Securing Grid Data Transfer Services with Active Network Portals*

Onur Demir, Michael R. Head, Kanad Ghose and Madhusudhan Govindaraju
Department of Computer Science
State University of New York, Binghamton, NY 13902-6000
e-mail:{onur, mike, ghose, mgovinda}@cs.binghamton.edu

Abstract

Widely available and utilized Grid servers are vulnerable to a variety of threats from Denial of Service (DoS) attacks, overloading caused by flash crowds, and compromised client machines. The focus of our paper is the design, implementation and evaluation of a set of admission control policies that permit the server to maintain sustained throughput to legitimate clients even in the face of such overloads and attacks. We propose several schemes to effectively, and importantly in an automated fashion, deal with these attacks and overloads. We discuss how these schemes can be efficiently implemented on an active network adapter based gateway that controls access to a pool of backend data servers. Performance tests conducted on a system based on a dual-ported active NIC demonstrate that efficient optimization schemes can be implemented on such a gateway to minimize the grid service response time and to improve server throughputs under heavy loads and DoS attacks. Our results, using the GridFTP server available with Globus Toolkit 4.0.1, demonstrate that even in adverse load conditions, the response times can be maintained at a level similar to normal, low-load conditions.

Keywords: *GridFTP, Active NIC, Intelligent Gateway*

1. Introduction

The rapid increase in network capacity, coupled with the requirements of data intensive applications, has fueled research on various optimizations for efficient large data transfers. Some examples include GFPS, XCP, XTP, Optiputer, and GridFTP. Among these, the GridFTP framework has emerged as a standard that provides a modular and extensible architecture to serve the needs of high performance applications. The features of GridFTP include support for striped data transfer, collective operations to transfer data between clusters, uniform interface across various data sources and sinks in a distributed environment, tuning of TCP parameters such as window size and number of parallel streams on a per connection basis, reliable re-start of transfers, and support for authentication, data integrity and confidentiality [1, 2, 16].

Even though the GridFTP framework provides a wide range of optimization options for high performance data transfer, its performance is vulnerable to various forms of Denial of Service (DoS) attacks, malicious clients that subvert the overall performance by deliberately tying up resources, flash events caused by legitimate clients, a few large requests monopolizing the available bandwidth, and constant interrupts to ongoing transfers. In this paper we discuss the design, rationale, and performance of a set of policies and optimization schemes that can serve as attractive solutions to this problem. We have implemented these schemes on an active network adapter based gateway that controls access to a pool of backend GridFTP (version 4.0.1) servers.

With the standardization efforts of GGF, it is expected that many popular grid services, such as GridFTP, Replica Location Services (RLS), and Metadata Catalog Services (MCS) will be available on well known locations for use by the grid community. The motivation is to facilitate the easy development of e-science applications. However, the availability of these servers on openly published locations can also expose them to a wide range of client abuses. For example, a sudden and heavy load on a server caused by a set of legitimate clients can severely impede its performance. This phenomenon, called flash crowd, is often in response to a specific event and requires the grid server to process an intense and overwhelming volume of

* This work supported in part by the NSF under award numbers EIA 9911099 and CNS 0454298

requests. Another well known vulnerability of such servers is a denial-of-service attack, which is usually launched by a set of compromised client machines or maliciously configured set of grid nodes. These attacks can prevent genuine users from utilizing grid servers by saturating the compute, network, and storage resources with bogus requests.

Distributed DoS (DDoS) attacks have in the recent past affected many popular web portals. Widely deployed grid servers are also susceptible to such attacks. In this paper we focus on the design of the GridFTP (also referred to as just servers in this paper) servers that will prevent a fluctuation in its performance when exposed to these kinds of attacks. The GridFTP server's performance is sensitive to a sustained or sudden momentary increase in the server's load. Moreover, an existing transfer by itself can demand additional processing (and memory) resources at the server if concurrent channels are used and when retries are automatically invoked on some of these channels because of networking errors or congestion. This problem is exacerbated when transfer requests for files that are significantly large or not have been cached.

An example of a possible DDoS attack on a GridFTP server could be the use the SYN flood attack. In this case the malicious host sends a series of SYN packets to the server, seemingly to initiate a TCP connection via the 3-way handshake protocol. The attacking host uses a spoofed source IP address and does not respond to the ACK packets sent by the grid server host. Ultimately, the grid server will timeout after 70 seconds (or a few minutes depending on the kernel configuration). In this process vital resources that could have been used by legitimate clients are instead tied up in the failed handshake protocol. A variation of this attack is also harmful wherein maliciously configured hosts repeatedly send requests for large files. In this case the hosts send their correct IP addresses and complete the TCP handshake, but waste precious bandwidth and processing power of the server.

It is essential to minimize the number of requests to the GridFTP server that time-out because of packets that are dropped during high load periods. Thus, a vital requirement to deal with attacks is to enable preferentially serving *active* (ongoing) data transfers over new transfer requests. In the absence of such preferential service, the ongoing transfer will time out and get resubmitted soon thereafter, adding to the server's load, further hurting the performance. Additionally, the server's utilization is also reduced as some or all of partial progress made on the transfer is aborted.

Grid services are usually designed such that resources within an organization are governed by local rules and policies. The gateway to the resources of a local organization is often responsible for handling load-balancing, minimizing response-time, maximizing throughput, and for verifying the security credentials of each incoming request. The schemes that we propose in this paper are consistent with the tenet of allowing the configuration of resources in accordance with local policies.

In this paper, we present a technique for selective admission control, implemented on an active network card based gateway (aka intelligent gateway) to a pool of GridFTP servers, which allow these servers to selectively process requests related to an ongoing transfer under heavy, unanticipated load conditions. The intelligent gateway relieves the actual servers from:

- Identifying packets that belong to ongoing packets and treating them separately from those that belong to new requests.
- Wasteful processing of packets that anyway will have to be dropped later on during heavy load due to the policy of prioritizing existing transfers.
- The bookkeeping overhead needed to resume an ongoing GridFTP transfer that was disrupted due to network errors/conditions.

Consequently, the utilization of the GridFTP server improves dramatically and the response time to transfer requests remain relatively stable under a DoS attack or on unexpected heavy load. We evaluate our technique using a prototype implementation and present the experimental results. Our tests involve running several concurrent downloads in striping mode under different conditions. The requesting scripts record the connection time (time from initiation until actual transfer begins) as well as the total transfer time and number of bytes received. The conditions under which we test include: DDoS attack (large number of spoofed SYN packets), high server CPU load, and high server I/O load. Our results show that in each case, we can provide a similar level of service to ongoing clients as during normal, "base case" conditions. We have designed a set of policies and described their rationale and experimental results to quantify the gains due to smart and adaptive admission control policies.

2. Smart Admission Control

We consider a locally distributed server configuration, such as the one shown in Figure 1, where a pool of server machines implements the GridFTP server. The GridFTP server's performance can be severely limited by sudden increases in the requests for its services. Such increases will result in long response times or even in request time-outs. In general, as the request rates increase, the resultant increase on the server load causes the server response time to go up commensurately. Additionally, existing transfers are also delayed. To provide stable transfer times under abrupt

increases in the load due to hostile events (such as a DoS attack) or due to rare but natural events (such as transfer resumption requests on network problems), an effective solution is to admit request packets selectively to the server. We now argue that such limiting is best performed by an intelligent gateway as it relieves the already-loaded server from the chores associated with such admission control.

To implement preferential admission control, the server has to track all ongoing transactions, the number of active service requests for each type of service (GridFTP and possibly others), and accept or deny incoming requests based on some criteria. However, this solution has some drawbacks. An individual server in a locally distributed server pool does not have information on the load and status of other servers. Consequently, server local decisions are not adequate in implementing load balancing or in inferring malicious events directed to the pool. Furthermore, under heavy load, the bookkeeping needed to monitor requests and to implement admission control policies can itself impose additional work on an already loaded machine. Finally, any malicious activity is hard to detect on individual servers.

Another solution may be to naively limit the incoming requests at the gateway leading to the server pool. This has some disadvantages. The ongoing GridFTP transfers are unknown to the gateway and associated packets may be dropped. It is also possible to deny the resumption request for an interrupted ongoing transaction request. A complete solution thus needs to take into account the context of a request. The load information of the servers is important as well; it is not possible to estimate a server machine's load by just examining the incoming packets. Load balancing can only be performed with accurate global knowledge of the load on each server machine.

It is precisely for the reasons listed above that we propose a solution of load and context conscious admission control to a GridFTP server pool using an intelligent gateway.

3. Prototype System Details

Figure 1 shows the overall configuration for our prototype. One port of a *dual-ported* active NIC (network interface card) based gateway acts as an interface to the GridFTP server. All admitted client traffic goes through the active NIC portal towards the server pool via the second interface on the active NIC. Responses from the server use a different path as shown, bypassing the gateway.

The active NIC is responsible for selecting and distributing incoming packets to the servers after subjecting them to a filtering rule. In particular, the intelligent gateway maintains information to prioritize ongoing transfers and information to perform load balancing. The server cluster provides a single IP (virtual IP, VIP) address to the Internet, which is assigned to the incoming port of active NIC. The incoming packet headers are modified by the gateway, which changes the VIP with the IP address of the selected server machine. When the server machine responds to the request it uses VIP as the source IP.

The host, where active NIC is mounted (called the active NIC host), runs a daemon called the *control agent*. The control agent periodically collects information from *server agents* that run on the servers. The control agent uses this information to determine the dynamic packet filtering rules that have to be deployed on the gateway and updates the existing filtering rule set on the active NIC. Keeping the control agent on the active NICs host significantly eases the processing load on the active NIC.

In our prototype implementation, we have used a Ramix PMC 694 active NIC with dual 100 Mbits/sec Ethernet interfaces, two autonomous DMA controllers, a 233 MHz. Power PC CPU and 32 Mbytes of RAM and 8 Mbytes of Flash memory [3]. The Ramix PMC 694 is a PCI card.

The proposed gateway should have at least three capabilities from the standpoint of performance. These are as follows:

1. The gateway should not impede the traffic directed at the servers. The gateway should be able to pass traffic at a rate that is equal to or higher than what it takes to saturate the servers.
2. The gateway should be able to react very quickly to attack traffic.
3. The path to the servers via the gateway must have a low latency - this is necessary for keeping the overall server response time down.

All of these requirements essentially call for a fast packet classification and filtering scheme, a low latency packet transport path from the input port to the server-side port and a simple packet dropping policy that allows the gateway to quickly clamp down on the attack traffic. We meet these requirements as follows:

- The packet filter used in our implementation is the widely used BPF+ packet filter [4]. We modified the native BPF+ code slightly to optimize the performance of the data cache on the Power PC processor on the PMC 694.
- The packet filter was embedded into the TCP/IP stack running on the active NIC immediately on top of the IP layer. The TCP layer was completely bypassed within the gateway. Although the packet filtering and classification module was deployed at the exit from the IP layer on the incoming side, one can still examine and classify packets using the TCP header and parts of the payload.
- Packets were forwarded from the incoming port/interface on the active NIC to the selected

server via the service side port/interface without any packet copying between the two interfaces.

- The load balancer within the gateway selected a server for the admitted request using a simple round-robin scheduling policy. However, alternative scheduling policies using the server feedback. Information can also be used [17, 18].
- To keep overall processing delays small, the traditional interrupt-driven packet-receiving interface was replaced by a polled mode of operation. A real-time task was created to poll the input port for an incoming packet. When a packet arrived, the packet classification, filtering and forwarding functions were completed before resuming the polling.

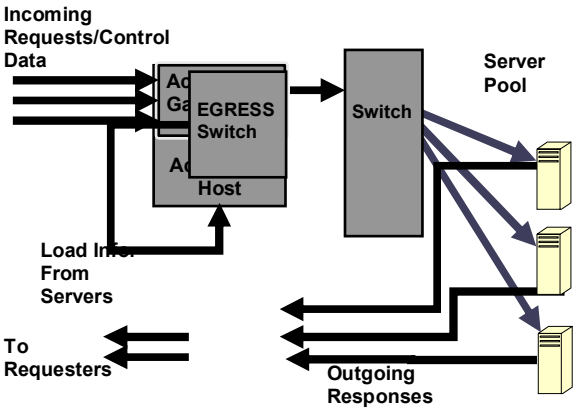


Figure 1: Active NIC enabled GridFTP Server Architecture.

Additional functions are provided on the machine hosting the intelligent gateway to quickly update the packet filtering rules and to read out packet classification statistics.

A proprietary library is used for communicating from the host PC to the PMC 694; this interface is not critical to the performance of our scheme.

The server agents gather the information used to classify incoming packets as admissible or non-admissible on a regular basis and pass this information to the control agent on the active NIC's host. The final decision for admission control and the dynamic alteration of the packet-filtering rule at the gateway is left to the control agent.

The data structure used to keep track of the IP addresses of hosts requesting a GridFTP transfer is a PATRICIA trie, which is extremely efficient for inserting and searching such information [5]. The control agent, the server agent and the active NIC all use this data structure. The IP addresses of the clients constitute the keys in this data structure. Each entry has a time stamp for last access time. Entries are aged according to this time stamp, and eventually removed from the data structure when the last access time becomes older than one hour.

4. Admission Control Policies

To implement admission control policies for the GridFTP server, the intelligent gateway classifies requesting hosts by their source IP addresses into the following categories:

- **Green:** These are hosts that are currently in the middle of a GridFTP transfer. Our aim is to keep servicing these addresses regardless of the DoS attacks and loading caused by (non-GridFTP) services. This class has a dynamic nature and has to be updated regularly.
- **Unknown:** These are the hosts that have not used the GridFTP server within the finite history of server logs.
- **Preferred:** This optional class of hosts is specific to the server. The server can choose the set of preferred hosts that request file transfers based on the GridFTP authentication information, the host's domain, or any other criteria. Preferred hosts can also be specified through a static list.

After classifying the requesting hosts into groups, the control agent transfers the corresponding filter rule updates to the active NIC gateway. The load on the server and the number of half-open connections are the main criterion to decide what packets are allowed to enter the server. We considered two types of loading information for each server machine in the pool: CPU load and I/O load. CPU load can be measured by monitoring CPU utilization and the I/O load is measured by monitoring number of I/O interrupts per second, and number of block operations done per second.

4.1 Admission Control Policy for Coping with Server Overloads

When a GridFTP host node provides other services also, we need to have policies that allow the GridFTP services to remain stable despite loading on the server caused by these other services. We have considered two different types of loading on the server: (a) "compute" loading caused by the execution of scripts (such as cgi) that mostly consume CPU resources, and (b) I/O loading caused by the file I/O accesses made by standard services (such as http).

The admission control policy implemented in this case requires the server agents to monitor the load level on their respective server machine. When the loading crosses a threshold level of L, the machine is considered to be heavily loaded and the server agent notifies the intelligent gateway to perform dynamic load balancing of the non-GridFTP requests at the gateway. As new

non-GridFTP requests arrive, they are preferentially directed to the machines that are not heavily loaded.

For the results reported later, we used L as 0.8 (that is 80%) of the peak compute or I/O load.

5. Scalability

The proposed solution was evaluated using a single active NIC gateway controlling access to a small server pool. A single gateway device may not be able to cope with the processing requirements for traffic directed at large server pools. The potential bottlenecks are the storage needed for the green or preferred class of IP addresses, the processing overhead for packet filtering and collection of statistical data at the gateway, and the performance of the gateway's network interfaces.

The proposed solution can be scaled up to meet the processing needs for protecting large-scale server pools as follows:

- Several active NIC gateways operating in parallel can be used. Multiple active NICs can be hosted on the common PCI bus of a single host. The PCI driver for the active NICs need to be modified to support the "broadcast" of status information to all cards on a common PCI bus. This can be easily done by passing on the status information via a common memory-mapped buffer in the RAM of the host of the gateways. Additionally, a front-end load balancing switch can be used to direct the incoming server traffic to a specific gateway. Alternative configurations that using independent gateway hosts can be used to improve overall reliability.
- The memory requirements for the IP address classes on each gateway can be prohibitive as the number of attacking clients increase. A solution here will be to use a dynamic data structure like MULTOPS [7]. This will limit the storage usage and switch dynamically between maintaining information on a per IP address basis or on a subnet address basis depending on the amount of traffic data and offered traffic volume. We are currently implementing this alternative on our prototype.
- The processing capabilities on the active NIC gateways continue to increase steadily, and this offers some relief for the solutions targeting larger scale systems and traffic volumes. The emerging generation of cards from Ramix has such capabilities (dual or quad 1 GBits/secs interfaces, faster CPU, additional RAM etc.).

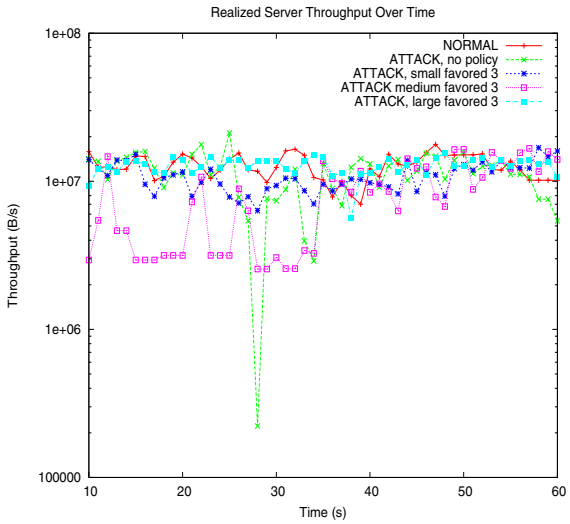
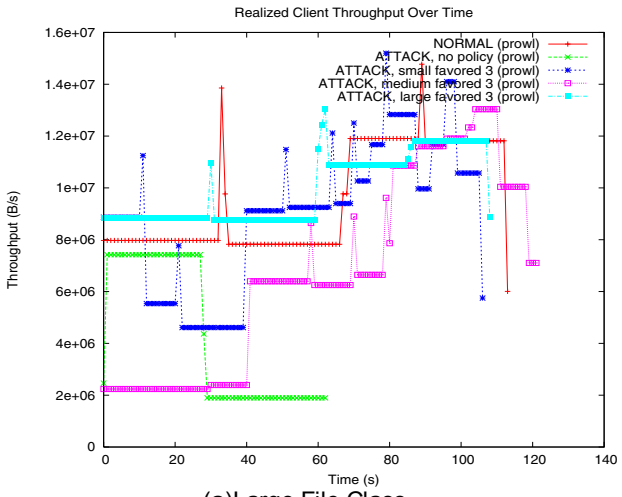


Figure 2: shows the server throughput under different conditions. We ran the test under five different conditions on our network: (1) normal conditions (Base Case) when there is no load on the server, (2) Base: Under heavy load and the server is not protected against attacks, (3) when small file class requests are favored and (4) when large file class requests are favored, (5) when the medium file class requests are favored.

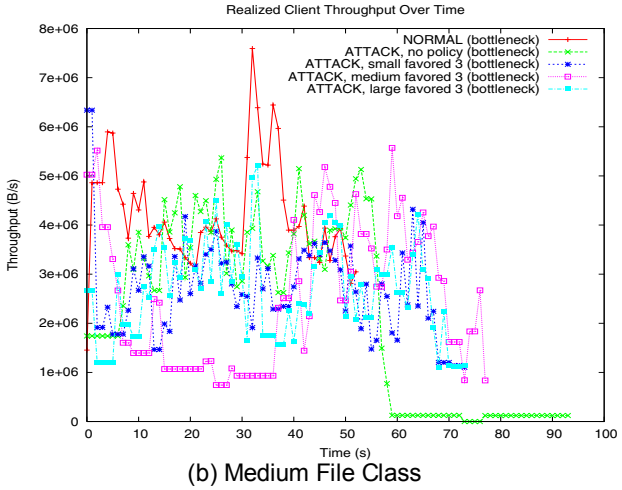
An alternative to using active NIC gateways is to use network processors. We have an ongoing effort using the Intel IXP2400 NPU.

6. Experimental Results

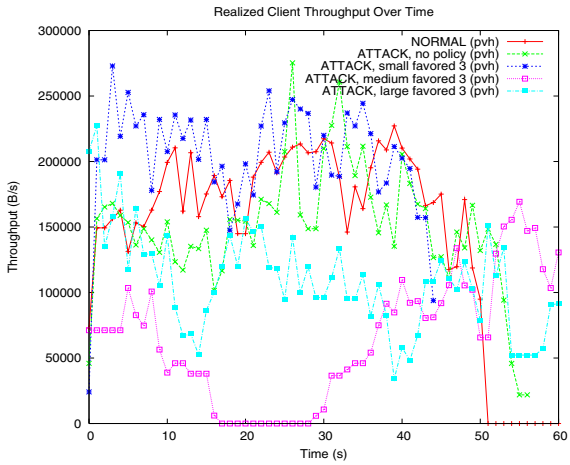
The servers used for the evaluation system are Pentium IV PCs running a modified version of Linux kernel 2.4.18. We used two switches and constructed two subnets with 100 Mbits/sec Ethernet. The server pool constitutes one subnet and the client GridFTP machines are from another subnet (representing the outside world). The active NIC is positioned as a gateway with its two ports connected to the two subnets. Multiple addresses are assigned to network interfaces of client and load machines to extend the IP range. For each request, clients are able to select an IP assigned to the interface.



(a)Large File Class



(b) Medium File Class



(c) Small File Class

Figure 3 (a), (b), and (c): shows the effective bandwidth of each file class under different conditions for (a) large file class (b) medium file class and (c) small file class. We ran the test under four different conditions on our network: (1) normal conditions (Base Case) when there is no load on the server, (2) Base: Under heavy load and the server is not protected against attacks, (3) when small file class requests are favored and (4) when large file class requests are favored (5) when the medium file class requests are favored.

Requestors connect directly to the gateway, which in turn forwards packets to the server, rewriting packet headers, or potentially dropping packets when necessary. Connections are normal GridFTP connections requesting transfers of files be sent in four parallel data connections in the "Extended Block Mode". The GridFTP requestor and server are both the official Globus 4.0.1 `globus-url-copy` and gridftp daemon, while the active NIC gateway software is written in C.

Our tests consist of running requestor scripts repeatedly under the different experimental conditions. A client script connects, authenticates, sets up the connection parameters, and requests the transfer of a file. The server transfers the file as requested after which the client quits and reports connection latency (time from initiation of the connection until transfer begins), total time (time from initiation of the connection until transfer completes). To gather performance results, the client script is run several times on each of the client machines.

We designed and implemented an admission request policy that classifies files served by the GridFTP servers based on the file sizes. We defined three different classes of files: (1) small, (2) medium, and (3) large. The main motivation of applying the admission control policies based on file classes is to maintain acceptable performance under heavy load conditions for specific classes of client requests. Assuming that the clients expect a bound service time based on the file size, a transfer should finish within a time-out period whose duration is based on the expected service time. In order to determine a time-out period for each file class, we measured the average transfer times for a wide range of file sizes within each class during lightly-loaded conditions and multiplied this duration by a factor of K to obtain the expected service time under heavy loading. A value of K=2 was found to be useful for the scenarios studied. In order to minimize the number of time-outs during intervals of heavy server load, we use the Active NIC gateway to allow/deny requests based on the class of the requested file. A quota is set for requests belonging to each class (based on the file size). The quotas for each class were determined through experimentation. A given class' quota is determined as the number of requests that can be fulfilled in a certain window without causing time-outs. The gateway applies

access control policies based on the quotas of each file class: requests submitted for a particular class of files are dropped at the gateway when the quota for that class is (momentarily) exceeded. Figure 2 shows how the server throughput varies under several different conditions. During an attack or when the server is heavily loaded, the server throughput, as perceived by the clients, varies quite dramatically. By using policies favoring a certain class, we can give guaranteed performance to the requests in that class. In Figure 2, we can see that the class-quota based admission control policy ensure that the server throughput is more reliable and predictable.

We used 48 KB sized test files for small files, 2 MB sized test files for medium files, and 64 MB sized test files for large files. Our tests consist of running requestor scripts (calling globus-url-copy) repeatedly under different experimental conditions. We measure the time to run globus-url-copy to collect the effective client throughput data. We used three admission control policies in which a certain file class, chosen statically, is favored. (The file classes are defined by the size of the file in the GridFTP request, as defined earlier.) However the classes can be constructed by any type of criteria such as content, type, modification time, etc. As can be seen on Figure 3 (a), (b) and (c), the policies favoring each class give reliable performance, timeliness, and predictability for requests in the favored class. However, predictability of requests within a class drops when the admission control policies do not favor that class.

.

7. Related Work and Conclusions

Our work involves providing differentiated service to GridFTP. A substantial amount of related work has been developed in support of these techniques for web servers, though little, as yet, as been developed for the GridFTP service.

There is a plethora of work in supporting differentiated services on web servers. Some examples follow. Operating System facilities for supporting differentiated services are explored in [8]. The work of [9] uses transcoding technique to vary content resolution/quality to meet QoS needs on a per-client basis. The work of [10] proposes a technique for dynamically partitioning a server pool into classes and assigning servers to a specific class. In [11], session-based relationships among http requests are used to device traffic conformation functions that are used for resource allocation to limit server overloading. In [12], an adaptive technique for determining the number of servers needed to service requests with specific targets is introduced and evaluated through simulations against optimal configurations. All of these techniques allow packets to enter the server and then are differentiated within the server. This implies that the servers take a performance hit to examine an incoming request and then either rejecting it or delaying its service. The performance hit can be substantial under flash crowd traffic or when a DoS attack is in progress. We filter low priority requests at the gateway, freeing up the server resources to perform the services for the high priority classes. Our work in this paper represents a significantly enhanced version of the results and policies introduced in [15]. We have also included new policies, their rationale, and experimental results to quantify the gains from these policies.

A complete solution for dealing with DDoS attacks, by necessity has to be distributed and requires the coordination of several entities on the network. Since many types of DDoS attacks use spoofed IP source addresses, a rather naive prevention mechanism is to use simple egress filtering - filters in switches take the traffic out of a subnet to ensure that the source addresses of packets going out corresponds to valid host IP addresses within the subnet. Although it sounds simple, this solution is not practical - the large majority of subnets do not have egress routers with this capability; neither will this scheme be of any use unless the filters are configured correctly. IP-traceback - tracing packets back to the source - and similar techniques can be used to trace a large and unusual influx of packets from a specific port (or set of ports). With the use of traceback, controlling or limiting packet flow is a more sophisticated and distributed mechanism for coping with DDoS [13, 6]. Mazu networks offers a commercial product for defending against DDoS attacks, that relies on traffic flow monitoring [14]; some other vendors offer similar products as well. Other distributed solutions for coping with DDoS are possible, including the use of trusted network components. Until these distributed solutions are standardized and widely adopted, servers have to deploy local solutions to protect themselves. Traceback and similar solutions (based solely on the monitoring of packet flow towards the servers) are generally incapable of dealing with load attacks, which do not always manifest themselves as a sudden burst of unusually heavy traffic. Furthermore, to detect such attacks, the en-route routers need to have the capability of examining the payload in the requests. Load attacks can be better dealt with by using the actual loading information at the servers. Distributing such loading information to en-route routers can be time consuming and complex - and, perhaps, practically infeasible. Solutions implemented on gateways closer to the server that incorporate the servers' loading information to perform dynamic packet filtering, as proposed in this paper, appear to be more attractive in coping with DDoS attacks.

We presented an intelligent gateway based solution for supporting differentiated service for a GridFTP server that preferentially services known clients under DDoS attack, and actively manages server load

distribution based on the servers' systems' statistics. The capabilities of the active NIC-based gateway permit a dynamic mechanism to react intelligently to a denial of service attack, as well as external load on the servers to be efficiently implemented. The packet filtering rules at the gateway are dynamically altered based on the incoming packet rate and dynamic loading information periodically collected from each of the servers in the server pool.

We demonstrated how a flexible admission control policy can be implemented at the gateway to provide differentiated service to various client classes. The clients are classified based on whether or not they are known to the server. We also showed that the desired degree of real-time performance (bounded response time) can be guaranteed even under heavy server loading and denial of service attack by choosing rate limits appropriately. The proposed system is scalable, flexible and provides continuous service of the servers by performing dynamic request rate limiting at the active NIC-based gateway.

8. References

[1] W. Allcock (editor), "GridFTP Protocol Specification", *Global Grid Forum Recommendation GFD.20*, March 2003.

[2] W. Allcock, et al. "Data Management and Transfer in High Performance Computational Grid Environments". *Parallel Computing Journal*, Vol. 28 (5), May 2002, pp. 749-771

[3] Ramix Inc., "Intelligent Ethernet Adapter Product Guide", available at: http://www.ramix.com

[4] A. Begel, S. McCanne and S.L. Graham, "BPF+: Exploiting Global Data flow Optimization in a generalized Packet Filter Architecture", in *Proc. of SIGCOMM 99*, 1999

[5] D.R. Morrison. "Patricia-practical algorithm to retrieve information coded in alphanumeric". *Journal of ACM*, 15(4):514--534, Jan 1968.

[6] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical Network Support for IP Traceback", in *Proceedings of ACM SIGCOMM'2000*, August 2000.

[7] T.M. Gil, and M. Poletteo, "MULTOPS: a data-structure for bandwidth attack detection", *Proceedings of USENIX Security Symposium'2001*, August 2001.

[8] M. Aron, "Differentiated and Predictable Quality of Service in Web Server Systems", *PhD Dissertation, Rice University*, 2000. Available at: http://cs-tr.cs.rice.edu/Dienst/UI/2.0/Describe/ncstrl.rice_cs/TR00-369.

[9] S. Chandra, C.S. Ellis, and A. Vahdat, "Differentiated Multimedia Web Services Using Quality Aware Transcoding", *Proc. of Infocom 2000*, 2000.

[10] J. Zhang, et al, "A New Mechanism for Supporting Differentiated Services in Cluster-based Network Servers", *Proc. of MASCOTS 2002*, 2002.

[11] H. Chen, and P. Mohapatra, "Overload control in QoS-aware web servers", *Computer Networks: The International Journal of Computer and Telecommunications Networking*, Vol 43, No. 1, May 2003.

[12] S. Ranjan, et al, "QoS-Driven Server Migration for Internet Data Centers", in *Proc. Intl. Workshop on QoS 2002*, 2002.

[13] S.M. Bellovin, "ICMP Traceback Messages", *Internet Draft: draftbellovin- itrace-00.txt*, March 2000.

[14] Mazu Networks, *Enforcer*, Product information and white papers at: http://www.mazunetworks.com, 2004.

[15] O. Demir, M. R. Head, K. Ghose, and M. Govindaraju, "Protecting Grid Data Transfer Services with Active Network Interfaces," In proceedings of *Grid 2005 - 6th IEEE/ACM International Workshop on Grid Computing*, pp: 9-16, Seattle WA, November 2005.

[16] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, I. Foster.The Globus Striped GridFTP Framework and Server. *Proceedings of Super Computing 2005 (SC05)* , November 2005.

[17] O. Kremier, J. Kramer. Methodical analysis of adaptive load sharing algorithms. *IEEE Trans. Parall. Distrib. Syst.* 3, 6 (Nov.), 747–760.1992

[18] N. G. Shivaratri, P. Krueger, M. Singal. Load distributing for locally distributed systems. *IEEE Computer 25*, 12 (Dec.), 33–44. 1992