Troubleshooting User Systems in IP Services and Case Studies

Abstract

This article describes a method for troubleshooting user systems in Internet protocol (IP) services and presents case studies demonstrating its application. It is the third in a bimonthly series on the theme of practical field information about telecommunication technologies. This month's contribution is from the Network Interface Engineering Group, Technical Assistance and Support Center, Maintenance and Service Operations Department, Network Business Headquarters.

1. Introduction

Internet protocol (IP) services such as IP-phone and IP-TV have been growing rapidly. Although problems that occur because of faults in home devices or local area network (LAN) cables can be resolved by replacing them, problems caused by other factors have been increasing. This situation has generated a need to investigate the flow of packets exchanged in actual IP communications and examine the data in those packets. This article introduces a method for analyzing faults by capturing and analyzing packets and presents case studies demonstrating its application.

2. Packet capture method

A typical packet capture method is to insert a switching hub having a mirror port between network devices from which the target packets enter and leave and to connect the mirror port to a personal computer (PC) running packet capture software such as Wireshark (formerly known as Ethereal) in order to copy and grab the packets (**Fig. 1**). If a repeater hub is used for packet capture, one must keep in mind that packet loss can occur as a result of collisions since communication here is performed in half-duplex mode (**Fig. 2**).

Here, we describe the basic functions of packet

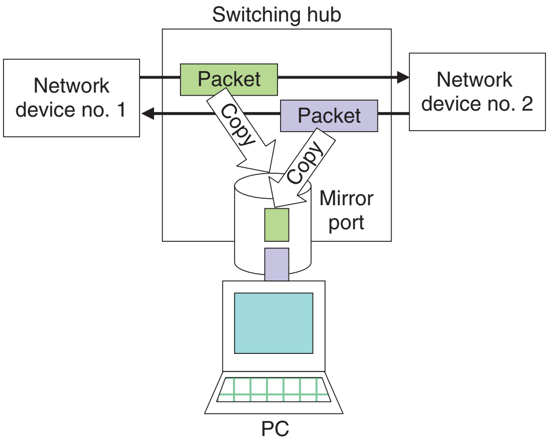


Fig. 1. Packet capture method.

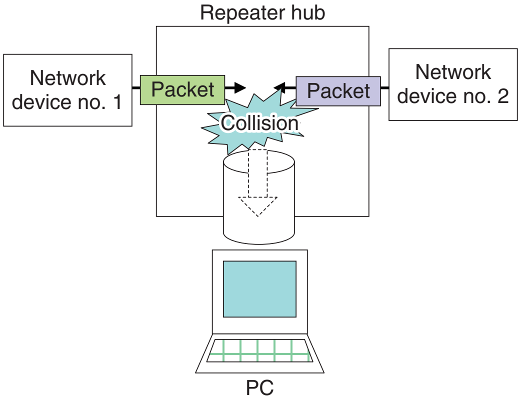


Fig. 2. Packet capture when using a repeater hub.

† NTT EAST
Ota-ku, 144-0053 Japan

capture software using Wireshark (ver. 1.2.3), which has found widespread use as free software that can run on a variety of operating systems including Windows and UNIX (**Fig. 3**).

Items marked (1)–(6) on the operation screen of the packet capture software shown in Fig. 3 correspond to items (1)–(6) described below.

(1) Packet capture start/stop

The receive mode in performing packet capture is set to promiscuous mode in order to capture not only packets addressed to oneself but also those addressed to others.

(2) Packet List pane

This pane displays the receive time, transfer direction (source/destination addresses), type of protocol used, additional information, etc. about captured packets in the order in which they are collected.

(3) Packet Details pane

This pane expands parameters and displays details of the packet selected in (2) for each protocol layer.

(4) Packet Bytes pane

This pane displays the content of the packet selected in (2) as a hexadecimal dump and ASCII (text) display. For the parameter selected in (3), it displays the corresponding data in highlighted form. It also shows the address of this data within the packet (its byte from the front).

(5) Packet Filter

Since the packet capture process grabs all packets, the pane in (2) displays all of those packets including unneeded packets, which can be inconvenient in actual problem analysis. To alleviate this problem, the software enables only those packets of interest in problem analysis to be displayed by setting filtering conditions such as IP address or port number (display filter). For example, a filtering formula like “IP.addr= =192.168.1.1” can be defined and applied by using the Display Filter function from the pull-down menu or by entering it in the Filter toolbar on the operation screen ((5)’ in Fig. 3).

(6) Packet statistics

Packet statistics can be displayed to aid in understanding the traffic conditions in the communications environment being monitored. These include the number of sent/received packets, number of bytes, and bit rate, and so on for each protocol layer, each endpoint, and each set of endpoints. These statistics can also be displayed versus time in graph form.

3. Data analysis method

In this section, we describe a method for analyzing captured data by referring to a case study. Packet capture can be performed between a router and client PCs to analyze the cause of delays that suddenly occur in network communications (**Fig. 4**).

Specifically, to determine whether traffic is concentrating in a specific server or router, traffic statistics can be investigated from captured data in units of IP addresses. This is done by selecting Endpoints from the Statistics menu on the menu bar of the Wireshark operation screen to launch the Endpoints viewing screen. Then, by selecting the IPv4 tab on that screen ((1) in **Fig. 5**) and clicking on the Tx/Rx Packets (or Bytes) column ((2) in **Fig. 5**), one can obtain traffic

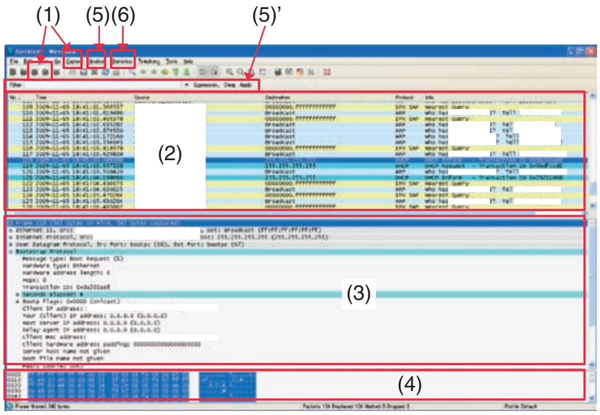


Fig. 3. Operation screen of packet capture software (using Wireshark as an example).

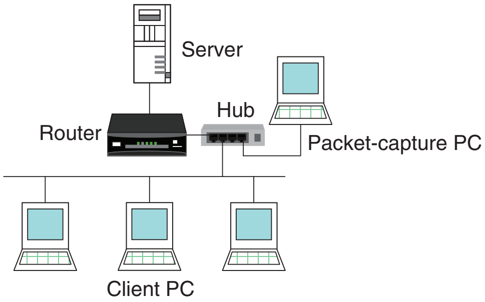
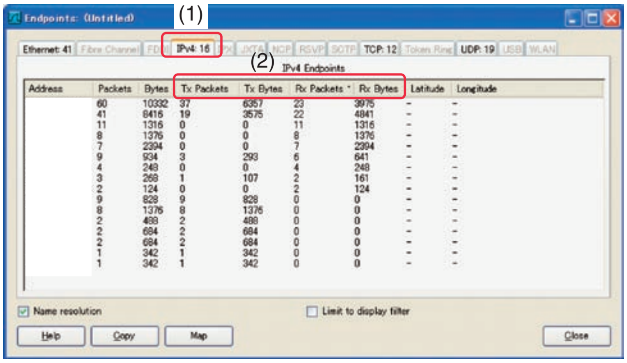


Fig. 4. Device configuration and capture location.



Endpoints: (Untitled)

Ethernet: 41 | Fibre Channel | FDDI | IPv4: 16 | IPX | JXTA | NCP | RSVP | SCTP | TCP: 12 | Token Ring | UDP: 19 | USB | WLAN

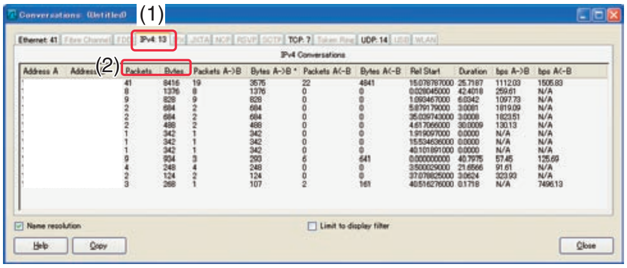
IPv4 Endpoints

Address	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes	Latitude	Longitude
	60	10332	37	6357	23	3975	-	-
	41	8416	19	3575	22	4841	-	-
	11	1316	0	0	11	1316	-	-
	8	1376	0	0	8	1376	-	-
	7	2394	0	0	7	2394	-	-
	9	934	3	293	6	641	-	-
	4	248	0	0	4	248	-	-
	3	268	1	107	2	161	-	-
	2	124	0	0	2	124	-	-
	9	828	9	828	0	0	-	-
	8	1376	8	1376	0	0	-	-
	2	488	2	488	0	0	-	-
	2	684	2	684	0	0	-	-
	2	684	2	684	0	0	-	-
	1	342	1	342	0	0	-	-
	1	342	1	342	0	0	-	-

☑ Name resolution　　☐ Limit to display filter

Help　Copy　Map　Close

Fig. 5. Traffic statistics by network device.



Conversations: (Untitled)

Ethernet: 41 | Fibre Channel | FDDI | IPv4: 13 | IPX | JXTA | NCP | RSVP | SCTP | TCP: 7 | Token Ring | UDP: 14 | USB | WLAN

IPv4 Conversations

Address A	Address B	Packets	Bytes	Packets A->B	Bytes A->B	Packets A<-B	Bytes A<-B	Rel Start	Duration	bps A->B	bps A<-B
		41	8416	19	3575	22	4841	15.078787000	25.7187	1112.03	1505.83
		8	1376	8	1376	0	0	0.028045000	42.4018	259.61	N/A
		9	828	9	828	0	0	1.093467000	6.0342	1097.73	N/A
		2	684	2	684	0	0	5.879179000	3.0081	1819.09	N/A
		2	684	2	684	0	0	35.039743000	3.0008	1823.51	N/A
		2	488	2	488	0	0	4.617066000	30.0009	130.13	N/A
		1	342	1	342	0	0	1.919097000	0.0000	N/A	N/A
		1	342	1	342	0	0	15.534636000	0.0000	N/A	N/A
		1	342	1	342	0	0	40.101891000	0.0000	N/A	N/A
		9	934	3	293	6	641	0.000000000	40.7975	57.45	125.69
		4	248	4	248	0	0	3.500029000	21.6566	91.61	N/A
		2	124	2	124	0	0	37.078825000	3.0624	323.93	N/A
		3	268	1	107	2	161	40.516276000	0.1718	N/A	7496.13

☑ Name resolution　　☐ Limit to display filter

Help　Copy　Close

Fig. 6. Traffic statistics for communications between endpoints.

statistics for each IP address ranked in order of highest traffic volume (Tx: transmitted, Rx: received).

The ranking described above enables the user to determine whether traffic is concentrating in a specific server or router.

Next, to determine whether any of the endpoints from among those communicating with one of the above servers or routers has a particularly large traffic volume, one can investigate traffic statistics from captured data in units of communications between end points. With Wireshark, for example, this is done by selecting Conversations from the Statistics menu on the operation screen to launch the Conversations viewing screen. Then, by selecting the IPv4 tab ((1) in **Fig. 6**) and clicking on the Packets (or Bytes) column ((2) in **Fig. 6**), one can get traffic statistics for pairs of endpoints in order of highest traffic volume.

In this way, one can assess which pairs of network devices have a large traffic volume and whether they are monopolizing network bandwidth. At this time, the TCP (transmission control protocol) or UDP (user

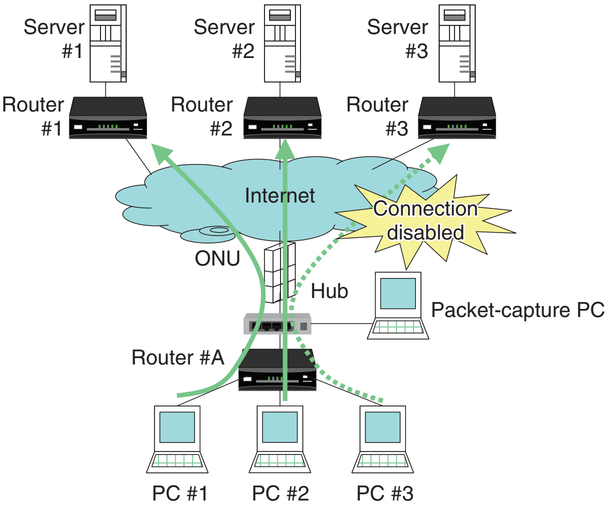
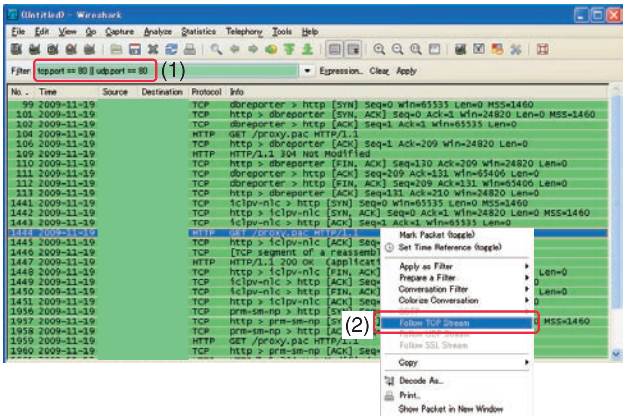


Fig. 7. Case of disabled Internet connection.

datagram protocol) tab can be selected and Packets (or Bytes) clicked to obtain traffic statistics for each port. This lets one investigate which ports are experiencing high traffic, and if the port number in question has a value of “0–49,151” (where 0–1023 are Well Known Ports and 1024–49,151 are Registered Ports) to identify the application being used by referring, for example, to the Internet Assigned Numbers Authority (IANA) site.

So far, we have described cases in which the cause of problems is traffic concentration between specific parties communicating between specific network devices. We now describe an analysis method that probes as far as packet content to determine whether an anomaly has occurred in packets or communication sequences in IP-packet communications between a server and client PCs.

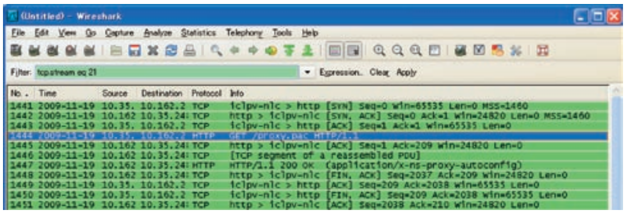
We take as an example the inability to make an Internet connection (assuming that DNS (domain name system) name resolution is operating normally). In **Fig. 7**, a connection cannot be made from PC #3. Therefore, to check whether HTTP (hypertext transfer protocol) communication (port number = 80) is being performed normally from this and the other PCs as well, a hub is inserted between the optical network unit (ONU) and router #A in the figure and a packet-capture PC is connected. Then, after the packet data collected by Wireshark has been expanded, only that corresponding to HTTP communications is displayed by selecting Filter = “TCP or UDP port is 80 (http)” prepared as a default in Display Filters of the Analyze menu or by directly entering



Filter: tcp.port == 80 || udp.port == 80

```
No. .  Time        Source  Destination  Protocol  Info
  99 2009-11-19                         TCP       dbreporter > http [SYN] Seq=0 Win=65535 Len=0 MSS=1460
 101 2009-11-19                         TCP       http > dbreporter [SYN, ACK] Seq=0 Ack=1 Win=24820 Len=0 MSS=1460
 102 2009-11-19                         TCP       dbreporter > http [ACK] Seq=1 Ack=1 Win=65535 Len=0
 104 2009-11-19                         HTTP      GET /proxy.pac HTTP/1.1
 106 2009-11-19                         TCP       http > dbreporter [ACK] Seq=1 Ack=209 Win=24820 Len=0
 109 2009-11-19                         HTTP      HTTP/1.1 304 Not Modified
 110 2009-11-19                         TCP       http > dbreporter [FIN, ACK] Seq=130 Ack=209 Win=24820 Len=0
 111 2009-11-19                         TCP       dbreporter > http [ACK] Seq=209 Ack=131 Win=65406 Len=0
 112 2009-11-19                         TCP       dbreporter > http [FIN, ACK] Seq=209 Ack=131 Win=65406 Len=0
 113 2009-11-19                         TCP       http > dbreporter [ACK] Seq=131 Ack=210 Win=24820 Len=0
1441 2009-11-19                         TCP       iclpv-nlc > http [SYN] Seq=0 Win=65535 Len=0 MSS=1460
1442 2009-11-19                         TCP       http > iclpv-nlc [SYN, ACK] Seq=0 Ack=1 Win=24820 Len=0 MSS=1460
1443 2009-11-19                         TCP       iclpv-nlc > http [ACK] Seq=1 Ack=1 Win=65535 Len=0
1444 2009-11-19                         HTTP      GET /proxy.pac HTTP/1.1
1445 2009-11-19                         TCP       http > iclpv-nlc [ACK] Seq
1446 2009-11-19                         TCP       [TCP segment of a reassembl
1447 2009-11-19                         HTTP      HTTP/1.1 200 OK  (applicat
1448 2009-11-19                         TCP       http > iclpv-nlc [FIN, ACK]                        Len=0
1449 2009-11-19                         TCP       iclpv-nlc > http [ACK] Seq                         0
1450 2009-11-19                         TCP       iclpv-nlc > http [FIN, ACK]                        Len=0
1451 2009-11-19                         TCP       http > iclpv-nlc [ACK] Seq                         0
1956 2009-11-19                         TCP       prm-sm-np > http [SY
1957 2009-11-19                         TCP       http > prm-sm-np [SY                               MSS=1460
1958 2009-11-19                         TCP       prm-sm-np > http [AC
1959 2009-11-19                         HTTP      GET /proxy.pac HTTP/1.1
1960 2009-11-19                         TCP       http > prm-sm-np [ACK] Seq
```

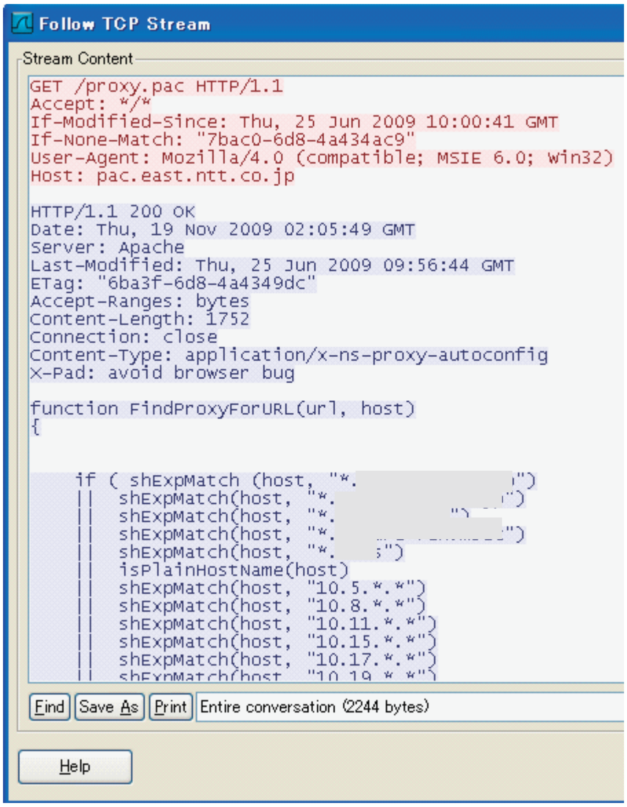
Fig. 8. Extraction of all HTTP communications by display filtering.


(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Expression... Clear Apply

Filter: tcp.stream eq 21

```
No. .  Time        Source  Destination  Protocol  Info
1441 2009-11-19 10.35. 10.162.2 TCP   iclpv-nlc > http [SYN] Seq=0 Win=65535 Len=0 MSS=1460
1442 2009-11-19 10.162 10.35.24 TCP   http > iclpv-nlc [SYN, ACK] Seq=0 Ack=1 Win=24820 Len=0 MSS=1460
1443 2009-11-19 10.35. 10.162.2 TCP   iclpv-nlc > http [ACK] Seq=1 Ack=1 Win=65535 Len=0
1444 2009-11-19 10.35. 10.162.2 HTTP  GET /proxy.pac HTTP/1.1
1445 2009-11-19 10.162 10.35.24 TCP   http > iclpv-nlc [ACK] Seq=1 Ack=209 Win=24820 Len=0
1446 2009-11-19 10.162 10.35.24 TCP   [TCP segment of a reassembled PDU]
1447 2009-11-19 10.162 10.35.24 HTTP  HTTP/1.1 200 OK  (application/x-ns-proxy-autoconfig)
1448 2009-11-19 10.162 10.35.24 TCP   http > iclpv-nlc [FIN, ACK] Seq=2037 Ack=209 Win=24820 Len=0
1449 2009-11-19 10.35. 10.162.2 TCP   iclpv-nlc > http [ACK] Seq=209 Ack=2038 Win=65535 Len=0
1450 2009-11-19 10.35. 10.162.2 TCP   iclpv-nlc > http [FIN, ACK] Seq=209 Ack=2038 Win=65535 Len=0
1451 2009-11-19 10.162 10.35.24 TCP   http > iclpv-nlc [ACK] Seq=2038 Ack=210 Win=24820 Len=0
```

Fig. 9. Extraction of particular HTTP communications.



```
GET /proxy.pac HTTP/1.1
Accept: */*
If-Modified-Since: Thu, 25 Jun 2009 10:00:41 GMT
If-None-Match: "7bac0-6d8-4a434ac9"
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)
Host: pac.east.ntt.co.jp

HTTP/1.1 200 OK
Date: Thu, 19 Nov 2009 02:05:49 GMT
Server: Apache
Last-Modified: Thu, 25 Jun 2009 09:56:44 GMT
ETag: "6ba3f-6d8-4a434 9dc"
Accept-Ranges: bytes
Content-Length: 1752
Connection: close
Content-Type: application/x-ns-proxy-autoconfig
X-Pad: avoid browser bug

function FindProxyForURL(url, host)
{


    if ( shExpMatch (host, "*.[redacted]")
     ||  shExpMatch(host, "*.[redacted]")
     ||  shExpMatch(host, "*.[redacted]")
     ||  shExpMatch(host, "*.[redacted]")
     ||  shExpMatch(host, "*.[redacted]s")
     ||  isPlainHostName(host)
     ||  shExpMatch(host, "10.5.*.*")
     ||  shExpMatch(host, "10.8.*.*")
     ||  shExpMatch(host, "10.11.*.*")
     ||  shExpMatch(host, "10.15.*.*")
     ||  shExpMatch(host, "10.17.*.*")
     ||  shExpMatch(host, "10.19.*.*")
```

Entire conversation (2244 bytes)

Fig. 10. Content of HTTP communications (command requests and replies).

“tcp.port= = 80|| udp.port==80” into the Filter toolbar on the operation screen ((1) in **Fig. 8**). This procedure lets one assess whether all HTTP communications have failed or only those of a particular site.

If it is now assessed in the above way that anomalies have occurred in only HTTP communications between PC #3 and server #3 in Fig. 7, then displaying only that TCP session will facilitate the analysis process. This is done by making any packet within that TCP session into an active display and then right-clicking and selecting Follow TCP Stream ((2) in **Fig. 8**), which causes all packets in that session to be automatically extracted and displayed (**Fig. 9**) simultaneously with the content of HTTP communications from GET requests to success replies (200 OK) (**Fig. 10**). Analysis can therefore proceed in an efficient manner while the communication sequence and packet contents of a particular communications session are being checked. This procedure also makes it possible to check the TCP session for segment loss on the server side or problems in HTTP communications on upper-level web servers.

4. Case study reflecting a typical problem

In recent years, there has been a dramatic increase in the popularity of peer-to-peer (P2P) file sharing software and online games. In the former case, a large number of users download and upload large quantities of files from and to each other over the Internet, and in the latter case, many users compete with each other over the Internet. This flourishing of applications that exploit the inherent convenience of the Internet to the limit is being accompanied by performance demands on routers and other devices on the customer’s premises that could not be envisioned in normal use. As a result, problems caused by processing capacity limits and resource depletion have begun to appear. Here, we present a case study obtained from consultations made with the Technical Assistance and Support Center.

A customer who uses online games occasionally experienced the phenomenon in which the console’s screen suddenly froze and the Internet connection was lost. The same customer also uses P2P file

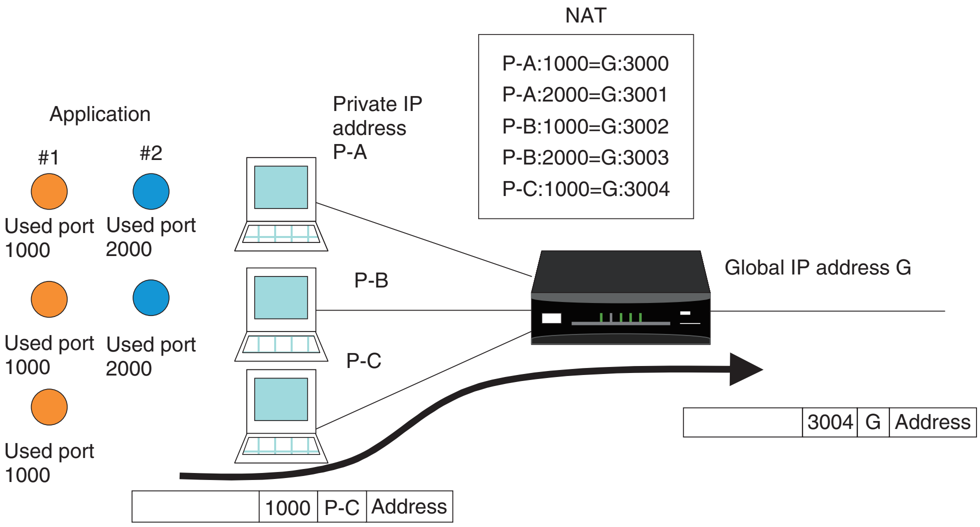
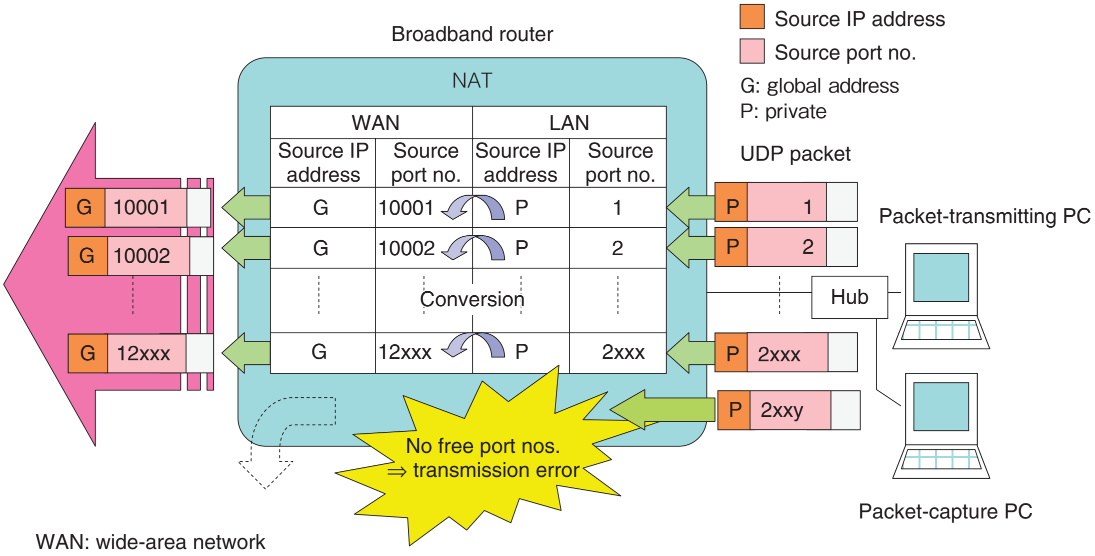


Fig. 11. NAT mechanism.



WAN		LAN	
Source IP address	Source port no.	Source IP address	Source port no.
G	10001	P	1
G	10002	P	2
G	12xxx	P	2xxx

Fig. 12. Problem reproduction test.

sharing software. When the communications log of the broadband router was checked, it was found that a transmission error occurred in conjunction with the network address table (NAT) in the time period during which the problem occurred. The NAT dynamically converts IP addresses and port numbers so that multiple applications can run simultaneously on multiple PC terminals using a single global address (**Fig. 11**). The details of this error indicated that the number of entries registered in the NAT exceeded the table's capacity, which suggested that the address/port conversion pattern could not be registered in the NAT. To test this hypothesis, we connected a PC to that broadband router, and using a packet-transmission program that we created, we transmitted IP packets continuously while incrementing the destination port number by one for each packet. In this way, we could check whether the same kind of transmission error would occur in the broadband router (**Fig. 12**). At this time, we also set up one more PC between the packet-transmitting PC and broadband router to perform packet capture.

We found from the communications log that a transmission error occurred at the NAT after the destination port number on the LAN side entered the 2000 series. Thus, to determine the maximum number of entries that could be registered in the NAT, we divided captured data into port number ranges in the manner of port no. = 1–2750, 1–2500, 1–2250, etc. and using a PC packet-generator tool, we transmitted that data in sequence to the broadband router and checked for a transmission error. We found that a transmission error did indeed occur upon exceeding port no. = 2XXX and that the maximum number of entries for the NAT was therefore 2XXX.

On the basis of this case study, we could infer that performing large-volume TCP/UDP communications simultaneously by online games and P2P-type file sharing software caused the number of NAT registration entries to reach 2XXX so that subsequent communications failed owing to insufficient NAT capacity. We could also infer that this problem could be solved by replacing the broadband router with another model having a larger NAT capacity or by changing the settings of the customer's P2P file sharing software.

5. Conclusion

In this article, we introduced a troubleshooting method targeting user systems in IP services and presented case studies of its use. We hope that readers found this article useful and interesting.