# Cyber Disease Monitoring with Distributed Hash Tables: A Global P2P Intrusion Detection System

Zhichun Li, Yan Chen, Aaron Beach, Jason Skicewicz
Department of Computer Science, Northwestern University

***Abstract*—Traffic anomalies and distributed attacks are commonplace in today's networks. Single point detection is often insufficient to determine the causes, patterns and prevalence of such events. Most existing distributed intrusion detection systems (DIDS) rely on centralized fusion, or distributed fusion with unscalable communication mechanisms. In this paper, we propose to build a distributed IDS based on the emerging decentralized location and routing infrastructure: *distributed hash table (DHT)*. We embed the intrusion symptoms into the DHT dimensions so that alarms related to the same intrusion (thus with similar symptoms) will be routed to the same sensor fusion center (SFC) while evenly distributing unrelated alarms to different SFCs. This is achieved through careful routing key design based on: 1) analysis of essential characteristics of three common types of intrusions: DoS attacks, port scanning and virus/worm infection; and 2) distribution and stability analysis of the popular port numbers and those of the popular source IP subnets in scans. We further propose several schemes to distribute the alarms more evenly across the SFCs. Evaluation based on one month of DShield firewall logs (600 million scan records) collected from over 2200 worldwide providers show that the resulting system, termed *Cyber Disease DHT (CDDHT)*, can effectively fuse related alarms while distributing unrelated ones evenly among the SFCs.**

## I. Introduction

Traffic anomalies and attacks are commonplace in today's networks, and identifying them rapidly and accurately is critical for large network/service operators. It was estimated that malicious code (viruses, worms and Trojan horses) caused over $28 billion in economic losses in 2003, and will grow to over $75 billion in economic losses by 2007 [1].

The current state of the art in intrusion detection research is to use a combination of network-based intrusion detection systems (NIDS) and host-based intrusion detection systems (HIDS) to protect computer systems from compromise. However, many of these systems still suffer from high false positive and false negative rates due to the quick emergence of new attacks and the limited view of the local IDS.

To this end, distributed IDS (DIDS) are purposed to leverage the diversity and strengths of existing third-party IDS technology to a distributed architecture in order to make global decisions about attacks observed in disjoint locations throughout the world [2], [3], [4], [5], [6]. Each IDS generates the attack symptom report based on its local detection, and sends to a global decision center, termed as *sensor fusion centers* (SFC). The SFC will correlate and analyze the prevalence, cause and patterns of the attack on a global scale. However, existing DIDS rely on centralized fusion techniques (*e.g.*, the root of a hierarchical DIDS) or distributed fusion techniques with un-scalable communication mechanisms for large groups (*e.g*, flooding to every SFC in its group for publish/subscribe model).

The exponential growth in the number of anomalies/attacks in the Internet demands the following features for a scalable DIDS infrastructure: 1) efficient routing of alarms related to different intrusion events to different SFCs without consulting any central directory server or flooding mechanisms among peering points; 2) distributed queries support over multiple SFCs to aggregate information at various levels, *e.g.*, IP address prefixes, port number ranges; 3) load balancing; 4) robustness and fault-tolerance; and 5) attack resiliency

To address these challenges, we propose to build a distributed IDS fusion system based on emerging distributed hash table (DHT) technologies [7], [8], [9], [10]. DHT provides decentralized routing and location infrastructure to many applications, such as p2p file sharing [11], information retrieval [12], content distribution networks [13], and even streaming media [14]. To the best of our knowledge, we are the first to apply DHT to route alarms for DIDS.

A recent DARPA research agenda calls for building a Cyber Disease Control Center (CDCC) to defend against worms [15]. Here we name our DHT-based distributed IDS fusion system the Cyber Disease DHT (CDDHT), which embeds intrusion symptoms into the DHT dimensions so that alarms related to the same intrusion (thus with similar symptoms) will be routed to the same SFC for a global view on the prevalence, cause, and patterns of such attacks.

DHT provides some very nice properties for distributed systems, such as fault-tolerance, robustness [7], [8], [9], [10], and DoS attack resilience [16]. However, the following challenges remain for building the CDDHT system.

- How to design the routing key so that all related alarms are fused to the same SFC, while reducing other irrelevant alarms?
- How to achieve load balancing among SFCs?
- How to support queries regarding intrusions?
- How to provide attack resilience when some set of IDS(s) and/or SFC(s) are compromised?

This paper provides some preliminary solutions and an evaluation to address the questions above. We make the following contributions.

- We design the routing key for three common types of intrusions: DoS attacks, port scanning and virus/worm propagation, based on their essential characteristics (Section 4).
- With one-month of DShield firewall log data consisting of over 600 million scan records from over 2200 worldwide providers, we are among the first to study the popularity dynamics of top port numbers and top source IP subnets of scans, and leverage it to design routing keys with good load balancing characteristics (Section 5).
- We propose several other dynamic load balancing schemes, such as *load-aware node bootstrapping*, *node migration*, *virtual nodes*, *IDS alarm rate limitation* and *aggregation tree* to distribute the alarms more evenly

across the SFCs (Section 5).

We have built the CDDHT system on top of Chord [8], and evaluate it by simulation with daily DShield firewall scan logs as in Section 6. The results show that we can effectively fuse the related alarms while distributing unrelated reports evenly among the SFCs. Open questions on querying and attack-resilience are also discussed.

Regarding other parts of the paper, we discuss the related work in Section 2, give the CDDHT architecture in Section 3, and finally conclude in Section 7.

## II. Related Work

The goal of DIDS systems is to send related intrusion alerts to SFCs for analysis and to obtain a global view of attacks seen on the monitored enterprise network or even the whole Internet. One basic challenge is to route the correlated alerts to the appropriate SFCs. Existing work on DIDS alert routing can be classified into three categories: hierarchical [2], [3] , publish/subscribe [4], [5], [6] models, and distributed passive query [17], [18].

In the hierarchical model, the IDS sensors form a hierarchical structure, where high-level sensors receive the alerts from lower-level sensors. The early work of DIDS [2] used the star topology to route alerts together, which is essentially a two-level hierarchy. After that, there are several variants of multi-level hierarchical DIDSs (*e.g.*, [3]).

However, there are several intrinsic drawbacks about this approach in terms of scalability and reliability. First, all the alerts have to be fused at the root node, which can be easily overloaded. Second, the failure of any non-leaf node will isolate all the nodes in its subtree from the rest. To make things worse, the higher-level the node, the more load it tends to receive and thus the more likely to become heavily loaded and crash.

DIDS based on the publish/subscribe model include Indra [4], COSSACK [5], and DOMINO [6]. Basically, users sign on to different groups based on their interests, and then they share the information within a group with application-level multicast.

Though working well for small or moderate scale DIDS, the publish/subscribe model suffers $O({n_i}^2)$ communication where $n_i$ is the number of nodes in group $i$, if every member in the group is attacked, *i.e.*, they would thus send alerts to everyone else. Given todays global-scale attacks, a publish/subscribe group for an emerging attack may have hundreds of thousands of subscribers or even more. In contrast, in our scheme, each node only sends alert to one SFC. After fusion, that SFC sends the fusion report to all the nodes which have sent related alerts in the past. Thus, the total communication is only $2 \times n_i$. In Section 5-A, we design several schemes for load balancing without incurring much extra communication costs.

Recently, Netbait [18] is proposed as a P2P IDS system on top of PIER [17], a DHT-based distributed rational database which supports SQL-like queries. In Netbait, every node maintains its local worm alarm database, and use the PIER-supported distributed query to collect the desired information from all other nodes. However, to understand the global-scale attacks like viruses/worm propagation, Netbait has to query *every* node, and suffers the scalability problem. In contrast, our CDDHT system proactively fuse the alarms to a small set of SFCs. Thus, for query over any attack, the request only needs to be routed to a few SFCs.

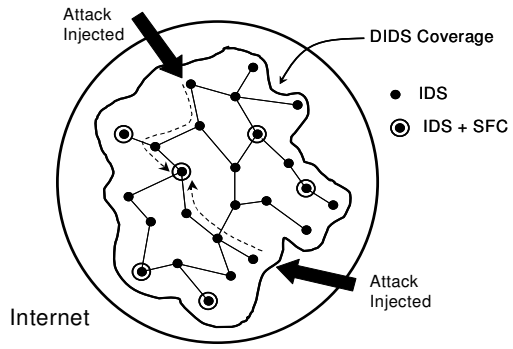


Fig. 1. Architecture of the CDDHT system.

## III. CDDHT System Architecture

The architecture of the CDDHT system is shown in Figure 1. There are two types of nodes in the DIDS system: multiple heterogeneous IDS sensors and SFCs. One strategy for selecting an SFC is to choose a subset of IDSs, who have more resources (*e.g.*, more CPU and bandwidth) and better security measures as the SFCs. Alternatively we can use some dedicated hosts to be the SFCs. In the CDDHT system, we use some pre-defined rules to recognize SFCs, *e.g.*, we can use the first bit of the node ID, "1" for SFCs, and "0" for normal IDSs. In the CDDHT, the number of IDSs and especially router-based IDSs, which monitor a large IP space, determines the overall coverage of the Internet used for gleaming the presence of widespread attacks.

All DHT systems provide two basic interfaces: insertion (`put`) and retrieval (`get`). The interface for insertion, `put(key,object)`, causes the DHT to route the given `object` to the node with a node identifier closest to the `key`. The interface for retrieval, `object=get(key)`, causes the DHT to obtain the `object` from the node with a node identifier closest to the `key`.

In CDDHT, when an attack is launched, each IDS sensor attempts to locally detect the attack. Upon detection, it generates a symptom report that will be forwarded to the appropriate SFC with the `put(key,object)` operation. Thus, `object` is the attack symptom report, and `key` is the the routing ID which is generated from the report, and deterministically maps to a node on the DHT. We term the `key` as *disease key*. Similarly, for an attack with *a prior* disease key, every IDS in the CDDHT can query its prevalence with the `get(key)` operation. The node ID of an SFC is the concatenation of the SFC identifier (the first bit "1") plus the disease key, while the node ID of a normal IDS can be the concatenation of the normal IDS identifier (the first bit "0") plus some random hash (*e.g.*, SHA-1) of their IP addresses.

For instance, as shown in Figure 1, there is a attack detected by two IDSs, and both send alarms to the same SFC. The SFC will fuse alarms together to better characterize the attack. Finally the SFC can send the result back to the IDSs which have contributed to the detected attack scenario.

## IV. Disease Key Design

As events are generated locally at each IDS node, the attack symptoms must be reported and routed to one SFC

| Intrusion | ID | Characterization Field(s) | | | Original Length |
|---|---|---|---|---|---|
| DoS Attack | 0 | Victim IP (subnet) | | | 34 bits |
| Scans | 1 | 0 (for vertical & block scan) | Source IP address | | 35 bits |
| | | 1 (for horizontal & coordinated scan) | Scan port number | Source IP (for horizontal scan) | 51 bits |
| | | | | 0 (for coordinated scan) | |
| Viruses/Worms | 2 | 0 (for known virus/worm) | | Worm ID | 19 bits |
| | | 1 (for unknown virus/worm) | | Destination port number | 19 bits |

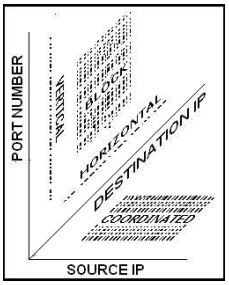


Fig. 2. Disease Key of CDDHT (left) and the visual representation for four types of scans (right)

which can then perform data fusion and inferences about such an attack. In this section, we design the routing keys to construct the CDDHT.

The challenge is: for a single intrusion, given the vast, diverse symptoms perceived from many heterogeneous IDSs around the world, how is the key effectively designed to fuse these events to a single SFC? For instance, while a worm is propagating, a router-based IDS may see much larger ranges of source/destination IPs along with a higher scan rate than that seen at a host-based IDS. For a distributed DoS attack, the network IDS (NIDS) for a zombie subnet; the NIDS for a victim subnet; and the NIDS for a third-party subnet have completely different views of attack and response traffic at either ingress, egress, or both.

Such key should also immune from attack pattern changes, like the number of zombies, the speed of attacks, the scope of attack, *etc.*. Furthermore, the design should strive to achieve load balancing while ensuring aggregation of related symptoms. All of these properties must be accomplished through key generation by each peer IDS in a decentralized and deterministic manner.

Given the requirements above, to ensure that related event symptoms will be routed to the same SFCs, we only use the intrinsic characteristics of each type of attacks, *i.e.*, the information that uniquely identifies related events, for the fields of the routing ID. Extraneous identifiers that may have clarified routing in some cases but confuse in others are thrown out. Since worms, DoS and port scans are the largest percentage of large scale attack on the Internet, we focus on these three categories.

The choice of characterization fields is based on an extensive survey of Internet intrusion literatures (see [19] for the full list of literature). The format of the disease key is shown in Figure 2. The first element of the routing ID is a two bit identifier (which is extensible) differentiating the three major types of intrusions we wish to better understand. We discuss each as follows.

*1) DoS attacks:* Ideally, the disease key for DoS attacks should properly distinguish and correlate millions of DoS attack events, but DoS attacks are very hard to characterize [20]. There may be one or many attack agents sending attack traffic, and these agents often have spoofed IP addresses, or may even be servers sending legitimate responses (*e.g.*, distributed reflection attacks [21]). Other than application-specific DoS attacks (*e.g.*, DNS request attack), DoS attacks do not have a fixed port number or protocol.

Thus the only invariant for a DoS attack is the victim, which is used in our design of the disease key. For application or host based attacks, the victim IP address is used as the key. For network attacks, which consume the bandwidth of a target network subnet, the key is the subnet (longest prefix) with all remaining bits set to 0. Note that the victim has to be recognized by each IDS, and the "victim" IP address can be source or destination depending on the situation. Take a TCP SYN flood attack for example. For the IDSs closest to the attack agent or the victim machine, the victim is the destination IP address. While for the third party IDSs which receives "backscatter" traffic [22], the victim is the source IP address. In this way, we can correlate the alerts from multiple sources to track the zombies, even when their IP addresses are spoofed.

*2) Port scan:* Scan is probably the most common and versatile type of intrusion. Based on source/dest IP and the port number involved in the scans, there are four well known types of scans: *horizontal scan, vertical scan, coordinated scan*, and *block scan* [23], [24] as designated in Figure 2.

Horizontal scans are the most common type of scan, and scans ports on IP addresses in some range of interest. We denote it as an identifier bit "1" and include the scan port number. The port number is unique because it reflects the vulnerability the virus/worm or attackers try to exploit. Unlike DoS attacks, the attacker needs to use a real source IP address, since she needs to see the result of the scan in order to know what ports are actually open [24], [23]. Thus, we use the source IP address as the last field of the scan disease key. We can find the amount of scans for each source IP. Moreover, with range queries described in Section 5-C we can aggregate some of the hirizontal scans to coordinated scans, and get a more comprehensive view of attacks. In addition, we also can aggregate solely by port number, to get the most popular scanning port.

Coordinated scans can be viewed as horizontal scans from multiple sources [23]. We use the same disease key as the horizontal scan except that the source IP address is set to zero because of its intrinsic variance in a coordinated attack.

A vertical scan is a scan of some or all ports on a single host, with the rationale that the attacker is interested in this particular host, and wishes to characterize its active services to find which exploits to attempt, or to find a suitable exploit via her network of contacts and resources [24]. We represent this scan with the identifier bit 0, and the source IP in the disease key. We do not include the port set because such a set is not stable and is hard to represent accurately within the small space in the disease key.

The fourth type of scan, a block scan, is a combination of horizontal and vertical scans over numerous services on numerous hosts [24]. This scan can be regarded as a vertical scan without a fixed destination. We use the same disease key as with vertical scans.

More complicated scans than these four are possible in

principal, but we have not seen much in practice, and for now we leave them as subsets of these major classifications.

*3) Viruses/worms:* Nowadays, viruses and worms are severe threats to the whole Internet [25], [26]. For existing ones, most IDSs use signature based approaches to detect them. Since their names can be unique to distinguish them [27]. Here, we use an SHA-1 hash of the names of viruses/worms as the disease key. Unknown viruses/worms can be detected through scanning, or approaches like SDC [28]. But given any virus/worm, the intrusion (destination) port number is normally unique, and we use that as the disease key.

These carefully chosen characteristics form the *heterogenous* dimensions for the disease key. Thus the keys have different length for different symptoms: viruses/worms may only need 19 bits, while the keys for horizontal scans can have up to 51 bits. For most DHT systems [8], [9], [10], given an $n$-node DHT, in an even node distribution, the effective key length is $\log n$. Then even a $10^6$ node DHT can only have 20-bit effective keys. We can increase the key length by a constant to accommodate all keys, but the bigger the constant, the more virtual nodes one real DHT node has to cover. To support the aggregate queries discussed in Section 5-C, we cannot hash the key randomly to distribute the load. A large key space can easily lead to unbalanced load distribution. In following sections,we will address these problems.

## V. Features of CDDHT

In this section, we discuss load balancing, querying, and attack resilience of CDDHT.

### A. Load Balancing

Internet attacks are increasing in frequency, severity and sophistication. When a global attack occurs, many IDSs will detect it and send alerts, which may easily overload some SFC(s). Furthermore, some characterization fields in the disease key have strongly uneven popularity distribution, like port numbers 80 and 135 for scans. We use as much essential characteristics of attacks as possible for the disease key to spread the alert loads among the SFCs. In addition, we design the following techniques.

*1) Proactive Balancing Scheme with Stable Hot Spots:* Since the number of SFCs is much less than the size of disease key space, for the characterization fields with hot spots, we want to design the disease key so that each hot spot gets at least one dedicated SFC because we have to somehow fuse all related alerts to one SFC to obtain a global view. There are more mechanisms to further alleviate the load of SFC as we will describe later.

However, this scheme is only useful when hot spots are stable. Since port scanning is the most common type of Internet event alarm, we study the popularity distribution and its stability of the characterization fields as an example.

While previous work suggests the popularity of scan port number and source IP address follows heavy tail distribution, *i.e.*, a small number of entities are responsible for a large number of scans [23], it remains unclear how stable these hot spots are. To this end, we study the distribution stability of the popular scanned ports, and those of top source IP subnets, based on the stability of the scan record coverage from

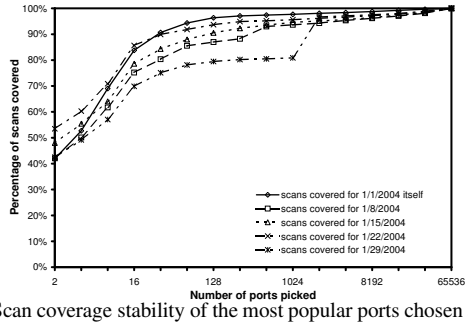


Fig. 3. Scan coverage stability of the most popular ports chosen at 1/1/2004

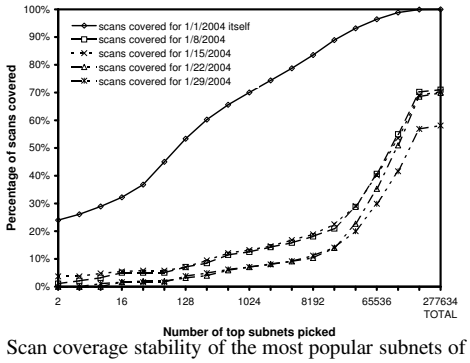


Fig. 4. Scan coverage stability of the most popular subnets of 1/1/2004

previous popular entities, *e.g.*, how many scan records from today are covered by the top 10 ports of 10 days before. Our analysis is based on one-month (January 2004) of intrusion logs from DShield [29] consisting of over 600 million scan records from over 2200 providers around the world.

As shown in Figure 3, given a sufficient training time period (one day in our example) the most popular ports will cover a large percentage of scans in the future. For instance, 64 popular ports remain to cover 85% of the scan records for up to three weeks. We further study the stability of the class C (/24) subnets of source IP as shown in Figure 4. Unlike port numbers, the popular source IP subnets of scans keep changing. That is, there is no steady blacklist. See [19] for more results and analysis.

Leveraging stable ports for load balancing depends on the type of underlying DHT. In Chord, we use 7 bits for the port number with 128 buckets. We map each of the 64 popular ports into one unique bucket, and map the other ports randomly into the remaining buckets. This effectively improves the load balancing especially when the SFC nodes are relatively few and thus sparse in the node ID space.

*2) Node Migration and Virtual Node:* Since each SFC $s$ is on the DHT routing paths to many other SFCs, it can continuously collect the load information of other SFCs (when there is an alert sending through $s$) as well as the load of its neighbors. If $s$ notices some SFC overloaded, it can migrate to that region for load sharing under the following conditions: 1) the load of $s$ has been below a certain threshold for a certain period; 2) $s$ can dump its load to its neighbors without overloading them. $S$ can leave, and then rejoin the system using migration.

Alternatively, an SFC with low load can spawn a virtual node on itself and have the virtual node join the CDDHT through the "load-aware bootstraping" described below.

*3) Load-aware Node Bootstraping:* Load-aware-bootstraping happens when we add an IDS node or any other

node as an SFC. The new SFC first finds a bootstrap node $b$ in the CDDHT with approaches described in [7], [10], and queries $b$ for the SFC node with the heaviest load from the locally maintained load table of $b$. It will then join and split the load with the overloaded SFC.

*4) IDS Alarm Rate Limiting:* A router IDS may generate many alarms for one specific attack. We can set some alarm rates for the IDS so that it executes some temporal aggregation to combine several alarms within one time interval to a single alarm.

*5) Aggregation Tree:* When a global-scale attack occurs, the number of alerts sent to an SFC may be overwhelming because all these alerts may have the same unique disease key. To address this issue, the IDSs and SFCs on the routing path can fuse the alerts locally before passing it on when it observes a big burst of alerts to the same destination. This will be only triggered by very large burst of alerts.

Actually, by using this technique, the alarms will follow an aggregation tree towards the final destination SFC. Each node in the tree has $O(\log n)$ neighbors, where $n$ is the total number of nodes in the CDDHT system. Thus for any IDS/SFC, the amount of alarms received per aggregation time interval (*e.g.*, 10 seconds) for each attack is bounded by $O(\log n)$.

Next, we will discuss some open questions.

### B. Attack Resilience

The DIDS systems themselves are very likely to be targets of attacks. Here we introduce several mechanisms to make the CDDHT system more resilient to attacks.

*1) Authenticity of Alarms:* To enforce the authenticity of alarms, when each IDS joins the CDDHT, we will validate its request by querying *whois* servers and checking the BGP tables from multiple vantage points. The purpose is to check whether the origin of the IDS is correct, and whether the range of the IP addresses it covers is the same as it claims. Furthermore, we may contact the administrator of that domain to further validate the authenticity of the request. With such validation, when an IDS reports attacks, we can check whether the involved IP addresses belong to the domain of the IDS.

Moreover, when every node (including both SFC and IDS) joins CDDHT, we will generate their public-private key pair based on RSA [30] or DSA [31] algorithms. The trusted certification authority (CA) of the CDDHT will generate the certificate for the public key of each node. When an IDS $d$ sends a symptom report, it appends its digital signature and its public key certificate to the symptom report. The receiver SFC will check $d$'s public key with the CA, and then uses it to verify the signature of the report. When a IDS/SFC on the path needs to do aggregation when it receives bursty alarms that need to be forwarded, it will sign such changes and append its own public key certificate to the aggregated report.

With such mechanisms, unless an IDS or SFC is compromised, we can always detect the bogus alarms.

*2) Dealing with Compromised Nodes:* Sometimes an IDS can be compromised by attackers. They can use these nodes to send bogus symptom reports or make up aggregated reports for the alarms routed through. To minimize their influence to the overall CDDHT system, we evaluate the severity of one global attack, not only by the total number of alarms, but also by the range of IP addresses influenced by the attack, and the number of IDSs reporting that attack. For aggregated reports, the SFC which is responsible for the attack will randomly check some IDSs which have attack reports aggregated. Each IDS/SFC will keep the original alarm reports received (from which the aggregation reports are generated) for certain period of time. Thus once cheating is detected, we can trace back to find the compromised node. Then we can update the routing table of all its neighbors to remove it from CDDHT. This is much easier to fix than that of the hierarchical based approach.

There is an more severe problem when a SFC is compromised. we cannot query any attacks of which information is collected by that SFC. But usually a SFC is much better secured than a IDS. And in contrast to the hierarchical approaches, each SFC is only responsible for a small set of attacks, the remaining CDDHT system still function well.

| scan type | Vertical | Horizontal | Block | Coordinated |
|---|---|---|---|---|
| # of scans | 3364 | 8486 | 22 | 25711 |

TABLE I
NUMBER OF SCANS AND THEIR TYPES IN THE ATTACKS

### C. Querying on CDDHT

CDDHT can support two types of queries for the fusion results: with a given disease key or *aggregate queries*. The former is easy: simply calling `get(disease key)`. The latter can answer queries like "show me all the horizontal and coordinated scans with port number $x$". This is particularly useful to detect a complicated worm propagation where some nodes perceive it as a horizontal scan while other nodes perceive it as a coordinated scan. The query is issued to the SFC which is responsible for the coordinated scan. That SFC will serve as a collector, and query all other nodes in the "zone" which have the same node ID as the collector except the last few bits for source IP addresses. We also plan to leverage "range queries" over DHT [32] for more complicated query requests.

## VI. EVALUATION

We implemented a preliminary CDDHT system based on the Chord [8] simulator with hot scan port load balancing, and simulated the scan alert fusion with DShield firewall logs. The data only contain scan records, which arguably represent the largest number of intrusions in the current Internet. It is our future work to collect data on other attacks for a more comprehensive evaluation of the system.

We have evaluated our work using daily DShield data, and have found that the results from each day are similar. For this reason, we will show the results for that of January 2nd, 2004. It contains over 25 million scan logs from nearly 1417 providers. These scan logs are classified into the events as in Table I. See [19] for the details on simulation set up.

We then used these classified scan events to generate the disease keys based on the design in Section 4. Each of the disease keys is simulated to route on the Chord DHT [8] of $n$ nodes where $n$ is the number of unique providers (IDS nodes) in the DShield data. For simplicity, we assume any DHT node (*i.e.*, IDS node) can be an SFC.

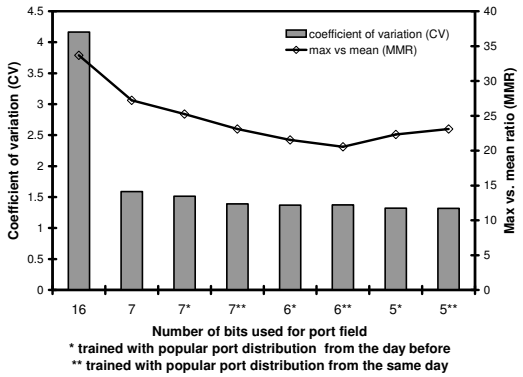


Fig. 5. Statistical Distribution of reports across the DHT nodes

The *metrics* to evaluate our work include the effectiveness of fusion, and the variation of load among the SFCs. The effectiveness of fusion is the percentage of related alarms (defined by the disease key) fused in the corresponding SFC. Because Chord has deterministic routing, if given the same disease key, it always routes to the same SFC node. Therefore, we can get 100% effectiveness of fusion in our CDDHT system. For each SFC, the load is denoted by the number of symptom reports received. The load variation is measured in terms of the *coefficient of variation* (CV)

$$CV(x) = \frac{standard\ deviation(x)}{mean(x)} \qquad (1)$$

and the *maximum vs.mean ratio* (MMR), as shown in Figure 5. The CV is a standard metric for measuring inequality of $x$, while the MMR checks if there is any single node whose load is significantly higher than the average load.

The stability study in Section 5-A.1 suggests we can map top 64 most popular ports to the first 6 bits of port evenly. In practice, we found that it can significantly improve the load balacing: it reduces the CV by more than 60%, and reduce the MMR by about 40%, compared with using the entire 16 bit port number, as shown in Figure 5. There are still certain hot spots because some scan events are much more popular than the rest, as indicated in the MMR ratio. The aggregation tree will further solve the problem by fusing the requests along the path and bound the load of SFCs to $O(\log n)$. We will evaluate its effectiveness as part of our future work.

Since the popular ports remain very stable, there is little difference between training the hashing function with the history or with the current dataset (like the oracle case). We also tried training the hashing function with older datasets (up to a month old), which gave similar results. See [19] for more detailed results. It is our future work to evaluate other load balancing techniques in Section 5-A.

## VII. Conclusion

In this paper, we propose to build a distributed IDS based on *distributed hash table (DHT)*. We embed the intrusion symptoms into the DHT dimensions so that alarms related to the same intrusion will be routed to the same SFC with good load balancing. In future work, we will focus on evaluating the load balancing schemes we designed and the open questions on querying and attack-resilience of CDDHT.

## References


[1] E. Mars et al., "Email defense industry statistics," `http://www.mxlogic.com/PDFs/IndustryStats.2.28.04.pdf`.

[2] S. R. Snapp et al., "DIDS (distributed intrusion detection system) - motivation, architecture and an early prototype," in *the 14th National Computer Security Conference*, 1991.

[3] P. A. Porras and P. G. Neumann, "Emerald: Event monitoring enabling responses to anomalous live disturbances," in *the 20th NIS Security Conference*, 1997.

[4] R. Janakiraman et al., "Indra: A peer-to-peer approach to network intrusion detection and prevention," in *IEEE WETICE Workshop on Enterprise Security*, 2003.

[5] C. Papadopoulos et al., "Coordinated suppression of simultaneous attacks," in *DARPA Information Survivability Conference and Exposition (DISCEX)*, 2003.

[6] V. Yegneswaran et al., "global intrusion detecction in the DOMINO overlay system," in *ndss*, 2004.

[7] S. Ratnasamy et al., "A scalable content-addressable network," in *ACM SIGCOMM*, 2001.

[8] I. Stoica et al., "Chord: A scalable peer-to-peer lookup service for Internet applications," in *ACM SIGCOMM*, 2001.

[9] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *ACM/USENIX Middleware*, 2001.

[10] B. Y. Zhao et al., "Tapestry: A resilient global-scale overlay for service deployment," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, 2004.

[11] B. Cohen, "Incentives build robustness in bittorrent," `http://bitconjurer.org/BitTorrent/bittorrentecon.pdf`, May 2003.

[12] C. Tang et al., "Peer-to-peer information retrieval using self-organizing semantic overlay networks," in *the ACM SIGCOMM*, 2003.

[13] M. Castro et al., "Splitstream: High-bandwidth multicast in a cooperative environment," in *ACM SOSP*, 2003.

[14] S. Ratnasamy et al., "Application-level multicast using content-addressable networks," in *NGC*, 2001.

[15] N. Weaver et al., "Large scale malicious code: A research agenda," Tech. Rep., DARPA Sponsored Report, 2003.

[16] Y. Chen et al., "Quantifying network denial of service: A location service case study," in *Proc. of International Conf on Info. and Comm. Security (ICICS)*, 2001.

[17] R. Huebsch et al., "The architecture of PIER: an internet-scale query processor," in *Conference on Innovative Data Systems Research (CIDR)*, 2005.

[18] B. N. Chun et al., "Netbait: a distributed worm detection service," Tech. Rep. IRB-TR-03-033, Intel Research Berkeley, 2003.

[19] Y. Chen, A. Beach, and J. Skicewicz, "Cyber disease monitoring with distributed hash tables: A global peer-to-peer intrusion detection system," Tech. Rep. Northwestern Technical Report, NWU-CS-04-040, 2004.

[20] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and defense mechanisms," *ACM CCR*, vol. 34, 2004.

[21] S. Gibson, "Drdos: Distributed reflection denial of service," 2002, `http://grc.com/dos/drdos.htm`.

[22] D. Moore, G. M. Voelker, and S. Savage, "Inferring Internet denial-of-service activity," in *USENIX Security Symposium*, 2001.

[23] V. Yegneswaran, P. Barford, and J. Ullrich, "Internet intrusions: Global characteristics and prevalence," in *ACM SIGMETRICS*, 2003.

[24] S. Staniford, J. A. Hoagland, and J. M. McAlerney, "Practical automated detection of stealthy portscans," *Journal of Computer Security*, vol. 10, no. 1-2, 2002.

[25] N. Weaver et al., "A taxonomy of computer worms," in *the First Workshop on Rapid Malcode (WORM)*, 2003.

[26] D. M. Kienzle and M. C. Elder, "Recent worms: A survey and trends," in *the First Workshop on Rapid Malcode (WORM)*, 2003.

[27] "Virus names could be standardized," 2004, Computer Business Review Online.

[28] D. Dagon et al., "Honeystat: Local worm detection using honeypots," in *RAID*, 2004.

[29] SANS Institute, "Dshield.org: Distributed intrusion detection system," `http://www.dshield.org/`.

[30] M. E. Locasto et al., "Collaborative distributed intrusion detection," Tech. Rep. CUCS-012-04, Columbia University Computer Science Department, 2004.

[31] M. E. Locasto et al., "Collaborative distributed intrusion detection," Tech. Rep. CUCS-012-04, Columbia University Computer Science Department, 2004.

[32] S. Shenker S. Ratnasamy, J. M. Hellerstein, "Range queries over DHTs," Tech. Rep. Intel Research Technical Report, IRB-TR-03-011, 2003.