



# An Effective Layered Load Balance Defensive Mechanism against DDoS Attacks in Cloud Computing Environment For Smart City E-Governance Systems

Nimesh Vaidya ( Research Scholar , Swaminarayan University ) ,

Email: nimesh.vaidya11@gmail.com

Prof.(Dr.) Vijaykumar B. Gadhavi, Dean, Faculty of Engineering, Swaminarayan University Email: deanengineering@swaminarayanuniversity.ac.in

## Abstract

Cloud computing is a technology which completely shifts the data to unaware Datacenter (DC) where the cloud service provider (CSP) is responsible for the subscribers' data and its protection. Distributed Denial of Service (DDoS) is a kind of overload threat aims to subvert DC and their resources which leads to resource unavailable to legitimate requestors. In this paper we proposed an effective layered load balancing mechanism which scrutinizes the incoming requestors' traffic at various layers and each layer outwits some kind of attack traffic. The early network traffic condition prediction paves the way to detect the threats earlier which in turn improves the availability. The significance of the proposed mechanism is detecting the higher rate of overload threats at earlier layers. Constant monitoring and stringent protocol setup for incoming traffic strengthens the proposed mechanism against several kinds of overload threats. Based on the traffic pattern of incoming requestors, the vulnerability is observed and outwitted at various layers. The simulation proved that the mechanism proposed is deployable at an attack-prone DC for resource protection, which would eventually benefit the DC economically as well.

**Keywords:** Cloud Computing; Availability; Load Balancing; DDoS; Flash crowd; Botnets

## 1. Introduction

Cloud computing is a technology that suffered from its security breaches, of which availability is the most serious security issue. Distributed Denial of Service (DDoS) is a kind of resource-availability related attack for subverting the Data Center (DC) for resource unavailability to the legitimate clients. DDoS is one of the most high traffic rate overload conditions towards DC. The incoming huge traffic consists of both legitimates and attacking traffic. Segregating them dynamically is a challenging task. Moreover, this DDoS attack is easier to launch from the client end and harder to withstand or detect at DC end. This made us pursue our research towards direction of DDoS Detection and DC resource protection in a cloud network.

Cloud Flare experienced the largest DDoS attack flooding in February 2014, which is a record-breaking from 300 up to 400 Gbps attack. This attack is launched by anti-

spammers Spamhaus, which was the largest DDoS attack up to date (2014) [1]. Some

other notorious DDoS attack experiences are: Mafia boy who succeeded in bringing down the world's most popular websites, namely Yahoo, CNN, eBay, Dell and Amazon in

February, 2000. Similar attacks were made towards South Korean's largest newspaper,

bank and United States forces created as a botnet over hundred thousands of computers in July, 2009 [2]. Still, several serious DDoS events lead to long outages to be identified and prevented.

The rest of the paper is organized as follows. Section 2 describes surviving techniques. Section 3 presents overview of the proposed architecture and mechanism. Section 4 explains the working mechanism. Section 5 discusses the performance of the proposed mechanism. Section 6 deal with the advantages of the proposed approach and Section 7 provides the conclusions with an outline for our future work.

## 2. Surviving Techniques

Sporas, which is a “reputation” based methodology, has the reputation value of any user from 0 up to 3000 [3]. Obviously, the new user will have the minimal reputation value of 0. The current user’s reputation will always be higher than the new users. Any two users should have only one rating value range from 0 up to 3000. In case the two users have several interactions, then only the recent value is considered. The evaluation is based on the more recent computed reputation value, because the more recent computed values results are current or much closer to the current behavior. The reputation value increases over the period upon good behavior and does not influence the initial low score.

Eschenauer proposed a framework for the evidence based trust management [4]. This considers the trust as a set of relationships between any two parties with the support of evidence. One way to generate evidence is through public-key cryptography. One of the entities in the network can create evidence for itself and for others. In order to create the evidence, the creator entity creates a piece of entity and signs it with the private key. It mentions its validity period and shares it to others with a public key for identification. Here the drawback is that an entity could also revoke the shared evidence. Since the revoking option could allow any anonymous partner to create evidence and to revoke it, this would create chaos in the network.

Multilevel trust filtration [5] mechanism consists of four modular detection algorithms. Firstly, Link prefetch which attempts to identify the location of the incoming requestor. Secondly, the Requestor scrutinizer which verifies the network-specific data to authenticate the incoming requestor. Thirdly, Traffic data log which logs the request rate and request type and distinguishes the attack category and eventual access after the right approval decides the differential treatment for any different type of overload conditions and incoming traffic.

Unlike conventional DDoS detection and protection mechanisms, the Trilateral Trust mechanism [6] is a scheme which is extending in direction of detection with three sidelong functionalities. They are the preliminary traffic signaller, Authentic Trust launcher, historical trust Analyzer that scrutinizes several cases of requestors and continuously monitors their behavior and updates the trusteeship of the requestor towards the CSP.

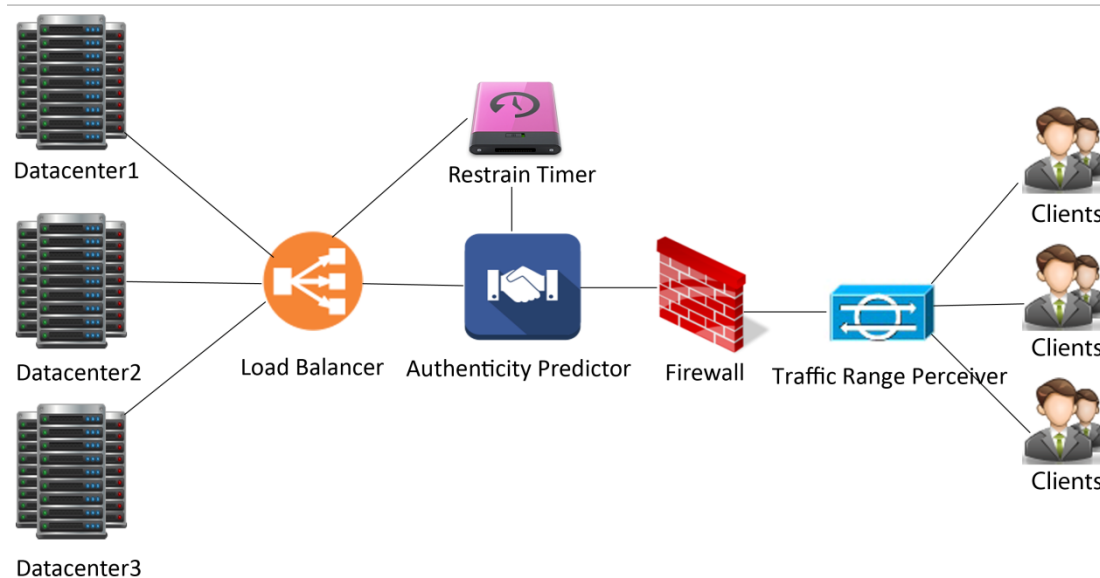
A stochastic hill climbing approach [7] which is centralized load balancing mechanism and it is an incomplete approach for solving optimization problem. Genetic Algorithm [8] based load balancing proposes the three step operation namely population generation, cross over to select the best fit resource configuration and mutation for choosing the probable best resource configuration. For insider threat detection, log management [9] has been proposed which uses log analysis with event correlation. Certain rule with timestamp is continuously monitored using log watcher for better detection.

Network flow based entropy approach [10] analyzes requests per flow and improves detection accuracy by three steps. They are flow aggregation analysis based on traffic flow, fast entropy computation, and threshold refresh for maintaining and improving the accuracy rate of detection. Application layer DDoS poses critical threats to web server [11]. For such detection, the traffic has to be classified in order to avoid considering flashcrowd as DDoS. This has been deployed with real-time frequency vector. Fuzzy logic based DDoS defense mechanism [12] considers the network packet parameters which attempts to detect the DDoS attack scenario, but there is a limitation or pitfall when the

rate of traffic is huge as they process each traffic requests’ packet parameters. DDoS Defense for web services in cloud discussion reveals the threat of XML attacks at application layer [13]. Here the feature extraction and attack request model construction typically detects malicious requests.

All the current surviving techniques either detect DDoS attack or create a secured architecture to protect the DC resource with certain a time lag. The proposed overload detection mechanism detects various kinds of attacks and authenticates them at different levels, and eventually makes the cloud environment free from attackers.

### 3. Overview of Proposed Mechanism



**Figure 3.1 Architecture of Proposed ELB Mechanism**

Requestors can be of several groups requesting several types of requests. Once they initiate the request, all the requests are fed into the Traffic Range Perceiver (TRP). TRP helps in sensing the incoming request traffic rate. This TRP acts as the early alarm system which identifies and classifies the traffic rate as normal and abnormal. The traffic rate is identified and then fed into firewall which at the initial stage directly allows the requests to bypass as the firewall has not informed about the incoming requests. Once the incoming request bypass firewall and reaches Authenticity Predictor (AuthPr), the incoming client requests types are identified and processed accordingly. The incoming request types are recognition requests, acknowledge requests, service requests. All the incoming request types are configured at web services for dynamic identification at AuthPr. If the incoming requests are new requestor they are queued separately at Restrainer Timer which is to authenticate the incoming requestor activity. Restrainer Timer (ResT) is contains two distinguishable queues to increase the service rate for legitimate clients. One of the queues holds the new requestor requests, and another queue is for registered requestor requests which not only optimizes the service rate and also allows new requestors at appropriate time. The authenticated requestors bypass the Restrainer Request Manger (RRM) module and reaches Load Balancer (LB), Here the LB is again broken up into two modules: they are overload Diminishment (oDim) and Instantaneous Load Balancer (ILB).

The reason of LB being splitted into oDim and ILB is because of the new registration requestor joins the traffic at Authenticity Predictor, which does not give any information about the character of incoming requestor. So the oDim, helps in detecting the overload if incase the new requestor attempts to overload the DC. Other kind of overload condition is detected at ILB. ILB is a very fast and efficient module to handle with almost any kind of request as it already gets some kind of information from the previous modules as buffer information and reports firewall to filter the requestor for certain period of time.

#### 3.2. Special Cases of Overload Threats

**Botnets:** Botnets are machines usually range from tens to thousands which continuously flood requests towards target to subvert the victim target. The impact depends on the flooding rate and number of machines joins the flooding rate. As this kind of threat is the highest degree of overload, this traffic overload must be outwitted sooner to serve the incoming requestor effectively otherwise this overload would deplete network and prevent legitimate requestor's entry. Bots are traffic requests simply overload DC with junk requests and as they are machines they cannot respond to AuthPr which turned as a characteristic to detect and prevented at firewall for certain time period. TRP identifies the abnormal network condition at the earliest at the time Botnet traffic flood.

**DDoS:** DDoS attacks are initiated and continued by some hundreds of attackers grouped and distributed globally who start populating the unwanted traffic packets with enormous size in order to acquire the memory, network resources and completely deplete them. Next to Botnet machine oriented overload, DDoS is higher rate of overload. DDoS attack is detected at earlier stage at the time of AuthPr, At the time of DDoS, TRP predicts abnormal traffic condition. As the DDoS attackers only aim is to subvert the DC performance, they continuously increase the traffic rate. When they attempt to do so, they are already queued at AuthPr, consecutive incoming request within timeout period recognizes the abnormal request traffic behavior.

**Spoofers:** This is a kind of threat which is impersonation of legitimate requestor who attempts to join the traffic and attempt to acquire illegitimate access. Spoofers are illegitimate requestor who attempt to gain access to DC by impersonating legitimates. This attack group is identified at AuthPr as they fail to acknowledge. Though they acquire registration at ICL, they could not be valid at ACL of AuthPr because the AuthPr response will be sent back to request initiator where only legitimate client only can respond to AuthPr as they communicate with ciphered session initiation and impersonator fails to decipher. This provides no way to acknowledge AuthPr with valid session initiation and thus prevented at firewall for certain timeout period.

**Aggressive Legitimates:** This kind of threat is an internal or registered requestor attempting to overload DC which often unnoticed but contaminates CSP network resources if undetected pollutes memory and other resources of CSP. Aggressive legitimates are the type of requestors who are often named as internal over loaders, as they are valid for CSP and successfully bypass the validation at AuthPr and ResT even at the time of overload. After successful validation, they are forwarded to load balancer which implicitly probes the incoming traffic with overload diminishment module by analyzing the requestor traffic behavior and the aggressive legitimates are prevented at firewall.

**Flash Crowd:** Flash crowd is an overload condition caused by simultaneous incoming of large number of legitimates but operated over a short period of time. Since flash crowd is rare event and though the traffic rate increases suddenly but will drop quickly and thrive for considerably short period. More over the requestors joining the flash crowd event are large in number but follow legitimate traffic profile, so each requestor will be allowed to access DC and bypass load balancer's overload diminishment as they follow legitimate traffic rate.

## 4. Working of Proposed Mechanism

### 4.1. Traffic Range Perceiver

---

#### Algorithm 1. Traffic Range Perceiver

---

```

Input: Incoming requests
Output: Network Traffic Condition
BEGIN
  FOR each time interval, t
    Requests are asserted for traffic Rate computation
    IF( $RT_{in} \leq BW_{in}$ ) Normal Ingress
    Traffic
  ELSE
    Abnormal Ingress Traffic
  ENDIF
END FOR
END

```

---

**$RT_{in}$ —Incoming request traffic;  $BW_{in}$ —ingress bandwidth;**

Whenever any clients interested in acquiring DC resource as a service, they are allowed to access DC only when they successfully bypass TRP. TRP acts as an early alarming system which continuously monitors the incoming traffic requests and predicts the traffic condition in an outline. This monitoring system aptly recognizes the traffic condition and fed the traffic condition to further level of overload detection system to improve the processing time of the proposed detection mechanism which further helps in improving the optimal load balancing to satisfy the requestor's requirement. If the incoming request traffic exceeds cloud network bandwidth, then the requests are queued, the limit of queue is configurable by CSP which is based on the traffic prone DC zones. This TRP Module can differentiate the new registration requestor and already registered requestors based on the web service initiation from client end and can also recognize the registered clients based on the unique  $ID_{client}$ .

### 4.2 Authenticity Predictor

---

#### Algorithm 2. Authenticity Predictor.

---

```

Input: Requests bypassed TRP with the alarming traffic condition
Output: Requestor type based authentication
BEGIN
  FOR each  $RT_{in}$ 
    type = IdentifyRequestType();

```

```

IF(type == RequestACC) IF(type ==
RequestNR)
Forward to Restrain Timer for further Processing
ELSE IF(type == RequestAuth)
AuthenticateRequestor();
ELSE
IF(type == RequestSER)
Forward to LB for further processing
END IF

```

---

```

END IF END FOR END
IdentifyRequestType(){
The requestor's webservice at DC end identifies the request type whether new
or registered. return type;}
AuthenticateRequestor(){ Create IDsession and
update ICL
Send IDsession encrypted with KDC-X
IF(At client end, when client is able to successfully decrypt)
Obtain IDsession, forward it to DC by encrypting with secret key by appending
IDclient.
END IF
IF(At DC End when the IDclient at ICL matches the decrypted IDsession) Ksession generated
for IDclient and forwarded to client.
IDclient destroyed at ICL and updated at ACL END IF}

```

**Request<sub>ACC</sub> - Access Request; Request<sub>NR</sub> - New Registration Request for those who attempt to sign up for the first time at CSP; Request<sub>Auth</sub> - Registered clients who attempts to sign-in to access DC resources; Request<sub>SER</sub> - Authenticated clients who bypass to reach LB and access the specific service type; ACL - Alive Connection Log; ICL - Idle Connection Log; K<sub>DC-X</sub> - Secret key shared between DC and client (X); ID<sub>client</sub> - Client ID; ID<sub>session</sub> - Session ID maintained at DC end; K<sub>session</sub> - Session key generated for clients at DC end.**

Authenticity Predictor is responsible for identifying the type of request and to authenticate the requestor. Basically the request types are access requests and service requests. Access requests are type of requests which process the authentication of the incoming requestor. Service requests are the authenticated requests which specify the type of request in need for further provision. If the request type is service request, then AuthPr simply forwards the requests to LB without any delay. Once the access request is authenticated and service request is approved, then the requestor is not much intervened and periodically monitored by provisioning the intended service. This reduces overhead of continuously monitoring of authenticated requestor instead of processing the arriving requestor traffic. Access requests again classified as registered requestor for arriving for authentication and the new requestor arriving for new registration request for accessing DC resources. New Registration requests are forwarded to RRM module for further processing as they would take time in proving the legitimacy behavior which has been dealt in detail at section 4.3. The requestor who had already been queued and again sending the request within timeout period is considered as DDoS attack traffic requestor and they are forbidden to firewall. The same is applicable at RRM to sync with the queued requestors.

When any registered client sends Request<sub>Auth</sub>, it bypasses TRP and reaches AuthPr. AuthPr creates ID<sub>session</sub> for each incoming ID<sub>client</sub> and maintains at ICL also triggers timeout period by forwarding the ID<sub>session</sub> encrypted with shared secret key to prevent the overhead of encryption at highly traffic prone DCs deployed at any public cloud environments. Once the client with ID<sub>client</sub> is able to decrypt at client ID and obtains ID<sub>session</sub>, returns the same ID<sub>session</sub>. When ID<sub>session</sub> is received successfully at AuthPr before timeout period, then ID<sub>client</sub> is deleted from ICL and is generated, K<sub>session</sub> maintained in ACL to identify them as active client count and the K<sub>session</sub> is forwarded to appropriate client for further valid communication with DC until session timeout which is a configurable parameter depends

on traffic prone zones. ICL is a (ID<sub>client</sub>, ID<sub>session</sub>) tuple. Whereas ACL is (ID<sub>client</sub>, K<sub>session</sub>) tuple.

### 4.3 Restrain Request Manager

Restrained Request Manager (RRM) is an extension of AuthPr but activated only for new registration requests. Based on the network traffic condition, the requests are queued and act as a buffer zone for request processing. At the time of abnormal traffic condition predicted by TRP, this RRM is activated and requests are queued. Here, the requests cannot be simply queued as the incoming requestors can be the new registration requests and also the registered requests. When these requests are simply queued, it results starvation and sometimes deadlock situation with DC resource and unnecessary waiting time for all other following registered requestors. So,

separate queue is maintained for new registration requests namely  $Q_{NR}$  and the queue for registered clients namely  $Q_{RC}$ . Requestors queued at  $Q_{RC}$  will wait for very short time; this queue is enqueued only when the traffic condition is predicted as abnormal. This procedure of differential treatment for new registration requests and registered requests helps in recognizing and ejecting the resource hunger activity based on the behavior at oDim which is dealt in section 4.4.

### Algorithm 3. Restrain Request Manager

```

Input: Request Type, Traffic Condition.
Output: Requests Queued based on their arrival for nominal network performance at DC.
BEGIN
FOR each Request
IF(Traffic Condition == Abnormal Traffic) IF(Request Type==
RequestNR)
EnqueueRequestNRwithgenerated newIDclient&trigger timeout period at QNR. ELSE
EnqueueIDclientwithIDsessionat QRC. END IF
ELSE
END IFIF(Traffic Condition == Normal Traffic&& Request Type== RequestNR) EnqueueRequestNRwithgenerated
newIDclient&trigger timeout period at QNR. ENDIF
END FOR END

```

#### $Q_{RC}$ - Queue data structure to maintain the registered client; $Q_{NR}$ - Queue data structure to maintain the new registration requests;

This Procedural process of handling the requests towards server farms or DCs at cloud network helps in maintaining resiliency at CSP DC network even at the time of overloaded network with unknown requests traffic. The timeout period at  $Q_{NR}$  helps to destroy the request which does not respond with the appropriate  $ID_{client}$ . Instead of directly assigning the  $ID_{session}$  the new requestors are validated based on the response of initial trial communication shoot back  $ID_{client}$  from the requestor's timestamp. If failed to receive the  $ID_{client}$  back within timeout period, the registration request is rejected. The reason of disallowing the session allocation for new registration requestor is to give higher priority to registered clients, so after the validation, the new requestors are again queued at  $Q_{RC}$  in case of abnormal traffic condition.

#### 4.4. Overload Diminishment.

oDim is most important module as it is the last layer of filter which recognizes the type of overload and diminish the traffic that reaches load balancer for accessing the DC resources. Malicious and legitimate incoming requestors' request type resembles similar whereas their traffic pattern deviates significantly which remains as the characteristic to differentiate the legitimacy of incoming requestor. Ingress traffic can be uniquely identified, computed and updated based on the  $ID_{client}$ . The request traffic recording helps in computing the traffic rate for each incoming  $ID_{client}$ . This traffic rate pattern helps in differentiating the ingress requestors. So, the default or nominal traffic profile is to be set by the CSP which is configurable based on the network bandwidth. By following the nominal request traffic profile is set not based on the conventional overall requests per second, whereas nominal request is based on the requests per second per requestor. So, it is much easier to detect the traffic behavior of each individual requestor. Traffic rate computation and traffic behavior prediction of each incoming requestor paves the way to differentiate the overloading requestor.

oDim maintains Traffic Behavior Repository (TBR) which contains client ID to uniquely identify the requestor and their traffic rate and session key for revoking the access against traffic rate violation. All the unnecessary traffic had been monitored and filtered appropriately at each level and the bypasses subsequent level.

---

### Algorithm 4. Overload Diminishment

---

```

Input: Requests bypassedAuthPr, IDclient Output:
Classified Legitimate traffic BEGIN
FOR each incoming requestor
IF (IDclientexists in TBR) Update traffic rate
ELSE
Create record for IDclient
END IF
IF (IDclientfollows nominal traffic profile)
Forward requests of IDclienttoInstantaneous LB Module
ELSE
Defer IDclientfor certain time period by forwardingIDclientto firewall to block them.

```

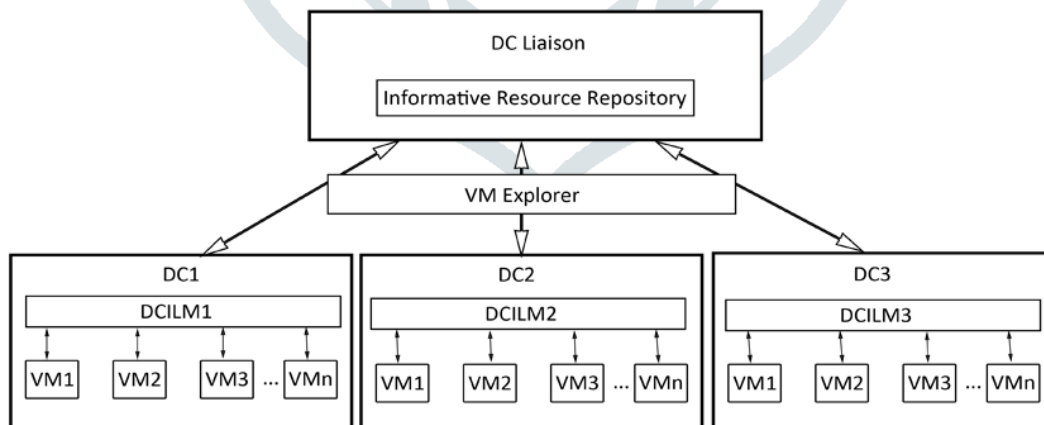
END IF END FOR  
END

**Table 1. Traffic Behavior Repository (TBR)**

Client ID	Traffic Rate ( $\chi$ ) (Request/Sec)	Session key
Client <sub>1</sub>	$\chi_1$	$K_{S(1)}$
...	...	...
Client <sub>N</sub>	$\chi_N$	$K_{S(N)}$

When the requestor actual traffic rate appears abnormal or exceeds the limit of nominal traffic profile, that particular requestor information is now removed from TBR and handed over to firewall to prevent the requestor until the session expires. This approach has an improved detection rate of abnormal requestors' traffic as it monitors traffic rate which reveals the direct behavior of requestor, as and when the requestor attempts to overload the DC, concern requestor is blocked by TBR and blocked further, penalizing them by reporting to firewall to deny further service requests initiated from the requestor with certain session key until that session expires. Then after the session expires, the requestor can join the queue to behave with nominal traffic profile to access DC resource. If the requestor again misbehaves, they are blocked. Again, the session expiration plays key role because increasing and decreasing session timeout has its own advantages and disadvantages which is the trade-off between continuously authenticating the requestor instead of serving them or block the misbehaved requestor for longer period. Maintaining optimal session improves performance along with improved detection rate. Otherwise, they are considered as legitimate requestor and forwarded to actual load balancer for quicker response with highly minimal waiting time.

#### 4.5. Instantaneous Load Balance Scheme



**Figure 2. Architecture of Instantaneous Load Balance Scheme**

#### Algorithm 5. Instantaneous Load Balance Scheme

Input: Incoming legitimate request Traffic  
Output: Balanced Load among DC  
BEGIN  
DCInstanceLoadMonitor(); IF(Load<sub>DC(x)</sub> >= ProcCap<sub>DC(x)</sub>) DCLiaison(); DCVMEexplorer();  
ELSE  
Resource Provision at requested DC.  
END IF END

**ProcCap<sub>DC(x)</sub>**: Processing Capability of DC<sub>(x)</sub>; **Load<sub>DC(x)</sub>**: Load at DC<sub>(x)</sub>

The requests that reach load balancers are thoroughly monitored at oDim where the cause of overload is found and the threats are precisely recognized based on their traffic pattern and prevented by blocking at firewall. Once the traffic bypasses oDim, then the request traffic is considered as complete legitimate traffic. This way the DC resources are protected against overloading threats without contaminating the network resources. It is better to serve requestors by forwarding the requests based on the closest DC but this scheme fails at overloading scenario as the DDoS attackers simply targets to subvert the DC. Instead of simply forwarding the requests to DC, the load is to be balanced effectively so as to prevent the target DC from being hunted. At each

DC, there are DCILM which is shown in figure2 which is responsible for monitoring the processing load at each VM.

#### 4.6 DC Instance Load Monitor (DCILM)

The requests from oDimare fed to DCILM of one of the DC. AT DCILM, the load of each VM is monitored and acts as load balancer within the DC among VMs. The load information of each VM can be found at DCLFR which contains the tuple of VM ID at DC, its memory utilization, Processor utilization, and the percentage of Criticality.

Algorithm 6. DC Instance Load Monitor

```

Input: Incoming requests
Output: Load Information from DCInstance Load Factor Repository

BEGIN
  IF(DCLFR == empty)
    Requested resource can be allocated
  ManageDCLFR
    Compute Availability Compute PoC
    IF(PoC >= threshold PoC)
      RA = Non-Critical
    ELSE
      RA = Critical
    END IF END IF
    IF(DCLFR != empty)
      Sense DCLFR for Load Factor; IF(Load Factor (each
VM) == Critical) DC_Liaison();
      Obtain DCxwith VMy
      Transfer Request to DCx
    ELSE
      Allocate VM Resource within the requested DC Update DCLFR
    END IF END IF
END
    
```

**PoC - Percentage of Criticality;RA –Resource Availability;DCLFR - DCInstance Load Factor Repository;**

**Table 2. Data Center Load Factor Repository (DCLFR)**

VM ID	Memory ( $\alpha$ )	CPU( $\beta$ )	Load Factor ( $\phi$ )	PoC	Timestamp ( $\rho$ )
VM <sub>1</sub>	$\alpha_1$	$\beta_1$	$\phi_1$	PoC <sub>1</sub>	$\rho_1$
...	...	...	...		...
VM <sub>N</sub>	$\alpha_N$	$\beta_N$	$\phi_N$	PoC <sub>N</sub>	$\rho_N$

If DCLFR is empty then there is no load at any VMs. So, the incoming requestor’s required resource can be allocated at suitable VM based on configuration. Create new resource allocation activity/ loadstatus of VM in DCLFR and also availability status is to be computed for allocated VM which is a useful metric for subsequent quicker allocation.

$$\text{Total resource available at DC (A)} = \sum_{i=1}^n VM$$

$$\text{Resource allocated at DC (B)} = \sum_{i=1}^n V A_{i} \rho_{i}$$

Resource Availability at DC (C) = A-B PoC = ((A - C) / A) \* 100;

100;

Here, the threshold PoC is CSP configurable parameter which depends on time taken to balance the load and number of DCs associated with load balancing activity. If the criticality percentage is lesser than threshold limit, then the resource is available and can be allocated, if not the VM is left as is and new VM is searched for allocation.

If the DCLFR is non-empty, the load factor of VM is non-critical and the requested resource limit is available at VM then the resource is allocated. If the load factor is found to be critical, then find next VM that suits the requestor configuration. If none of the VM matches, then request DC Liaison which acts as a networked load balancer among available DC. Based on the response obtained from DC Liaison, allocate the resource through VM migration and update DCLFR of appropriate DC.

#### 4.7 DC Liaison

Whenever the DC<sub>x</sub> does not find the requestors' resource configuration then it transfers the request to DC Liaison which acts a load balancer among DCs. Now, on receiving the request for VM configuration from any DCILM, DC Liaison (DCL) simply broadcasts the VM configuration information to all DCs. The responses from DCs are recorded at IRR, which is analyzed and optimized configuration is sent back to DC<sub>x</sub>. All the new VM allocations are recorded at IRR and the latest responses of DCs by allocated VMs are also updated at IRR. At each session expiry, IRR is updated along with each DCILM to maintain updated load/ allocation information. DCL holds all the information about VM migration across DCs. After the VM Exploration at each DC, if the notification received as RRA, then allocate the resource and update IRR. Otherwise find the minimum PoC of DC to match the required configuration. If match found, then find the optimal PoC for VM migration. If the minimum PoC does not match the required configuration, then skip searching this DC and check other DCs and allocate the max PoC of VM with optimal PoC of all DCs suitable for resource allocation. This DCL's IRR holds updated PoCs of all DCs. So it is also possible to check with IRR for any matched configuration available before searching with DC VM Explorer.

#### Algorithm 7. DC Liaison

Input: VM configuration from DCILM

Output: Availability from Informative Resource Repository

BEGIN

Initiate DC\_VM\_Explorer();

Receive the resource availability notification (VM<sub>Notified</sub>) from notification =

notification = DC\_VM\_Explorer();

Manage IRR; IF(notification == RRA)

Identify PoC of VM<sub>Notified</sub>

Update IRR ELSE

Identify VM<sub>LC</sub>

For( x = 1 to n in IRR)

IF (VM<sub>config</sub> < VM<sub>x</sub> && VM<sub>x</sub> < VM<sub>LC</sub>) VM<sub>opt</sub> = VM<sub>x</sub>

END IF

END FOR

IF (VM<sub>opt</sub> found)

Update IRR and balance load by VM Migration

ELSE Continue with other DC

END IF

END IF END

IRR - Informative Resource Repository; VM<sub>Notified</sub> - VM notified by VM Explorer as VM with matched configuration; VM<sub>LC</sub> - VM with Largest Configuration; VM<sub>Config</sub> - Required VM Configuration; VM<sub>opt</sub> - optimized VM suitable for requested configuration.

Table 3. Informative Resource Repository

Exploration ID	DC ID ( $\lambda$ )	VM ID	Load Factor ( $\phi$ )	PoC	Timestamp ( $\rho$ )
Exp <sub>1</sub>	$\lambda_1$	VM <sub>1</sub>	$\phi_1$	PoC <sub>1</sub>	$\rho_1$
...	...	...	...		...
Exp <sub>N</sub>	$\lambda_N$	VM <sub>N</sub>	$\phi_N$	PoC <sub>N</sub>	$\rho_N$

IRR Efficiency = max (min PoC(DC<sub>1</sub>), min PoC(DC<sub>2</sub>), ..., min PoC(DC<sub>N</sub>))

This way of modularized communication between DCs and VMs within DCs and recording the latest allocation information at IRR which acts as a cache until next IRR update. This scheme improves the load balancing efficiency by improving response time and with no processing time at all at certain cases where the required configuration available at IRR.

#### 4.8. DC VM Explorer

---

##### Algorithm 8. DC VM Explorer

---

Input: VM Configuration

Output: Explore the VM among available DCs

BEGIN

Receive VM Configuration request from DC Liaison

Search for a DC; Check DCLFR;

IF (VM Configuration found)

return RRA ELSE

return RRUA

END IF

---

END

---

RRA –Requested Resource Available; RRUA –Requested Resource UnAvailable

When any requestor's configuration is not found at IRR, then the VM Explorer begins exploration at each DC's DCLFR and obtain the resource configuration availability by parsing the complete list of VMs thereby returns the resource availability notification. This iterative approach helps in not only identifying the requested resource configuration for allocation but also updates IRR about the resources available at DCs. The performance improvement is directly proportional to the number of DCs involved in LB. At this VM Exploration phase, each DC's resource availability can be obtained and this information is used if they are required for any other LB activity in near future. If incase of RRUA status found in all the available DC, it is considered as blocking condition of load balancer which results in increased waiting time for requestors by continuously updating the sessions of legitimacy requestors at load balancer.

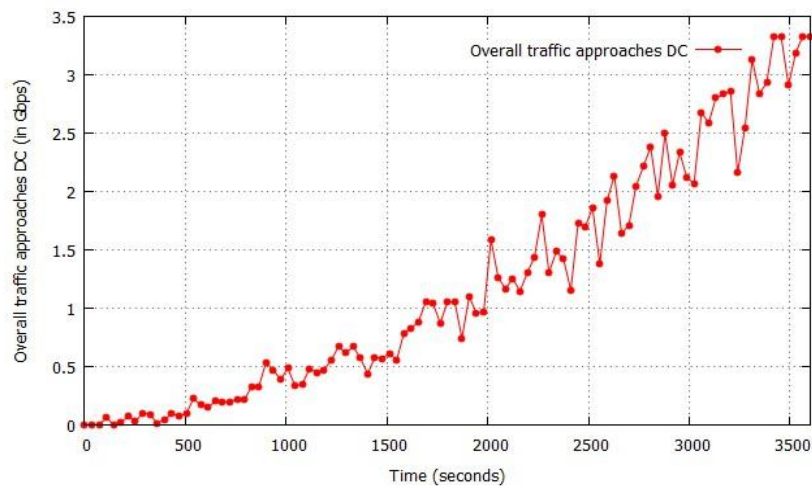
## 5. Performance Evaluation

### A. Experimental Setup

Jeyanthi and Iyengar [14] used OPNET as simulator to test the cloud computing environment. Jeyanthi *et al.* [15] experimented DDoS in cloud computing. We tested our proposed mechanism as simulation experiment in OPNET Modeler. The experiments are performed in a campus network where DC requesters are grouped in five subnets and each subnet has got 100 workstations, 100 attackers and 200 legitimate clients requesting for application-specific requests at each subnet. This way we created the attacker and legitimate profile and other devices, which would be needed to test our algorithm as an experiment. The traffic represents internet and the group of spoof attackers is activated at varying time intervals. The attack profile is replicated to increase the attack strength to engage the DC resources like bandwidth, CPU and memory. On the whole, our experiment has 1000 clients and 500 attackers. The experiment is carried out with three different scenarios, namely the network with no attackers, networks with attackers and no detection mechanism in place, and finally the network with attackers and the proposed oDim& ELB (Enhanced Load Balancer) mechanism where the nominal request traffic rate is configured as 0.2/sec, session timeout period configured as 120sec and threshold PoC configured as 90%. The configuration would vary based on the network settings.

## B. Performance Evaluation

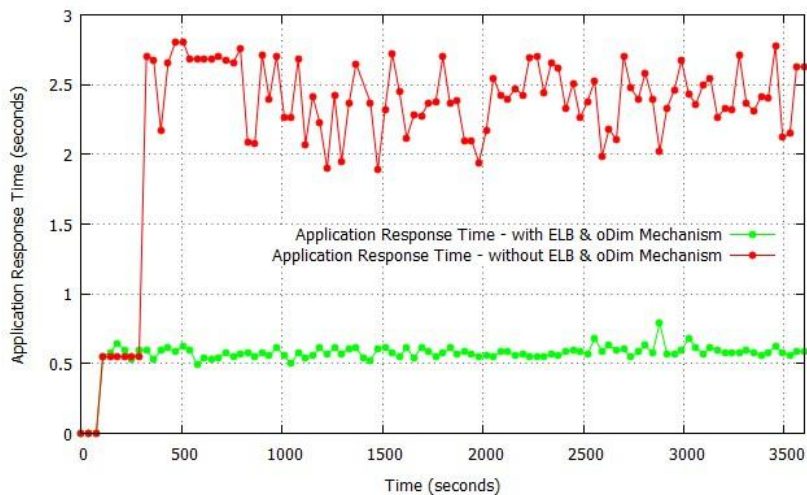
**5.2.1 Overall Traffic Rate:** Overall traffic rate is the statistic that represents the average number of request packets received or transmitted by the receiver or transmitter channel per second. This raw traffic includes the both legitimate and illegitimate requestors' traffic.



**Figure 3. Overall Network Traffic Rate**

Figure 2 shows the overall traffic approaches CSP *i.e.*, the traffic reaching TRP. It is notable that the traffic reaching TRP grows exponentially. The traffic pattern can be seen oscillating which is because the incoming DDoS attackers starts flooding requests but they are outwitted continuously as they do not follow the legitimate traffic profile. The oscillation appears more and more as more number of misbehaved or overloading requestors is continuously outwitted at firewall until the session expiry. Based on the random number of DDoS attackers' activation, the attack traffic grows at a maximum range of 3.5 Gbps exponentially.

**5.2.2 Application Response Time:** Application Response time is the statistic measured as the time elapsed between sending requests and receiving response from DCatCSP network. Figure 4 shows the application response time at the time of DDoS with the proposed ELB and without ELB.



**Figure 4. Application Response Time**

**5.2.3 Authenticity Predictor Response Time:** Authenticity Predictor response time is the statistic measured as the time elapsed between sending requests and receiving response from DC in the network, which includes signaling delay for the connection setup. Figure 5 shows the AuthPr response time and also its processing time.

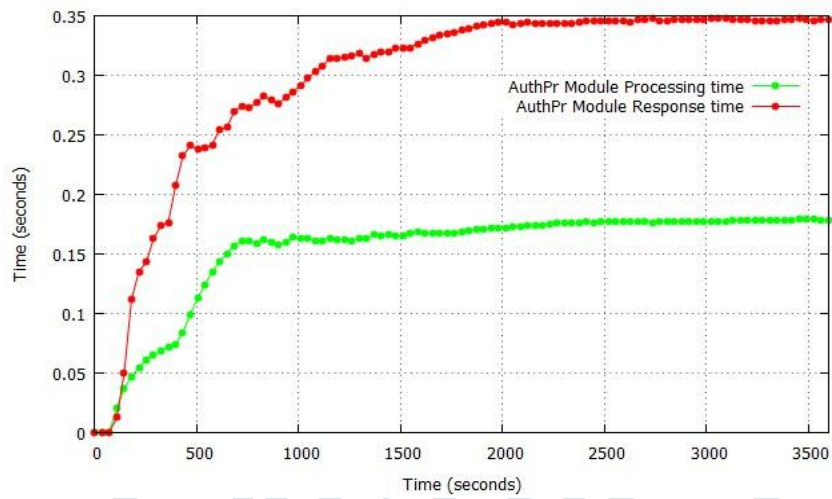


Figure 5. Authenticity Predictor Response Time

**5.2.4 Restrain Request Manager Efficiency:** Restrain Request Manager efficiency is the statistic which is measured as the delay incurred at active legitimate requestors at the time of DDoS overload towards victim DC. Figure 6 shows the Restrain Request Manager efficiency of the proposed mechanism for the registered requestors and un-registration requestors.

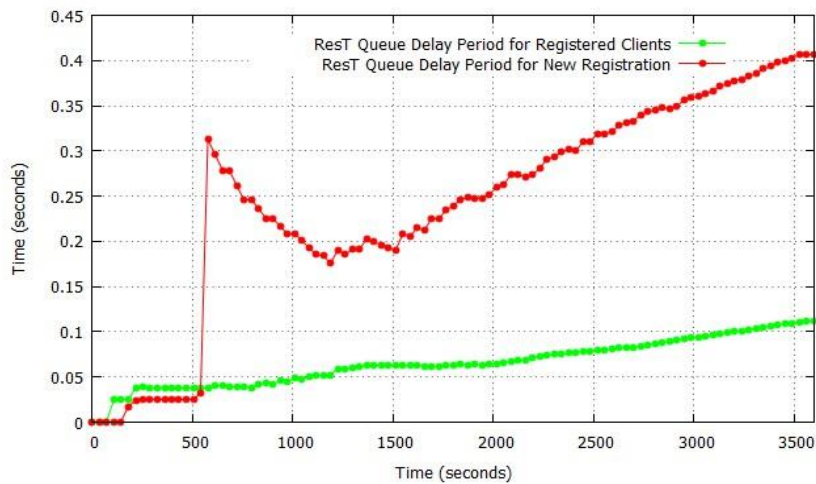


Figure 6. Restrain Request Manager Efficiency

**5.2.5 oDim Efficiency:** oDim Efficiency is the statistic measured as the time taken to process the incoming requestor traffic behavior to initiate the differential treatment. Figure 7 shows the oDim efficiency of the proposed mechanism at the time of network traffic overload.

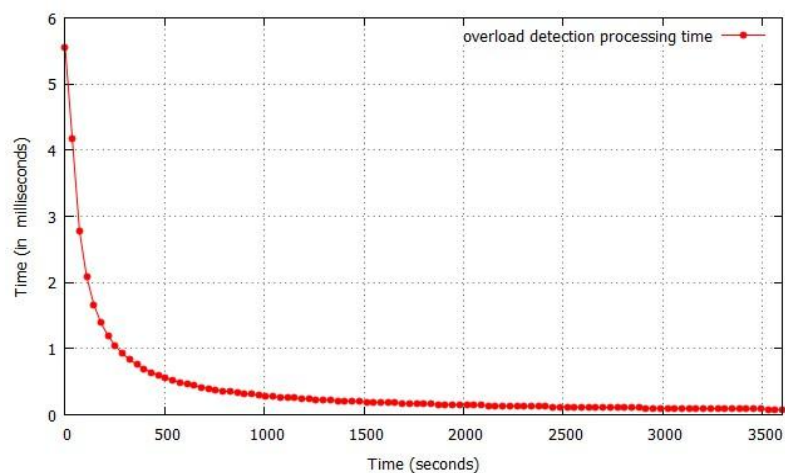


Figure 7. oDim Efficiency

**5.2.6 Load Balancing Efficiency:** Load Balancing Efficiency is the statistic measured as the processing time taken to provision the necessary resource to the legitimate requestors. Figure 8 shows the load balancing efficiency of the proposed mechanism at the time of DDoS overload.

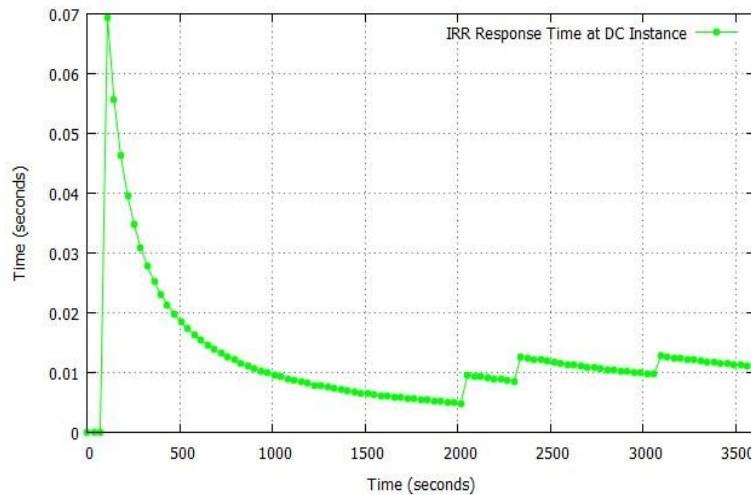


Figure 8. Load Balancing Efficiency

**5.2.7 Firewall Occlusion:** Firewall Occlusion is the statistic measured as the time taken to forbid the requestor based on the illegitimate traffic behavior. Figure 9 shows the firewall occlusion of the proposed mechanism at the time of DDoS overload.

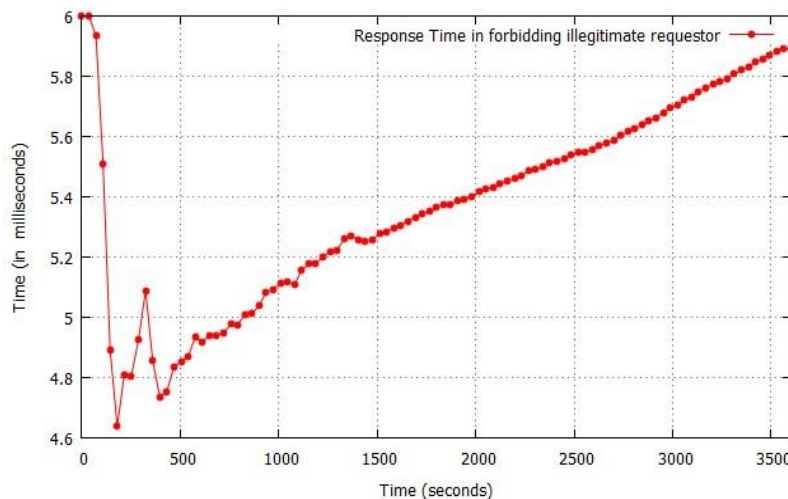


Figure 9. Firewall Occlusion

**5.2.8 Detection Efficiency:** Detection Efficiency is the statistic measured as the traffic flow rate towards DC bypassing several levels of protection layers of the proposed architecture. Figure 10 shows the detection efficiency of the proposed mechanism at the time of network overload.

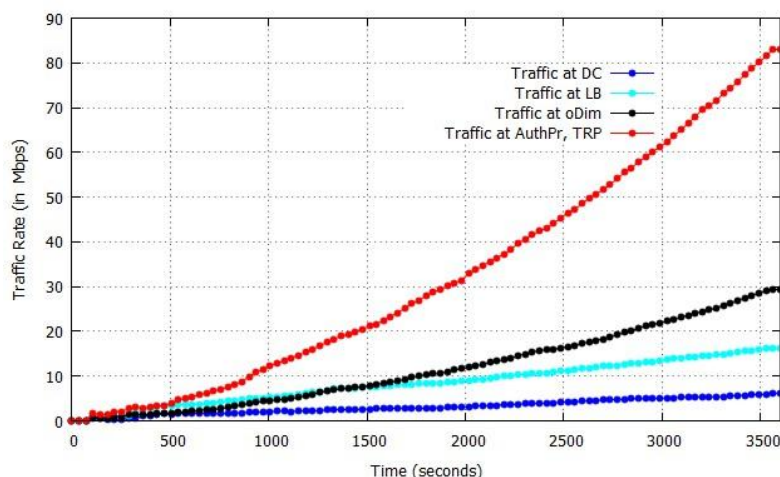


Figure 10. Detection Efficiency

## 6. Advantages of Proposed Mechanism

**Layered Load Balancing** - Load balancing activity has passively begun right from the TRP module. Figure 10 shows the detection efficiency of the proposed mechanism where the incoming traffic is continuously processed at various levels and the rate of traffic reaching DC is probed in order to maintain resiliency.

**Suits at any Deployment Model**—usually the proposed mechanism include AuthPr which is responsible for authenticating the flooding traffic. Activating ResT at the time of overload and at the time of new registration request arrival, it is safer and faster to allow only registered requestor to bypass the load balancer. This procedural way of traffic processing suits at Public / private/ hybrid cloud deployment models.

**Improved performance**—The performance improvement can be observed higher at when the number of DC associated for load balancing activity is higher. Because, the IRR stores the latest load status of each and every DC which results reduced processing time for subsequent load balancing activity. Repositories like DCLFR, IRR stores accurate and timely status.

**High availability**—High availability is achieved only when the resource availability in terms of network. The layered and distributed overload detection improves resiliency. Separate queue management for new registration requestors and registered requestors at various modules improves reliability and at the time of DDoS, the registered requestors could be given priority and the new requestors requests can be dropped for safer consideration improves the resource availability.

**Reduced overhead**— Traditional overload detection schemes imposes crypto-overhead and also bombards at DC for requestor behavior classification. Whereas the proposed detection scheme involve reduced crypto- symmetric overhead only at initial authentication. Processing the requestor traffic by delegating at various levels improves the network resiliency to serve requestors better.

## 7. Conclusion and Future Work

DDoS attacks are the easier to launch but tough to detect. Moreover this type of completely deplete network resource and disallows the legitimate requestors to reach DC. Detecting such kind of overload threats at earlier stage improve the availability. Detecting the overload threats along with better load balancing policy improves performance to greater extent. Instead of load balancing based on round-robin, closest DC, the proposed overload detection and effective load balancing policy has much better advantages because former simply distributes load without considering the victim, whereas latter recognizes the overload and act appropriately. This procedural way of processing traffic through TRP, AuthPr not only improves performance but also alarms the network traffic condition at the earliest to proactively process the incoming request traffic.

Our proposed mechanism suits well in all deployment models as it continuously filtering the traffic and differentiating the traffic pattern and allowing only the legitimate traffic pattern towards DC. Our future work is to improve the load balancing efficiency and to reclaim the resource chunks to allocate them back to the DC for optimized resource usage.

## References

- [1] <https://www.cloudflare.com/under-attack> (accessed on December 2014).
- [2] <http://www.thedailybeast.com/articles/2010/12/11/hackers-10-most-famous-attacks-wormsand-ddos-takedowns.html> (accessed on December 2014).
- [3] G. Zacharia, "Trust Management through Reputation Mechanisms", Workshop in Deception, Fraud and Trust in Agent Societies, Third International Conference on Autonomous Agents (Agents'99), ACM, (1999).
- [4] L. Eschenauer, V. D. Gligor and J. Bara, "On Trust Establishment in Mobile Ad Hoc Networks", Security Protocols Springer, (2004), pp. 47-66.
- [5] N. Ch. Sriman Narayana Iyengar, Gopinath Ganapathy, P. C. Mogan Kumar, and Ajith Abraham, "A multilevel thrust filtration defending mechanism against DDoS attacks in cloud computing environment", International Journal of Grid and Utility Computing, vol. 5, no. 4, (2014), pp. 236-248.
- [6] N. Ch. S. N. Iyengar and Gopinath Ganapathy, "Trilateral Trust Based Defense Mechanism against DDoS Attacks in Cloud Computing Environment", Cybernetics and Information Technologies, vol. 15, no. 2, (2015), pp. 119-140.
- [7] B. Mondal, K. Dasgupta and P. Dutta, "Load Balancing in Cloud Computing using Stochastic Hill Climbing-A Soft Computing Approach", Procedia Technology, vol. 4, (2012), pp. 783-789.
- [8] K. Dasgupta, B. Mandal, P. Dutta, J. K. Mandal and S. Dam, "A Genetic Algorithm (GA) based Load Balancing Strategy for Cloud Computing", Procedia Technology, vol. 10, (2013), pp. 340-347.
- [9] A. Ambrea nd N. Shekokar, "Insider Threat Detection Using Log Analysis and Event Correlation", Procedia Computer Science, vol. 45, (2015), pp. 436-445.
- [10] J. David and C. Thomas, "DDoS Attack Detection Using Fast Entropy Approach on Flow-Based Network Traffic", Procedia Computer Science, vol. 50, (2015), pp. 30-36.
- [11] W. Zhou, W. Jia, S. Wen, Y. Xiang and W. Zhou, "Detection and defense of application-layer DDoS attacks in backbone web traffic", Future Generation Computer Systems, vol. 38, (2014), pp. 36-46.

- [12] N. Ch. S. N. Iyengar, A. Banerjee and G. Ganapathy, "A Fuzzy Logic based Defense Mechanism against Distributed Denial of Service Attack in Cloud Computing Environment", International Journal of Communication Networks and Information Security (IJCNIS), vol .6, no. 3, (2014), pp. 233-245.
- [13] T. Vissers, T. S. Somasundaram, L. Pieters, K. Govindarajan and P. Hellinckx, "DDoS defense system for web services in a cloud environment", Future Generation Computer Systems, vol. 37, (2014), pp. 37-45.
- [14] N. Jeyanthi, N. Ch. S. N. Iyengar, Mogan Kumar, P.C. and Kannammal, A., "An enhanced entropy approach to detect and prevent DDoS in cloud environment", International Journal of Communication Networks and Information Security, vol. 5, no. 2, (2013), pp. 110–119.
- [15] N. Jeyanthi and N. Ch. S. N. Iyengar, "Packet resonance strategy: a spoof attack detection and prevention mechanism in cloud computing environment", International Journal of Communication Networks and Information Security, vol. 4, no. 3, (2012), pp. 163–173.

