DHT-based security infrastructure for trusted internet and grid computing

Kai Hwang*, Min Cai, Yu-Kwong Kwok,
Shanshan Song, Yu Chen and Ying Chen

University of Southern California, Los Angeles, CA 90086 USA
E-mail: kaihwang@usc.edu E-mail: mincai@usc.edu
E-mail: ykwok@hku.hk E-mail: shanshan.song@oracle.com
E-mail: cheny@usc.edu E-mail: chen2@usc.edu
Website: http://gridsec.usc.edu
*Corresponding author

Abstract: We designed a distributed security infrastructure with self-defence capabilities to secure networked resources in Grids and internet applications. This paper reports new developments in fuzzy trust management, game-theoretic Grid models, security-binding methodology, as well as new Grid performance metrics, defence architecture and mechanisms against intrusions, worms, and low-rate pulsing Distributed Denial of Service (DDoS) attacks. The design is based on a novel Distributed Hash Table (DHT) for security enforcement among Grid sites scattered over the internet.

Keywords: security infrastructure; internet computing; grid computing; intrusion detection; worm containment; DDoS defence.

Reference to this paper should be made as follows: Hwang, K., Kwok, Y-K., Song, S., Cai, M., Chen, Y. and Chen, Y. (xxxx) 'DHT-based security infrastructure for trusted internet and grid computing', *Int. J. Critical Infrastructures*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Kai Hwang is a Professor of Electrical Engineering and Computer Science at the University of Southern California. He specialises in computer architecture, parallel processing, internet security, and distributed computing systems.

Min Cai is a PhD student in Computer Science at USC. His research interests include distributed worm-detection and DDoS-defence systems, peer-to-peer, grid computing, and semantic web technologies.

Yu-Kwong Kwok is an Associate Professor of Electrical and Electronic Engineering at the University of Hong Kong. His research interests include grid and mobile computing, wireless communications, and network protocols.

Shanshan Song is a Member of Technical Staff at Oracle Corporation. Her research interests include trust management in grid and P2P systems, scheduling algorithms for computational grids, and high availability in database systems.

Yu Chen is a PhD student in Electrical Engineering at USC. His research interests include internet security, DDoS attack detection and defence, internet traffic analysis, and distributed security infrastructure.

Ying Chen is a PhD student in Electrical Engineering at USC. Her research interests include intrusion detection, alert correlation, and security systems for Grid computing.

1 Introduction

Over the past few years, a new breed of network worms like the CodeRed, Nimda, SQL Slammer, and love-bug have launched widespread attacks on the Whitehouse, CNN, Hotmail, Yahoo, Amazon, and eBay, etc. These incidents created worm epidemic (Moore et al., 2003) by which many internet routers and user machines were pulled down in a short time period. These attacks had caused billions of dollars loss in business, government, and services. Open resource sites in information or computational Grids could well be the next wave of targets. Now more than ever, we need to provide a secure Grid computing environment over the omni-present internet (Hwang et al., 2005b).

Network-centric computing systems manifest as Grids, intranets, clusters, Peer-to-Peer (P2P) systems, etc. Malicious intrusions to these systems may destroy valuable hosts, network, and storage resources. Network anomalies cause even more damages. Internet anomalies found in routers, gateways, and distributed hosts may hinder the acceptance of Grids, clusters, and public-resource networks (Nagaratnam et al., 2002). Our work is meant to remove this barrier from Grid insecurity. This paper reports our latest research findings in advancing security binding and building self-defence systems tailored for protecting Grid resource sites.

- architectural design of trusted Grid security infrastructure in Section 2 (Hwang et al., 2005b)
- the CAIDS Distributed IDS (DIDS) and alert correlation system in Section 3 (Hwang et al., 2005a, 2005c)
- DHT-based overlay networks for collaborative worm containment in Section 4 (Cai et al., 2005)
- filtering of shrew, low-rate, pulsing DDoS attacks in Section 5 (Chen et al., 2005 and Chen and Hwang, 2006)
- security binding for trusted Grid job scheduling in Section 6 (Song et al., 2005a, 2005b, 2006)
- game theoretic modelling of selfish or non-cooperative Grids in Section 7 (Kwok et al., 2004, 2005).

1.1 Security Demands (SDs) in grid computing

The internet has become so deeply integrated into our daily lives that any disruption of services can lead to great monetary losses. Indeed, trusted e-commerce activities are now routinely carried out over the internet. As a major communication medium, the internet must now provide confidentiality to correspondences in the form of e-mail messages, FTP transferred files, etc. Recently, P2P file sharing has become one of the dominant forms of internet traffic and thus, must also be protected. In summary, from the intruder's point of view, there are simply many vulnerable targets to attack or take advantage of.

On the defence side, a plethora of tools have been developed to detect system intrusions and network traffic anomalies: firewalls, packet filters, Virtual Private Network (VPN) gateways, traffic monitors, security overlays, Public Key Infrastructure (PKI) services, to name only a few. Nevertheless, it seems that the arsenal of the intruders can always make faster advancements compared with that of the defence. Thus, a new breed of 'evolutionary' type of defence kits has been suggested: self-defence middleware based on network overlays to combat viruses, worms, etc.

1.2 Three generations of defence technology

We have witnessed three generations of network defence technologies. In the first generation, tools were designed for preventing or avoiding intrusions. Thus, these tools usually manifested as access control policies or tokens, mathematically sound cryptographic systems, etc. However, as mentioned above, the intruder can always penetrate a 'secure' system because there is always a weakest link in the security provisioning process.

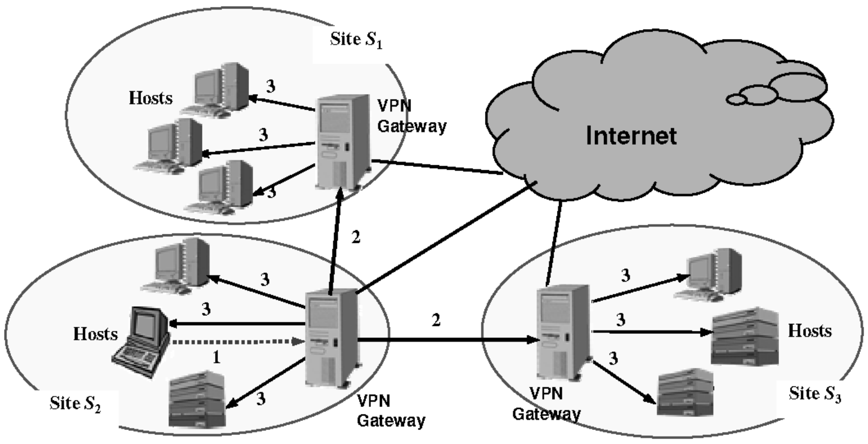
For example, human beings, always in the loop in the first generation design, were a major weakest link (e.g., easily leaked or guessed passwords). This motivated the second generation techniques, which were designed based on the assumption that intrusions are bound to occur. Thus, the goal was not to avoid intrusions but to efficiently and accurately detect intrusions so that appropriate and timely remedial actions can be taken. These second generation techniques include: firewalls, Intrusion Detection Systems (IDSs), PKI services, reputation systems (Kamvar et al., 2003), etc.

Undeniably, this is a consequence exactly targeted by the intruders. Thus, the third generation techniques are recently proposed to provide more intelligent responses to intrusions. Specifically, the third-generation techniques can make the attacked system tolerate the intrusions so as to continue providing services. This is relatively new form of network defence technology that is still intensively under research. The fortified Grid infrastructure and new models and metrics apply to many security-sensitive Grid applications including digital government, electronic commerce, and distributed supercomputing, etc.

2 GridSec security architecture

Figure 1 shows the distributed security enforcement scheme in our GridSec project. The security policy is distributed to all cluster nodes under the coordination of VPN gateways. A *gateway firewall* is installed at the front-end, playing the role of screening between the cluster network and the external world of the internet. Here, individual nodes act as enforcers of the central security policy. All the cluster nodes form a single security domain.

Figure 1 GridSec: a network security research project at USC, where the intrusion is detected at Step 1, reported to all VPN gateways at Step 2, and responsive counter-measures taken at the Step 3



A *micro-firewall* is built on each host by OS kernel extensions aided by some intrusion detection and policy updated mechanisms. The distribution of security functions removes some of the constraints associated with conventional gateway firewall. This distributed architecture is meant to cope with insider attacks. In addition, the cluster is supported by a policy update mechanism that responds to new attacks dynamically.

The functions of intrusion prevention, detection, and responses are now distributed to two levels of security control, namely the VPN gateway (security manager) and micro-firewalls at local hosts. Collectively, they carry out the security enforcement and update the security policy dynamically. The main purpose is to repel intrusions from all sources in real-time.

In Figure 1, we show three steps of defence responses to attacks detected at any host machine in the Grid system.

Step 1: Intrusion detected by a local micro-firewall.

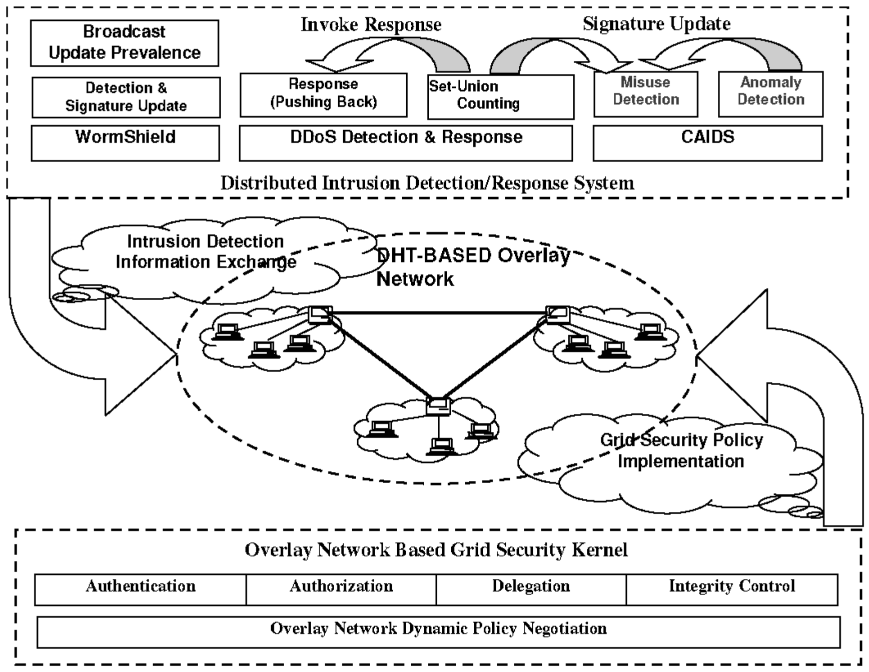
Step 2: All security managers alerted with the intrusion

Step 3: Security managers broadcast response command to all hosts in their jurisdiction so that updated central policy is enforced locally (Hwang et al., 2005a; Nagaratnam et al., 2002).

2.1 NetShield defence system

Our GridSec security architecture is designed to be a wide-area defence system that enables high degree of trust among the Grid sites in collaborative computing over the internet. As illustrated in Figure 2, GridSec adopts DHT-based overlay architecture as its backbone. As a virtual communication structure lay logically on top of physical networks, our overlay network maintains a robust virtual inter-networking topology.

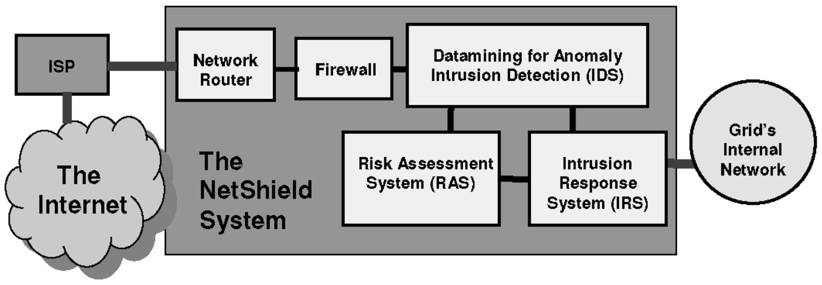
Figure 2 DHT-based security infrastructure against network worms and DDoS attacks


Broadcast Update Prevalence
Invoke Response
Signature Update
Detection & Signature Update
Response (Pushing Back)
Set-Union Counting
Misuse Detection
Anomaly Detection
WormShield
DDoS Detection & Response
CAIDS
Distributed Intrusion Detection/Response System
Intrusion Detection Information Exchange
DHT-BASED Overlay Network
Grid Security Policy Implementation
Overlay Network Based Grid Security Kernel
Authentication
Authorization
Delegation
Integrity Control
Overlay Network Dynamic Policy Negotiation

Through this topology, trusted direct application level functionalities facilitates inter-site policy negotiation and management functions such as authentication, authorisation, delegation, policy exchange, malicious node control, job scheduling, resource discovery and management, etc. (Cai et al., 2005). The GridSec system functions as a Cooperative Anomaly and Intrusion Detection System (CAIDS) (Hwang et al., 2005a). As shown in Figure 2, currently available functional blocks include the WormShield (Cai et al., 2005), CAIDS (Hwang et al., 2005a) and DDoS pushback scheme (Chen et al., 2005).

Figure 3 shows the network setting, where the NetShield system is deployed. The NetShield core interacts with the gateway firewall, network router, and Internet Service Provider (ISP) at the upstream. Smart filtering is done at the router stage to drop packets from suspicious or verified IP sources. The IDS, Risk Assessment System (RAS), and Intrusion Response System (IRS) work together to perform automated intrusion and anomaly detection, risk assessment and responses.

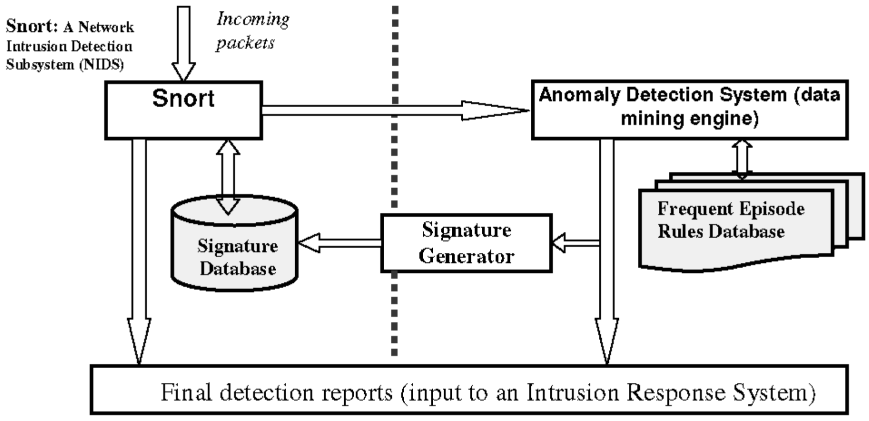
Figure 3 The NetShield system works with the firewall, network router, and Internet Service Provider (ISP) against network attacks on Grid's internal networks


ISP
Network Router
Firewall
Datamining for Anomaly Intrusion Detection (IDS)
The Internet
The NetShield System
Risk Assessment System (RAS)
Intrusion Response System (IRS)
Grid's Internal Network

3 Collaborative alert correlation

The CAIDS architecture is conceptually shown in Figure 4 (Hwang et al., 2005c). We integrate the IDS and Anomaly Detection System (ADS) by making them supporting each other. The signature-matching IDS (Snort) is connected in cascade with a custom-designed ADS. These two subsystems join hand to cover all traffic events initiated by both legitimate and malicious users.

Figure 4 Basic idea of a Cooperative Anomaly and Intrusion Detection System (CAIDS), built with an Intrusion Detection System (IDS) in cascade with an Anomaly Detection System (ADS) and a feedback loop for automated signature update


Snort: A Network Intrusion Detection Subsystem (NIDS)
Incoming packets
Snort
Anomaly Detection System (data mining engine)
Frequent Episode Rules Database
Signature Database
Signature Generator
Final detection reports (input to an Intrusion Response System)

Unknown, burst, or multi-connection attacks are detectable by the ADS. These include the attacks Apache2, Guess-telnet, Neptune, UDP-Storm, Dictionary, and Smurf, which cannot be detected by the Snort IDS. A novel *signature generator* is proposed to bridge the two subsystems. Our new approach performs much better than the IDS or the ADS alone.

3.1 Collaborative alert correlation

The CAIDS is deployed at Grid sites to form a DIDS supported by alert correlation sensors. These sensors are scattered around the Grids. They generate a large amount of low-level alerts. These alerts are transmitted to the alert correlation modules to generate high-level intrusion reports, which can provide a broader detection coverage and lower false alarm rate than the localised alerts generated by single IDS. Figure 5 shows the alert operations performed by various modules locally and globally (Cuppens and Miege, 2002).

In Figure 6, we plot the ROC curves corresponding to four attack classes. The detection rate grows quickly to its peak value with a small false alarm rate. The detection rate gets saturated after large false alarms. For example, to achieve a detection rate above 75% of DoS attacks, we have to tolerate 5% or more false alarms. CAIDS has the second best performance against R2L attacks, and has about the same performance against probe attacks. CAIDS has the lowest 25% detection rate against U2R attacks at 10% false alarms due to U2R attacks' stealth nature.

Figure 5 Alert operations performed in local Grid sites and correlated globally

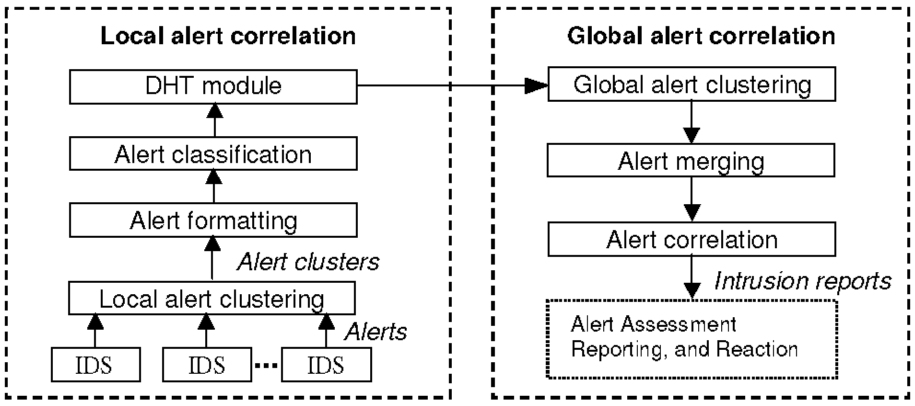
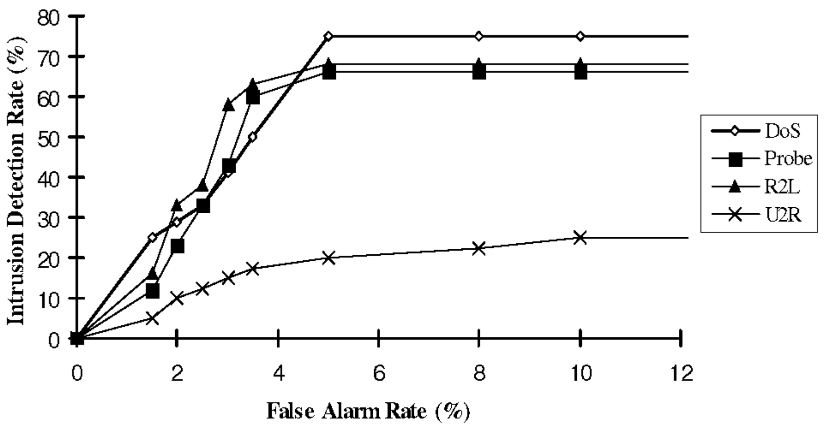


Figure 6 ROC plot of intrusion detection rate vs. false alarm rate in using the CAIDS



4 DHT-based worm containment

Large-scale worm outbreak is one of the major security threats to today's internet and tomorrow's Grids. A network worm is a self-propagating program that exploits the vulnerability of widely-deployed homogeneous software and the unrestricted connectivity of the internet. With different payloads, they can be employed to access or damage sensitive data, host spam-relays or HTML proxies, launch DDoS attacks, or open a backdoor for remote access.

Recent incidents in worm epidemic clearly demonstrate that they are realistic and pressing threats to the internet's security. The CodeRed worm infected about 359,000 web servers in 14 hours in 2001 and its consequent recovering cost is estimated at $2.6 billion. Slammer was the fastest worm in the wild in that it achieved its maximum internet-wide scanning rate (55 million scans per second) in a few minutes. Recent studies by Moore et al. (2003) suggested that containing worms effectively from widespread outbreak is quite challenge since the reaction time of containment is only a few minutes.

4.1 Worm containment strategies

In general, the following strategies are often adopted to prevent and contain worms from infecting millions of vulnerable hosts or paralysing many of the hundreds of thousands of edge networks in the internet (Singh et al., 2004).

- *reduce vulnerability*: preventing worms by upgrading software quality as well as reducing window of vulnerabilities
- *scan detection*: filtering traffic destined at detected ports where worms are scanning and spreading
- *hygiene enforcement*: discovering infected hosts and keep susceptible hosts off network
- *signature inference*: detecting payload content sub strings to generate and disseminate signatures automatically and throttle connection rate to slow down spread.

4.2 DHT-based WormShield architecture

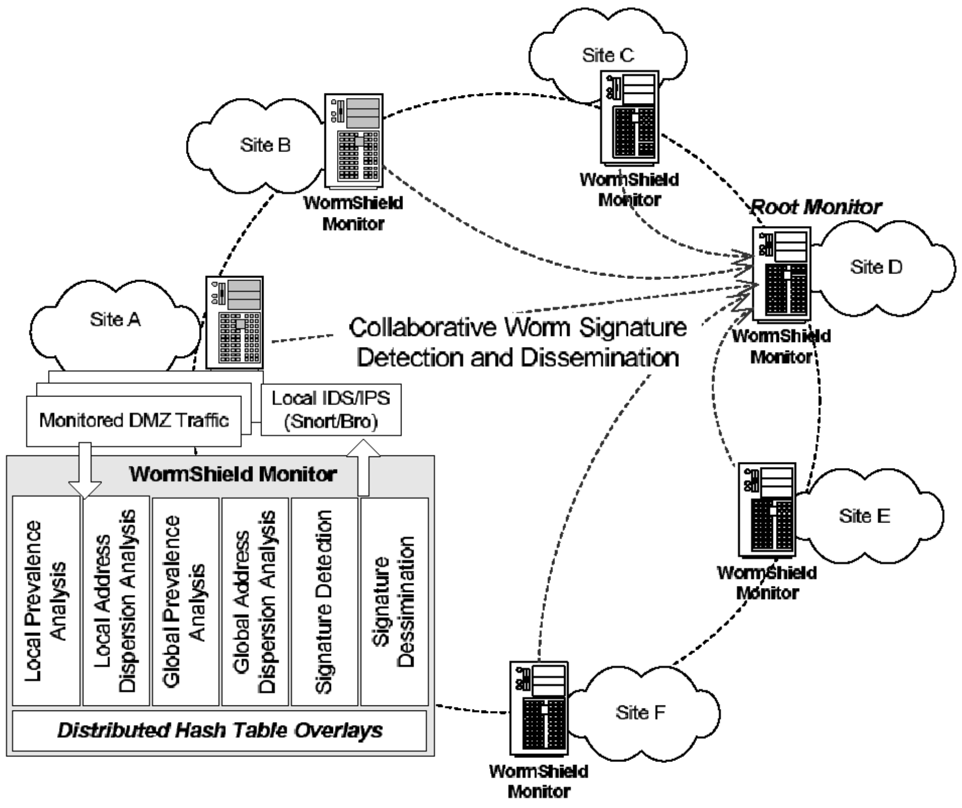
To facilitate high-speed intrusion detection and alert information exchanges, we propose a DHT (Kodialam et al., 2004; Stoica et al., 2001) based WormShield architecture that employs fast and scalable overlay networks over different edge networks. In WormShield, DHT-based overlays are used for monitoring, detection, and containment of worm spreading as well as for defence against related DDoS flooding attacks. This security overlay approach emphasises on robust and scalable inter-domain solutions since network-wide worms are intrinsically difficult to prevent and defend at individual hosts, routers or even edge networks.

4.3 Collaborative worm containment

We build a scalable DHT overlay to protect a large number of autonomous domains in edge networks. Our WormShield system (Cai et al., 2005) consists of a set of geographically distributed monitors that self-organise into a structured P2P overlay network (Figure 7). In WormShield, each monitor analyses all network traffic passing through it and partitions packet payloads into small blocks. It then calculates locally the prevalence and address dispersion of each content block. A content block's prevalence is the number of its occurrences in network traffic for a given time period. Its address dispersion is the number of distinct source and destination addresses corresponding to that block.

Once the local prevalence and address dispersion of a content block are both greater than thresholds, the local monitor will send its information to a *root monitor* for global aggregation. In WormShield, The root monitor of a content block is automatically selected using the same consistent hashing as Chord (Stoica et al., 2001). Therefore, different content blocks will be assigned to different root monitors in a load-balanced manner. The root monitor then computes the global prevalence and address dispersion of a suspicious worm signature by aggregating its information from all other monitors.

Figure 7 The WormShield architecture deployed at six sites for collaborative worm monitoring, detection, and containment


Site C
WormShield Monitor
Site B
WormShield Monitor
Root Monitor
Site D
WormShield Monitor
Site A
Collaborative Worm Signature Detection and Dissemination
Local IDS/IPS (Snort/Bro)
Monitored DMZ Traffic
WormShield Monitor
Local Prevalence Analysis
Local Address Dispersion Analysis
Global Prevalence Analysis
Global Address Dispersion Analysis
Signature Detection
Signature Dessimination
Distributed Hash Table Overlays
Site E
WormShield Monitor
Site F
WormShield Monitor

If both the global prevalence and address dispersion exceed predefined thresholds, the content block will be identified as a potential worm signature. Then, the root monitor will construct an efficient broadcast tree on top of the WormShield overlay network and disseminate the signature to all other monitors. Once receive the detected worm signatures, each monitor could automatically import them into its local IDS, such as Snort or Bro.

4.4 Simulated containment results

We simulated the CodeRed-like worm propagation over a large-scale internet configuration with 105,246 edge networks and 338,652 vulnerable hosts. In this simulated network, there are 61,216 vulnerable edge networks in which there is at least one vulnerable host. We compare the signature detection speeds of isolated and collaborative monitors when different percentages of edge networks are monitored.

We measure the signature detection speed by using the number of vulnerable hosts infected at the signature detection time. Table 1 shows the number of infected vulnerable hosts when 0.1%, 1%, 10% and 50% vulnerable edge networks are monitored.

Table 1 Comparison of using isolated and collaborative monitors for containing CodeRed worm attacks

		Number of infected hosts		
Percentage of deployment	*Local threshold*	*Average of isolated monitors*	*Best of isolated monitors*	*Collaborative monitors*
0.1	1,000	253,312	40,638	10,837
	10,000	316,978	237,734	77,890
1	1,000	223,510	35,220	4,064
	10,000	316978	222,156	8,128
10	1,000	232,993	35,220	4,064
	10,000	316,978	222,156	4,064
50	1,000	232,993	35,220	4,064
	10,000	316,978	222,156	4,064

For the cases of isolated monitors that are independent to each other, the signature detection speed of an average monitor and that of the best monitor do not improve significantly with increasing number of monitors. By contrast, for WormShield monitors, the infected hosts at detection time decreased from 77,890 to 8,128 when the number of monitored edge networks increases from 61 to 612 and the local threshold is 10,000. Other interesting worm control schemes include the Earlybird system (Singh et al., 2004) and the Autograph system (Kim and Karp, 2004).

5 Spectral shrew DDoS defence

Recently, a variant category of DDoS attack has been identified by on the Internet2 Abilene backbone. This periodic pulsing attack exploits the transients of a system's dynamic behaviour. These low-rate pulsing DDoS attacks introduce system inefficiencies that tremendously reduce the system capacity or service quality.

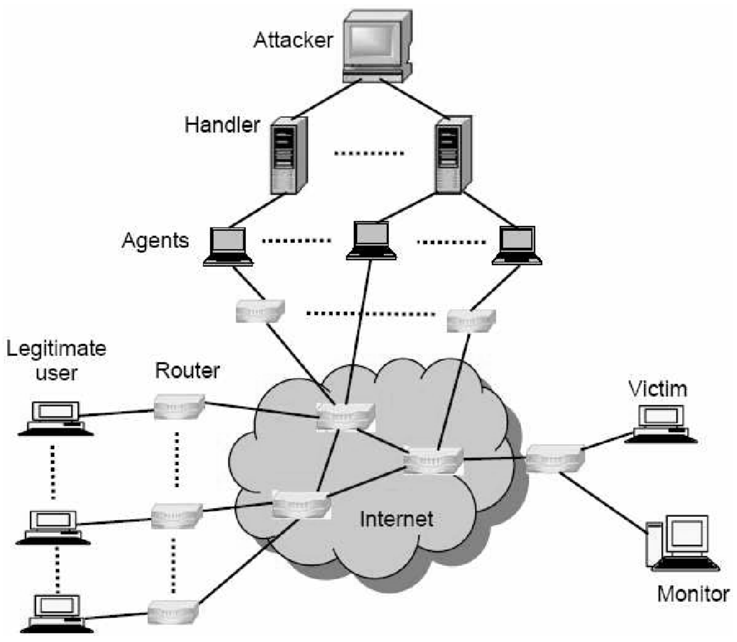
In the literature, this kind of DDoS attack is called shrew attack, pulsing DoS attacks, or the Reduction of Quality (RoQ) attack. Comparing to traditional DDoS flooding attacks, shrew attacks are even harder to detect. The shrew attacks can damage the victim for a long time without being detected.

5.1 Filtering of shrew DDoS attacks

We report a new approach to cope with shrew DDoS attacks. By converting the sample series into the frequency domain using Discrete Fourier Transform (DFT), we find that traffic containing shrew attacks has more energy in the low-frequency band than legitimate traffic does.

A shrew-filtering algorithm is thus developed to identify malicious flows. The performance results of the shrew DDoS defence scheme are based on NS-2 simulation experiments. As shown in Figure 8, attacks could be launched in a distributed manner. The distributed attack sources could decrease their average traffic either by reducing peak rates or increasing attack periods.

Figure 8 Shrew DDoS attack scenario and the simulation setting of attacking experiments



Although it is very challenging to detect and response to the low-rate attacks using defence measures developed against DDoS attacks, the periodicity itself provides a clue for new defence mechanism. We take the number of packet arrive as the signal and sample it every 1 ms, Nyquist sampling theorem indicates that the highest frequency of our analysis is 500 Hz. We sample the arriving packets number $x(n)$ and then convert the time-domain series into its frequency domain representation using DFT:

$$DFT(x(n),K) = \frac{1}{N}\sum_{n=0}^{N-1} x(n) \times e^{-j2\pi kn/N}, \quad k = 0,1,2,\ldots,N-1. \tag{1}$$

In Figure 9(a), the normalised amplitude spectrum of a shrew stream and a TCP flow at low frequency band of [0 Hz, 50 Hz] show that a low frequency band biased energy distribution can be used as the signature of shrew attacks. We can see that the 20 Hz frequency point is the end of the major peak of amplitude spectrum of shrew stream. As such, shrew attack stream could be segregated from legitimate TCP flows by comparing their normalised amplitudes.

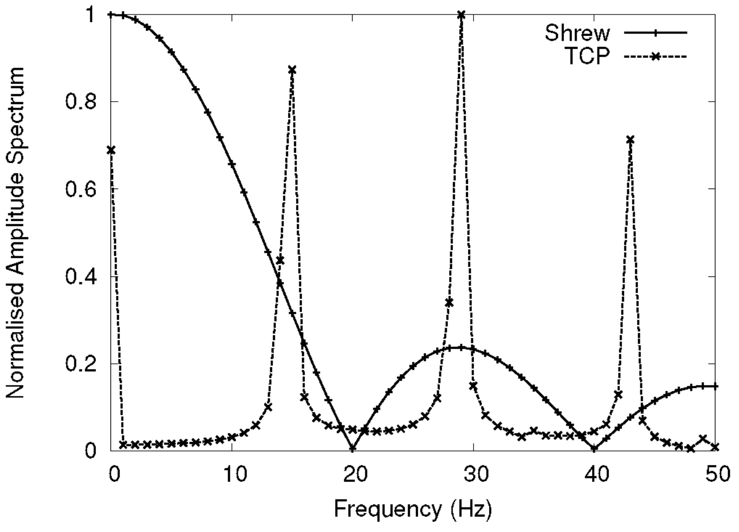
Our NS-2 simulations are carried out with the topology shown in Figure 8 for different combinations of legitimate TCP flows and shrew attack streams. We compared the TCP throughputs achieved by the shrew-filtering algorithm with the well-known Active Queue Management (AQM) algorithm *Drop Tail* using normalised throughput (ρ) defined by:

$$\rho = \frac{Xput_d}{Xput_n} \tag{2}$$

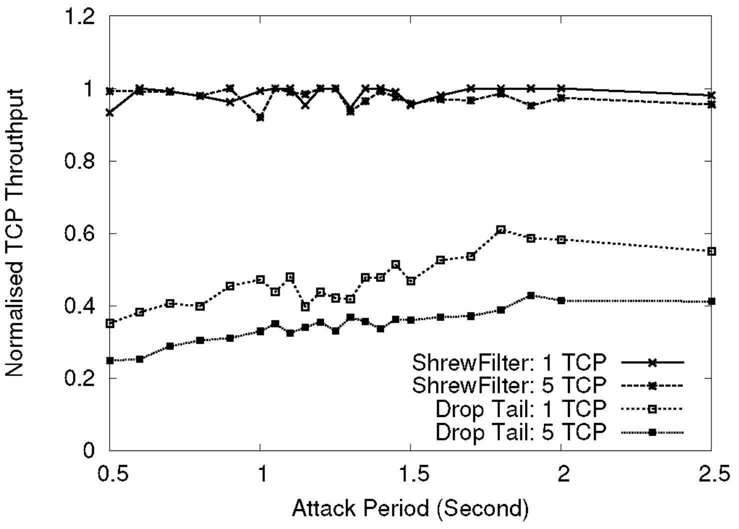
where $Xput_d$ is the average throughput achieved by the TCP flow (or aggregate) with DDoS stream and $Xput_n$ is the throughput achieved without DDoS stream.

Figure 9(b) compares the scenarios of single and 5 TCP flows under attack of four shrew streams distributed in space and time. Under the Drop Tail algorithm, the throughput of legitimate TCP flow is far below the actual attainable throughput and the link utilisation is very low. With our shrew-filtering algorithm, the gain in TCP throughput is significant. It reaches what legitimate flows can reach when there is no shrew stream.

Figure 9 The filtering of pulsing shrew DDoS attacks from normal TCP traffic flows (a) normalised amplitude spectrum of the shrew pulse stream and of the TCP flow and (b) normalised throughput TCP vs. four spatial shrew attacks



(a)



(b)

6 Fuzzy trust management in grids

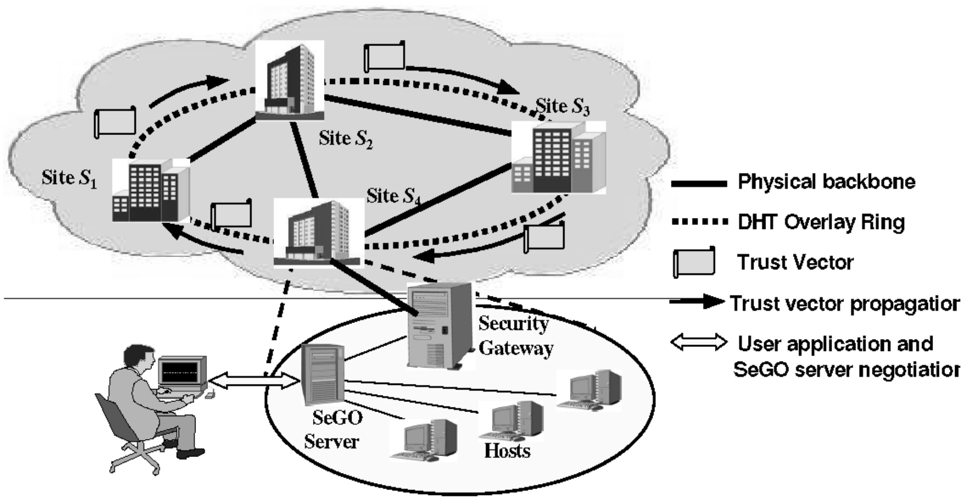
The trust relationships among Grid sites are hard to assess due to uncertainties involved. Fuzzy theory is capable of quantifying imprecise data or uncertainty in measuring the Trust Level (TL) of resource sites. Specifically, we have proposed a fuzzy-logic based trust model to enable the aggregation of numerous trust parameters and security attributes into easy-to-use scalar quantities (Song et al., 2005a).

In our model, the job Security Demand (SD) is issued by a user program. The demand may appear as a request for authentication, data encryption, access control, etc. The TL of the target system is aggregated through a fuzzy-logic inference process over the contributing parameters. A trusted resource allocation scheme must satisfy a *security-assurance condition*: *TL* ≥ *SD* during mapping of user jobs onto the Grid resource sites.

6.1 Fuzzy trust integration

The TL is normalised as a single real number with 0 representing the condition with the highest risk and 1 representing the condition that is totally risk-free or fully trusted. The fuzzy inference is done by four steps: *fuzzification, inference, aggregation* and *defuzzification*. Fuzzy aggregation process is conceptually illustrated in Figure 10.

Figure 10 Fuzzy aggregation for trust integration over a DHT-based overlay network in which cooperating gateways working together between the Grid resource sites



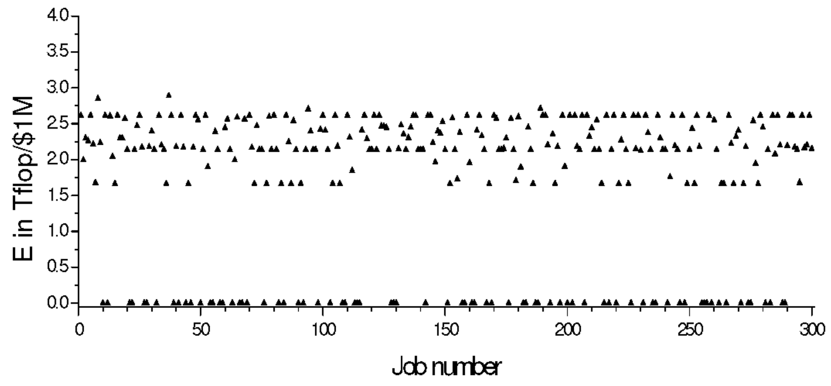
The site TL is an aggregation of site reputation and site defence capability. The *reputation* is an aggregation of four major attributes: *prior job execution success rate, cumulative site utilisation, job turnaround time* and *job slowdown ratio* (Song et al., 2005b).

The defence capability is attributed to intrusion detection, firewall, anti-virus/worm, and attack response capabilities. Both site reputation and defence capability jointly determine the TL of a resource site. In Song et al. (2005a), we have suggested a fuzzy-logic based approach to generating the local TL from the above-mentioned attributes.

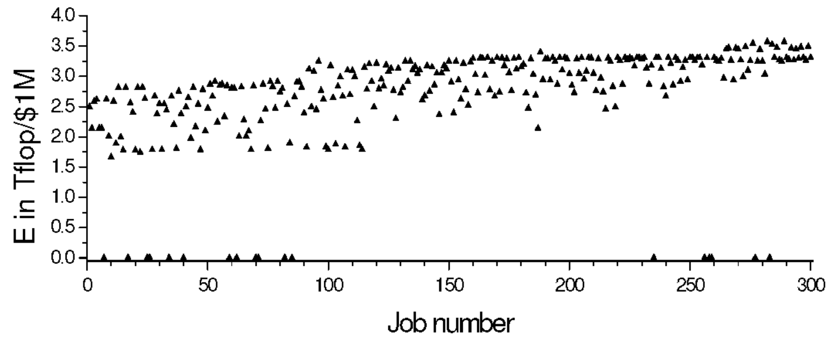
6.2 Simulated performance results

We evaluate the trust integration performance by simulating job execution on multiple Grid sites. Figure 11 compares performance/cost ratio *E* of 300 jobs with fixed trust and integrated trust. One job group consists of those jobs failing the qualification test or short of resources before the deadline expired. These are job drops with *E* = 0 along the X-axis. The dropped jobs are not allocated with any resources. The second job group contains the successful jobs meeting controlled security and thus allocated with ample resources.

Figure 11 Comparison of the Grid performance/cost ratio of using Grid with six resource sites to execute 300 independent jobs (a) performance under no security upgrade and (b) performance after trust integration



(a)



(b)

In Figure 11(a) and (b), the job drop rates are 25.3% and 6% for without and with trust integration, respectively. The average performance/cost ratio *E* are 2.27 Tflop/$1M and 2.92 Tflop/$1M. The job drop rate is reduced by (76 – 18)/76 = 75% in favour of trust-integration solution. On the average *E*, a performance gain of 28% = (2.92 – 2.27)/2.27 was resulted from trusted resource allocation. The results clearly demonstrate the effectiveness of trust integration. Security-upgraded resource sites are able to accommodate 94% = 1.6% of 300 user jobs satisfactorily in these experiments.

6.3 Grid performance metrics

When the *security-assurance condition: TL* $\geq$ *SD* can be satisfied, the job can expect to finish successfully. Otherwise, the job may fail and has to be restarted on the same or sent to different securer site. The above section studies job scheduling with security assurance.

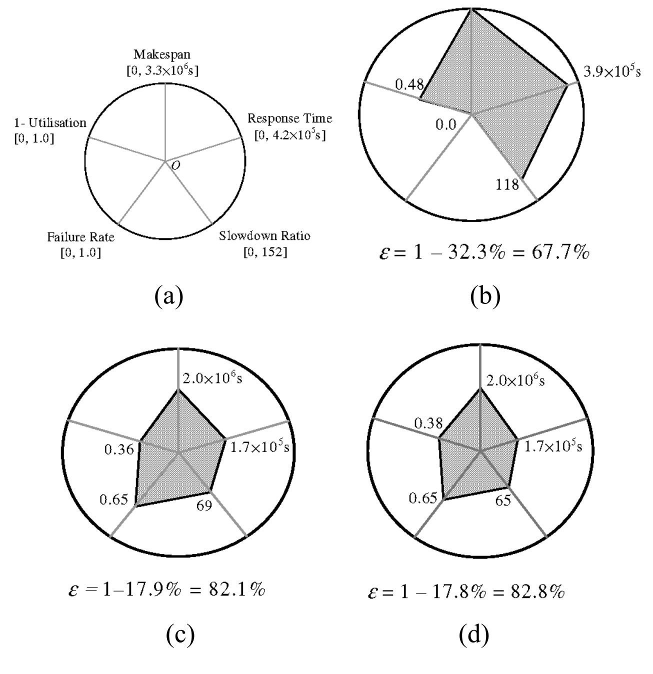
In this section, we study three risk modes to match the *TL* of Grid sites to *SD* of jobs. These modes are designed to apply in realistic Grid scheduling platforms:

- *Secure mode*. A job is conservatively allocated to a site, only if the risk-free condition *TL* $\geq$ *SD* is met.
- *Risky mode*. Allocate jobs to any available Grid sites and thus take all possible risks at the resource sites.
- *f-risky mode*. This is a partial risky mode, by which the job takes some calculated risk. Allocate jobs to available sites to take at most *f* risk, where *f* is in the range (0, 1) with *f* = 0 for the secure mode and *f* = 1 for the risky mode.

We evaluate the performance of each scheduling mode of the scheme by considering five performance metrics: *makespan, job failure rate, site utilisation, response time* and *slowdown ratio* etc. These definitions were original given in Song et al. (2005b). In Figure 12, we use a 5-D Kiviat diagrams to evaluate trusted resource allocation under three risk conditions. The five dimensions correspond to the above five performance metrics. We define a *Grid efficiency function* as follows:

$$\varepsilon = (1 - A_{\text{shaded}}) / A_{\text{circle}}. \quad (3)$$

Figure 12 Grid efficiency measured along five dimensions of the Kiviat diagrams corresponding to five performance metrics shown: (a) scales and ranges of five measures; (b) secure mode with *f* = 0; (c) *f*-risky mode with *f* = 0.5 and (d) risky mode with *f* = 1



The smaller is the shaded polygon area at the centre of the Kiviat diagram, the better is the Grid efficiency. The risky mode results in the best Grid efficiency. The secure mode actually leads to the poor efficiency. This implies that more efficient Grid has shorter makespan, average job response time and lower slowdown, failure rate, and under-utilisation rate (i.e., 1 – utilisation rate).

Our NAS simulation results (Song et al., 2005b) shows that it is more resilient for the global job scheduler to tolerate job delays introduced by calculated risky conditions, instead of resorting to job preemption, replication, or unrealistic risk-free demand. Higher Grid utilisation is observed after the integration process.

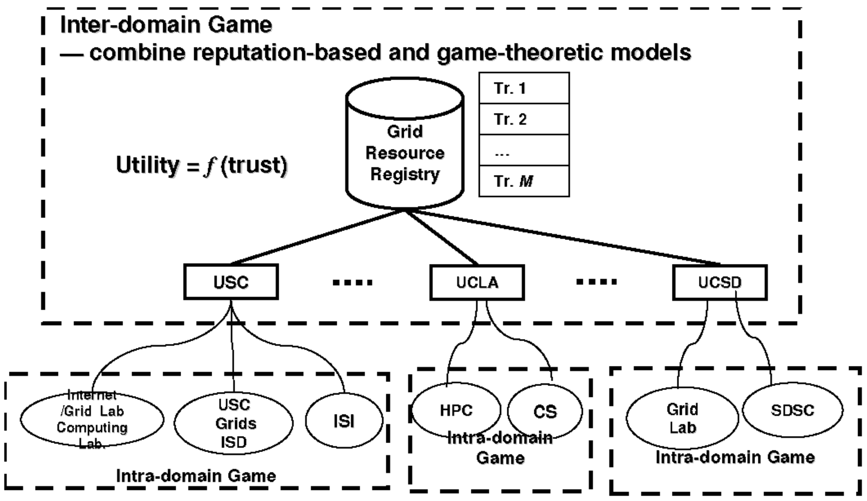
7 Game-theoretic grid modelling

Selfish behaviours of individual machines in a Grid can potentially damage the performance of the system as a whole. However, scrutinising the Grid by taking into account the non-cooperativeness of machines is a largely unexplored research problem. Recently we have witnessed an intensive interest in using game theoretic and market-oriented approaches in the analysis and design of distributed computing and networking algorithms. An open question to ask is how to structure and maintain such a gigantic Grid computing platform.

7.1 Hierarchical grid architecture

We believe that a hierarchical structure, as depicted in Figure 13, is the only feasible solution. In our study, we envision that each 'Grid site' is not going to be a single computer but rather a network of computers, each of which is a cluster of machines or a tightly-coupled massively parallel machine. Thus, eventually we may have hundreds of Grid sites, each of which consists of tens of multiprocessors (i.e., clusters and parallel machines). Indeed, such a structure, again resembling the internet itself, closely matches the 'administrative' structure of computing resources in real-life organisations.

Figure 13 Hierarchical structured Grid modelled by inter-domain and intra-domain games



With the hierarchical structure shown in Figure 13, there are also two levels of job scheduling and dispatching. Specifically, the job submission system, which is implemented as a global middleware, channels user submitted jobs to the global scheduling system.

In practice, such a job submission middleware can be easily constructed using Web services tools (e.g., WSDL and SOAP messages). Equipped with a global Grid processing resources registry (again could be based on the UDDI protocol), the global scheduler performs job allocation, according to a certain scheduling algorithm.

Most importantly, at the inter-site level, the scheduler has only the knowledge of the processing capability of each Grid site as a whole, without regard to the details within the site. In this manner, the scalability of scheduling at the global Grid level can be efficiently handled. Furthermore, again this scheduling model conforms well to the administrative structure of the Grid community in the sense that the global scheduler probably should not 'micro-manage' the execution of jobs down to the machine level.

The global scheduler makes use of the capability parameters supplied by the Grid sites as the inputs to the scheduling algorithm. These capability parameters are, in turn, mediated by the local job dispatcher at each Grid site based on its information about the local participating machines. From the hierarchical model, we can formulate three different game theoretic job allocation and execution problems (Osborne and Rubinstein, 1994).

7.2 Intra-site job execution strategies

This problem concerns about the strategies of the participating computers inside a Grid site. Specifically, each individual computer is selfish in that it only wants to execute jobs from local users but does not want to contribute to the execution of remote jobs. For example, a computer cluster administrators and/or users may prefer to dedicate the computing time to satisfy local requests first. However, if no participating computer contributes, the Grid site as a whole will fail to deliver its promise to serve the Grid community, thereby defying the original motive of forming the Grid.

7.3 Intra-site bidding

This problem concerns about the determination of the advertised 'execution capabilities' for jobs submitted to the global scheduler. Recall that for the scheduler to allocate jobs using a certain scheduling technique, it needs to know all the sites' execution capabilities – in our study, these are modelled as the execution times needed for the pending jobs. To determine the execution time needed for a certain job, within a Grid site each participating computer can make a declaration and a notification to the local job dispatcher specifying the time needed to execution the job.

The local job dispatcher can then 'moderate' all these declarations to come up with a single value to be sent to the global scheduler. For example, if the local job dispatcher is aggressive in job execution, it could use the 'minimisation' approach – taking the minimum value of the declarations from all the member computers. On the hand, a conservative approach is to perform 'maximisation' – taking the maximum value instead. This problem is interesting in that we need to analyse, possibly using auction theory, to determine the best strategies for each member computer in bidding (Kwok et al., 2004, 2005).

7.4 Inter-site bidding

Similar to the intra-site situation, at the inter-site level, the various local job dispatchers also need to formulate game theoretic strategies for computing the single representative value of the job execution time to be sent to the global scheduler. Indeed, different combinations of the above games will result in different Grid structures. For a *semi-selfish Grid*, the intra-site games are non-cooperative while the inter-site game is cooperative.

This model fits most nowadays' Grid situation because a Grid is usually formed after some cooperative negotiations at the organisation level. However, the individual machines operated by bottom-level departments may not cooperate among each other. For a *fully-selfish Grid*, the games are assumed to be non-cooperative at all levels. This model is the most general model. Finally, the ideal Grids are modelled by cooperative games at all levels.

In our intra-site modelling, when a job is assigned to a Grid site, it is up to the member machines within the site to decide who should take it up. We model a 'wait-and-see' mechanism in this job execution game. Each machine in the site selfishly waits to see if some other machine would be so kind to take it up and execute it on the site's behalf. With the incentive of protecting the system reputation, some machine will eventually take up the job.

To analytically model this 'wait-and-see' situation, for each machine *i*, we define a utility function $U_i = P_i^t / P_i^r$, where P_i^t is the total number of processing elements within the machine and P_i^r is the number of processing elements used for remote jobs. We assume that $P_i^r > 0$ because there is always some overhead incurred in participating in the Grid. With this utility function, each machine decides whether to take on a job based on a probability s_i. That is, we assume mixed strategies with randomised decisions instead of deterministic actions.

7.5 Game optimisation strategies

The parameter s_i is also called the Degree of Cooperation (DoC) of the game. To model the penalty incurred when a job is not taken after each round of waiting, we use a parameter $\alpha(1 > \alpha > 0)$, called *selfishness penalty factor*, to compute the extra processing elements required to finish the job in time despite the delay incurred. Specifically, after each round of waiting, αP extra processing elements are required to catch up with the job execution deadline, where P is the original required number of processing elements. With n machines in a Grid site, we can formulate three different strategies:

- *Nash strategy*. This strategy corresponds to the randomised action taken by all the machines uniformly at the Nash Equilibrium (Osborne and Rubinstein, 1994). Specifically, at a Nash equilibrium, there is no incentive for any machine to unilaterally deviate from the equilibrium strategy. Every machine uses a value of s_i decided by:

$$s = \frac{(1-\alpha)\xi^2 - (\alpha+2)\xi + 1}{(1-\alpha)\xi^2 - \xi} \tag{4}$$

where $\xi = (1 - s)^{n-1}$.

- *Optimal strategy*. We found that (Kwok et al., 2004) the Nash strategy is suboptimal in the sense that the utility function is not optimised. Thus, we derived (Osborne and Rubinstein, 1994) the optimal strategy by finding the maximum point of the utility function with respect to the strategy space. At this optimal point, every machine also uses a uniform strategy s_i according to the following equation:

$$\begin{aligned}&1+[(1-\alpha n)+(n-2)s](1-s)^{2n-1}\\&\quad-[(\alpha n+2)+(n-2)s](1-s)^{n-1}=0.\end{aligned} \tag{5}$$

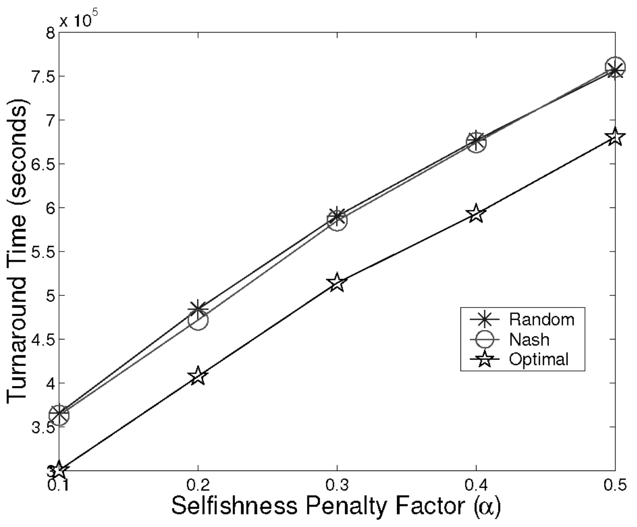
- *Random strategy*. This corresponds to a totally uncoordinated situation wherein the machines heterogeneously use different random strategy values of s_i.

We performed an extensive simulation study to investigate the efficacy of the above three strategies. We used three months of accounting records for the 128-node iPSC/860 located in the Numerical Aerodynamic Simulation (NAS) Systems Division at NASA Ames Research Center. This trace contains 92 days data, gathered in year 1993. There are 16,000 jobs in the whole trace. We have simulated a 12-site Grid in which each site has eight machines. Each machine has eight processing elements.

7.6 Simulated grid game performance

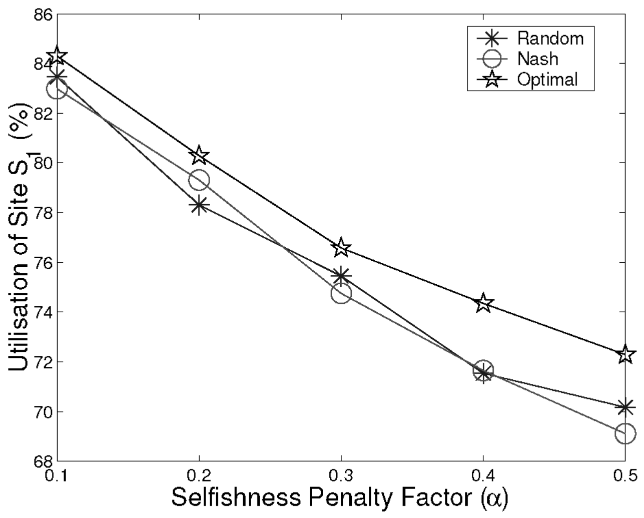
Figure 14 shows the turnaround time and utilisation results of the three strategies. As can be seen from Figure 14(a), the turnaround time of the optimal strategy is significantly shorter than those of the random and Nash strategies. This indicates that achieving an optimised utility value at each machine, albeit in a selfish manner, can also produce a much better system performance than using a more natural uncoordinated random strategy.

Figure 14 The NAS performance using the three game strategies (Random, Nash, and Optimal) under various values of the selfishness penalty factor (a) turnaround time and (b) utilisation of a single site



(a)

Figure 14 The NAS performance using the three game strategies (Random, Nash, and Optimal) under various values of the selfishness penalty factor (a) turnaround time and (b) utilisation of a single site (continued)



(b)

Another important observation is that apart from being suboptimal in terms of utility value, the Nash strategy's system performance is also poor – almost the same as that of a purely random strategy. In general, the turnaround times increased as the selfishness penalty was increased. Figure 14(b) illustrates the utilisation of a single site *S* (chosen in an unbiased manner). We can see that the performance of the optimal strategy was again the best. More results can be found from Kwok et al. (2004).

8 Conclusions

We have reported the recent progress made by the GridSec project at USC. This covers trust management, security-driven job scheduling, trusted resource allocation, DIDS, collaborative alert correlation, worm containment, distributed DDoS pushback, and game-theoretic modelling of realistic Grids that are selfish and non-cooperative in nature. We offer a scalable security overlay architecture, experimental validation of DIDS design, and new schemes to capture network worms and pushback DDoS attacks. The GridSec system offers early warning of internet worm spreading and launching effective pushback operations to protect Grid resources.

The major threats come from software vulnerability and naive users. Today's Windows, Unix and Linux variants are by no means immune from worm attacks, let alone free from DDoS flood attacks. Outbreaks must be dealt with immune response swiftly. The major research challenge lies still in the worm containment area. In particular, we need automated signature generation and fast suppression of malicious flows. Other interesting approach is to use cardinality counting to cope with DDoS attacks (Durand and Flajolet, 2003; Hwang et al., 2005b).

Internet outbreak detection and monitory are other big challenges. The reaction time, containment strategies, deployment scenarios are all yet to be worked out. We have identified the requirements of robustness, resilience, cooperativeness, responsiveness, efficiency, and scalability. The DHT-base security overlays offer a viable approach towards a fast cybersecurity solution. Of course, further advances in operating-system security, active networks, and trust management are also important.

The game-theoretic model and the optimal strategies suggested in our study are designed to rescue selfish Grids from becoming useless in real-life applications. A broader impact of this work is the restoring of faith of cooperative distributed computing in open Grids organised by virtual organisations. The performance results under Nash equilibrium and optimal strategies over the NAS workload offer the first set of quantitative data towards the design and evaluation of practical computational, information, data, and business Grids.

Acknowledgements

We are grateful to GridSec team members, Xiaosong Lou and Runfang Zhou, for useful discussions. This work was fully supported by National Science Foundation (NSF) under Information Technology Research (ITR) grant 0325409 at the USC internet and Grid Computing Laboratory.

References

Cai, M., Chen, Y. and Hwang, K. (2005) 'Integrating misuse and anomaly-based intrusion detection systems with weighted signature generation', Technical Report (TR-2004-16), USC Internet and Grid Computing Lab, Submitted to *IEEE Transactions on Dependable and Secure Computing*, September.

Cai, M., Hwang, K., Kwok, Y-K., Chen, Y. and Song, S. (2005) 'Collaborative internet worm containment', *IEEE Security and Privacy*, IEEE Computer Society Press, June, Los Alamitos, CA, USA, pp.25–33.

Chen, Y. and Hwang, K. (2006) 'Collaborative detection and filtering of shrew DDoS attacks using spectral analysis', *Journal of Parallel and Distributed Computing*, September, Vol. 66, No. 9.

Chen, Y., Kwok, Y-K. and Hwang, K. (2005) 'MAFIC: adaptive packet dropping for cutting malicious flows to pushback DDoS attacks', *Proc. Int'l Workshop on Security in Distributed Systems (SDCS-2005)*, Ohio, June, pp.123–129.

Cuppens, F. and Miege, A. (2002) 'Alert correlation in a cooperative intrusion detection framework', *IEEE Symposium on Security and Privacy*, pp.187–200.

Durand, M. and Flajolet, P. (2003) 'LogLog counting of large cardinalities', *Proc. European Symposium on Algorithms*, pp.605–617.

Hwang, K., Chen, Y. and Liu, H. (2005a) 'Protecting network-centric computing system from intrusive and anomalous attacks', *Proc. IEEE Workshop on Security in Systems and Networks (SSN'05)*, in conjunction with IPDPS 2005, April 8.

Hwang, K., Kwok, Y-K., Song, S., Cai, M., Zhou, R., Chen, Y., Chen, Y. and Lou, X. (2005b) 'GridSec: trusted grid computing with security binding and self-defense against network worms and DDoS attacks', *ICCS-2005 International Workshop on Grid Computing Security and Resource Management (GSRM'05)*, May, Atlanta, GA, pp.187–195.

Kamvar, S., Schlosser, M. and Garcia-Molina, H. (2003) 'The Eigentrust algorithm for reputation management in P2P networks', *Proc. 12th International World Wide Web Conference*, pp.640–651.

Kim, H.A. and Karp, B. (2004) 'Autograph: toward automated distributed worm signature detection', *Proc. USENIX Security Symposium*, pp.271–286.

Kodialam, M., Lakshman, T.V. and Lau, W.C. (2004) 'High speed traffic measurement and analysis methodologies and protocols', *Bell Labs Technical Memo*, August.

Kwok, Y-K., Song, S. and Hwang, K. (2004) 'Non-cooperative grids: game-theoretic modeling and performance optimization', *IEEE Transactions on Parallel and Distributed Systems*, accepted to appear in 2006, November 30.

Kwok, Y-K., Song, S. and Hwang, K. (2005) 'Selfish grid computing: game-theoretic modeling and NAS performance results', *Proc. CCGrid 2005*, Cardiff, UK, May 9–12, pp.1143–1150.

Moore, D., Shannon, C., Voelker, G.M. and Savage, S. (2003) 'Internet quarantine: requirements for containing self-propagating code', *Proc. INFOCOM*, pp.1901–1910.

Nagaratnam, N., Janson, P., Dayka, J., Nadalin, A., Siebenlist, F., Welch, V., Tuecke, S. and Foster, I. (2002) *Security Architecture for Open Grid Services*, http://www.ggf.org/ogsa-sec-wg, July.

Osborne, M.J. and Rubinstein, A. (1994) *A Course in Game Theory*, MIT Press, Cambridge, MA, USA.

Singh, S., Estan, C., Varghese, G. and Savage, S. (2004) 'Automated worm fingerprinting', *Proc. USENIX Symposium on Operating System Design and Implementation*, San Francisco, December, pp.45–60.

Song, S., Hwang, K. and Kwok, Y-K. (2005a) 'Trusted grid computing with security binding and trust integration', *Journal of Grid Computing*, Vol. 3, Nos. 1–2, pp.53–73.

Song, S., Hwang, K. and Kwok, Y-K. (2006) 'Risk-resilient heuristics and genetic algorithms for security-assured grid job scheduling', *IEEE Trans. Computers*, Vol. 55, No. 6, pp.703–719.

Song, S., Kwok, Y-K. and Hwang, K. (2005b) 'Security-driven heuristics and a fast genetic algorithm for trusted grid computing', *Proc. IEEE IPDPS-2005*, April 4–9, Denver, CO.

Stoica, I., Morris, R., Karger, D., Kaashoek, M.F. and Balakrishnan, H. (2001) 'Chord: a P2P lookup protocol for internet applications', *Proc. ACM SIGCOMM*, pp.149–160.