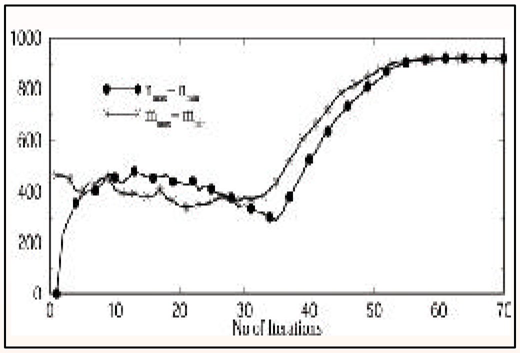
5

High Performance Computing and Networking (HPCN)

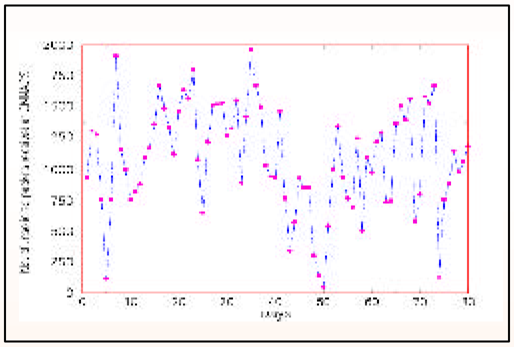
Mathematical modelling and computer simulation in the fields of ocean, atmosphere, earth science and engineering involve computational tasks which can only be provided by High Performance Computing(HPC). The need for computational power, measured in terms of Giga Floating Point Operations per Seconds (FLOPS), grows exponentially with every bit of increase in the complexity of problem. C-MMACS today has one of the best computing facilities in the country.

Highlights

The year 2004-05 has been a year of growth and expansion for HPCN both in terms of computing resource and areas of research. A substantial enhancement of C-MMACS computing platform took place through installation of an ALTIX-350 12-processor system. The two prominent research areas under HPCN: Network Security and Cryptography, provided new results in these important areas.



(n_{max}-n_{min}) and (m_{max}-m_{min}) with k for 250,000 sets [n =10000-bits & l =16-bits]



Unsolicited packets received at C-MMACS. Day corresponds to Jan. 22, 2005

Inside

5.1 Towards a Measurement Framework for DoS Attack Prevalence

Internet has become a platform where unethical activities such as Denial-of-Service (DoS) attacks, port-scanning and automatic worm spreading etc. are being practiced massively. "Backscatter analysis", a technique by which unsolicited packets are collected and analyzed to gain insight into the nature of malicious activities on the Internet, was applied for the first time by David Moore, Cooperative Association for Internet Data Analysis (CAIDA). The main objective of our work is to design, develop and deploy a measurement system, similar to that of Moore's, for in-depth understanding of the nature of malicious activities including DoS attacks.

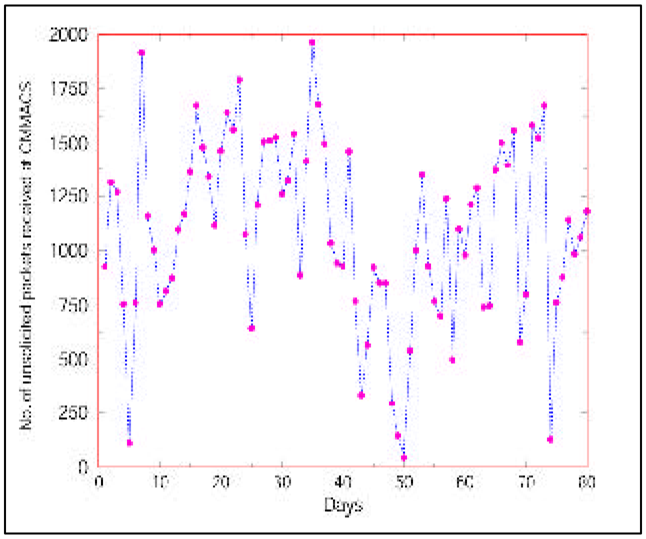


Fig 5.1 Unsolicited packets received at C-MMACS. Day 1 corresponds to Jan. 22, 2005

Backscatter analysis is based on a key factor that attackers normally spoof source address of each attack packet with a random source address picked from a list of 2^{32} possible IP address space. Consider a case where an attacker, A, sends an attack packet, P, to a target, T, by spoofing the source address of an innocent machine, I. Communication protocols are designed to respond appropriately to each incoming packet, and the address to send the response is picked from the source address of the incoming packet. Since the source address of P belongs to I, the response from T, which is referred as R, goes to I.

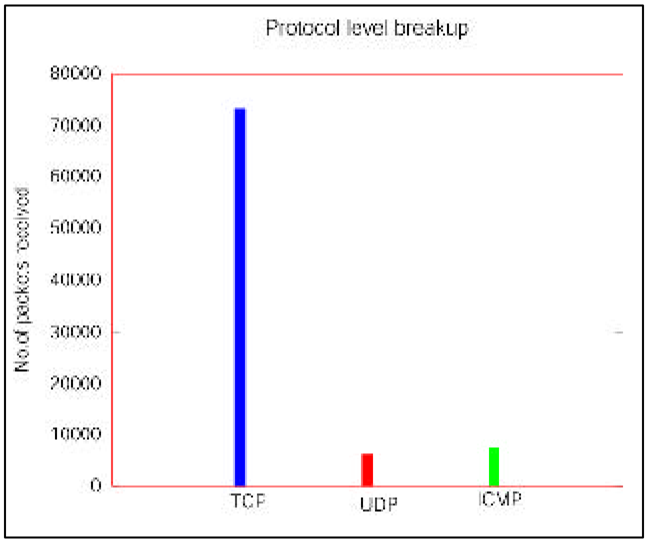


Fig 5.2 Protocol level breakup of unsolicited Packet

Such responses are called backscatters. While the source address of the response packet R can reveal the attack target, other contents of R can be used to derive additional information about the attack such as its intensity, duration and protocol used etc.

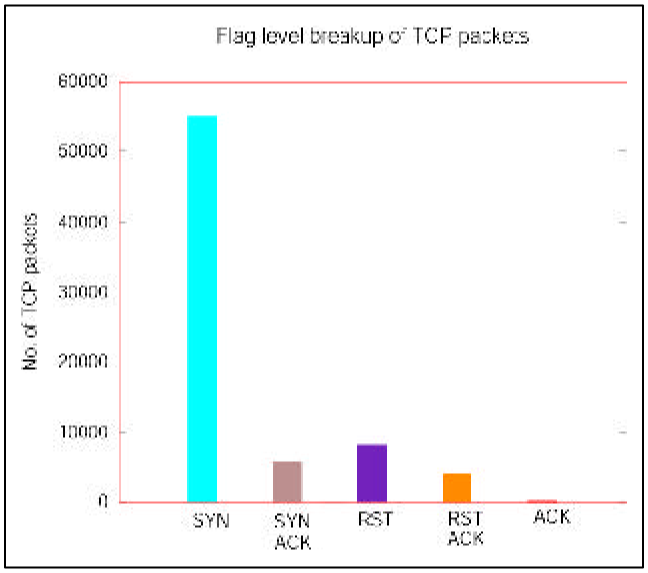


Fig 5.3 Flag level breakup of unsolicited TCP packets

We have initiated the development of a tool to collect and inspect each inbound packet from the internet and classify them as normal or unsolicited packet. The list of unsolicited packets includes backscatters as well as other malicious packets resulting from port-scanning etc. A preliminary version of this tool had been deployed at C-MMACS since January 22, 2005, and it collected more than 86000 unsolicited packets during 80 days. Fig 5.1 shows the breakup of no. of unsolicited packets received in each day. Our preliminary analysis reveals that out of 86720 unsolicited packets, 73224 (84.43%) were TCP packets. We also observed that 6.89 % were UDP, 8.67 % were ICMP and 6 (negligibly small) were unknown packets. Fig 5.2 provides protocol level breakup of number of packets received. We further sub-divided TCP packets based on the TCP flags set in each packet. As shown in Fig 5.3, we received 54977 (75.08%) TCP packets with SYN ON, 5782 (7.90%) with SYN & ACK ON, 8250 (11.27%) with RST ON and finally 4017 (5.48%) with RST & ACK ON. High value of unsolicited SYN packets is likely an indication of heavy port-scanning activities of C-MMACS address space. This reveals that external agencies are regularly using automated port-scanning tools to identify open TCP ports at C-MMACS so that malicious code insertion can be attempted. The SYN & ACK packets are the backscatters of the well known SYN flood attack happening somewhere else on the Internet.

V Anil Kumar, G K Patra and R P Thangavelu

5.2 An Experimental Study on Practicability of Acknowledgment Spoofing in TCP Servers

Our previous work had identified the possibility of a malicious TCP receiver constituting Denial-of-Service (DoS) attack scenarios by exploiting the features of TCP congestion control process, and demonstrated the same through simulation. The focus of the current work is to explore how widely deployed Operating Systems (OS) respond to maliciously spoofed duplicate acknowledgements (ACKs). We implemented necessary modifications to the TCP source code in Redhat Linux to imitate an attacker who spoofs large number of duplicate ACKs. Using the modified kernel, we tested three popular UNIX OSs, SGI IRIX 6.5, IBM AIX 4.3 and SUN Solaris 9.0, in a test bed environment consisting of the above three servers on one side of an IP router and the modified TCP receiver on the other side of the router. We observed that all the three OSs can be forced to eject large number of new data packets in response to spoofed duplicate ACKs. The results of these experiments are summarized in Fig 5.4, Fig 5.5 and Fig 5.6.

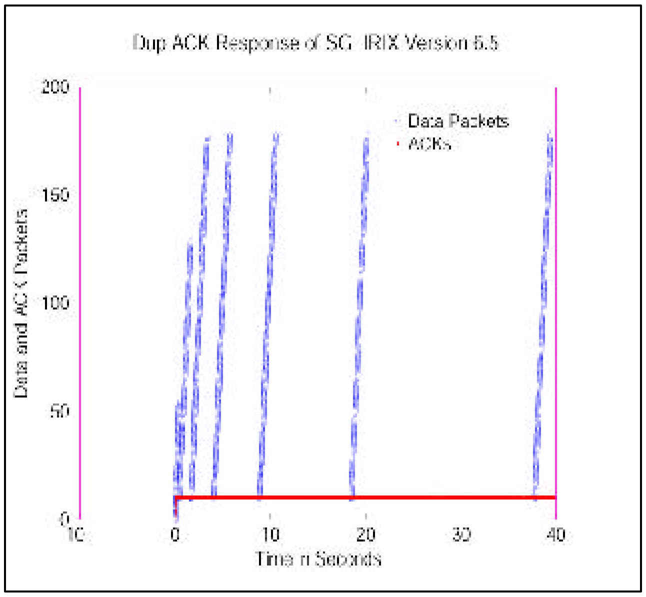


Fig 5.4 Data packets ejected by IRIX TCP sender in response to spoofed duplicate ACKs

In all the three cases, the receiver establishes a TCP connection with the server and then request for a file transfer from the server to the receiver. The receiver behaves normally and acknowledges packets up to and including packet no. 9 and then stops generating normal ACKs, but sends large number of duplicate ACKs to packet number 9. The sender sends new data packets in response to spoofed duplicate ACKs. The number of packets sent by the server is same as the receiver window advertised by the receiver at the time of establishing the connection. The server, upon retransmission timeouts, retransmits packet no. 10 as it never gets an acknowledgement for packet no. 10. The duplicate ACKs that reach the sender after each retransmission of packet no. 10 force the server to perform repeated fast recovery and this generates additional bursts (only in the case of Fig 5.4 and Fig 5.5).

An important observation is that though these servers are implemented based on standard specifications and guidelines, there exist deviations in their behaviour. For example, while IRIX and AIX repeat the ejection of same sequence of packets after each retransmission timeouts, Solaris does

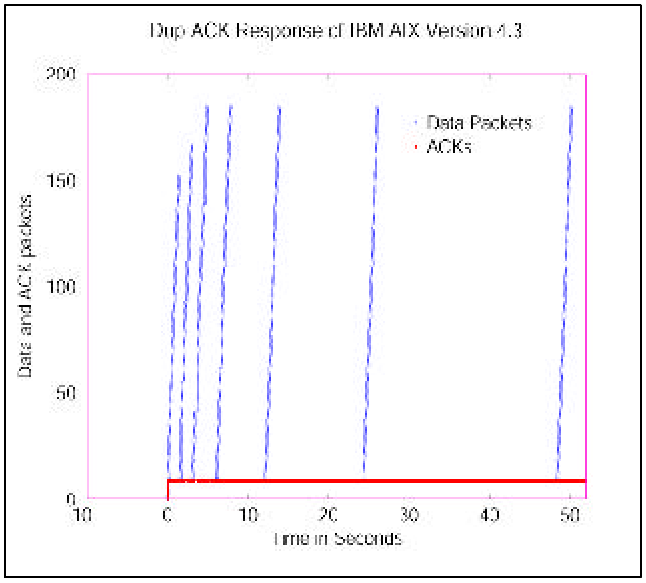


Fig 5.5 Data packets ejected by AIX TCP sender in response to spoofed duplicate ACKs

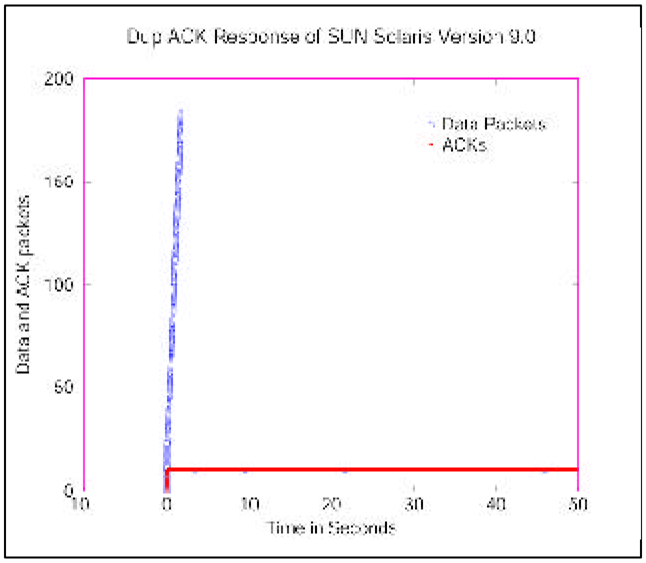


Fig 5.6 Data packets ejected by Solaris TCP sender in response to spoofed duplicate ACKs

not follow this. Such deviations are not so easily detectable and may get exposed only through this sort of experiments. We propose to include more operating systems like Microsoft Windows XP, FreeBSD and Linux to our test bed and conduct experiments on these platforms also. We also plan to evaluate how these operating systems respond to optimistic ACK spoofing, another exploitable feature of TCP congestion control process.

V Anil Kumar, D Sisalem and Pradeep

5.3 Multiparty Secure Key Exchange Algorithm using Neural Cryptography

Secret key exchange protocol based on neural network has been designed such that an opponent who tries to obtain the key is unable to do so. As an extension of the concept of supervised and unsupervised learning, mutual learning of more than two neural networks exhibits this novel phenomenon, where the networks synchronize to a state with identical time-dependent weights. The main motivation of this work is to have secure communication for distributed applications like collaborations, video conferencing, replicating servers and grid computing. The neural network topology determines the number of inputs, the number of outputs, the number of hidden layers and the learning rule. Three multilayer feed forward networks (A, B and C) with n input units and k hidden units are considered here for explanation.

The output of each hidden layer unit is denoted by s_i^A, s_i^B and s_i^C depending on the network belonging to A, B or C. The synaptic weights are represented by n-dimensional weight vectors *'W' [-L, L]* and n-dimensional input binary vectors represented by 'X' [-1, 1]. The inner product computation yields the values of the hidden bits *s [= sign(w.x)]*, which are combined to form an output bit t for the entire network as shown below

$$t^{A/B/C} = s_1^{A/B} s_2^{A/B/C} s_3^{A/B/C} \qquad (1)$$

The output bits are exchanged and used for the mutual learning and synchronization. At each training step, the machines A, B, C receives identical input vectors X. In the training process, if all output bits are equal i.e $\tau^A = \tau^B = \tau^c = \tau$ the weights can be changed. In this case, only the hidden unit 'i' whose output bit s_i is identical to the actual output bit τ changes its weights using the Hebbian learning rule as follows

$$w_i^{A/B/C}(t+1) = w_i^{A/B/C}(t) + x_i \qquad (2)$$

If the learning step pushes any component weight out of the interval [-L, L] then the component is replaced by ±L.

The attacking strategy is intended to prove the protocol's robustness. The attacker's design is

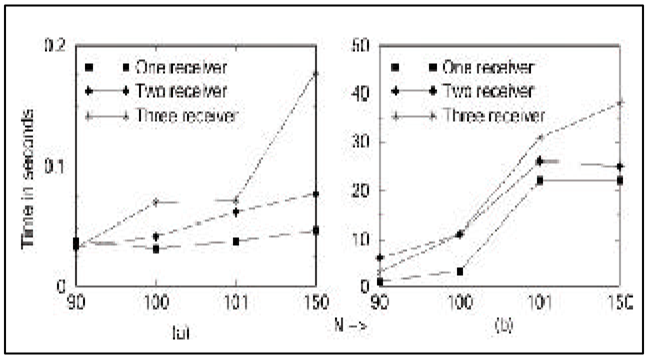


Fig 5.7 (a) Synchronization (b) Learning time for two, three and four party communication for different numbers of input units

similar to that of the communicating parties. It uses the same algorithm as the sender and the receiver. The neural network topology and the Hebbian learning rule are also the same. The only difference lies in the fact that the attacker modifies its weights only when the outputs of the two actually communicating networks are equal. Then, the weights are changed only for those hidden layer units whose outputs match the output bits of the communicating parties (instead of matching the output bit of the attacker's own network). The learning time required for the attacker to synchronize with the sender and the receiver is calculated by allowing the sender, receiver and attacker to wait until all the three synchronize. The attacker is not expected to learn the weights before the communicating parties synchronize, because of two reasons. Considering the case where, $\tau_A=\tau_B=\tau_c=1$, there are four possible configurations of hidden units in each network (+1,+1,+1), (+1,-1,-1), (-1,+1,-1) and (-1,-1,+1). The attacker and the communicating parties may have different configurations and thus modify different sets of weights. This may result in the network weights of the attacker and the communicating parties getting separated by such errors made by the attacker. The second reason is that the attacker cannot influence the communicating parties. Fig 5.7 shows the difference in the synchronization and learning time for different numbers of input units. Further, Similar results are obtained by changing the numbers of hidden units and range of discrete weights.

G K Patra, Thahir Ali, V Anil Kumar and R P Thangavelu

5.4 Statistical Analysis of Random Key Generation Algorithm

To demonstrate the robustness of the random key generation algorithm (where two authenticated communicating parties can generate a common secret key from two random binary strings by publicly using well known Error Correction and Elimination mechanisms), a statistical analysis was carried out of a large number of experiments. Initially we generated 250,000 sets of binary strings of size 10,000 bits. Let $P(n,m,k)$ be the probability of distribution of n and m at the k^{th} iteration. It is quite clear that $P(n,m,k)$ is governed by a Markov process, since the current probability depends only on the preceding probability. Then the probability density function of n and m of the k^{th} distribution can be represented as

$$P(n,m,k) = P(n,m,k/n',m',k-1) \times P(n',m',k-1)$$

where $P(n, m, k/n', m', k-1)$ is the conditional probability density function of $n, m, k, n', m', k-1$ which transforms the probability distribution from one iteration to the next iteration. We followed the evolution of $d(A,B)$ (Hamming distance $d(A, B)$ between A and B is defined as the number of locations in which A and B do not match) and also the distribution $P(n,m,k)$ as a function of k. At each k, we divide the range of values of n and m into a fixed number of bins and we plotted the two dimensional histogram of the frequency of the occurrence of a given n, and m within $n \pm \frac{(n_{max} - n_{min})}{2 \times 20}$.

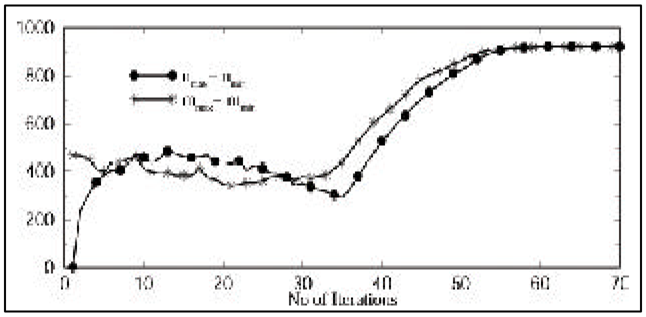


Fig 5.8 $(n_{max}-n_{min})$ and $(m_{max}-m_{min})$ with k for 250,000 sets [n =10000-bits & l=16-bits]

This yields an estimate of $P(n, m, k)$ at each k. We found that the form of $P(n, m, k)$ when normalized as above did not change significantly (at least visually) with k. This is reasonable since the locations of the bits that match and are connected, at each iteration are likely to be random. We note that we can get an estimate of the conditional probability density function $P(n, m, k/n', m', k-1)$ from the estimates of $P(n, m, k)$ and $P(n', m', k-1)$. To demonstrate that the $d(A,B)$ goes to zero for all initial conditions considered in our numerical experiments, we have plotted $(n_{max}-n_{min})$ and $(m_{max}-m_{min})$ as a function of 'k'. We note that in our experiments $(n_{max}-n_{min})$ converges to $(m_{max}-m_{min})$ within 60 iterations indicating that in all the sets considered n converges to m as k increases (shown in Fig 5.8).

G K Patra, V Anil Kumar, R P Thangavelu and T R Ramamohan

5.5 High Performance Computing Resources

A significant improvement in the computing power of C-MMACS was achieved with the installation and commissioning of an SGI Altix 350 server configured with 12 numbers of Itanium2 processors, 60 GB of main memory and 1 TB of fibre channel RAID storage. This server runs on 64-bit Linux operating system and has SGI ProPak & utilities, Intel Fortran & C++ compilers, Intel VTune performance analyzer, and MPI libraries for parallel computing. Key applications on this server are GFDL MOM, LMD GCM, and ABAQUS. Porting and validation of applications such as GFDL MOM, LMD GCM were carried out on Altix 350.

The Origin 3000 and Altix 350 servers were maintained with an uptime efficiency of 99.8% during the year while the network infrastructure was maintained with 100% uptime efficiency. The cumulative utilisation of the 24 processor Origin 3000 server so far has exceeded 4,50,000 CPU hours.

An additional 2 TB RAID storage was added to the Origin 3000 server to provide adequate storage space for HPC users. The Origin 3000 server continues to be the most preferred platform for many users and the same is being upgraded to Origin 3900 with 32 processors (MIPS R16000 @ 1 GHz, 16 MB cache) and the upgraded system will deliver 166% more performance than the present one. The upgrade is expected to be completed in June 2005. Enhancement of the Altix 350 server has been taken up and a Purchase Order has been placed for additional 20 processors, 40 GB memory and 1 TB RAID storage for the Altix 350 server. With this enhancement, the total computing power on the HPC servers will exceed 250 Gflops.

A detailed performance evaluation of HPC systems using LMD GCM and GFDL MOM4 has been carried out to identify suitable computer systems to meet the current and upcoming computing needs. The following high-end systems were evaluated.

1 Cray XD1 server based on Opteron processor
2 HP Integrity servers based on the Itanium2 processor
3 IBM 570 and 720 servers based on Power5 processor
4 SGI Altix 3700 server based on the Itanium2 processor
5 Sun Fire V20z server based on Opteron processor

Based on the performance evaluation, the Cray XD1 and the SGI Altix 3700 servers were short listed. Two numbers of SGI Altix 3700 BX2 servers with 24 processors each have been chosen based on the price / performance. These two systems together will deliver a computing power of 300 Gflops.

5.5.1 *Storage Virtualisation Solution*

Recognising the need for long term storage and retrieval of scientific data in a user friendly and transparent manner, an effort has been initiated to set up a Storage Area Network (SAN) based high performance 3-tiered storage virtualisation solution leading to efficient data life cycle management. A detailed technical study of currently available storage solutions and the trends in storage technologies was carried out by the HPC team. Based on the technical study, the architecture and basic design for the storage virtualisation solution have been completed and tenders were invited. Evaluation of technical bids to identify suitable solutions is in progress.

5.5.2 Other Hardware & Software Enhancements

Twenty numbers of Pentium4 processor based Linux workstations were installed and configured on the LAN. Many users have switched to Linux environment from Windows. In order to provide enhanced network services and midrange application services, a set of Xeon and Pentium4 based servers are being procured. A web based mail service has been introduced enabling users to access their e-mails from anywhere in the Internet.

Application software such as CFD-ACE+, NISA, IDL, GAMIT/GLOBK and Intel compilers have been upgraded. Totalview debugger has been installed for source level debugging of parallel codes. PBSpro workload management software has been installed for efficient management and utilisation of computing resources. Under the CSIR network project, ABAQUS software for finite element analysis has been procured and installed on the HPC servers, which is available to all participating laboratories in the network project. A current list of hardware and software in the computing environment can be accessed from the C-MMACS website http://www.cmmacs.ernet.in.

5.5.3 Other Services

Computing services were provided to the practical sessions of the Ocean Modelling and GPS courses that were conducted in October and November 2004 respectively. The computing requirements for the practical sessions of the Ocean Modelling course were very demanding in nature and were successfully managed using the PBSpro software. Further, a software environment that is similar to the one in GFDL was created for ease of use by both the faculty, members and the participants. Further, a large number of students from various academic institutions have availed the computing services at C-MMACS.

R P Thangavelu, V Anil Kumar, G K Patra and N Prabhu