Measurement and Diagnosis of Address Misconfigured P2P Traffic

Zhichun Li, NEC Laboratories America, Inc.
Anup Goyal, Yahoo! Search
Yan Chen and Aleksandar Kuzmanovic, Northwestern University

Abstract

Through measurement study, we discover an interesting phenomenon, P2P address misconfiguration, in which a large number of peers send P2P file downloading requests to a "random" target on the Internet. Through measuring three large datasets spanning four years and across five different /8 networks, we find address-misconfigured P2P traffic on average contributes 38.9 percent of Internet background radiation, increasing by more than 100 percent every year. To detect and diagnose such unwanted traffic, we design the *P2PScope*, a measurement tool. After analyzing about 2 Tbytes of data and tracking millions of peers, we found that in all the P2P systems, address misconfiguration is caused by resource mapping contamination: the sources returned for a given file ID through P2P indexing are not valid. Different P2P systems have different reasons for such contamination. For eMule, we find that the root cause is mainly a network byte-order problem in the eMule Source Exchange protocol. For BitTorrent misconfiguration, one reason is that anti-P2P companies actively inject bogus peers into the P2P system. Another reason is that the KTorrent implementation has a byte-order problem.

Peer-to-peer (P2P) traffic has grown to be one of the dominant sources of Internet traffic. Given the unprecedented amount of P2P traffic flowing through the Internet, misconfiguration, due to software bugs or attackers with malicious motives, is a serious problem of both the Internet and P2P systems.

Usually it is believed that so-called Internet background radiation [1] — unsolicited traffic sent to all IP addresses including unused ones — is mainly scanning traffic or denial-of-service (DoS) backscatter traffic. As part of the contributions of this article, however, after we analyze about 2 Tbytes of data from three institutions spanning four years on five different /8 networks, we discover that address-misconfigured P2P traffic is one of the major sources (an average of 38.9 percent in terms of the number of connections) of Internet background radiation. Address-misconfigured P2P traffic is caused by a large number of peers sending P2P file downloading requests to a target that has never been part of the P2P network. In addition, we observe that from 2004 to 2007, address-misconfigured P2P traffic increased more than 100 percent each year (as shown in Fig. 1). To the best of our knowledge, we are the first to discover and study address-misconfigured P2P traffic with large-scale datasets.

For end users, such unwanted traffic affects their P2P system performance and can involve innocent users in distributed DoS (DDoS) attacks unconsciously [2].

From the Internet service providers' (ISPs') perspective, such misconfigured traffic wastes Internet resources. As a first order estimation, extrapolating the address-misconfigured P2P traffic observed in our datasets to the whole Internet, we find it consumes modest bandwidth, 7.9 Gb/s globally. Moreover, this traffic is mostly intercontinental. Therefore, given a 10 Gb/s intercontinental link (e.g., between Los Angeles and Tokyo) costs *$1.4 million* per year to lease, ISPs might want to remove such traffic for a more "green" Internet.

Our first contribution, discovering and measuring address-misconfigured P2P traffic, further motivates us to build tools for *detection* and *diagnosis* of such unwanted traffic. Here detection refers to attributing the misconfiguration to a particular variant or version of the P2P software that contains bugs, or a particular peer group (e.g., peers from anti-P2P companies who are hired by the movie and music industries to protect their copyrights). Detection is of crucial importance, because without it, nobody will take responsibility for fixing the problem, given the large number of P2P software variants. The root causes can be either internal (bugs or software misbehavior) or external (e.g., injection attacks). Diagnosis means inferences of the root causes, say, by locating the software code that contains the bug or identifying the anti-P2P peers that trigger the misconfiguration.

We design two general principles:
- P2P software testing by tracking its information flow in a controlled environment
- P2P traffic measurement including both passive monitoring and active backtracking to identify misconfiguration events in the wild

Applying the principles above, we design and implement *P2PScope*. Our analysis shows that *all* address misconfigurations are caused by resource mapping contamination, that is, the peers returned through indexing are invalid, thus causing a large amount of incorrect file downloading requests. Different systems have different reasons for contamination. We have performed root cause analysis for the two largest P2P systems, eMule and BitTorrent.

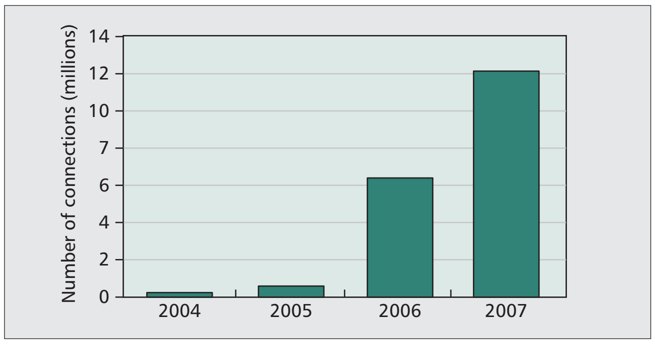


Figure 1. *The total number of connections that match the P2P payload signatures from 2004 to 2007 (LBL).*

For eMule, one major root cause is a network byte-order problem in the source exchange protocol. We confirm that aMule (an eMule variant) has the byte-order bug. For BitTorrent, address misconfiguration is mostly disseminated by the Peer Exchange (PEX) protocol implemented by uTorrent-compatible clients. We find two reasons:

- Some anti-P2P companies actively inject bogus peers through the PEX protocol using modified Azureus.
- KTorrent has a byte ordering problem. We have confirmed the bug with the developers.

Passive Measurement Analysis

When monitoring the Internet background radiation, we discover an interesting traffic pattern that has not been described in the literature before. We call it P2P address misconfiguration. In this section, we study its basic behavior through passive measurement. In the next section, we further design the P2PScope system to diagnose its root causes.

Data Collections

In our study, we mainly use three datasets collected at different Honeynet/Honeyfarm sensors. Honeynets are unused address blocks on which we deploy honeypot responders in order to elicit information about incoming probes. Those honeypots are protocol simulators, which respond to incoming probes with faked responses as if the responses are from real machines. Honeyfarm goes one step further, channeling the incoming traffic to virtual machines as if real machines are running at the addresses (Table 1).

The LBL Honeynet Dataset — The LBL Honeynet sensor is at the Lawrence Berkeley National Laboratory with five continuous /24 IP blocks in one /16 network. We have data from 2004 through 2007. The honeynet simulates the common protocols and acts as an echo server (sending back the same requests to the clients) for other port numbers. In 2008, traffic collection was stopped because the Motion Picture Association of America (MPAA) repeatedly complained that the Honeynet IPs were hosting their latest movies (actually false positives).

The NU Honeynet Dataset — The NU Honeynet sensor at Northwestern University has 10 discontinuous /24 IP blocks within three different /8 IP prefixes. It used the same configuration as the LBL Honeynet until 23 August, 2007. After that, we developed the P2P-enabled Honeynet for the P2PScope system, which simulates P2P protocols on all port numbers.

The GQ Honeyfarm Dataset — We have 26 days of traffic from the GQ honeyfarm at the International Computer Science Institute, which was originally designed to capture worms and botnets. The GQ honeyfarm has four /16 networks in one /8 network. Since the GQ honeyfarm runs a different response scheme, we use the GQ trace only for prevalence analysis.

Address Misconfiguration Event Identification

To identify address-misconfigured P2P traffic, we first filter out the scanning traffic caused by botnets or worms based on whether they have malicious payloads that match known worm/botnet signatures or they scan a large number of IP addresses (≥ 10). We have also checked the removed traffic against P2P protocol signatures, and have not found any matches. For remaining traffic, we detect P2P connections based on whether their payloads match P2P protocol signatures.

After that, we aggregate the address-misconfigured P2P traffic to events. We find all the peers target a few hotspots in Honeynets. Ninety-seven percent of peers only contact one Honeynet IP per day (all contact less than four per day). Therefore, we group the P2P connections into address-misconfiguration events based on their target destination. We mainly analyze the events with more than 100 unique sources seen in a six hour interval. For smaller events, it is hard to profile their patterns. The event time boundaries are the time intervals that the number of unique sources reduces to less than one-third of the peak.

The above method only conservatively estimates the lower bound of the number of P2P connections and number of events seen in the Honeynets. We address this by adding a "Likely" category in Table 2 for the event in which we cannot identify which protocol is being used, but its traffic pattern is similar to that of an address-misconfiguration event. These "likely" events follow the same pattern—a large number of sources contact a few hotspots. We believe that they are actually address-misconfiguration events, because there are no similar patterns known for Internet background radiation. The likely cases account for only 9.6 percent of the total events, showing that we are able to identify most cases accurately.

P2P System Diversity

Table 2 shows the percentage of P2P connections that match P2P signatures and the number of address-misconfiguration events found. Each row of the table is for one P2P protocol. The "other P2P" category is for connections that matched P2P protocols other than the six protocols listed in Table 2.

We find that address-misconfiguration events mainly come from six different P2P systems. BitTorrent and eMule are the most popular P2P systems, and thus also generate the most traffic and events. We also find a number of address misconfigurations from other P2P software, such as Gnutella, Soribada, Xunlei, and VAgaa.

These P2P systems include centralized, decentralized distributed hash table (DHT)-based, and decentralized unstructured P2P systems. This diversity suggests that the prevalence of address misconfiguration is not confined to any single type of P2P system.

	LBL	NU	GQ
Sensor size	5 /24	10 /24	4 /16
Trace size	901 Gbytes	916 Gbytes	49 Gbytes
Starting time	2004/03/07	2006/09/01	2006/01/03
End time	2008/01/21	2007/12/31	2006/01/28

Table 1. *Data description.*

	event (≥ uniq srcs) 100		P2P connections by payloads		comments
	LBL	NU	LBL	NU	
eMule	143	416	5.96%	76.52%	Popular, especially in Europe
BitTorrent	74	211	75.1%	19.79%	Popular
Gnutella	4	3	2.48%	3.65%	Popular, unstructured
Soribada	6	0	15.8%	.0001%	Popular in Korea
Xunlei	12	0	0.34%	0%	Popular in China
VAgaa	1	1	0.14%	0.013%	Popular in China
Other P2P	0	0	0.17%	0.018%	
Likely	73	20	N/A	N/A	

Table 2. *Address-misconfigured P2P traffic distribution and event distribution (LBL and NU Honeynet datasets).*

Characteristics of Address Misconfiguration Events

We also discover that such events are quite heterogeneous. The scale of events (in terms of duration and the number of unique sources involved) varies substantially. Some events are as short as one to three hours; some are as long as one month. The average number of peers in eMule events is 1404 for LBL and 1028 for NU. BitTorrent events are on a smaller scale: the average number of peers in BitTorrent events is 894 for LBL and 291 for NU.

Prevalence in Time and Space

We use the four-year LBL data to analyze the temporal trend of address-misconfiguration. Since the scale of the events differs substantially, instead of using the number of events, we use the total number of connections that match the P2P signatures in a year as the metric for studying temporal trend. This metric gives us the lower bound of the number of P2P connections. Figure 1 shows the increasing trend of address-misconfigured P2P traffic. Additionally, we analyze the annual trend of the total volume of Internet background radiation in the LBL sensor. The total traffic is stable while the address-misconfigured P2P traffic increases quickly.

We also observe that address misconfiguration is prevalent across the IP space. As we mentioned earlier, the LBL and NU Honeynets are in four different /8. In all the four /8 prefixes, we find misconfiguration traffic. We also observe similar behavior in the GQ Honeyfarm traces. Therefore, we believe the behavior is not confined to a single subspace of the IP space.

Global Address-Misconfigured P2P Traffic Estimation

We try to estimate the percentage of address-misconfigured P2P traffic in Internet background radiation traffic. It is very challenging, given the limited data we have. We use the data (June–December 2007) from the LBL and NU Honeynets for this analysis. We treat the percentage estimated in each /8 prefix of the Honeynet sensors as an independent sample. We then use the average percentage from all the samples as a more representative result. As a first order estimation, we find the average percentage of address-misconfigured P2P traffic in the Internet background radiation is 38.9 percent.

System Design

General Design Principles

In general, two reasons can potentially cause P2P misconfigurations: software bugs, and external attack or misconfiguration injection. Two approaches are useful for detecting misconfiguration (attributing the problem to a particular software version or peer group):

- Measurement, including both passive monitoring and active backtracking
- Tracking the information flow for the suspicious P2P software in a controlled environment

The first approach helps quickly identify which software version, communication mechanism, or peer groups is/are potentially the sources of the misconfiguration. This approach also helps identify the possible attacks or misconfiguration injected, such as those from anti-P2P companies. The second approach can further validate misconfiguration caused by software bugs.

After detection, we want to further diagnose the root causes of the misconfiguration. This diagnosis step is harder than detection, and usually requires manual inspection. Information flow tracking is still very useful in this stage. Another useful approach is to form the hypotheses regarding the root causes and then to conduct experiments for validation.

Per the principles above, P2PScope has two major subsystems, detection and diagnosis, as shown in Fig. 2.

Detection Subsystem

The detection system has three modules:

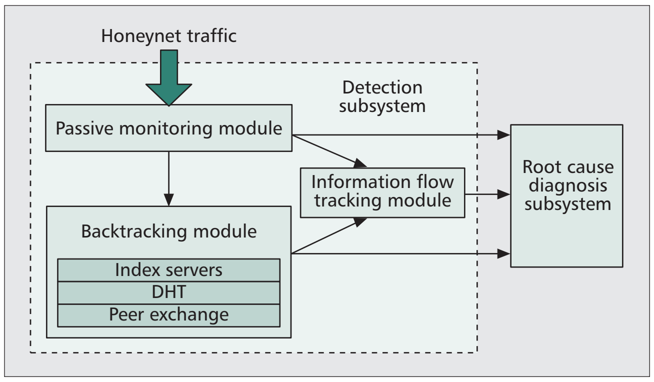
- Passive monitoring module
- Backtracking module
- Information flow tracking module

The passive monitoring module detects the address-misconfigured P2P traffic from Honeynets. Further, the backtracking module collects more information about the misconfigured peers and the files related to address misconfiguration. When these two measurement modules detect some suspicious P2P software version through protocol analysis, the software will be passed to the information flow tracking module. In the information flow tracking module, we install that version of P2P software and check its peer propagation. Finally, all collected information will be output to root cause diagnosis subsystem.

Passive Monitoring Module — We find that Honeynets are useful for monitoring the address-misconfigured P2P traffic on unused IP blocks. The Honeynets respond to the incoming connections and generate "faked" responses. We use Honeyd-based [3] lightweight Honeynets. Honeyd uses port numbers to identify protocols, but P2P traffic is on random port numbers. To overcome this limitation, we apply a *payload signature based* approach. We first identify the P2P protocol based on payload signatures and then call appropriate P2P responders for further processing. Currently, we have developed BitTorrent and eMule protocol responders.

Backtracking Module — To understand how peers get misconfigured, the backtracking module tracks the peers in real time. In P2P systems, a peer needs to download from a list of peers that have the given resource. Three methods can facilitate the resource to peer mapping: central index servers (tracker in BitTorrent terminology), distributed hash indexing, and peer exchange (query more peers from known peers). For eMule and BitTorrent, we implement all three approaches. In the backtracking module, for a given resource we periodically query the index servers or DHT for the peer lists. Then, for any peer, we use peer exchange protocols to query more peers. The whole process stops when the total number of peers remains stable.

Information Flow Tracking Module — Whenever a certain P2P


Honeynet traffic
Detection subsystem
Passive monitoring module
Information flow tracking module
Root cause diagnosis subsystem
Backtracking module
Index servers
DHT
Peer exchange
Figure 2. *Architecture of the P2PScope system.*

client version is suspicious, we install it in our controlled environment. In the controlled environment, by interpreting the protocols (index server, DHT, and peer exchange protocol) used for peer propagation, we monitor the peers in and out through protocol analysis. If the software gives out a peer that never comes in, we know the software will cause address misconfiguration.

Diagnosis Subsystem

Tracking down the root causes of address misconfiguration is a challenging problem. We have developed the following tools for understanding the root causes:

- Track the information flow within the suspicious P2P software.
- Trace the Honeynet IP addresses propagated in the P2P systems.
- Check the routability of the peers returned.
- Check whether the peers are on the anti-P2P IP lists and analyze their behavior.
- Check the reverse byte order of peer IPs, because it is one of the most frequent mistakes in networked system implementations.

To understand why the bogus peers can be produced by the suspicious P2P software, we trace how the bogus peers are produced. Another useful toolkit is the routability checking. In [4], Cooke et al. report 66.8 percent of all possible IPv4 addresses are unroutable. If we assume that the bogus peers are produced randomly, we would find a reasonably large percent of unroutable bogus peers. If the percentage is small, it implies that bogus peers are not produced randomly. The percentage of unroutable peers can serve as a low bound of the bogus peers. Furthermore, since we suspect the anti-P2P companies, which try to take down the P2P networks, might be related to address misconfiguration, we also check whether the peers belong to the anti-P2P IP list.

With the above tools, we mainly try to answer the following questions:

- Which peers spread misconfiguration?
- What is the root cause?
- How is misconfiguration disseminated?
- What is the percentage of bogus peers in misconfigured P2P networks?

Detection and Diagnosis Results

We have deployed P2PScope in the NU Honeynet since August 2007.

We first study the general characteristics that are related to the root causes of all P2P systems. Then we focus on two case studies for eMule and BitTorrent, the most popular P2P software generating most of the address-misconfigured P2P traffic.

Data Plane Traffic Radiation

Usually, there are a number of different protocol messages involved in a P2P file sharing network. Address misconfiguration in all the six P2P systems that we explored has a common feature: the traffic is caused by file downloading requests, i.e., data plane traffic which is directly related to data transfer. Different P2P systems use different ways for exchanging control plane information (e.g., index server, Peer Exchange Protocol, and DHT), but all have the same problem of giving bogus IPs to peers. These bogus IPs propagate quickly in P2P systems and cause large amounts of address-misconfigured P2P traffic.

eMule Misconfiguration Diagnosis

Which Peers Spread Misconfiguration? — Hypothetically, the misconfigured peers that target the Honeynets might include anti-P2P peers. However, among 1,771,296 misconfigured peers we observed, 99.90 percent of peers are normal peers (non-anti-P2P peers).

What Is the Root Cause? — When analyzing the network-level behavior of the eMule misconfiguration, we notice that the IP addresses formed by reversing the byte order of the targets were mostly alive. With the real-time P2PScope system, we have done two experiments to verify the byte-order problem. Furthermore, we have confirmed the bug in source code.

In the first experiment, we tested 13 targeted destinations in real time, and observed that 61 percent of reverse IPs were indeed running eMule with the same port number of the targets. In the second experiment, we checked whether the reverse IPs of unroutable peers observed in the peer list of a given file ran eMule. We aggregated the result of 100 files found in real time. 10.3 percent of backtracked peer IPs are unroutable. Among them, we observe 69.6 percent whose reverse IPs run eMule. This, again, is more strong evidence in favor of the byte order hypothesis.

We further located the bug in the source code of aMule, a popular eMule alternative. eMule has a complex client ID to IP mapping system called HybridID. In this mapping system, to represent IP in index servers or DHTs different byte orders are used. The source (peer) exchange protocol needs to consider both cases. However, in aMule with versions before 2.1.0, this problem was ignored, causing the byte-order bug.

How Is the Misconfiguration Disseminated? — eMule offers three ways to obtain peers for a file: index servers, eMule source exchange protocol, and eMule Kademlia DHT. We would like to know which one disseminates the misconfiguration. Since we have not observed any misconfigured DHT traffic in the approximately 2-Tbyte trace we have, we believe the DHT does not have the byte-order problem. To attribute the problem to index servers or the source exchange protocol, we queried 100 files and got a total of 37,079 unique peers. We find that 10.7 percent of the peers from the source exchange have Honeynet IPs in their neighbor list, but the index servers never return Honeynet IPs. We have also done a routability check, and found that 12.8 percent of the peers returned by the source exchange protocol are unroutable; none of the peers returned by the index servers is unroutable. Also, 15.7 percent of peers from source exchange have reverse IPs that run eMule; none of the peers returned by index servers has such behavior. Therefore, it is not an index server but the source exchange protocol that is used for disseminating the misconfiguration.

What Is the Percentage of Bogus Peers in the Misconfigured P2P Networks? — We try to estimate the percentage of peers

	Normal peer events	Anti-P2P peer events
Number of events	136-NU, 36-LBL	75-NU, 39-LBL
Client software — NU	90% uTorrent 10% others	100% Azureus
Client software — LBL	57% Bit Spirit, 31% BitComet 12% others	100% Azureus
Message length	Variable lengths	Similar lengths
Files queried	Diverse	Latest music/movies
Avg. no. of connections/IP	25	400
Source correlation	Little coordinated	Highly coordinated
Arrival and departure	Gradually	All together
Average event duration	106.1 hours	4.5 hours
IP distribution	Diverse	Small number of /24

Table 3. *Comparison between normal peer events and anti-P2P peer events.*

(from the source exchange protocol) that have the byte-order problem. This is challenging because there are certain cases, as we show below, in which we cannot determine whether they have a byte-order problem. Instead, we estimate the lower and upper bounds.

If a peer is not running eMule but its reverse IP is, we believe it definitely has a byte-order problem. Such peers are 12.7 percent of the total peers (37,079 peers for 100 files).

If a peer is unroutable but its reverse IP is routable, we believe it is likely to have a byte-order problem. Because for those peers we cannot verify whether their reverse IPs run eMule directly, we only count them in the upper bound, which is 25 percent of the total peers.

BitTorrent Misconfiguration Diagnosis

Which Peers Spread Misconfiguration? — When analyzing the events discussed earlier, we found two groups of BitTorrent events. In one group, the majority of peers (> 90 percent) are anti-P2P peers. We call them anti-P2P events. In the other group, the majority of peers are normal peers. We call them normal events. There are no other events between these. Anti-P2P events surprise us, since they show that the peers from the anti-P2P companies somehow get themselves solely misconfigured.

As shown in Table 3, we compare the properties of these two types of events and discover that their characteristics are indeed different. Peers from anti-P2P peer events all use Azureus. We conjecture that anti-P2P companies modify Azureus and add their functionalities because Azureus is a popular open source BitTorrent client with a nice plug-in architecture. On the other hand, peers from normal peer events run on a diverse set of clients. In addition, the peers in anti-P2P peer events are highly coordinated. They start contacting the Honeynet IPs at almost the same time. They also depart at the same time. By contrast, the peers in normal peer events arrive and depart randomly and gradually. The messages sent in anti-P2P peer events are almost the same; they have a similar message length and even a similar bitmap that annotates the content availability distribution. To some extent, the group of anti-P2P peers is like a clone army with identical soldiers. Moreover, the peers in anti-P2P peer events are from a small number of networks, which belong to server hosting companies. The metric source correlation measures the degree to which the sources of different events are overlapped. Ninety-seven percent of pairs of anti-P2P peer events share more than 25 percent of sources. However, the normal peer events are little coordinated.

What Is the Root Cause? — We have not captured any real-time anti-P2P peer events. Thus, we cannot diagnose their root causes. In this section and later, we mainly focus on normal peer events. Normal peer events can still be caused by anti-P2P peers, as discussed next.

Root Cause I: Anti-P2P Companies Deliberately Inject Bogus Peers in P2P Systems — Our P2PScope system tracks the misconfigured peers in real time. We detect some interesting anti-P2P hosts coming from the same server farm, 72.172.90/24, owned by an anti-P2P company called Artist-direct. Through further analysis on the server farm, we discover that each of these IPs runs hundreds of BitTorrent clients simultaneously. All of the clients run a version of Azureus 2500. Moreover, in the normal case, the peers only respond to a peer exchange message when they download (or own) the file so that they can return the other peers from which they download. However, these anti-P2P clients respond to any arbitrary file; that is, they declare they have any file the remote peer wants and exchange the bogus peers with it. Actually, they also respond to any random file ID even when no file is associated with that value. Artist-direct might have modified the Azureus client to implement this.

We have checked the neighbor lists returned by these anti-P2P IPs. Most returned peers are bogus peers. Among the peers, 56.7 percent are unroutable, 1.8 percent are from the same serverfarm, and we could not connect to any of the remaining 41.5 percent of the peers. We suspect anti-P2P companies want to slow down the downloading process by supplying bogus peers randomly.

The *anti-P2P related normal peers* are the normal peers that query the files in which anti-P2P peers are interested. Most of these peers (> 90 percent) are within two IP hops of anti-P2P peers. The anti-P2P related normal peers are 19.7 percent of the total misconfigured peers and are responsible for 20.7 percent of the probes sent to the Honeynet. Therefore, *the anti-P2P companies are responsible for 1/5 of the traffic observed*. Since it is impossible to identify all the anti-P2P peers, this estimate is quite conservative.

Root Cause II: The Byte-Order Problem of KTorrent Clients — We further actively track peers related to the files requested by the misconfigured peers. We find that a large portion of unroutable peers are supplied by the KTorrent peers, which are suspicious. We also study uTorrent and Azureus, since they are the most popular clients. We set up a controlled environment to test these three clients. We evaluate the top seven torrent files seen in Honeynets.

We classify the peers seen by clients into two types:
• *Incoming peers*: peers that come to the client
• *Outgoing peers*: peers that are sent out by the client

Outgoing — incoming peers are the peers that go out from the client but never come in. These are the bogus peers created by the client itself. Among all outgoing KTorrent peers, 56 percent of them are bogus. Thus, it shows that KTorrent is one of the sources of misconfigurations. µTorrent and Azureus

```
void UTPex::encode(…) // UTPEX.CPP
{       …
        // Use KDE API to get an IP in Network Byte Order.
        // WriteUint32 swap the byte order again!!!
        WriteUint32(buf,size,addr.ipAddress().IPv4Addr());
        … }
void WriteUint32(Uint8* buf,Uint32 off,Uint32 val) //FUNCTIONS.CPP
{
        // swap the byte order
        buf[off + 0] = (Uint8) ((val & 0xFF000000) >> 24);
        buf[off + 1] = (Uint8) ((val & 0x00FF0000) >> 16);
        buf[off + 2] = (Uint8) ((val & 0x0000FF00) >> 8);
        buf[off + 3] = (Uint8) (val & 0x000000FF);
}
```

Figure 3. *Code snippet of KTorrent byte order bug.*

do not have any outgoing — incoming peers.

After that, we study the source code of KTorrent and pinpoint the exact bug. The problem is also a byte-order problem. In Fig. 3, we show the code snippet related to this bug. The uTorrent PEX protocol requires the IP addresses in peer exchange messages to be in network byte order. When KTorrent sends out the peer exchange messages, in its *encode* function it reverses the byte order twice, so finally IP addresses become host byte order again. *addr.ipAddress().IPv4Addr()* is a KDE library method that returns IP addresses in network byte order. However, in the *WriteUint32* function implemented by KTorrent, it reverses the byte order again, and thus causes the problem. To our knowledge, we have not found anyone else reporting this bug.

How Is the Misconfiguration Disseminated? —Similar to the eMule study, we find that BitTorrent DHT and Index servers (trackers) are unlikely to spread or to generate misconfigurations. Unlike eMule, BitTorrent has several incompatible peer exchange protocols proposed by popular BitTorrent clients, such as µTorrent, Azureus, and BitComet. The µTorrent PEX protocol is the most popular, and we find it is the source for misconfiguration dissemination.

To understand which µTorrent PEX compatible clients propagate the misconfigurations, we study the behavior of different clients. We mainly study, when a client obtains peers from uTorrent PEX protocol, how it checks the availability of the peers before it further propagates the peers. We classify the behavior into three categories:

- No checking
- Requiring peers to respond to SYN-ACK (only active port regardless of whether it is running BitTorrent)
- Requiring peers to respond to BT-HANDSHAKE messages

KTorrent propagates anything that comes to it without even checking for an open port. µTorrent propagates anything that has an open port. Azureus only propagates peers that are active BitTorrent clients. Therefore, KTorrent will fully propagate the bogus peers. µTorrent will propagate the bogus peers if they happen to have the port open. Azureus usually will not propagate any bogus peers. To reduce bogus peers, we believe all the clients should have strict checking like Azureus. In this way, the address-misconfigured P2P traffic can be greatly reduced.

Related Work

Misconfiguration Studies

Misconfiguration is widely spread across different network systems on the Internet. Labovitz *et al.* studied wide-area backbone failures and concluded that misconfiguration could be responsible for 12 percent of the incidents [5]. There has also been much recent work that considers the various instabilities and misconfiguration in Border Gateway Protocol (BGP) [6]. However, we find little literature on the study of P2P misconfiguration.

P2P Measurement

Most existing measurement studies of P2P networks are based on the measurement of *normal* P2P traffic [7]. On the other hand, we measure the *abnormal* P2P traffic observed in Honeynets. This is also observed by Yegneswaran *et al.* in [1]. Their work focuses on the comparison of network-level characteristics of P2P address misconfiguration with botnets and worms, and places an emphasis on accurately classifying botnet sweeps and worm outbreaks. We mainly study the prevalence and trend of address-misconfigured P2P traffic, and its root causes. Liang *et al.* show that P2P systems like Fasttrack and Overnet are vulnerable to index poisoning attacks [8]. In contrast, we investigate the root causes for *both* intentional and unintentional index misconfiguration.

Distributed System Debugging

Distributed systems are extremely hard to debug. People have proposed various ways to debug distributed systems, including replay-based predicate checking [9], online monitoring [10], log forensic analysis [11], network trace based inference [12], and model checking [13]. Although some of the approaches might be useful for the root cause diagnosis of P2P address misconfiguration, the following properties make leveraging any of the above existing techniques hard. First, the P2P systems we diagnose are heterogeneous. Multiple different implementations exist for a given P2P protocol such as BitTorrent. Therefore, it is hard to apply model checking techniques. Second, the peers are out of our control. We cannot modify the peers to add the logging functionality required by replay-based predicate checking, online monitoring, and log forensic analysis. It is also very difficult to log network traces from remote peers. Third, the P2P systems we monitor are very large, consisting of millions of peers. All the existing techniques cannot handle this scale yet.

Conclusions

In this article, we first discover a new traffic pattern, P2P address misconfiguration. Its traffic is about 38.9 percent of Internet background radiation and is growing quickly. Such traffic is harmful for both end users and ISPs. As the second contribution, we design P2PScope to understand the root causes. We find that data plane traffic radiation is the common characteristic of the six P2P systems. For eMule, the problem is mainly caused by a byte-order problem. For BitTorrent, we find two reasons:

- Anti-P2P companies deliberately injecting invalid peers
- KTorrent byte-order bug

Acknowledgments

Our thanks to Vern Paxson and the Lawrence Berkeley National Laboratory for support of the LBL Honeynet operations. This work was supported in part by NSF Awards 0627715, DoD Young Investigator Award FA9550-07-1-0074, and DoE Career award DE-FG02-05ER25692/A001. Opinions, findings, and conclusions or recommendations are those of the authors, and do not necessarily reflect the views of the funding sources.

References

[1] V. Yegneswaran *et al.*, "Using Honeynets for Internet Situational Awareness," *Proc. ACM Hotnets IV*, 2005.
[2] Ruben Torres *et al.*, "Inferring Undesirable Behavior from P2P Traffic Analysis," *Proc. ACM SIGMETRICS*, 2009.
[3] N. Provos, "A Virtual Honeypot Framework," Proc. USENIX Security, 2004.
[4] E. Cooke *et al.*, "The Dark Oracle: Perspective-Aware Unused and Unreachable Address Discovery," *Proc. USENIX NSDI*, 2006.
[5] C. Labovitz *et al.*, "Experimental Study of Internet Stability and Backbone Failures," *Proc. FTCS: Int'l. Symp. Fault-Tolerant Computing*, 1999.
[6] R. Mahajan, D. Wetherall, and T. Anderson, "Understanding BGP Misconfiguration," *Proc. ACM SIGCOMM*, 2002.
[7] K. Gummadi *et al.*, "Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload," *Proc. ACM SOSP*, 2003.
[8] J. Liang *et al.*, "The Index Poisoning Attack in P2P File-Sharing Systems," *Proc. IEEE INFOCOM*, 2006.
[9] D. Geels *et al.*, "Friday: Global Comprehension for Distributed Replay," *Proc. NSDI*, 2007.
[10] X. Liu *et al.*, "D3s: Debugging Deployed Distributed Systems," *Proc. NSDI,* 2008.
[11] M. K. Aguilera *et al.*, "Performance Debugging for Distributed Systems of Black Boxes," *Proc. ACM SOSP*, 2003.
[12] P. Bahl *et al.*, "Towards Highly Reliable Enterprise Network Services Via Inference of Multi-Level Dependencies," *Proc. ACM SIGCOMM*, 2007.
[13] C. Killian *et al.*, "Finding Liveness Bugs in System Code," *Proc. NSDI*, 2007.

Biographies

ZHICHUN LI (zhichun@nec-labs.com) is a research staff member with NEC Laboratories America, Inc. He received his Ph.D. in computer science from Northwestern University in December 2009. His research focuses on network security, system security, network measurement and monitoring, and distributed system diagnosis.

ANUP GOYAL is an engineer with Yahoo Search since October 2008. He received his M.S. degree in 2008 from the Computer Science and Engineering Department at Northwestern University and his B.Tech degree from the Indian Institute of Technology, Kharagur.

YAN CHEN is an associate professor in the Department of Electrical Engineering and Computer Science at Northwestern University, Evanston, Illinois. He got his Ph.D. in computer science from the University of California at Berkeley in 2003. His research interests include network measurement, monitoring, and security, and P2P systems. He won the DOE Early CAREER award in 2005, and Microsoft Trustworthy Computing Awards in 2004 and 2005.

ALEKSANDAR KUZMANOVIC is an associate professor in the Department of Electrical Engineering and Computer Science at Northwestern University. He received his B.S. and M.S. degrees from the University of Belgrade, Serbia, in 1996 and 1999, respectively. He received the Ph.D. degree from Rice University in 2004. His research interests are in the area of computer networking with emphasis on design, measurements, analysis, denial-of-service resiliency, and prototype implementation of protocols and algorithms for the Internet. He received the National Science Foundation CAREER Award in 2008.