S17/CW/IN/66

1999 NASA/ASEE SUMMER FACULTY FELLOWSHIP PROGRAM

JOHN F. KENNEDY SPACE CENTER
UNIVERSITY OF CENTRAL FLORIDA

Performance Evaluation of Reliable Multicast Protocol for Checkout and Launch Control Systems

Wei Wennie Shu
Department of Electrical and Computer Engineering
University of Central Florida

In Conjunction with John Porter
Checkout and Launch Control Systems
John F. Kennedy Space Center

ABSTRACT

The overall objective of this project is to study reliability and performance of Real Time Critical Network (RTCN) for checkout and launch control systems (CLCS). The major tasks include (a) reliability and performance evaluation of Reliable Multicast (RM) package and (b) fault tolerance analysis and design of dual redundant network architecture.

Performance Evaluation of Reliable Multicast Protocol for Checkout and Launch Control Systems

Wei Wennie Shu

1 INTRODUCTION

1.1 Project Definition

The overall objective of this project is to study reliability and performance of real time critical network (RTCN) for checkout and launch control systems (CLCS), with two major components of work to be focused:

- Reliability and performance evaluation of reliable multicast (RM) package;
- Fault tolerance analysis and design of dual redundant network architecture.

1.2 Background Overview

CLCS includes four major subsystems:

- RTPS, *Real-Time Processing Systems*
- SDC, Shuttle Data Center
- SIM, Simulation System
- BIN, Business Information Network

Our project is focused on the real-time processing subsystem, which in turn consists of four major processing components:

- Gateways
- DDPs/CCPs, Data Distribution Process/Command Control Process
- SDC, Shuttle Data Center
- CCWs, Command Control Workstations

To interconnect these processing components together, it involves with construction of three major network components:

- RTCN, Real time critical network
- DCN, Display and control network
- UN, Utility network

1.3 Application Characteristics

Applications associated with RTCN are mainly information exchanges, which include the 10ms synchronous rate to send messages from gateways to DDPs, CCPs, and SDC, with the pattern of many-to-many multicasting [1], and the 100ms synchronous rate to send messages from DDPs/CCPs to CCWs, gateways, and SDC, also with the pattern of many-to-many multicasting.

There are two message protocols supported, ACK-based and NACK-based. In an NACK-based message stream, a sender does not wait for acknowledgment of the receiver and a receiver sends NACK back if any message is out of order. The sender will perform retransmission upon receiving NACK. In an ACK-based message stream, a receiver sends ACK back for every message received, and the sender waits for ACK, or time-out for retransmission.

1.4 Software Architectures

It is basically a multithreading client-server model. The server is a Reliable Multicast (RM) package running on top of UDP, and clients are application programs migrated from the old LPS and communicate exclusively via RM server.

On each machine, there exists a single RM server with multiple threads and multiple clients running as concurrent processes/threads. It utilizes many operating system features, such as Pthread package, POSIX.4 real-time extension to accomplish priority scheduling, shared message queue to establish request and response flow, and shared memory to eliminate excessive message copying.

1.5 Network Infrastructure

There are many currently available technologies [2,3,4,5,6]. Among them, the Switched Fast Ethernet has been selected due to its reasonable cost-performance and adequate functionality. Major products include Catalyst 2900 Switches by Cisco, BayStack 450 Switch by Bay Network, and SuperStack II 3300 by 3Com. A brief *evaluation report* is available online.

2 PERFORMANCE AND RELIABILITY EVALUATION

2.1 Testing Goal and Levels

We decide to use the synthetic load to test system at different levels. Consequently, we will compare the measured capacity limits to the real-word worst case analysis to determine the safety margin. The three different testing levels are described here and the task of this project is concentrated at the third level of testing:

- ***Level I***: Underlying network architecture testing will use Smartbits to determine port-to-port capacity
- ***Level II***: Network infrastructure testing uses the standardized transport interfaces, UDP and TCP to determine the available bandwidth on the top of operating system
- ***Level III***: Network application testing will use the RM package with synthetic communication and CPU load to determine the available bandwidth from the application interface.

2.2 Modeling and Performance Evaluation of ACK-based Message Protocol

2.2.1 Testing environment

The testing is set up to have SUN Ultra60 as the sender and SUN Enterprise3500 as the receiver. In each machine, the RM server is always running as the top priority process and the application clients are running as the processes with the second highest priority. Each sender is periodically sending the messages with specified sizes to the receiver. Here, the synchronous rate can be varied from 1ms to 10ms and the message size can range from 1Kbytes to 64Kbytes, which is the upper limit the RM can handle.

2.2.2 Testing send/receive of single message stream

We define three important types of metrics in testing of behavior of the ACK based message protocol.

- ***Response time***: the time from sending a message by the application client to reach the receiving side's RM server until receiving the ACK message back at the sending side. It includes a round trip time of message transmission to assure the arrival of message at the receiving side, but not guarantee the receipt of message by the application client at the receiving side.
- ***End-to-end delay time***: the time from sending a message by the application client to reach the receiver until receiving the message by the application client at the receiving side
- ***Throughput***: the amount of messages to be sent without loss of messages. Here, throughput can be calculated and measured based on the back-to-back message transmission or periodical message transmission with the fixed synchronous rate.

In the ACK-based message protocol, the response time is pretty close to the end-to-end delay time. If the receiving side is heavy loaded, it can have great impact on the end-to-end delay time. On the other hand, the response time depends on the network traffic. Throughput can be calculated and measured based on the back-to-back message transmission or periodical message transmission with the fixed synchronous rate.

2.2.3 Modeling and analysis

Here, we models the response time t_r as a function of message sizes:

$t_r = \alpha + \beta * size / \gamma$

where, $\alpha = 680$ μs, startup time

$\beta = 103$ μs/Kbytes, transmission time

$\gamma = (1 + 0.001*size)$, adjustment factor for large messages

size = message size in Kbytes

And the end-to-end delay time t_d is basically proportional to the response time t_r.

$t_d = \lambda * t_r$, where λ is about 1.05 to 1.15

By ignoring adjustment factor γ, the throughput T can be obtained by

$T \approx 1 / (\alpha + \beta * size)$

Theoretically speaking, the throughput can be approximately calculated by assuming the back-to-back message transmission.

- For a small message of 1K
  $T \approx 1 / (\alpha + \beta) = 8$ Kbits / 783 μs ≈ 10 Mbps
- For a message of 10K
  $T \approx 1 / (\alpha + 10\beta) = 80$ Kbits / 1710μs ≈ 47 Mbps
- For a large message of 50K
  $T \approx 1 / (\alpha + 50\beta) = 400$ Kbits / 5830 μs ≈ 67 Mbps

Figure 2.1 gives comparison of measured & calculated data.

Msg size in Kbytes	1	10	30	50	64
Bandwidth in Mbps	0.8	8	24	40	51.2
	1K	10K	30K	50K	64K
tr in us, measured	779	1670	3669	5594	6950
tr in us, calculated	783	1700	3680	5585	6875
td in us, measured	783	1779	3878	6045	7533
td in us, calculated	861	1870	4048	6143	7563

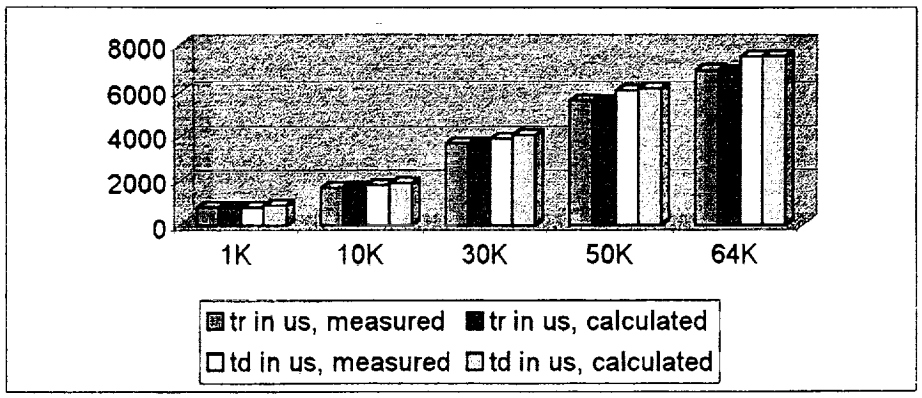

Figure 2-1 Comparison of measured and calculated data for single ACK message stream

2.2.4 Testing send/receive of multiple message streams

If there are multiple senders in the testing configuration, the background traffic has impact on the response time of the message stream to be tested, as well as the end-to-end delay time. Both the background traffic and the message stream to be tested will compete for network switcher's bandwidth and CPU resources at the receiving side. The end-to-end delay time and throughput defined in the above can also be applied here.

$t_{r,multi} = t_{r,single} * (1 + \delta / 100)$

where, δ = background traffic modifier in Mbps

$t_{r,single}$ = response time for the single message stream

Figure 2.2 shows how the response time is affected by the various background traffics.

Background Traffic in Mbps	Msg size in Kbytes	1	10	30	50	64
	Bandwidth in Mbps	0.8	8	24	40	51.2
		1K	10K	30K	50K	64K
0	tr in us, measured	779	1670	3669	5594	6950
	tr in us, calculated	783	1700	3680	5585	6875
10	tr in us, measured	819	1744	4123	6371	7824
	tr in us, calculated	857	1837	4036	6153	7645
20	tr in us, measured	878	1923	4416	7134	8630
	tr in us, calculated	935	2004	4403	6713	8340
40	tr in us, measured	1051	2129	5425	7464	
	tr in us, calculated	1091	2338	5137	7832	9730

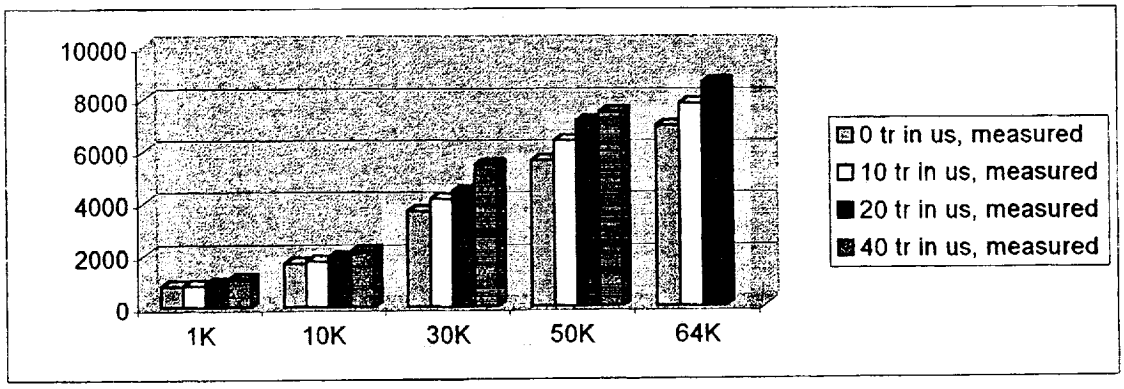

Figure 2-2 Response time of multiple ACK message streams

Notice that it makes difference when testing ACK-based messages with one primary receiver or with one primary receiver and one non-primary receiver. How does the number of receivers have impact on ACK's bandwidth? Next, what is capacity of receiving many small messages at Enterprise3500 side? Currently, due to the resource limitation, we have tested 15 streams with message size of 1K, 2K, 4K, but not 5K. More importantly, the best priority settings to RM server as well as client processes need to be determined.

2.3 Modeling and Performance Evaluation of NACK-based Message Protocol

2.3.1 Testing send/receive of single message stream

In addition, sending time is a newly defined metric, particularly defined for the NACK-based message protocol.

- ***Sending time***: the time from sending a message by the application

In the NACK-based message protocol, the sending time is very different from the end-to-end delay time

2.3.2 Modeling and analysis

Here, we models the end-to-end delay time t_d as a function of message sizes:

$t_d = \alpha + \beta * size / \gamma$

where, α = 500 μs, startup time

β = 110 μs/Kbytes, transmission time

$\gamma = (1 + 0.001*size)$, adjustment factor for large messages
size = message size in Kbytes

And the sending time t_s is less dependent on the size of messages.

$t_s = \alpha' + \beta' * size / \gamma$

where, $\alpha' = 280$ μs, startup time
$\beta' = 20$ μs/Kbytes, transmission time
$\gamma = (1 + 0.001*size)$, adjustment factor for large messages
size = message size in Kbytes

By ignoring adjustment factor γ, the throughput T_{NACK} can be obtained by

$T_{NACK} \approx 1 / (\alpha + \beta * size)$

Figure 2.3 gives comparison of measured & calculated data

Msg size in Kbytes	1	10	30	50	64
Bandwidth in Mbps	0.8	8	24	40	51.2
	1K	10K	30K	50K	64K
ts in us, measured	337	439	894	1262	1550
ts in us,calculated	300	480	880	1280	1560
td in us, measured	613	1728	3735	5873	7294
td in us, calculated	610	1589	3704	5738	7117

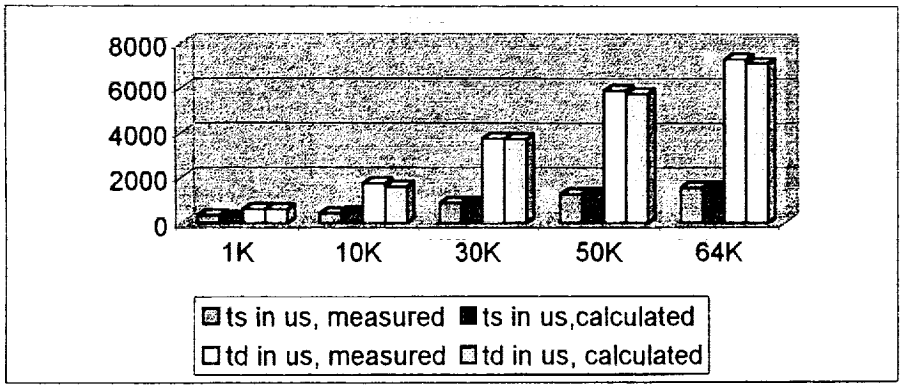

Figure 2-3 Comparison of measured and calculated data for singel NACK message stream

2.3.3 Testing send/receive of multiple message streams

Similarly, if there are multiple senders in the testing configuration, the background traffic has impact on the end-to-end-delay time of the message stream to be tested.

$T_{d,multi} = t_{d,single} * (1 + \delta / 90)$

where, δ = background traffic modifier in Mbps
$t_{d,single}$ = end-to-end delay time for the single message stream

Figure 2.4 shows how the end-to-end delay time is affected by the various background traffics.

Background	Msg size in Kbytes	1	10	30	50	64
Traffic in Mbps	Bandwidth in Mbps	0.8	8	24	40	51.2
		1K	10K	30K	50K	64K
0	td in us, measured	613	1728	3735	5873	9429
	td in us, calculated	610	1589	3704	5738	7117
10	td in us,measured	751	2034	4142	7073	
	td in us, calculated	681	1920	4150	6526	10477
20	td in us, measured	779	2153	4539	10784	
	td in us, calculated	749	2112	4565	7178	11524
40	td in us, measured	799	2363	5340		
	td in us, calculated	885	2496	5395	8483	13620

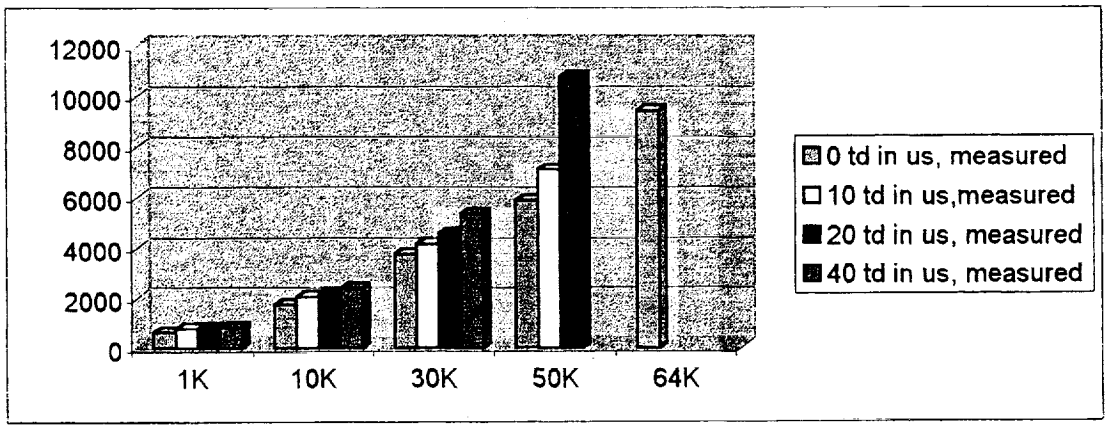

Figure 2-4 End-to-end delay time of multiple NACK message streams

2.4 Measurement of Throughput

2.4.1 Throughput in ACK-based single/multiple message stream

For the single stream case, the upper limit of throughput without missed messages is measured when the synchronous rate is varied. Since the maximum allowable size of messages in RM package is of 64Kbytes, the longer synchronous rate cannot saturate the network bandwidth.

- 10ms, 64KBytes, 64*8/10 = 51.2Mbps
- 5ms, 40KBytes, 40*8/5 = 64Mbps
- 2ms, 16KBytes, 16*8/2 = 64Mbps

Therefore, the measured throughput is about 64Mbps. Similarly, for the multiple stream case, upper limit of throughput is measured:

- with 10Mbps, 64KBytes, 64*8/10+10=61.2Mbps
- with 20Mbps, 50KBytes, 50*8/10+20=60Mbps
- with 40Mbps, 25KBytes, 25*8/10+40=60Mbps

The measured throughput is about 60Mbps

2.4.2 Throughput in NACK-based single/multiple message stream

For the NACK-based message streams, the upper limit of throughput without missed messages is measured when the synchronous rate is varied.

- 10ms, 64KBytes, 64*8/10 = 51.2Mbps
- 5ms, 30KBytes, 30*8/5 = 48Mbps
- 2ms, 12KBytes, 12*8/2 = 48Mbps

Therefore, the measured throughput is about 50Mbps. Similarly, for the multiple stream case, upper limit of throughput is measured:

- with 10Mbps, 50KBytes, 50*8/10+10=50Mbps
- with 20Mbps, 50KBytes, 50*8/10+20=60Mbps
- with 40Mbps, 30KBytes, 30*8/10+40=64Mbps

The measured throughput is about 60Mbps, same as ACK one. However, for the single stream case, the throughput of NACK is even lower than one of ACK. It needs more investigation for performance verification or implementation explanation.

3 FAULT TOLERANCE ANALYSIS AND DESIGN

3.1 Different Design of Dual Redundant Network

In general, the dual network can be used in different ways to improve fault tolerance of single point failures. It has been decided to construct a complete dual redundant network: RTCN-A and RTCN-B. However, many varieties of design choices exist.

The first approach is called the Active/Standby redundant network. RTCN-A is assigned as the active network whereas RTCN-B acts as the standby network. In normal cases, only one network is fully operational, thus, no extra load is added to network or CPUs.

The second approach is called fully duplicated redundant network. Both RTCN-A and RTCN-B are fully operational. For every message, CPU will send it to both networks and the receiving side CPU will receive whichever comes first and ignore the second arrival one. In this way, it will almost double the CPU load on both sending and receiving sides.

The third approach is newly proposed in this project, called Ping-Pong alternation redundant network. More details are described follows.

3.2 Ping-Pong Alternation Dual Network

The basic idea of Ping-Pong alternation approach is to use dual networks, RTCN-A and RTCN-B, alternatively. The design is motivated since there is no increase of CPU load on sending and receiving sides, especially important for Ultra60 with single CPU. Another design motivation it to low the network traffic to its half, lighter traffic of network always enhances its performance.

As one of major drawbacks, it makes RM protocol a little more complex. Additional information is needed on each RM server. We define a boolean variable ***flag,*** *being 0 for RTCN-A and 1 for RTCN-B;* In each message, we also need a boolean variable ***flag,*** *being 0 for RTCN-A and 1 for RTCN-B;* We outline the protocol modification to the Ping-Pong Alternation approach as follows.

- ACK-base message stream
  - message send:

```
if (flag==0)
   then send to RTCN-A
   else send to RTCN-B
flag = !flag
```

  - message receive:

```
select to receive messages from either RTCN-A or RTCN-B
if (message.flag==0)
   then send ACK back via RTCN-A
   else send ACK back via RTCN-B
```

  - message send time-out:

if (message.flag==0)
then resend to the other network RTCN-B
increment failCountA to keep record
if failCountA > threshold perform fail over

- NACK-base message stream
  - message send:

if (flag==0)
then send to RTCN-A
else send to RTCN-B
flag = !flag

  - message receive:

select to receive messages from either RTCN-A or RTCN-B

  - message receive out of order

if (majority of missed message.flag==0)
then send NACK back via RTCN-B
else send NACK back via RTCN-A

  - message send getting NACK:

resend the missed message to the network where NACK comes
increment failCount to keep record
if failCount > threshold perform fail over

Dual redundant network design is only in the initial stage. A prototype is needed to verify correctness and test different approaches for performance comparison and fail over time measurement.

4 DISCUSSION AND FUTURE WORK

All test data conducted in this project are only preliminary; many places need detailed analysis and verification [7]. Inadequate application traffic analysis makes thorough performance evaluation difficult. Detailed information about application characteristics, both for normal and worst cases, can be very helpful.

4.1 About Network Switches

The current broadcast-based multicast limits the total bandwidth to 100Mbps. The future VLAN implementation with the true multicast can help the system to scale up [8]. Prioritization is another future feature can be used to enhance the real-time critical performance

4.2 About Real-time Message (RM) Package

More formal design specification and verification are suggested [9, 10, 11]. Without loss of concurrency, the number of threads in the RM server should be minimized. Message buffers are allocated and de-allocated by different sides of client and server, being efficient but less safety. Use of many nested mutex locks needs careful investigation.

4.3 About Operating System Level Support

The operating system support is very weak in this project and needs substantial efforts [12, 13]. Solaris' real-time extension should be studied and evaluated in its behavior and performance. Assigning different priorities to the RM server and application clients shall be studied on single and multiple CPU machines, particularly in response time and frequency of context switches. Memory locking behavior and its impact on performance needs to be tested. Use of shared memory and message queues need more studies, particularly when created and accessed by different users and applications processes. Behavior of user threads bounding to kernel processes or lightweight processes should be clarified, especially for multiple CPUs. Any possible memory leakage needs to be checked and monitored, particularly for extended run-time.

4.4 Suggestions from Software Engineering Point of View

In general, risk analysis needs more attention in all kinds of levels, including misuse of RM package.

Formal specification on design, implementation, and testing can be further improved. Interaction and integration with the operating system supports should be much encouraged or even enforced to ensure the system integrity.

Hardware failure was/is a dominant consideration in developing reliable systems. Software complexity needs more attentions. RM package needs more tightly integration among various subsystems, such as Health Check, Data Center, Gateways, etc. In addition, is there any potential alternatives to RM package? The Xpress Transport Protocol (Xtp) provides reliable datagrams and reliable multicast connections. Many other reliable UDP can offer possibilities.

5 ACKNOWLEDGMENT

The author would like to thank the entire CLCS division, particularly to Vincent Mandese, John Porter, Steve Davis, Terry Ross, Jeff Crowder, and Steve Goodmark for their wonderful help. Many thanks also go to the KSC and UCF, who made this program possible, particularly Ramon Hosler, Gregg Buckingham, Jane Hodges, Judie Gilliam, and all other faculty teammates

REFERENCES

1. Kenneth C. Miller, *Multicast Networking and Applications*, Addison Wesley, 1999, ISBN 0-201-30979-3.
2. Darryl P.Black, *Building Switched Networks*, Addison-Wesley, 1999, ISBN 0-201-37953-8.
3. John J. Roses, *Switched LANs*, McGraw-Hill, 1998, ISBN 0-07-053413-6.
4. Martin Nemzow, *Fast Ethernet Implementation and Migration Solutions*, McGraw-Hill, 1997, IBSN 0-07-046385-9.
5. Jayant Kadambi, Ian Crayford, and Mohan Kalkunte, *Gigabit Ethernet: Migrating to High-Bandwidth LANs*, Prentice Hall, 1998, ISBN 0-13-913286-4.
6. Howard W. Johnson, *FastEthernet: Dawn of a New Network*, Prentice Hall, 1996, ISBN 0-13-352643-7.
7. Gilbert Held, *LAN Performance: Issues and Answers*, John Wiley & Sons, 1996, ISBN 0-471-96926-5.
8. Gilbert Held, *Virtual LANs: Construction, Implementation, and Management*, John Wiley & Sons, 1997, ISBN 0-471-17732-6.
9. K. Birman, *Building Secure and Reliable Network Applications*, Manning Publications, 1996, ISBN 1-884777-29-5.
10. Roger S. Pressman, Software Engineering: A Practitioner's Approach, 4th edition, McGraw-Hill, 1997, ISBN 0-07-114603-2.
11. Stephen R. Schach, Classical and Object-oriented Software Engineering with UML and C++, 4th edition, McGraw-Hill, 1999, ISBN 0-07-290168-3.
12. H. Majidimehr, *Optimizing UNIX for Performance*, Prentice Hall, 1996, ISBN 0-13-111551-0.
13. Cockcroft, *Sun Performance and Tuning*: SPARC & Solaris, Sun Microsystems, 1995, ISBN 013-149642-3.