INCREASED AVAILABILITY FOR NETWORKED SOFTWARE SERVICES

Peter Clutterbuck
Faculty of Business
University of the Sunshine Coast
Maroochydore, Queensland, Australia
Email: pclutter@usc.edu.au

ABSTRACT

The availability of networked software services is an increasingly important quality of service issue. A service user gauges service availability in terms of initial access time and subsequent service delivery time. This paper develops – from a user perspective – a quality of service framework for the availability of networked software services. The paper utilises this quality of service framework to suggest the functional characteristics of a new transport layer software specification that would facilitate the increased availability of networked software services.

Keywords

Availability, Quality of Service, Network, Services, Denial of service.

INTRODUCTION

Business computing is becoming more networked oriented. Universities allow candidates from faraway places to earn qualifications online. Service industries encourage customers to track delivery progress from their very own home of office PCs. Companies are taking their entire supply chains to the web. Networked software services are increasingly important within this business-computing paradigm. Name, file, mail and web services are examples of that software set considered essential for efficient business operation. The availability of these services is an increasingly important quality of service issue to all major stakeholders associated with the network. IDC (1998) reports that 99.999% availability equals five minutes of downtime per year, whilst 97% availability equals a loss of 263 hours per year. Whilst hardware availability has received significant research and development focus, no broadly accepted solutions exist for the comprehensive, automated management of networked software service availability.

Software availability research to date has produced two availability management paradigms. Initial availability studies described in Gligor (1983), Nyberg (1988), Che *et al* (1992), Amoroso (1990a, 1993b), Bacic *et al* (1991), and Millen (1992) pursued the goal of denial of service protection. These studies focused upon the services and resources directly managed by a single operating system within a multi-user, multi-tasking computing environment. Denial of service protection aimed to provide to all authorised users guaranteed rates of progress in completing their current computing task. User rates of progress were theoretically quantifiable by limiting the upper number of users operating within the system, and by establishing for each service or resource the maximum time interval for which it could be held for exclusive use by any one user. A second availability paradigm has subsequently developed with a focus upon services within networks and is described in Karamanolis *et al* (1995), Cristian *et al* (1994) and most completely in Pfister (1998). This paradigm has been shaped by two interlinked network characteristics: the frequency of hardware, system or application crashes and the resource cost in reconfiguring and recovering following these crashes. The paradigm proposes a service availability strategy pivoting upon service replication across the network. The service replicates combine as a group to ensure the service always exists at some accessible and publishable point within the network. This second availability paradigm, under the label 'Clustering Architecture', is gaining some acceptance as the choice of industry for business solutions.

This paper provides a comprehensive analysis of the networked software service availability challenge. The paper opens with a review of concepts and theory provided by denial of service research and also the clustering architecture availability solution. The paper continues by defining networked-service availability and by proposing a quality of service framework to support this definition. The availability expectations of a generic service user shape this definition and framework. The paper then proposes a conceptual blueprint for transport layer functionality that would support this quality of service. A brief discussion of future work to implement the quality of service framework concludes the paper.

EXISTING AVAILABILITY THEORY

Historically availability management research has not been prolific. Considered as a body of work, the existing research spans two paradigms: denial of service protection and clustered server availability.

Denial of Service Protection

Denial of service protection was initially examined in Gligor (1983), incrementally developed in Amoroso (1990a, 1993b), Nyberg (1988), Bacic *et al* (1991), and most fully described in Millen (1992) and Leiwo et al (1997a, 1998b). Denial of service involves deliberate or accidental software service monopolisation, destruction or withholding via the usage patterns of other software users. Denial of service is a security concern and is a subset of

availability management. Denial of service protection guarantees to credentialed users service access and provision within acceptable, well defined and advertised wait times.

Service access and provision guarantees are provided by a Denial of Service Protection Base (DPB) which forms part of, or works closely with, the Trusted Computing Base (TCB) or host operating system. The DPB is tamperproof, cannot be circumvented or prevented from operating, and guarantees authorised access to services under its control. The DPB consists of *waiting time policies, user agreements, a resource allocation system,* and a *resource monitor.* Waiting time policies quantify when a user may expect service access. A Maximum Wait Time prescribes a fixed time bound within which the required service access will be provided. A Finite Wait Time specifies that required service access will eventually be provided. A Probabilistic Wait Time specifies an averaged wait time for service access. User agreements or usage policies constrain the behaviour of users when accessing the service. A resource allocation system controls the timeliness of service allocation – the holding, surrender and revocation of resources. A resource monitor implements the service-scheduling algorithm appropriate to the waiting time policy of the DPB.

The operational logic of a DPB as described in Millen (1992) pivots upon several main points. Firstly, two user agreements are stipulated:- 1) a service request cannot exceed the system capacity for that service and the request must be announced by the user upon entry to the system, and 2) all service users must engage and use the service for a minimum time before user voluntary deactivation (this ensures users progress toward task completion / service completion). Secondly, each service or resource within the protected system has a Maximum Holding Time (MHT) associated with it. This MHT represents the longest continuous time for which the service may be 'held' or utilised by any user. The MHT does not cap the total service time a user may require – service requests exceeding the MHT will be provided as discrete multiples of the MHT. From these starting points, the DPB knows the maximum number of users permissible within the system and the feasible service needs of those users when they enter the system. The DPB (and operating system) then runs each user job – allocating the service required in time unit multiples of the service Maximum Holding Time. This allows the DPB to quantify an upper bound for the Maximum Wait Time that will be experienced by each user job.

The DPB as outlined in Millen (1992) is similar to the operation of real time operating systems as described in Levi *et al* (1990) and Agrawala *et al* (1992). The DPB and real time operating systems exercise *quota setting* on resources/services and *acceptance testing* on user needs prior to admitting new users to the system. This strategy effectively caps the number of users, their service requests that may be active at any one time - and the level of operational services existing within the system. The dynamic scaling of services existing within the system or the operational reality of unbounded numbers of service users is outside the DPB model.

Clustered Server Availability

Clustered server availability has been theoretically described in Karamanolis *et al* (1995), Cristian *et al* (1994), and most completely discussed in Pfister (1998). The paradigm guarantees service availability within distributed systems and is motivated by the frequency of hardware, system or application crashes and the resource cost in reconfiguring and recovering from these crashes.

Clustered server availability assumes an asynchronous network consisting of multiple host machines or nodes linked together into an arbitrary network topology. The primary entities within the model include the *distributed service*, a *service group*, an *availability policy*, and a *management service*. The distributed service is a facility of the network, and is characterised by its state, type and operational implementation. A service group is a collection of replicated operational implementations. The service group consists of a single primary implementation and several backup implementations. A group is formed by consensus on the single criteria of speedy communication amongst all nodes hosting the group members. Group membership is dynamic. An availability policy is a 2-tuple (replication, synchronisation). Replication defines the number of service backup implementations maintained within the distributed system. Synchronisation defines the number of updates by which backup state differs from the primary state. The model categories *close* (one to one update) and *loose* (update arrears) synchronisation. A management service is the system entity that implements the availability policy of the distributed service.

The model guarantees distributed service availability by ensuring operational primary and backup implementations exist within the distributed system at all times. The only threat to model viability is total network failure (the model suggests that the last surviving network node will hold the primary-backup group). The model makes no distinction between communication faults at a link level, faults within a node itself, or service implementation failure. When group reconfigurations occur for any of these failure reasons, the newly formed group select a replacement primary server and then create an additional backup on a suitable host node within the distributed system.

The model approaches availability management at the macro or system level. Service access is always available to users because the service implementation is replicated across the system. The model however does not address any form of service degradation other than service failure. As long as a service implementation is functioning and maintaining communication with group members, the model assumes service integrity and efficiency. Availability under this model is all about service access – service delivery is assumed to follow as a natural consequence.

A DEFINITION AND FRAMEWORK

There are two strategic issues most prominent in researching the service availability challenge – a definition of service availability, and the quality of service framework that would enable a networked-service achieve this definition.

Defining Service Availability

A comprehensive, clear definition of service availability underpins much of the discussion within this paper. An availability definition facilitates a more accurate appraisal of relevant completed research and contemporary industry trends - and may well suggest other appropriate solution strategies.

Within the computing vocabulary, *availability* is a sparingly used, and poorly defined, term. The international standards TCSEC (1985), ITSEC (1991), CTCEC (1993) that traditionally have served as a baseline for evaluating security and quality of service have lightly treated system and application availability. Of these standards, CTCEC (1993) is most specific – defining availability as the access to a specific resource or service within a time frame prescribed by the resource or service. The major distributed system standards and RM-ODP (1995) CORBA (1996) superficially discuss availability. RM-ODP (1995) declares availability to be an important quality of service consideration, and CORBA (1996) defines availability as service accessibility and usability upon demand by an authorised user. The availability research publications referenced in this paper contain differing availability definitions. The clustering architecture publications define availability as service access assurance, the goal being the high probability at all times of service operational existence at some contact location within the network. Clustering aims to guarantee service existence – user contact and usage progress are then assumed to follow. Alternatively, the denial of service research publications view availability in terms of initial access and subsequent user progress in using the service. The availability discussions considered as a body of work also reflect inconsistencies in describing the relationship between availability assurance and other computing architectural design and operational characteristics. Is availability assurance a security concern of the hosting operating system – or a quality of service challenge for each hosted service-providing entity? Does service availability encompass fault tolerance and reliability engineering? Can the obvious operational interdependencies of individual services, host software systems, hardware platforms and connecting networks be demarcated and related in describing the availability challenge?

The views, experiences and expectations of *any* service user may well help clarify this complicated scenario. Firstly, a service is viewed by a prospective user as a contact point for some functionality or capability. Secondly, service use is experienced by a user as a sequence: service contact, request feasibility, and finally the complete provision of this functionality or capability – all activities unfolding over some time line. This time line may be highly specified or loosely approximated – but is essential for meaningful service use. Thirdly, service availability is a gauge as to how this service usage time line meets that timetable expected by the user. This user timetable expectation may be individually derived or evolved via some explicit standard or generalised community understanding.

This generic user-driven definition of availability pivots on the timeliness of service contact and complete service provision. This must now be specifically reshaped within the context of a networked service. Figure 1 will assist with this development.

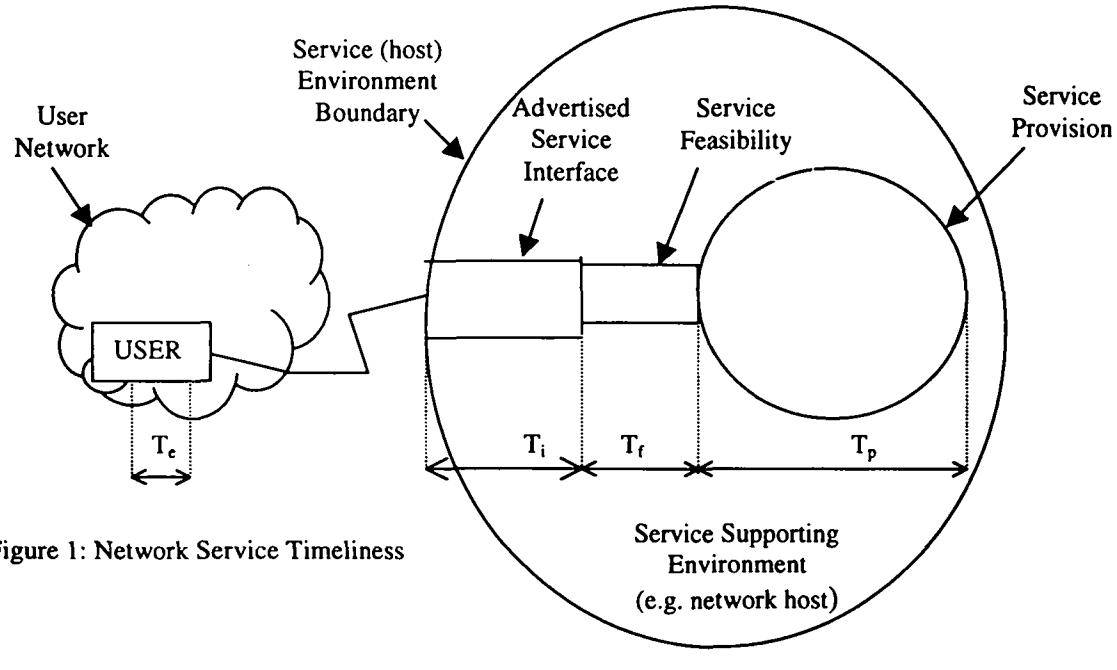

Figure 1: Network Service Timeliness

The generic distributed service is represented in Figure 1. The user requests service at the advertised service contact point. From this point, service availability pivots on the time taken to accept the request (T_i), verify request feasibility (T_f) and complete processing the request (T_p) – and how this total (plus reply times through T_i if appropriate) compares with the user's expectation of total service usage time (T_e). This scenario applies regardless of network topology, the physical positioning of user and service within this topology, or the type of service offered.

This analysis is summarized within the following definition:

> *Networked service timeliness is an accumulating measurement of the following activities:*
>
> *1) accepting a service request at the advertised service contact point AND*
> *2) analyzing the feasibility and validity of the service request AND*
> *3) establishing a contractual timetable for feasible service provision AND*
> *4) progressing and completing service provision as contracted.*

Networked service timeliness should be monitorable, verifiable, and publishable. Maximised networked service availability maps the timeliness needs and expectations of users to the historically achieved and forecast levels of distributed service timeliness.

Quality of Service Framework

Maximised service availability as defined in this paper maps user timeliness expectations to historically achieved and forecast levels of service timeliness. The historical and forecast service timeliness is an aggregated measurement of the time taken to admit, verify, progress and complete service requests. This definition serves as the single strategic goal for this availability-maximising analysis. It is now necessary to consider the quality of service framework that will achieve this goal.
The availability definition suggests five areas of generic functionality: *service contact point, service request validation, service throughput, service state, and service accountability*. Consequently a quality of service framework must directly focus on these functionality areas.

- The *service contact point or service interface* is viewed by users as the contact and delivery point for some specific behaviour or facility of the entire network. This interface consists of three design dimensions: a referencing representation or publishable dimension (e.g., a service name appropriately recorded within a network name server or trader server); a usage protocol or user-calling dimension (e.g., a TCP message of well defined format); and an instantiation or physically functioning dimension (e.g., a TCP program and port mapping to user space). Service users require the following quality of service framework for the service interface: -

  A highly publicised and well-known interface throughout the users' network. The interface representation must be readily registrable and detectable within all forms of service listings advertised through the network – regardless of whether these listings be based upon service name, service type or some other characteristics.

  A simple and easily used interface. Interface usage protocol must be concise, unambiguous and lightweight.

  A consistently addressable interface when viewed from any user location within the network. The interface representation must remain constant to all users across network space and time.

  A temporally efficient interface. The interface must quickly admit new service requests, dispatch both on-going service dialogue and completed service results. The interface must be operationally efficient for all traffic volumes – queues and delays must be avoided.

- *Request validation* describes a service capability whereby appropriate service requests are admitted for service provision – whilst inappropriate service requests are refused at the earliest opportunity. In general, a networked-service must manage two broad classes of inappropriate service requests. One class consists of requests that are infeasible, incomplete or incorrect in some way. A second class consists of requests that are appropriate in format and content – but whose volume or load stresses the service or its supporting infrastructure as in the case of spamming. Both classes may be traced to either accidental or malicious origins. The maximally available service must manage this traffic – any legitimate and certainly fee-paying customers have little appeal for cuts in service quality that result from the non-screening of inappropriate service requests. To this end, service users require a quality of service framework that centres upon the concept of usage agreements - defined as the combination of interface usage protocol with all relevant service utilisation conditions and quotas. These usage agreements must be concise, unambiguous and lightweight. In functional terms, it is difficult to be more prescriptive in terms of generalised usage agreements. It is clear however, that

additional functionality is required within the generalised networked-service if request validation is to be fully implemented. This additional functionality is a service awareness of non-corruptible user identity. This user identity enables the service to establish state information linking each user identity to a request history. With this state information, a networked-service may dynamically prioritise service requests down a range of decreasing service levels – the minimum of which is service refusal.

- *Request throughput* within this model describes a timetable for completed service provision. Throughput is a dynamically calculated cumulative time measurement of the following service activities: *request receipt*, *request validation*, and *complete service provision*. Service users have a very clear quality of service requirement in this area:

  Service delivery timeliness that meets or exceeds the users' timeliness expectations.
  Throughput consistency – service delivery timeliness standards are maintained during all peak workload times.

- *Service state*, to a user, facilitates seamless service completion – a very important quality of service expectation. Even in problematic instances, a single legitimate service request from each user must ultimately trace to completed service provision. In functional terms, *service statelessness*, however, will mean discontinuous service provision in problematic instances - several legitimate service requests will be required for completed service provision. The availability definition derived in section 3.1 views meaningful service provision as *completed* service provision. In the general case, users are not satisfied with access only to the service contact point, or with other partially completed service results. Whilst reliability engineering can never completely offset operational difficulties or failures, a graceful degradation of service is always preferable from a user's perspective. The ideally available service must always complete service provision, and this implies fault tolerance. This fault tolerance must exist at two levels. At the macro level, the service must always exist in some operationally ready form within the network to meet the needs of prospective users. At the micro level, the currently progressing service requests from users must also be cushioned so that the service quality for each request experiences managed degradation, not extinguishment, in the event of operational difficulties within the service or its environment.

- *Service accountability* defines how a service describes its availability levels to prospective and actual users. Service users, especially those operating within an economic paradigm, would have the following quality of service expectations:

  The ready supply of up-to-date, accurate, obligation-free '*quotes*' in respect of service timeliness.

  Service contracts that comprehensively define service usage. These contracts fully specify the usage conditions that must be met by the client, the timeliness guaranteed by the service, the conditions and mechanisms for service level renegotiation, and the domain or scope in which the contracts remain binding and enforceable.

  These accountability quality of service expectations clearly involve two different levels of implementation. A distributed service can readily issue quotes and contracts – but how are these recognised as credible and binding? A trusted third party is required to authenticate contracts and ensure they are honoured. Service accountability is an area where functionality is significantly shaped by the network security model – and in particular that model's strategies for confidentiality, correctness, and non-repudiation.

The five quality-of-service characteristics outlined above translate to four generic capabilities within the service contact point: *separation, redirection, relocation, and identification.*

Interface separation maximises the processing power for admitting service requests and dispatching request results. Separation describes an interface instantiation that is physically and operationally distinct from other service provision components. Separation means that the interface instantiation - whilst possibly initiated by, managed by, and certainly coupled with other service provision components – operates in an immediate resource space distinct from that of other service provision components. Consequently, the throughput of the separated interface instantiation is impacted by the admittance and dispatching of incoming/outgoing service requests – and not by the workload that the ensuing service provision entails.

Interface redirection facilitates replication – and therefore scaling – of the service at multiple points across the network. Redirection describes a capacity for an interface instantiation to be replicated at different, addressable points within the users' network – and for each replicated interface instantiation to redirect,

where necessary, service requests to other instantiations within the group. The service requires a single, high profile contact point within the users' network. This is the primary interface instantiation and it is supported by one or more secondary interface instantiations. Redirection within a group of replicated interface instantiations is hierarchical and rule based. The redirection rules are established initially, and subsequently varied, within the instantiation by that service component responsible for overall service throughput and load management. The instantiation, however, must autonomously exercise rule-based redirection decisions without further reference to other service provision components. An instantiation must also recognise and remember redirection 'chains' in which the instantiation is either the initial or final point of redirection.

Interface relocation implements address transparency for the overall service. Where redirection facilitates service availability via interface load management, relocation facilitates existential availability – ensuring the publicised service contact point (the primary interface) operationally exists within the users' network at all possible times.

Interface identification implements a basic level of protection against the accidental or malicious overuse or spamming of the networked-service. Interface identification also provides a basic framework upon which the networked-service may differentiate the levels of service attracted by individual service clients.

TRANSPORT LAYER FUNCTIONALITY

The definition and quality of service framework derived in this paper suggest a networked-service interface very similar to that achieved in business via the *telecentre* concept. The telecentre concept combines telephony infrastructure and business personnel to achieve a business public contact process that is versatile and professional. The telecentre involves mapping all telephone inquiries to a single, well-known, relocatable telephone contact point. From this point, incoming inquiry traffic is distributed to a well-defined group of operator contact points. These contact points may be geographically centralised or distributed. The set of contact points is sufficiently scalable to respond quickly to spikes or decreases in the numbers of received inquiries. The management of inquiry workload across the telecentre is very flexible – the differing characteristics and time requirements of individual calls can readily be factored into the workload distribution.

Clustered software services or *server farms* is the closest analogy that business computing has with the telecentre concept. This paradigm of service delivery is comprehensively discussed in Pfister (1998). Commercial clustering products are available from Microsoft, Sun, and a combined offering from Hewlett-Packard and Cisco. All of these solutions impose restrictions in terms of operating systems and/or supporting network topology. The basic operational logic of all these solutions is to share load across the cluster or farm on the basis of incoming request numbers. Existing cluster solutions lack the versatility and simplicity of the telecentre concept.

Transport layer protocols glue together networked software services. The ready implementation of the telecentre concept depends on the operational capabilities of the relevant transport layer protocol. The *defacto* standard, Transport Control Protocol (TCP), is well described in Feit (1998). TCP's popularity is largely due to the quality of service it offers to users. TCP provides a reliable, flow-controlled, connection-oriented communication service. TCP is easily utilised by applications and is implemented within all commercially available operating systems.

TCP does not support the full quality of service framework developed in this paper. TCP clearly offers interface separation – TCP executes as part of the host operating system and is not tied to the resources allocated to the user application. TCP offers a very coarse-grained identification facility – TCP identifies the type, and not the identity, of the application user. TCP does not support redirection or relocation.

TCP's relative simplicity has been significantly responsible for its popularity. It is felt, however, that minimal changes to TCP's specification would be needed for it to fully support the desired quality of service framework. These specification changes will evolve from the future work outlined in the next section.

FUTURE WORK

The distributed file server is usually the most heavily used service within the network – and consequently an excellent vehicle with which to benchmark and demonstrate maximised availability. Distributed file systems present a single name space to the user and hide the physical location of the file from the file system user. The user references the required file via a single pathname and cannot and need not differentiate physically local and remote file locations.

Microsoft's Distributed File System (DFS), Sun's Network File System (NFS), and the Coda File System are all influential file systems and follow the client/server paradigm. In NFS and DFS a small set (usually cardinality one) of server processes provide a storage repository utilised by a much larger set of users. This server set provides no file locking or synchronisation – consequently NFS and DFS do not provide one copy update semantics for files held uniquely within, or shared across the server set. The NFS and DFS server set aims to enhance scalability and also reduce the impact on users of server failure or network component problems. The Coda File System has evolved from the Andrew File System and is substantially more resilient to file server and network component difficulties. In Coda a

small set of replicated file servers provides one-copy file updates to a large set of distributed file users.

The future work with the research outlined in this paper will apply the quality of service framework previously described to distributed file server design. The resulting distributed file system will resemble Coda in its operational goals and its focus on replicated file storage – but differ significantly via a state-based but more lightweight client-side agent and greater server interface functionality.

CONCLUSION

Networked-service availability is an increasingly important quality of service issue and is essential in supporting the paradigms of network and user-pays computing. Despite this situation, no clear definition or solution strategies exist for managing service availability.

This paper has defined networked-service availability and described the service quality level expected by service users. The paper has broadly summarised the strategic changes necessary to transport layer protocols to provide these service levels. The paper also describes the construction of a distributed file system that is being used to demonstrate the practicality and accuracy of the model.

REFERENCES

Amoroso, E.G. (1993a) A Policy Model for Denial of Service. **Proceedings Computer Security Foundations Workshop III,** Franconia, N.H.

Amoroso, E.G. (1993b) *NDU(C) :* A Mandatory Denial of Service Model. **Proceedings of the 16th National Computer Security Conference.**

Agrawala, A.K. and Gordon, K.D and Hwang, P. (1992) Mission Critical Operating Systems. **Case Studies in OS.** IOS Press Netherlands.

Bacic, E. and Kuchta, M. (1991) **Considerations in the Preparation of a Set of Availability Criteria** Canadian System Security Centre, Ottawa Canada 1991.

Che, F. and Gligor, V.D. (1990) A Specification and Verification Method for Preventing Denial of Service **Proceedings of the IEEE Symposium on Security and Privacy.**

CORBA (1996) Object Management **Group Common Object Request Broker Architecture.** URL http://www.corba.org

CTPCEC (1993) **Canadian Trusted Computer Product Evaluation Criteria,** Version 3.0. Communications Security Establishment, Government of Canada.

Cristian, F. and Shivakant, M. (1994) Automatic Service Availability Management in Asynchronous Distributed Systems. **Proceedings of the Second International Workshop on Configurable Distributed Systems IEEE.**

DoD. (US Department of Defense, 1987) **Trusted Network Interpretation of TCSEC.** US National Computer Security Center

Feit, S. (1998) **TCP/IP.** McGraw Hill. New York.

Gligor, V.D. (1983) A Note on the Denial-of-Service Problem. **Proceedings of the Symposium on Security and Privacy, IEEE.**

IDC (1998) **Preparing for I-Commence.** URL http://www.idc.com

Karamanolis, C.T. and Magee, J.N. (1995) **Configurable Highly Available Distributed Services.** Imperial College of Science, Technology and Medicine, London.

Leiwo, J. and Zheng, Y. (1997a) **A Method to Implement a Denial of Service Protection Base.** Monash University.

Leiwo, J. and Zheng, Y. (1998b) **Layered Protection of Availability.** Monash University.

Levi, S.T. and Agrawala, A.K. (1990) **Real Time Operating Systems.** McGraw-Hill Publishing Company. New York.

Millen, J.K. (1992) Mitre Corporation. A Resource Allocation Model for Denial of Service. **Proceedings of the IEEE Symposium on Research in Security and Privacy.**

Nyberg, N. (1998) Denial of Service Flaws in SDI Software : An Initial Assessment. **Proceedings [of the IEEE Symposium on Security and Privacy.**

Pfister, G.F. (1998) **In Search of Clusters.** Prentice Hall, New Jersey.

RM-ODP (1995) **Reference Model – Open Distributed Processing.** ISO (International Standards Organization).

TCSEC. (1985) **DoD (US Department of Defense) Trusted Computer Security Evaluation Criteria (TCSEC).** US National Computer Security Center, Department of Defense.