconfig, and Finsh/MSH usage commands are provided. First open env tool, enter the BSP directory, enter menuconfig in the env command line to enter the configuration interface to configure the project, and select the appropriate NetUtils functions, as shown in the figure (Note: Ping and TFTP depend on lwIP, so you need to enable lwIP dependency before they can be displayed)

RT-Thread online packages
-> IoT - internet of things
-> netutils: Networking utilities **for** RT-Thread

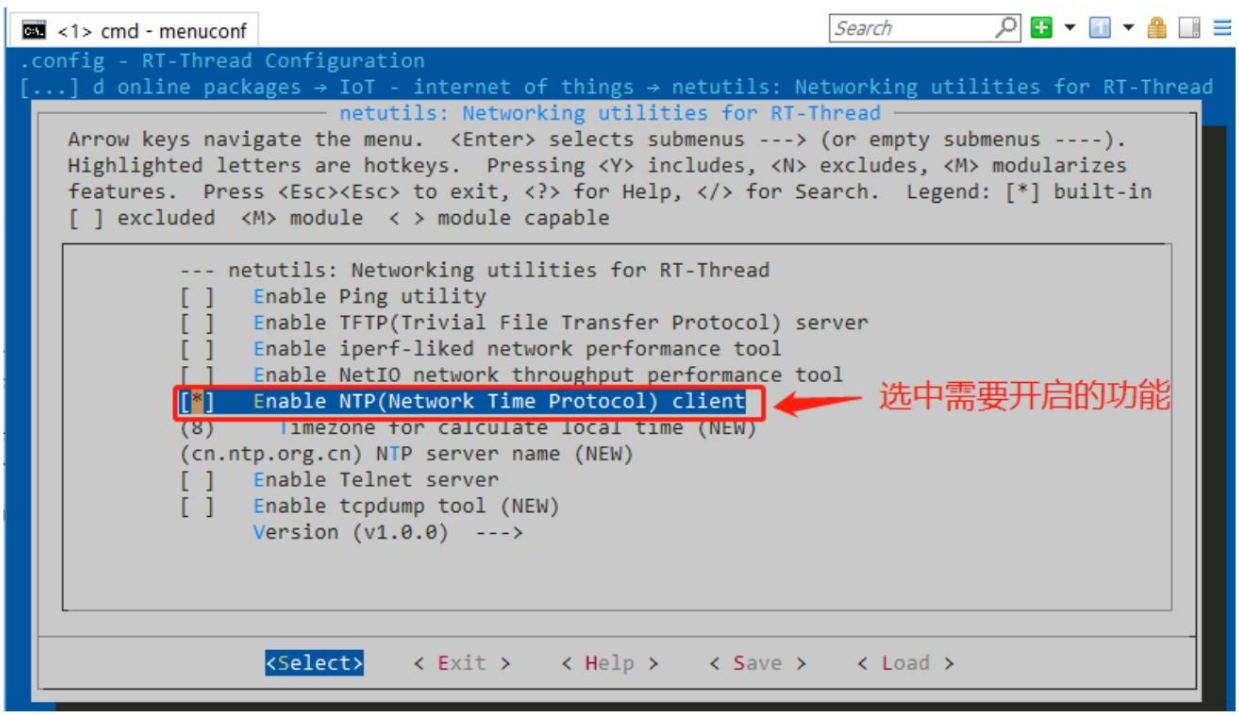

Figure **1:** *env* Configuration

4.2 Configuration and Use of Ping Tool

4.2.1. Introduction

Ping It is a network diagnostic tool used to test whether data packets can reach a specific host through the IP protocol. It estimates the packet loss rate (packet loss rate) and the round-trip delay time (network delay) between the host and the packet.

4.2.2. Usage

The Ping tool depends on lwIP. You need to enable the lwIP dependency in the env tool before it becomes visible. The steps are as follows:

```
-> RT-Thread Components
    -> Network stack
        -> light weight TCP/IP stack
            -> Enable lwIP stack
```

Enable the Ping option in the NetUtils menu bar:

```
RT-Thread online packages
    -> IoT - internet of things -> netutils:
        Networking utilities for RT-Thread
        [*] Enable Ping utility
```

Ping supports accessing IP addresses or domain names . Use Finsh/MSH commands for testing. The results are as follows:

- Ping domain name

```
msh />ping rt-thread.org 60 bytes from
116.62.244.242 icmp_seq=0 ttl=49 time=11 ticks 60 bytes from 116.62.244.242 icmp_seq=1
ttl=49 time=10 ticks 60 bytes from 116.62.244.242 icmp_seq=2 ttl=49 time=12 ticks 60 bytes
from 116.62.244.242 icmp_seq=3 ttl=49 time=10 ticks

msh />
```

- Ping IP

```
msh />ping 192.168.10.12 60 bytes
from 192.168.10.12 icmp_seq=0 ttl=64 time=5 ticks 60 bytes from 192.168.10.12 icmp_seq=1
ttl=64 time=1 ticks 60 bytes from 192.168.10.12 icmp_seq=2 ttl=64 time=2 ticks 60 bytes
from 192.168.10.12 icmp_seq=3 ttl=64 time=3 ticks msh />
```

4.3 Configuration and Use of NTP Tools

4.3.1. Introduction

NTP The Network Time Protocol (NTP) is a protocol used to synchronize the time of various computers on a network. An NTP client is implemented on RT-Thread. Once connected to the network, it can obtain the current UTC time and update it to the RTC. #### Usage #### Enable the NTP option in the NetUtils menu bar:

```
RT-Thread online packages
    -> IoT - internet of things -> netutils:
        Networking utilities for RT-Thread
        [*] Enable NTP(Network Time Protocol) client
```

- Get UTC timeUTC time Also known as World Standard Time, World Standard Time, and International Coordinated Time. Beijing time is UTC+8

  Time is 8 hours more than UTC time, or it can be understood as 8 hours earlier.

API: time_t time_t ntp_get_time(void)

parameter	describe
none	none
return	describe
>0	Current UTC time
=0	Failed to obtain time

Sample code:

```
#include <ntp.h>

void main(void) {

    time_t cur_time;

    cur_time = ntp_get_time();

    if (cur_time) {

        rt_kprintf("NTP Server Time: %s", ctime((const time_t*) &cur_time));
    }
}
```

- Get local time

Local time has the concept of time zone compared to UTC time. For example, Beijing time is in the Eastern Time Zone, which is 8 hours longer than UTC time.

The current time zone can be set in menuconfig , the default is 8

API: time_t ntp_get_local_time(void)

parameter	describe
none	none
return	describe
>0	Current local time
=0	Failed to obtain time

The usage of this API is similar to ntp_get_time()

• Synchronize local time to RTC

If the RTC device is enabled, you can also use the following commands and APIs to synchronize the local time of NTP to the RTC device.

The effects of the Finsh/MSH command are as follows:

```
msh />ntp_sync
Get local time from NTP server: Sat Feb 10 15:22:33 2018
The system time is updated. Timezone is 8.
msh />
```

API: time_t ntp_sync_to_rtc(void)

parameter	describe
none	none
return	describe
>0	Current local time
=0	Time synchronization failed

• Notes 1. The NTP API method will occupy a large amount of thread stack when executed. Ensure that there is sufficient stack space (fi1.5K) when using it; 2.

NTP API methods do not support reentrancy. Please ensure locking when using them concurrently.

4.4 Configuration and Use of TFTP Tools

4.4.1. Introduction

TFTP (Trivial File Transfer Protocol) is a protocol in the TCP/IP family used to transfer files between clients.

A protocol for simple file transfer between a computer and a server, providing a simple and low-cost file transfer service. The port number is **69 ,** which is much better than traditional

The FTP protocol is much lighter and suitable for small embedded products.

RT-Thread currently supports TFTP server.

4.4.2. Usage

The TFTP tool depends on lwIP. You need to enable the lwIP dependency in the env tool before it becomes visible. The steps are as follows:

```
-> RT-Thread Components
   -> Network stack
      -> light weight TCP/IP stack
         -> Enable lwIP stack
```

Enable the TFTP option in the NetUtils menu bar:

```
RT-Thread online packages
   -> IoT - internet of things -> netutils:
      Networking utilities for RT-Thread
      [*] Enable TFTP (Trivial File Transfer Protocol) server
```

- Install a TFTP client

The installation file is located in netutils/tools/Tftpd64-4.60-setup.exe . Please install the software before using TFTP.

- Start the TFTP server

Before transferring files, you need to use the Finsh/MSH command on RT-Thread to start the TFTP server. The effect is as follows:

```
msh />tftp_server TFTP
server start successfully.
msh />
```

- Transfer files

Open the newly installed Tftpd64 software and configure it as follows:

1. Select Tftp Client ; 2. In the Server interfaces drop-down box, be sure to select the network card in the same network segment as RT-Thread; 3. Enter the IP address of the TFTP server. You can use the ifconfig command under RT-Thread's MSH to view it; 4. Enter the TFTP server port number, default: 69

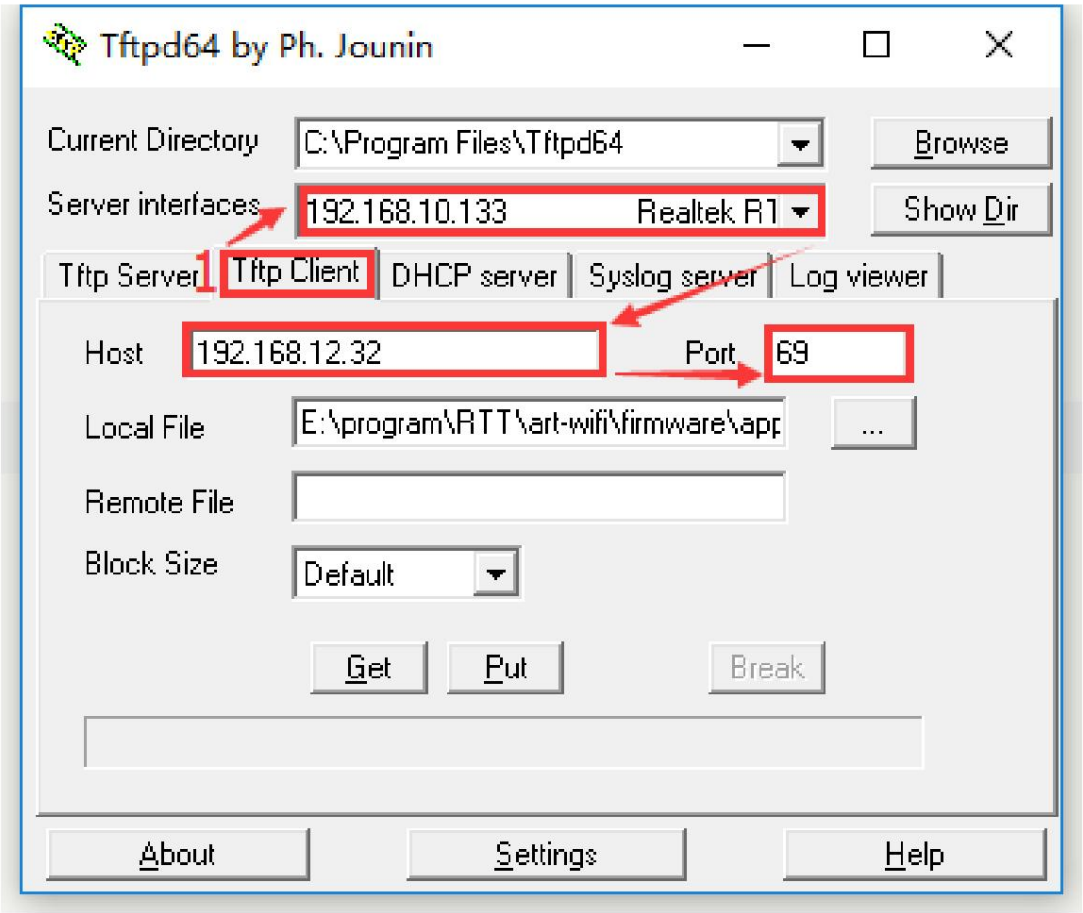

Figure **2:** *tftpd config*

• Send files to RT-Thread

1. In Tftpd64 , select the file you want to send. 2. "Remote File" specifies the path (including the file name) where the file will be saved on the server. This option supports both relative and absolute paths. Since RT-Thread uses the DFS_USING_WORKDIR option by default, relative paths are based on the directory currently accessed by Finsh/MSH. Therefore, when using relative paths, be sure to change the directory in advance. 3. Click the "Put" button.

As shown in the figure below, the file is sent to the directory currently entered by Finsh/MSH. The relative path is used here:

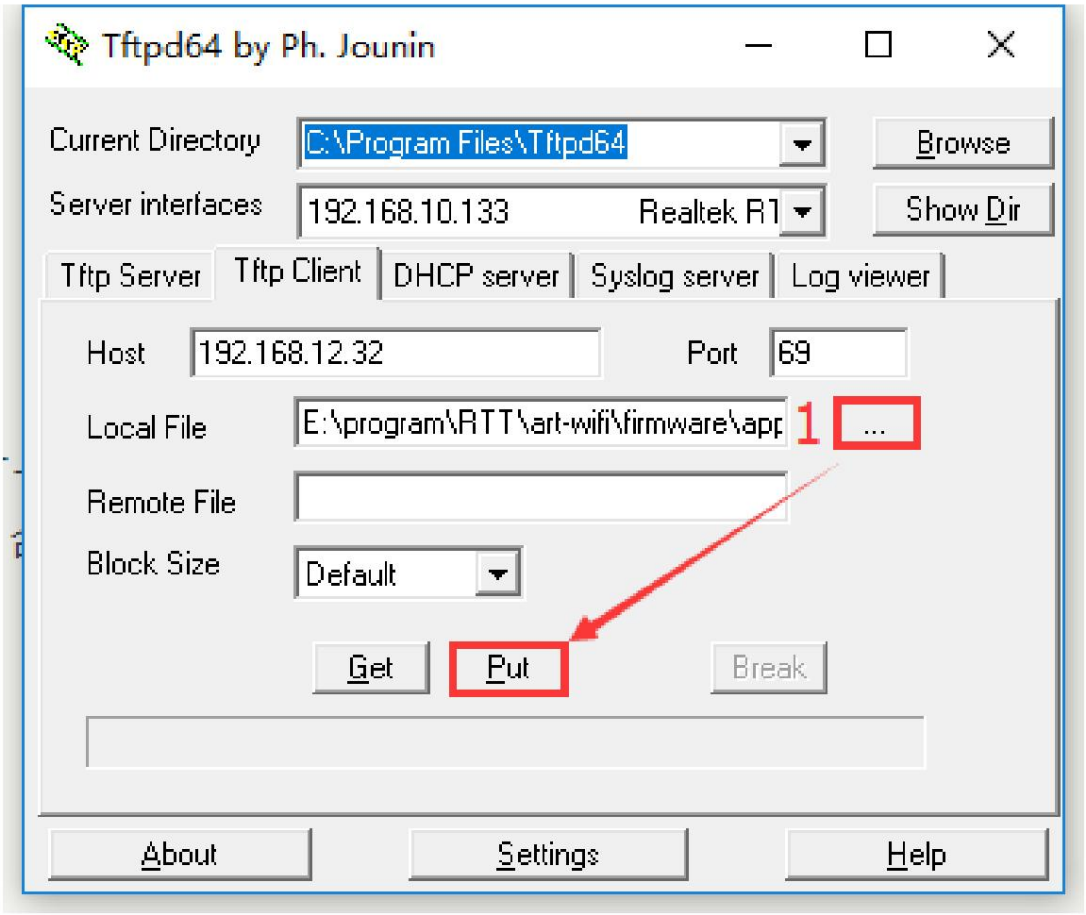

Figure **3:** *tftpd get*

Note: If DFS_USING_WORKDIR is not enabled and Remote File is empty, the file will be saved in the root directory.

• Receive files from RT-Thread

1. In Tftpd64 software, fill in the file path (including file name) to be received and saved; 2. Remote File is the server

The file path (including file name) to be received by the client. The options support relative and absolute paths.

DFS_USING_WORKDIR option, the relative path is based on the directory currently entered by Finsh/MSH. Therefore, when using relative paths, be sure to Be sure to switch the directory in advance; 3. Click the Get button.

As shown below, save /web_root/image.jpg to the local computer. The absolute path is used here:

```
msh /web_root>ls Directory /                ## Check if the file exists
web_root:
image.jpg msh /                 10559
web_root>
```

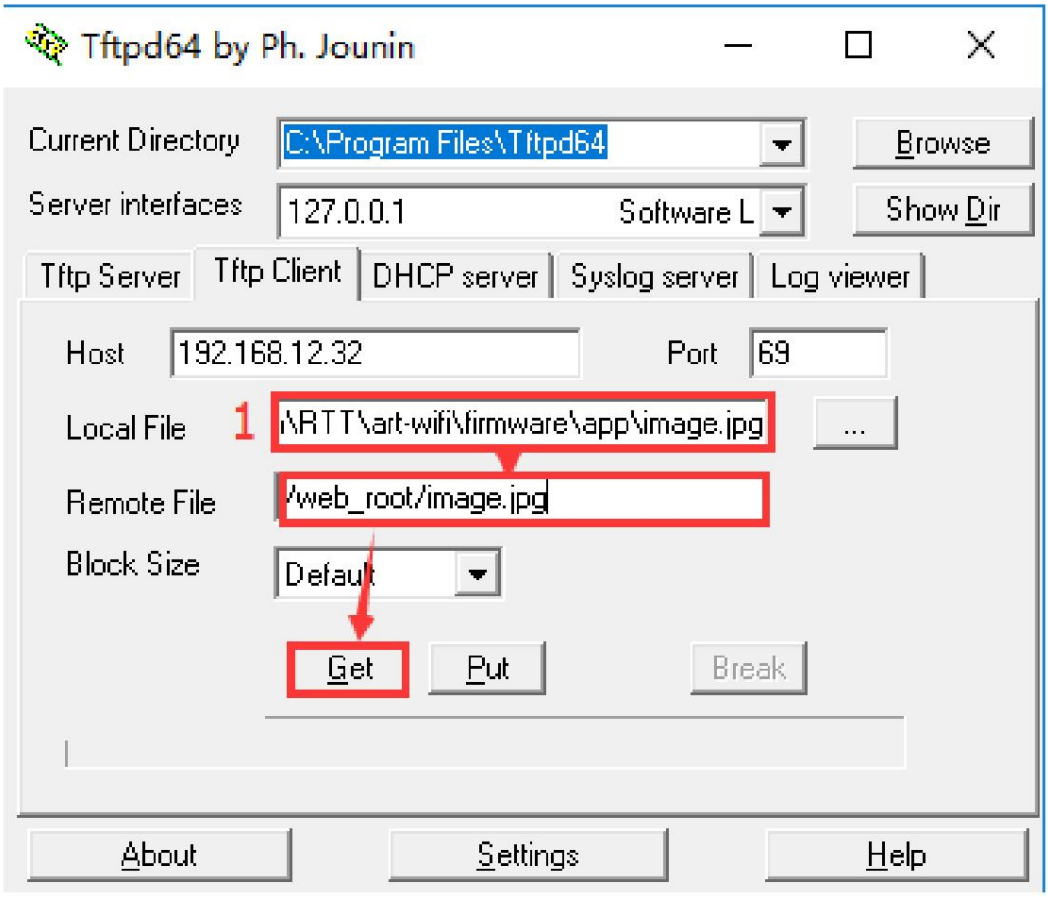

Figure **4:** *tftpd put*

4.5 Configuration and Use of Iperf Tool

4.5.1. Introduction

Iperf Is a network performance testing tool. Iperf can test the maximum TCP and UDP bandwidth performance, with a variety of parameters and UDP Features, which can be adjusted as needed, can report bandwidth, latency jitter, and packet loss.

4.5.2. Use

Enable the Iperf option in the NetUtils menu bar:

```
RT-Thread online packages
    -> IoT - internet of things -> netutils:
        Networking utilities for RT-Thread
        [*] Enable iperf-liked network performance tool
```

Iperf uses a master-slave architecture, that is, one end is the server and the other end is the client. The Iperf software package we provide implements TCP Server mode and client mode do not support UDP testing at this time. The following will explain how to use the two modes in detail.

Iperf server mode

- Obtaining an IP address

You need to use the Finsh/MSH command on RT-Thread to obtain the IP address. The effect is as follows:

```
msh />ifconfig
network interface: e0 (Default)
MTU: 1500
MAC: 00 04 9f 05 44 e5
FLAGS: UP LINK_UP ETHARP
ip address: 192.168.12.71 gw
address: 192.168.10.1
net mask : 255.255.0.0
dns server #0: 192.168.10.1
dns server #1: 223.5.5.5
```

Write down the obtained IP address 192.168.12.71 (record according to the actual situation)

- Start the Iperf server

You need to use the Finsh/MSH command on RT-Thread to start the Iperf server. The effect is as follows:

```
msh />iperf -s -p 5001
```

-s means start as a server -p means listen on port 5001

- Install JPerf testing software

The installation file is located in netutils/tools/jperf.rar . This is green software. The installation is actually a decompression process. Just decompress it to a new folder.

- Run jperf tests

Open the jperf.bat software and configure it as follows:

1. Select Client mode; 2. Enter the IP address you just obtained, 192.168.12.71 (use the actual address); 3. Change the port number to 5001; 4. Click run Lperf! to begin the test; 5. Wait for the test to complete. During the test, the test data will be displayed in the shell interface and JPerf software.

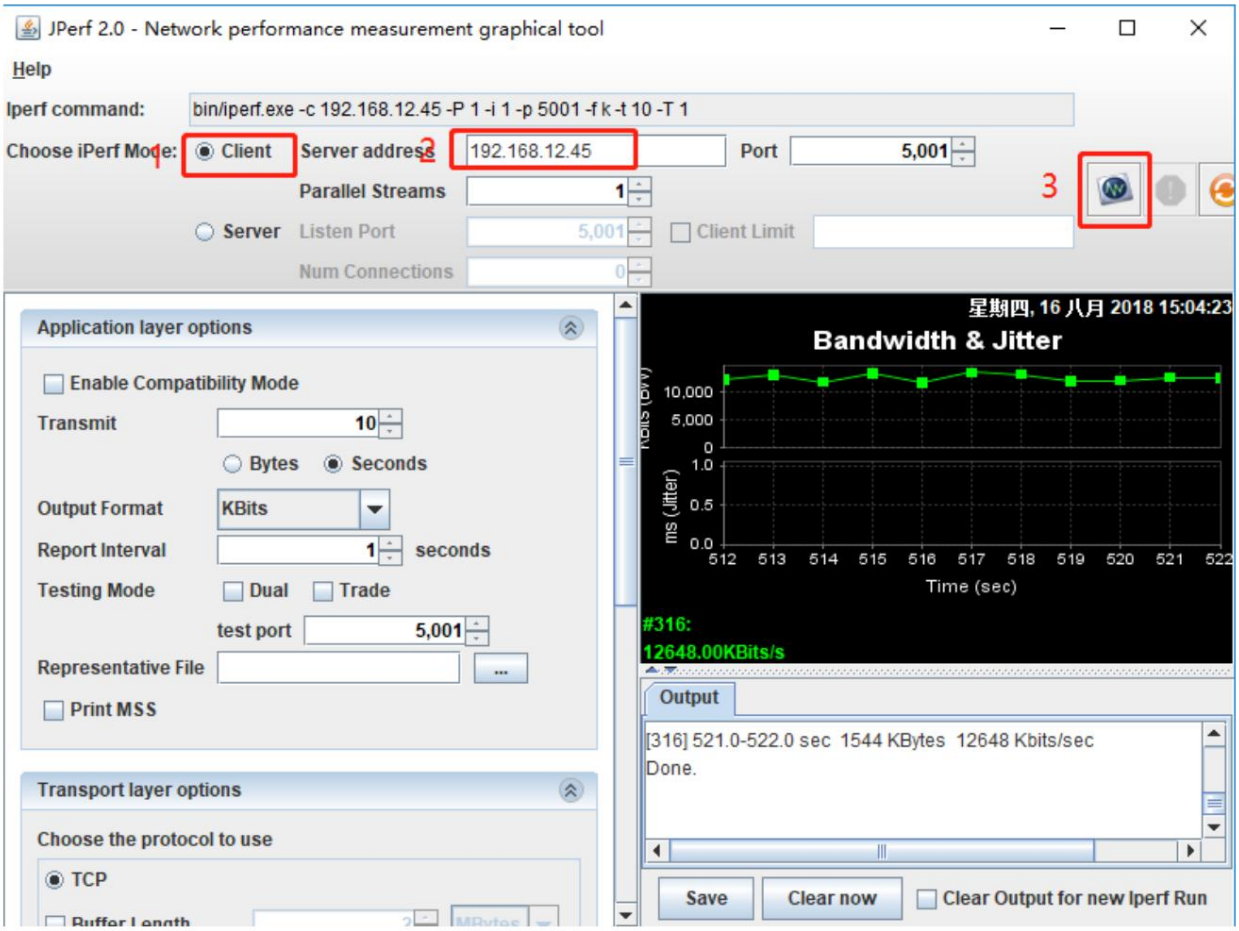

Figure **5:** *iperf server*

Iperf client mode

• Get the IP address of your PC

Use the ipconfig command in the command prompt window of the PC to obtain the IP address of the PC, and write down the obtained PC IP address as 192.168.12.45 (record according to actual situation).

• Install JPerf testing software

The installation file is located in netutils/tools/jperf.rar . This is green software. The installation is actually a decompression process. Just decompress it to a new folder.

• Start the jperf server

Open the jperf.bat software and configure it as follows:

1. Select Server mode 2. Change the port number to 5001 3. Click run Lperf! to start the server

• Start the Iperf client

You need to use the Finsh/MSH command on RT-Thread to start the Iperf client. The effect is as follows:

```
msh />iperf -c 192.168.12.45 -p 5001
```

-c means starting the test as a client, followed by the IP address of the PC running the server. -p means connecting to port 5001 and waiting for the test to end. During the test, the test data will be displayed on the shell interface and JPerf software.

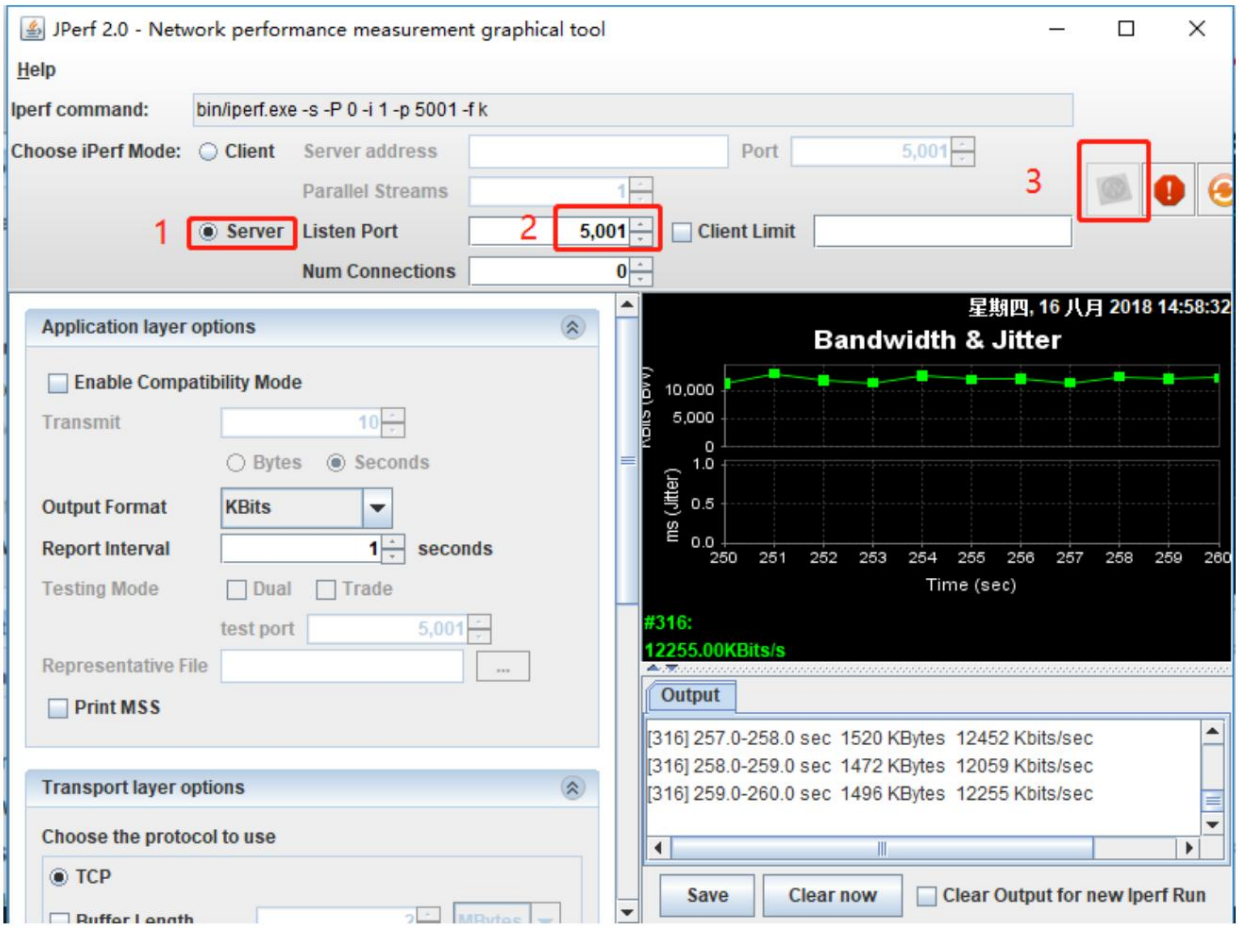

Figure **6:** *iperf client*

4.6 Introduction and use of other network debugging tools

In addition to the commonly used network tools mentioned above, RT-Thread also provides some more practical network tools for development and debugging, such as NetIO tools, Telnet tool and tcpdump tool.

NetIO Tools

NetIO A tool for testing network performance on OS/2 2.x, Windows, Linux, and Unix. It tests the network throughput using TCP/UDP packets of varying sizes.

RT-Thread currently supports the NetIO TCP server.

For the usage of NetIO, please refer to the README in the component directory, which will not be described here.

Telnet Tool

Telnet The protocol is an application layer protocol used in the Internet and local area networks, using the form of a virtual terminal to provide two-way, text-based

A key interactive function based on character strings. It is a member of the TCP/IP protocol suite and is the standard protocol and primary method for Internet remote login services. It is commonly used for remote control of web servers, allowing users to run tasks on remote hosts from their local hosts.

RT-Thread currently supports Telnet server. After the Telnet client is successfully connected, it will remotely connect to the device's Finsh/MSH to achieve remote control of the device.

For the usage of Telnet, please refer to the README in the component directory, which will not be described here.

4.6.3. tcpdump tool

tcpdump is a small tool based on RT-Thread to capture IP packets. The captured data can be saved through the file system or Import the data into PC through rdb tool and analyze it with wireshark software.

For the usage of tcpdump, please refer to the README in the component directory. I will not go into details here.