COPING WITH OVERLOAD ON THE NETWORK TIME PROTOCOL PUBLIC SERVERS[1]

DAVID MILLS, UNIVERSITY OF DELEWARE, JUDAH LEVINE, NIST, RICHARD SCHMIDT, USNO, AND DAVID PLONKA, UNIVERSITY OF WISCONSIN

Abstract

The public time servers operated by USNO and NIST provide time synchronization, directly or indirectly, to millions of Internet computers today. The load in the form of processor cycles and network traffic has doubled in the last two years and could eventually overwhelm the servers and the network infrastructure unless something is done about it. While both USNO and NIST operate multiple servers across the US, the aggregate load is highly unbalanced and the flagship servers at headquarters are nearing capacity. This paper discusses the current conditions at USNO and NIST and suggests technical defenses designed to protect their resources.

Surprisingly, a significant fraction of the total load is due to the occasional defective client design that spews an alarming number of packets without good reason. In one incident at the University of Wisconsin a defective NTP implementation in a router product resulted in a large scale denial of service attack on the university's network. At NIST and USNO most of the population are well behaved mice, but a significant proportion of the total traffic is due to a relatively few number of abusive elephants. The paper proposes that the best advice may be to find the elephants and shoot them.

Keywords: network time protocol, flooding attack, denial of service attack.

INTRODUCTION

The Network Time Protocol (NTP) has become the de facto standard means for synchronizing Internet computers to UTC as disseminated by national laboratories. There are currently well over 100 Internet public time servers synchronized by modem or GPS to the national timescale and probably thousands more private servers and millions of clients on corporate, academic and government networks. The NTP Version 4 (NTPv4) software distribution for Unix, Windows and VMS[2] contains a full featured NTP server and client implementation with features designed for the busy time servers operated by national laboratories in the US and several other countries. NTP client software with diminished features suitable for personal computers can be obtained from several sources.

The NIST and USNO national laboratories operate about three dozen Internet public time servers at various locations in the US and some overseas. By nature, these are very busy machines dedicated to time service and generally producing accurate and reliable time synchronization. As the NTPv4 support is not particularly invasive and requires minimal system resources, a relatively large population of clients can be supported. However, the client request loads have increased ten-

1. Sponsored by: Naval Surface Weapons Center (Dahlgren) Contract N00178-04-1-9001.

2. Available at www.ntp.org.

fold in the past two years, reaching thousands of packets per second (PPS[3]) at the busiest servers. The service has also become much more widely used, reaching populations in the tens of millions.

It has been the recent experience at NIST and USNO, as well as nongovernment institutions, that defective NTP client implementations can cause great havoc in the NTP server population by sending packets at unreasonably high rates. The most popular servers operated by both USNO and NIST have experienced traffic surges up to 7,000 PPS. This has also happened at other NTP servers operated by public institutions, including the University of Wisconsin, which has recorded aggregate NTP request rates exceeding 285,000 PPS. Sustainable packet rates on some IP provider networks are reaching 1 Gbps, so it is in principle possible to hammer a time server on a gigabit Ethernet at rates in the millions of PPS.

The origins, experience and lessons learned from these attacks are discussed in this paper. It describes defensive measures designed to deflect abusive traffic and to provide feedback advising the client to modulate its rate. It continues with an evaluation of these measures and concludes with a set of recommendations for a future defensive infrastructure.

HOW NTP OPERATES

The NTP network includes a number of primary servers which synchronize directly to an external reference, such as a GPS radio or telephone modem. Secondary servers synchronize to the primary servers, directly or via other intervening secondary servers. In this paper we are concerned only about the primary servers and those secondary servers that synchronize directly to them, collectively described as stratum-2 clients. There can be a large number of them, possibly in the tens of millions, ganging up on public primary servers operated by NIST, USNO and other institutions.

There are two kinds of clients using the public servers, NTP and SNTP (simple NTP). Almost all NTP servers operating today use software from the public archives at www.ntp.org. This software has been carefully groomed to be a good citizen; it begins by sending one packet per minute and then retreats to one packet every fifteen minutes. A typical NTP client is expected to maintain the time within 10 ms. SNTP clients are available from several sources and are integrated in several operating systems, including Windows NT and XP and Linux. They can be run manually or from a script at intervals of a day to a week. A typical SNTP client is expected to maintain the time within 1 s.

If all client implementations carefully followed these designs, this paper would not be necessary. However, It has been conventional experience that misbehaving NTP implementations can result in clogging attacks on the public time servers operated by NIST, USNO and others. These attacks can result in client requests in the range of 1 PPS to over 200 PPS. Such behavior is not only rude and counterproductive for accurate synchronization, but might be an opportunity for a cyberterrorist to mount a denial of service attack.

As do other packet services, an NTP request consumes resources on the network and at the server. A NTP service unit involves a packet request and a packet response, each of which is 90 bytes on the network wire. With processors of the Alpha class, this exchange takes about 100 μs of proces-

3. As it occurs so often in this paper, the abbreviation PPS stands for packets per second and not the usual pulse per second.

sor time. If we assume about fifty percent of the processor cycles can be devoted to NTP service, the processor can handle up to about 5,000 PPS. This in turn requires 3.6 Mbps on the network.

The following sections describe a number of incidents in which NTP time servers at NIST, USNO and Wisconsin were engaged in a clogging or denial of service attack, either due to sheer numbers or misguided client implementations. These are followed by a number of suggestions on how to deal with them now and in the future.

THE WISCONSIN INCIDENT

Ingress on the University of Wisconsin network in May 2003 was normally about 40,000 PPS aggregated over all campus computers[6]. Early on the morning of 13 May 2003 NTP traffic started a dramatic increase first to 80,000 packets and then to levels the Wisconsin campus network could not handle. It was discovered that most of this traffic was destined for a public campus NTP server and coming from an apparently unique port, so steps were taken by the commodity Internet service provider (ISP) to temporarily block traffic from this port to the server. Eventually, an average of 250,000 PPS second were discarded from over 500,000 different NTP clients.

The problem was traced to four models of home routers sold as a commodity item, which later were found with defective network code. If the configured time server could not be reached for some reason, the code would retry at 1-s intervals, which is far more frequent than expected. In addition, the server address was hard coded in the firmware and could not be changed.

If only a few or a few dozen of these routers were involved, even this bit of crass engineering would not be a problem. However, the manufacturer sold 700,000 routers and they were all sending traffic to the Wisconsin server at the same time. As long as the server and network path were operating normally, the situation was tolerable; however, if not, the traffic from the router population could result in an unsustainable network traffic rate over 400 Mbps.

To make matters worse, the four router models contained a firewall designed to protect against unwanted traffic. A naive user could inadvertently configure the firewall to reject NTP packets received from the server, which would have the same result as a broken server or network.

It is not the intent of this paper to pass judgement on the wisdom of this design or the lack of it. It is the intent to comment on how Wisconsin and the router manufacturer is dealing with the problem. It is not considered practical to notify all the users, recall the routers or update the firmware. Briefly put, the situation is tolerable as long as the Wisconsin time server is reachable and operating normally. However, if something breaks in the ISP or Wisconsin campus, a traffic splurge can occur with amplitude up to 700,000 PPS.

There were many lessons learned in the incident both by the Wisconsin operators and the router manufacturer. Some lessons are patently obvious, like allowing the user to select the server address, backing off the request rate when the server does not respond and opening the NTP port in the firewall by default. A collection of best practices designed to protect the Internet from naughty implementations is given in [4].

THE USNO EXPERIENCE

The USNO flagship time servers in Washington, DC, are certainly among the busiest in the world. There are three servers sharing an aggregate peak load of between 3,000 and 7,000 PPS and badly overloading a 10-Mbps Ethernet network. The load on each of the three servers requires between

10 and 12 percent of the available machine cycles. Another way of expressing this is to establish a redline equal to fifty percent of the available machine cycles, which means that, if the load could be balanced between the three machines, redline would occur at approximately 35,000 PPS and require over 25 Mbps on the network. Still a third way of putting it is to observe the potential capacity of the system is about five times the current load. The downside is that the network at present is not capable of handling even the current load. The results seen from a client at University of Delaware show packet loss of about ten percent and increased jitter in the tens of milliseconds up from the expected millisecond of a year ago.

One may well ask where those packets are coming from. If from a client running the NTP public software, packets are sent at intervals of 64 to 1,024 s. Usually, the client starts out at 64 s in order to quickly stabilize the frequency, then increases in steps to 1,024 s. If the aggregate rate is 35,000 PPS and all clients had slowed to 1,024 s, this would be sufficient for 36 million clients or over ten percent of the population of this country. The worst offender at the moment is spraying 14 PPS at the USNO servers, which is equivalent to the aggregate rate of 731 properly engineered clients.

An interesting thing to observe about the USNO traffic characteristic is that it peaks at 7,000 PPS on or about the hour and drops back to 5,000 PPS between the hours. Apparently a large number of clients are run from hourly scripts. A simple way to amortize the peaks would be to stagger the scripts over the hour. Further investigation shows that a large number of packets are coming from a university firewall. Apparently, 2,000 clients behind the firewall are using the USNO servers with a total impact of 9 PPS. There is no engineering rationale that can support this configuration. Only the firewall should synchronize to the USNO servers; the remaining campus clients should synchronize to the firewall.

THE NIST EXPERIENCE

The time servers operated by NIST as a group process 1.4 billion packets per day or about 16,000 PPS and growing by about 7 percent per month. The flagship servers in Boulder, CO, are in much the same shape as the USNO servers. There are three servers behind a load balancing device, running at about the same level of traffic as the USNO machines. However, in this case the network is apparently able to handle the load. There are suspicious that this is not the whole story, as there is evidence that implementations other than the public software are massively impolite and hammer the servers at intervals much less than 64 seconds.

In an effort to gauge the extent and severity of this problem, the flagship servers were surveyed over a period of two days. Each of the three machines collected data on recent packet arrivals using an algorithm similar to the page replacement algorithm used by a virtual memory operating system. Both algorithms use a list sorted in the order of usage from the newest to the oldest. When a new packet arrives the list is searched for an entry matching the IP address. If not found, the new entry is placed first in the list and the other entries shifted back to make room. If an old entry matching the IP address is found, the entry is moved first in the list and the entry updated to reflect the time since the entry was first found, the time since most recently found and the total number of packets in the burst.

The size of the list in each of the three machines is limited to about 1,200 entries and the arrival rate is close to 1,000 PPS per machine. The lists quickly fill up and a decision must be made either to discard the oldest entry or discard the new arrival. This is done on a probabilistic basis, so the actual operation is in effect a sampling process. However, the arrival rate is so large that entries

don't live very long before being swept out by new arrivals; in fact, new nonduplicate entries are swept out after only nine seconds. This is not necessarily a bad thing, since the real abusers generally use retry intervals less than nine seconds.

The data are retrieved by running a program on each machine at substantially the same time to retrieve and consolidate the three lists. This assumes the load balancer distributes incoming packets at random to the three machines. After a number of filtering and sorting operations the following interesting conclusions result. In the 9-s window on the three machines the aggregate list contained 3,595 packets (400 PPS), which represents about 13 percent of the total number of 3,000 arrivals during that period. Of the total, 1,094 represent bursts from 574 different clients where the spacing between packets is less than 5 s. Altogether, the burstmakers account for about 313 PPS. In effect, 14 percent of the clients account for 78 percent of the total load.

Of the 574 burstmakers, 15 were sending at rates greater than 1 PPS (28 PPS total), 253 sending at rates between 1 and 2 PPS (166 PPS total) and the remaining 36 sending at rates between 2 and 5 PPS (120 PPS total). The most bizarre observation is the length of the bursts. Of the 574 burstmakers, there were 379 that lasted less than one minute (214 PPS total), 189 that lasted less than one hour (93 PPS total) and 6 that lasted over one day (6 PPS). The worst two had been sending 2 PPS for over two days, which is the limit of observation and probably means they were sending continuously.

Here we have assumed the burstmakers are individual clients. However, it may well be that these represent multiple clients whose traffic traverses a firewall Network Address Translation (NAT) device. This causes all packets from multiple devices to appear to come from the same IP source address, so the burstmaker elephants might turn out to be mice hiding behind the firewall.

This level of abuse, whether due to misguided client implementations, as the case in the Wisconsin incident, or cyberterrorists may be the single worst enemy of a busy public server. Not only is the burstmaker traffic intrusive, sending at these rates has absolutely no value in precision timekeeping.

DEFENSIVE MEASURES

What can we do about this situation? So far as can be determined, the past and present clogging attacks are due to defective or ill-conceived client software designs. If so, it may be that an aggressive program of education can reduce the impact of these attacks. To that end, the SNTP specification document has been revised and expanded with explicit advice to the designers and implementors [4][4]. Where this is seen, believed and implemented, the problem may abate. However, many companies today, including router manufacturers, have outsourced the design as well as the manufacture and have little opportunities for quality control and product review. In addition, the explosive growth in the use of NTP in everything from print servers to uninterruptable power

4. Internet Drafts (IDs) are not normally citable, as they represent works in progress until the definitive document becomes a Request for Comments (RFC). As evident in this paper, it is highly important this information be available to potential implementors quickly. However, as the final draft was accepted in October 2003 and is yet to be published as RFC, the ID is cited anyway.

supplies suggests a more proactive approach. A medley of possible actions is summarized in following sections.

SOLUTIONS BASED ON CREATIVE DIVERSITY

A natural approach is to spread the load over geographically diverse servers, which has been done by both NIST and USNO. But, at the moment the 20 USNO servers deployed outside the Washington, DC, area see only about 12 percent of the Washington traffic. In principle, additional servers could be added as needed at whatever location is nearing redline. The problem with this approach is how to advise the generally naive user which server is best suited for each particular location. For instance, the Windows XP client is set by default to time.microsoft.com, but is easily set by a user to time.nist.gov and with no obvious provisions to use any of the other 16 servers operated by NIST.

Most SNTP clients obtain the server IP address using the Domain Name System (DNS), which returns the IP address when given the server name. In fact, the DNS response can contain several addresses if they all are associated with the same distributed service. Current DNS servers can rotate the order of these addresses, so if a dumb SNTP client takes the first one, it will probably get a different address each time the program is run. Of course, this scheme does not find the best or closest server, just equalizes the load over the available servers. However, an argument can be made today that the IP network infrastructure has evolved to a substantially uniform level of service anywhere in the US. So, to first order, it doesn't matter which NIST or USNO server is selected, only that they be selected with substantially equal probability.

A scheme such as this has been implemented to aid NTP server discovery in the general population. A distinguished name, in this case pool.ntp.org, is associated with a number of volunteer public servers in different areas of the world, each providing nominally equivalent service. The areas are distinguished by prefixes such as eu.pool.ntp.org for Europe and ca.us.pool.ntp.org for California. Since the addresses are randomized within the area, the effect is to spread the clients evenly over the population in that area. Currently, the scheme is used only for stratum-2 clients without access restrictions; however, the scheme could be used to spread clients over the busy public primary servers, but only if a uniform access policy is adopted.

SOLUTIONS BASED ON CLIENT REDIRECTION

It would be very useful to find a way where the client could be told to redirect traffic to another server, possibly in another place as load imbalances develop. If congestion developed at time.nist.gov, for instance, a client could be told to redirect to another NIST server elsewhere in the country. It is possible to do this in either of two ways. In the first, a server can encapsulate arriving packets and send to a selected server. The selected server would decapsulate the packet, process it and return to the client using its own IP adddress in the packet. A cooperating NTP client would notice the source address has been changed and update its configuration accordingly.

A key requirement in this scheme is that the client is able to associate the original request with a reply from a server with different IP address. This is the same challenge as with the Manycast scheme described in the next section and will be considered there.

In the second way a redirection feature could be incorporated in the protocol design through use of the newly defined extension field, normally used to convey cryptographic values used in server

authentication. One attractive feature in either scheme is that the redirection would be dynamic, not necessarily frozen at the time the server name is resolved.

SOLUTIONS BASED ON IP MULTICASTING

IP multicasting is designed to deliver a single packet to multiple destinations for applications such as teleconferencing and broadcasting. It has been a generic feature of the Internet architecture for many years, but not a popular offering by Internet service providers. Multicasting is an ideal paradigm to support very large client communities and is supported by the current NTPv4 public distribution. Multicast group addresses have been assigned by the Internet Assigned Numbers Authority (IANA) for both IPv4 and IPv6 address families [2].

There are problems hindering the universal deployment of an NTP multicasting service. First, the providers are reluctant to deploy it since, among other concerns, it can be expensive if the distribution tree changes rapidly. Second, routing policies are difficult to manage with the current interdomain routing architecture and protocol. In addition, the current router multicast implementations are sometimes frail under the best of circumstances, but also are vulnerable to denial of service attacks.

A problem believed solved in the current NTPv4 protocol model is time offset correction for the sometimes very different network paths used for the multicast (point-to-multipoint) path and the unicast (point-to-point) path. Special provisions have been incorporated in the NTPv4 protocol design in which the time is first determined using the unicast path, which requires a brief packet exchange between the client and server. Once the correct offset applicable to the multicast path has been determined, the client reverts to listen-only operation.

A feature has been implemented in current NTPv4 which provides for the discovery, configuration and mitigation of NTP servers that might be lurking in the nearby network neighborhood. Called NTP Manycast, it operates by selective multicasting of a message inviting participating servers to respond if they can provide service. Upon receipt of the server response, the client mobilizes associations for each server found and continues operation as in ordinary NTP. In general, several servers do respond, but using the engineered NTP mitigation algorithms all but the best three are discarded and operation continues with only these three.

A key requirement for NTP Manycast is that the clients associate a reply from a server with a request previously sent to a different IP address. This is done by comparing one of the timestamps in the request with the corresponding timestamp in the reply. As this is a very fine-grained value and unlikely to be accidently replicated in some other packet, it serves as a unique identifier to confirm the reply matches the request. This feature is useful for NTP Manycast and in fact for any scheme making use of dynamic server redirect.

NTP Manycast is well suited for a relatively dense network where IP multicast service can be supported, but probably not in the wider Internet of interest to national public time servers and clients. However, a possible solution might be represented by the Anycast paradigm [5] and Border Gateway Protocol (BGP) Anycast feature [7] and/or the Protocol Independent Multicast (PIM) Rendezvous (RP) feature [3]. In BGP Anycast any of a set of BGP routers advertises a distinguished IP address block and agrees to serve an associated function, most commonly host name to address translation. A client sends a request to this address and the first BGB router along the path intercepts it and returns the reply, avoiding further transit in the network. For wide area multicasting the RP feature of PIM [3] aggregates traffic to the same destinations, saving load on parallel paths.

A solution based on BGP Anycast has been suggested for Wisconsin and other locations near traffic confluences. It is designed to deflect NTP traffic at the border gateways to one or another locally connected NTP servers. With Anycast routing, multiple border gateways intercept traffic to a designated destination IP address and use a load sharing scheme similar to USNO and NIST to distribute packets over the available NTP servers. While this would work without multicast, the BGP Anycast scheme would allow the border gateways as a system to intercept traffic nearer the clients and spread the load over multiple servers.

SOLUTIONS BASED ON ACTIVE ACCESS LISTS

A mechanism long used to protect against unauthorized access is the access control list (ACL). The ACL contains a list of accepted IP addresses (whitelist) and/or denied addresses (blacklist), possibly including modifier conditions. ACL features have been included in the NTP public distributions for many years and have been used to control access for time services and monitoring services. While the ACL provisions are quite useful if only a small number of clients are to be included or excluded, the problem for a busy public server is how to construct the ACL dynamically. A scheme to do this might be as follows.

Usually, and especially for the USNO clients, clients can be distinguished by domain, as determined by a reverse-DNS query that returns the fully qualified domain name given the IP address. For instance, should the USNO servers be restricted to only military clients, only those with domain names ending in MIL would be accepted and added to the whitelist, while others would be added to the blacklist. The relatively expensive reverse-DNS query then needs to be done only once.

Schemes like this are relatively easy to implement, but can cause significant increases in memory and processor resource requirements. As said in the abstract, a better approach may be to find the elephants and shoot them. This approach is discussed in following sections.

PROACTIVE DEFENSE: THE KISS-O'-DEATH SCHEME

NTP client configurations tend to last awhile, some even persisting long after the server has ceased operation, moved or been replaced by something unresponsive to NTP clients. Such situations can and have gone on for years with little hope of contacting the responsible person and have the client shut down. A means is needed to advise the client on the occasion of a fatal error, either due to cryptographic mismatch, excessive zeal or other nonrecoverable situation.

Current and previous NTP versions include an error message used with symmetric key cryptography and called the crypto-NAK. It advises the client that cryptographic authentication has failed due to an incorrect or nonexistent key. Upon receiving a crypto-NAK the client is expected to cease operation and send a message to the system log. While this was useful in such a narrow scope, a generalized error message facility was needed. The solution was to take advantage of certain packet fields that were not used when the server clock was not synchronized. Because the first use of this message was to tell the client to stop transmitting, it is called the kiss-o'-death (KoD) packet and the message code it contains is called the kiss code. There are about a dozen kiss codes now in use, some notifying error, some requesting rate reduction and some requesting outright stop.

At the present time the KoD has been implemented only in the public software distribution and not yet in the various SNTP clients that might heed it. However, the SNTP specification document

now on the standards track makes strong recommendations for its use as well as strong recommendations on best practices in related traffic management areas. Now the KoD is probably best used to signal permanent failure conditions, like authentication failure and access violation. The KoD itself is not without hazard. A determined clogger could create the same overload as if the server responded in the ordinary way. For this reason the KoD packet rate is limited to a safe value and the excess thrown away.

If the NTPv4 server is so configured, it keeps the KoD address of an offending client indefinitely and discards all further packets from that client. Upon receiving a KoD, the client is expected to cease operation or moderate the rate, depending on the kiss code. Upon receiving a STOP kiss code, the client disables the server association and sends a message to the system log. The error message code is revealed to monitoring and logging means as well. While the client is free to ignore the KoD, of course, the standards documentation and current specification give strong advice that future NTP client implementations must respect the KoD.

SHOOTING THE ELEPHANTS: THE CALL-GAP SCHEME

It would seem a good defense against clogging attacks would be simply to discard packets and (possibly) to send a KoD packet. For a datagram protocol like NTP, this is the only way to protect the network from overload. The problem is that defective clients might try even harder if their packets are not answered, which did in fact happen in the Wisconsin incident. Telephone companies have long employed a scheme called call-gap designed to handle gross overloads due to a disaster or popular concert ticket opportunity. It's rather like a rotating power blackout where dial tone is withheld for varying periods for varying pools of customers. While it might not be possible in principle to throttle a determined clogger, it might be possible to withhold service, by dropping packets. The expectation is that a properly designed client might conclude that, if there is no reply to the request, the server is telling it to slow down or shut up. Coupled with a KoD that works, this approach might be an effective clogging defense.

Drawing from this example, a call-gap feature has been implemented in NTPv4. It operates by measuring the interval between packets for each recent client and dropping (gapping) requests if the interval is too small. It operates as follows:

1. If a packet arrives less than two seconds after the previous packet it is dropped and a KoD sent if configured.

2. If the exponentially averaged interval between packets is less than five seconds, succeeding packets are dropped and a KoD sent if configured.

The NTPv4 scheme is used at USNO; NIST uses a slightly different scheme, but the differences are not significant.

Experience with the call-gap scheme is mixed. Currently, about 6.5 percent of the packets arriving at USNO are gapped, while only 0.1 percent arriving at NIST are gapped. Judging from the data reported in this paper, these schemes are quite good at finding elephants to shoot, but more needs to be done to effectively aim the gun.

It was noted for both USNO and NIST that the load balancer distributes arriving packets randomly to the three servers. To be more precise, the packets are delivered round-robin fashion; however, at aggregate arrival rates in the thousands of PPS and the arrival rates for even the most ornery clients

only 14 PPS, the effect is the same. A problem with random distribution is that the probability that all or even most of the packets land on the same server is small, so that the KoD and call-gap schemes become less effective. A remedy for the load balancer would be to hash arriving packet source IP addresses and select the server using a mask on the result. So far as known, the load balancers provide no such feature.

CONCLUSIONS

We of course should have seen this coming. NTP is not the only distributed Internet service vulnerable to clogging attacks. A better example might even be the DNS, which has already taken steps to defuse and disperse even the normal load [1]. However, while DNS service is generally complete in one packet exchange, NTP invites a more or less continuous volley, although intended at low rate. The lessons learned with DNS and now NTP should serve as a wakeup call for some very serious protocol engineering designed explicitly for name resolution, network time and other distributed, ubiquitous protocols.

REFERENCES

1. Abley, J. Hierarchical anycast for global service distribution. Technical Note ISC-TN-2003-1, Internet Systems Consortium, March 2003.

2. Hinden, R., and S. Deering. Internet protocol version 6 (IPv6) addressing architecture. Request for Comments 3513, Internet Engineering Task Force, April 2003.

3. Kim, D., D. Meyer, H. Kilmer and D. Farinacci. Rendevous point (RP) mechansim using Protocol Independent Multicast (PIM) and Multicast Source Discovery Protocol (MSDP). Request for Comments 3446, Internet Engineering Task Force, January 2003.

4. Mills, D., D. Plonka and J. Montgomery. Simple network time protocol (SNTP) version 4 for IPv4, IPv6 and OSI. Internet Draft draft-mills-sntp-v4-00.txt, Internet Engineering Task Force, September 2003.

5. Partridge, C., T. Mendez and W. Milliken. Host anycasting service. Request for Comments 1546, Internet Engineering Task Force, November 1993.

6. Plonka, D. Flawed routers flood University of Wisconsin Internet time server, Technical Report http://www.cs.wisc.edu/~plonka/netgear-sntp/.

7. Thaler, D. Border gateway multicast protocol. Request for Comments 3913, Internet Engineering Task Force, September 2004.