RE 103663902

MRD 1/29/14

01/29/2014

103665717

Form PTO-1595 (Rev. 06-12)
OMB No. 0651-0027 (exp. 04/30/2015)

U.S. DEPARTMENT OF COMMERCE
United States Patent and Trademark Office

RECORDATION FORM COVER SHEET
PATENTS ONLY

To the Director of the U.S. Patent and Trademark Office: Please record the attached documents or the new address(es) below.

1. Name of conveying party(ies)

Open Text S.A.

Additional name(s) of conveying party(ies) attached? □ Yes ☒ No

2. Name and address of receiving party(ies)

Name: Open Text S.A.

Internal Address:

Street Address: 40 Avenue Monterey

City: Luxembourg

State:

Country: LU Zip: L-2163

Additional name(s) & address(es) attached? □ Yes ☒ No

3. Nature of conveyance/Execution Date(s):

Execution Date(s) 08/14/2013

□ Assignment □ Merger

□ Security Agreement □ Change of Name

□ Joint Research Agreement

□ Government Interest Assignment

□ Executive Order 9424, Confirmatory License

☒ Other Correction by Declaration for ownership of U.S. Pat. 6,615,215 at Reel/Frame 017663/0392

4. Application or patent number(s): □ This document serves as an Oath/Declaration (37 CFR 1.63).

A. Patent Application No.(s)

09/527,247

B. Patent No.(s)

6,615,215

Additional numbers attached? □ Yes ☒ No

5. Name and address to whom correspondence concerning document should be mailed:

Name: Sprinkle IP Law Group / OPEN
Customer No. 109422

Internal Address:

Street Address: 1301 W. 25th Street, Suite 408

City: Austin

State: Texas Zip: 78705

Phone Number:

Docket Number: OPEN1660

Email Address: docketing@sprinklelaw.com

6. Total number of applications and patents involved: 1

7. Total fee (37 CFR 1.21(h) & 3.41) $ 40.00
See USPTO Doc #103663902

☒ Authorized to be charged to deposit account

□ Enclosed

□ None required (government interest not affecting title)

8. Payment Information

Deposit Account Number 503183

01/30/2014 HTONI1 ~~00000005 503183~~ 6615215

Authorized User Name Katharina Schuster

01 FC:8021 40.00 DA

9. Signature: Katharina Schuster
Signature

Jan. 29, 2014
Date

Katharina W. Schuster, Reg. 50,000
Name of Person Signing

Total number of pages including cover sheet, attachments, and documents: 45

Documents to be recorded (including cover sheet) should be faxed to (571) 273-0140, or mailed to:
Mail Stop Assignment Recordation Services, Director of the USPTO, P.O.Box 1450, Alexandria, V.A. 22313-1450

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

DECLARATION	Atty. Docket No. **OPEN1660**
Applicant(s) **W. Clinton Petty**	
Application Number **09/527,247**	Filed **March 17, 2000**
Patent Number **6,615,215**	Issue Date **September 2, 2003**
For **METHOD FOR GRADUATED LOAD SENSITIVE TASK DISPATCHING IN COMPUTING SYSTEM**	
Group Art Unit **2175**	Confirmation No. **3220**

Mail Stop: Assignment Recordation Services
Director
U.S. Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

Certificate of Transmission Under 37 C.F.R. § 1.8

I hereby certify that this correspondence is being transmitted to the Commissioner for Patents, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22312-1450 on August 14, 2013.

Delia Narvaiz
Delia Narvaiz

Dear Sir:

STATEMENT OF FACTS

1. On July 7, 2000, an assignment paper was recorded by the U.S. Patent and Trademark Office at Reel/Frame 010952/0855. This recordation lists the sole inventor, Mr. W. Clinton Petty, of the above-identified U.S. Patent Application as the assignor and CommerceQuest Inc. as the assignee, evidencing a transfer of title and ownership of the above-identified U.S. Patent Application from the inventor to CommerceQuest Inc. See **Exhibit A**, "Assignment 1".

2. On October 12, 2005, an assignment paper was recorded by the U.S. Patent and Trademark Office at Reel/Frame 017073/0506. This recordation lists CommerceQuest Inc. as the assignor and Metastorm, Inc. as the assignee, evidencing a transfer of title and ownership of the above-identified U.S. Patent Application from CommerceQuest Inc. to Metastorm, Inc. See **Exhibit A**, "Assignment 3".

3. On July 11, 2012, an assignment paper was recorded by the U.S. Patent and Trademark Office at Reel/Frame 028559/0298. This recordation lists Metastorm Inc. as the assignor and Open Text S.A. as the assignee, evidencing a transfer of title and ownership of the above-identified U.S. Patent Application from Metastorm Inc. to Open Text S.A. See **Exhibit A**, "Assignment 5".

4. The recorded "Assignment 1," "Assignment 3," and "Assignment 5" reflect the true chain-of-title for the above-identified U.S. Patent Application.

5. On May 24, 2006, an assignment paper was recorded by the U.S. Patent and Trademark Office at Reel/Frame 017663/0392. This recordation lists Lantronix, Inc. as the assignor and Silicon Valley Bank as the assignee, showing a grant of security interest in the above-identified U.S. Patent Application from Lantronix, Inc. to Silicon Valley Bank. See **Exhibit A**, "Assignment 4".

6. On May 23, 2012, the undersigned reviewed the assignment history and, upon seeing Assignment 4 in the assignment history for the above-identified U.S. Patent Application, determined to order a copy of the assignment paper (the "Lantronix Security Agreement") recorded by the U.S. Patent and Trademark Office at Reel/Frame 017663/0392. A copy of the assignment history dated May 23, 2012 is enclosed herewith as **Exhibit B**. A copy of the Lantronix Security Agreement is enclosed herewith as **Exhibit C**.

7. Page 3 of the Lantronix Security Agreement describes U.S. Patent No. 6,615,215, September 2, 2003, as "Switch Node for Connecting a Keyboard Video Mouse to Selected Servers in a Interconnected Switch Node Network." See **Exhibit C**.

8. The above-identified U.S. Patent Application, issued as U.S. Patent No. 6,615,215 on September 2, 2003, is entitled "METHOD FOR GRADUATED LOAD SENSITIVE TASK DISPATCHING IN COMPUTING SYSTEM." A copy of U.S. Patent No. 6,615,215 is enclosed herewith as **Exhibit D**.

9. The description for U.S. Patent No. 6,615,215 on page 3 of the Lantronix Security Agreement (see **Exhibit C**) does not match the actual title of the invention for U.S. Patent No. 6,615,215 issued on September 2, 2003. See **Exhibit D**.

10. Open Text S.A. is the correct owner of U.S. Patent No. 6,615,215. Open Text S.A. has attempted to contact Silicon Valley Bank and the correspondent "CBCINNOVIS DBA FEDERAL RESEARCH" responsible for requesting the recordation of the Lantronix Security Agreement to request that they immediately file corrective papers with the U.S. Patent and Trademark Office to remove the improperly recorded Lantronix Security Agreement from the chain of title of U.S. Patent No. 6,615,215.

a. A letter addressed to Oleh Hereliuk, CBCInnovis dba Federal Research, 1023 Fifteenth Street, NW, Suite 401, Washington, DC 20005, was mailed on February 14, 2013 by certified mail, return receipt requested. It was returned by the U.S. Postal Service unsigned and with the address crossed out. A copy of the returned certified mail envelope with the unsigned return receipt is enclosed here as **Exhibit E**.

b. Separately, a letter addressed to Silicon Valley Bank, 3003 Tasman Drive, Santa Clara, California 95054, was also mailed on February 14, 2013 by certified mail, return receipt requested. The letter was accepted by Silicon Valley Bank. A copy of the signed return receipt is enclosed here as **Exhibit F**. However, as of June 15, 2013 (see **Exhibit A**), the improperly recorded Lantronix Security Agreement remains in the chain of title of U.S. Patent No. 6,615,215.

11. The previously recorded Lantronix Security Agreement was submitted with erroneous information. Lantronix, Inc. was **never** in the chain-of-title for the above-identified U.S. Patent Application and thus has no right, interest, or title whatsoever to grant any security interest in U.S. Patent No. 6,615,215 to Silicon Valley Bank. Consequently, Silicon Valley Bank is not an assignee of and does not own U.S. Patent No. 6,615,215 issued on September 2, 2003.

12. The Lantronix Security Agreement was improperly recorded at Reel/Frame 017663/0392. The improperly recorded Lantronix Security Agreement should not be in and should be removed from the chain of title of U.S. Patent No. 6,615,215.

13. The true chain-of-title for U.S. Patent No. 6,615,215 should not be considered altered by the incorrect recordation of the Lantronix Security Agreement or the Lantronix Security Agreement itself.

14. The last correct assignee, Open Text S.A., has been, and continues to be, the owner of U.S. Patent No. 6,615,215.

I, Katharina W. Schuster, am the attorney of record for the above-identified U.S. Patent Application. All statements of my own knowledge contained in this Declaration are true, and that all statements made on information and belief are believed to be true.

Respectfully submitted,

SPRINKLE IP LAW GROUP
Attorneys for Applicant

Katharina Schuster

Katharina W. Schuster
Reg. No. 50,000

Date: Aug. 14, 2013

1301 W. 25th Street, Suite 408
Austin, TX 78705
Tel. (512) 637-9220
Fax. (512) 371-9088

EXHIBIT A

United States Patent and Trademark Office

Home | Site Index | Search | Guides | Contacts | eBusiness | eBiz alerts | News | Help

Assignments on the Web > Patent Query

Patent Assignment Abstract of Title

NOTE:Results display only for issued patents and published applications. For pending or abandoned applications please consult USPTO staff.

Total Assignments: 5

Patent #: 6615215 **Issue Dt:** 09/02/2003 **Application #:** 09527247 **Filing Dt:** 03/17/2000

Inventor: W. Clinton Petty

Title: METHOD FOR GRADUATED LOAD SENSITIVE TASK DISPATCHING IN COMPUTING SYSTEM

Assignment: 1

Reel/Frame: 010952/0855 **Recorded:** 07/07/2000 **Pages:** 3

Conveyance: ASSIGNMENT OF ASSIGNORS INTEREST (SEE DOCUMENT FOR DETAILS).

Assignor: PETTY, W. CLINTON **Exec Dt:** 05/15/2000

Assignee: COMMERCEQUEST INC.
3550 WEST WATERS AVENUE
TAMPA, FLORIDA 33614

Correspondent: FOLEY & LARDNER
WILLIAM T. ELLIS
WASHINGTON HARBOUR
3000 K STREET NW, SUITE 500
WASHINGTON, DC 20007-5109

Assignment: 2

Reel/Frame: 013033/0197 **Recorded:** 06/26/2002 **Pages:** 6

Conveyance: SECURITY INTEREST (SEE DOCUMENT FOR DETAILS).

Assignor: COMMERCEQUEST, INC. **Exec Dt:** 06/24/2002

Assignees: BLACKBURN, F. SCOTT
16403 AVILA BOULEVARD
TAMPA, FLORIDA 33613

THOMAS, GUY RICHARD, AS COLLATERAL AGENT
16404 AVILA BOULEVARD
TAMPA, FLORIDA 33613

ROTH, PAUL, AS COLLATERAL AGENT
CAROLINE ROTH 2602 BROOKER TRACE LANE
VALRICO, FLORIDA 33259

ROSENHTAL, ALAN HY, AS COLLATERAL AGENT
902 SPRINGDALE ROAD, N.E.
ATLANTA, GEORGIA 30306

ALLEN HY ROSENTHAL IRREVOCABLE TRUST-DATE MAY 1, 2000, THE
902 SPRINGDALE ROAD, N.E.
ATLANTA, GEORGIA 30306

FORESTER, MICHAEL, AS COLLATERAL AGENT
109 EAST RIDGE DRIVE
LOUISVILLE, MICHIGAN 39339

SUAREZ, JACK, AS COLLATERAL AGENT
16401 AVILA BOULEVARD
TAMPA, FLORIDA 33613

TAYLOR, TODD, AS COLLATERAL AGENT

934 GUISANDO DE AVILA
TAMPA, FLORIDA 33613

STANLEY, PAUL, AS COLLATERAL AGENT
16407 AVILA BOULEVARD
TAMPA, FLORIDA 33613

STANLEY, SHERRI, AS COLLATERIAL AGENT
16407 AVILA BOULEVARD
TAMPA, FLORIDA 33613

Correspondent: FRED DUGUAY, ESQ.
KEVIN M. HANKS
2202 NORTH WEST SHORE BOULEVARD
SUITE 600
TAMPA, FLORIDA 33607

Assignment: 3

Reel/Frame: 017073/0506 **Recorded:** 10/12/2005 **Pages:** 5

Conveyance: ASSIGNMENT OF ASSIGNORS INTEREST (SEE DOCUMENT FOR DETAILS).

Assignor: COMMERCEQUEST, INC. **Exec Dt:** 10/05/2005

Assignee: METASTORM, INC.
8825 STANFORD BOULEVARD, SUITE 200
COLUMBIA, MARYLAND 21045

Correspondent: VENABLE LLP
P.O. BOX 34385
WASHINGTON, DC 20045-9998

Assignment: 4

Reel/Frame: 017663/0392 **Recorded:** 05/24/2006 **Pages:** 16

Conveyance: SECURITY AGREEMENT

Assignor: LANTRONIX, INC. **Exec Dt:** 05/17/2006

Assignee: SILICON VALLEY BANK
3003 TASMAN DRIVE
SANTA CLARA, CALIFORNIA 95054

Correspondent: CBCINNOVIS DBA FEDERAL RESEARCH
1023 FIFTEENTH STREET, NW, STE 401
ATTN: OLEH HERELIUK
WASHINGTON, DC 20005

Assignment: 5

Reel/Frame: 028559/0298 **Recorded:** 07/11/2012 **Pages:** 45

Conveyance: OFFICER'S CERTIFICATE

Assignor: METASTORM INC. **Exec Dt:** 05/29/2012

Assignee: OPEN TEXT S. A.
40, AVENUE MONTEREY
LUXEMBOURG, LUXEMBOURG L-2163

Correspondent: SPRINKLE IP LAW GROUP
1301 W 25TH STREET, SUITE 408
AUSTIN, TX 78705

Search Results as of: 06/15/2013 09:36 PM

If you have any comments or questions concerning the data displayed, contact PRD / Assignments at 571-272-3350. v.2.3.3
Web interface last modified: Apr 8, 2013 v.2.3.3

| HOME | INDEX| SEARCH | eBUSINESS | CONTACT US | PRIVACY STATEMENT

Attorney Docket No.
OPEN1660

Serial No. 09/527,247
Customer No. 109422

EXHIBIT B

United States Patent and Trademark Office

Home Site Index Search Guides Contacts eBusiness eBiz alerts News Help

Assignments on the Web > **Patent Query**

Patent Assignment Abstract of Title

NOTE:Results display only for issued patents and published applications. For pending or abandoned applications please consult USPTO staff.

Total Assignments: 4

Patent #: 6615215 **Issue Dt:** 09/02/2003 **Application #:** 09527247 **Filing Dt:** 03/17/2000

Inventor: W. Clinton Petty

Title: METHOD FOR GRADUATED LOAD SENSITIVE TASK DISPATCHING IN COMPUTING SYSTEM

Assignment: 1

Reel/Frame: 010952/0855 **Recorded:** 07/07/2000 **Pages:** 3

Conveyance: ASSIGNMENT OF ASSIGNORS INTEREST (SEE DOCUMENT FOR DETAILS).

Assignor: PETTY, W. CLINTON **Exec Dt:** 05/15/2000

Assignee: COMMERCEQUEST INC.
3550 WEST WATERS AVENUE
TAMPA, FLORIDA 33614

Correspondent: FOLEY & LARDNER
WILLIAM T. ELLIS
WASHINGTON HARBOUR
3000 K STREET NW, SUITE 500
WASHINGTON, DC 20007-5109

Assignment: 2

Reel/Frame: 013033/0197 **Recorded:** 06/26/2002 **Pages:** 6

Conveyance: SECURITY INTEREST (SEE DOCUMENT FOR DETAILS).

Assignor: COMMERCEQUEST, INC. **Exec Dt:** 06/24/2002

Assignees: BLACKBURN, F. SCOTT
16403 AVILA BOULEVARD
TAMPA, FLORIDA 33613

THOMAS, GUY RICHARD, AS COLLATERAL AGENT
16404 AVILA BOULEVARD
TAMPA, FLORIDA 33613

ROTH, PAUL, AS COLLATERAL AGENT
CAROLINE ROTH 2602 BROOKER TRACE LANE
VALRICO, FLORIDA 33259

ROSENHTAL, ALAN HY, AS COLLATERAL AGENT
902 SPRINGDALE ROAD, N.E.
ATLANTA, GEORGIA 30306

ALLEN HY ROSENTHAL IRREVOCABLE TRUST-DATE MAY 1, 2000, THE
902 SPRINGDALE ROAD, N.E.
ATLANTA, GEORGIA 30306

FORESTER, MICHAEL, AS COLLATERAL AGENT
109 EAST RIDGE DRIVE
LOUISVILLE, MICHIGAN 39339

SUAREZ, JACK, AS COLLATERAL AGENT
16401 AVILA BOULEVARD
TAMPA, FLORIDA 33613

TAYLOR, TODD, AS COLLATERAL AGENT
934 GUISANDO DE AVILA
TAMPA, FLORIDA 33613

STANLEY, PAUL, AS COLLATERAL AGENT

16407 AVILA BOULEVARD
TAMPA, FLORIDA 33613
STANLEY, SHERRI, AS COLLATERAL AGENT
16407 AVILA BOULEVARD
TAMPA, FLORIDA 33613

Correspondent: FRED DUGUAY, ESQ.
KEVIN M. HANKS
2202 NORTH WEST SHORE BOULEVARD
SUITE 600
TAMPA, FLORIDA 33607

Assignment: 3

Reel/Frame: 017073/0506 **Recorded:** 10/12/2005 **Pages:** 5

Conveyance: ASSIGNMENT OF ASSIGNORS INTEREST (SEE DOCUMENT FOR DETAILS).

Assignor: COMMERCEQUEST, INC. **Exec Dt:** 10/05/2005

Assignee: METASTORM, INC.
8825 STANFORD BOULEVARD, SUITE 200
COLUMBIA, MARYLAND 21045

Correspondent: VENABLE LLP
P.O. BOX 34385
WASHINGTON, DC 20045-9998

Assignment: 4

Reel/Frame: 017663/0392 **Recorded:** 05/24/2006 **Pages:** 16

Conveyance: SECURITY AGREEMENT

Assignor: LANTRONIX, INC. **Exec Dt:** 05/17/2006

Assignee: SILICON VALLEY BANK
3003 TASMAN DRIVE
SANTA CLARA, CALIFORNIA 95054

Correspondent: CBCINNOVIS DBA FEDERAL RESEARCH
1023 FIFTEENTH STREET, NW, STE 401
ATTN: OLEH HERELIUK
WASHINGTON, DC 20005

Search Results as of: 05/23/2012 05:11 PM

If you have any comments or questions concerning the data displayed, contact PRD / Assignments at 571-272-3350. v.2.3.1
Web interface last modified: Jan 26, 2012 v.2.3.1

| HOME | INDEX| SEARCH | eBUSINESS | CONTACT US | PRIVACY STATEMENT

EXHIBIT C

PATENT ASSIGNMENT

Electronic Version v1.1
Stylesheet Version v1.1

SUBMISSION TYPE:	NEW ASSIGNMENT
NATURE OF CONVEYANCE:	Security Agreement

CONVEYING PARTY DATA

Name	Execution Date
LANTRONIX, INC.	05/17/2006

RECEIVING PARTY DATA

Name:	SILICON VALLEY BANK
Street Address:	3003 Tasman Drive
City:	Santa Clara
State/Country:	CALIFORNIA
Postal Code:	95054

PROPERTY NUMBERS Total: 23

Property Type	Number
Patent Number:	4972368
Patent Number:	4972470
Patent Number:	5272558
Patent Number:	5410363
Patent Number:	6571305
Patent Number:	6615215
Patent Number:	6615272
Patent Number:	6881096
Patent Number:	6898660
Patent Number:	6922748
Application Number:	10791109
Application Number:	10791110
Application Number:	10896088
Application Number:	10929858
Application Number:	10931539

Application Number:	10909981
Application Number:	11060664
Application Number:	11075266
Application Number:	11084342
Application Number:	10712084
Application Number:	09780985
Application Number:	11031643
Application Number:	10229251

CORRESPONDENCE DATA

Fax Number: (866)459-2899
Correspondence will be sent via US Mail when the fax attempt is unsuccessful.
Phone: 202-783-2700
Email: Oleh.Hereliuk@federalresearch.com
Correspondent Name: CBCInnovis dba Federal Research
Address Line 1: 1023 Fifteenth Street, NW, Ste 401
Address Line 2: attn: Oleh Hereliuk
Address Line 4: Washington, DISTRICT OF COLUMBIA 20005

ATTORNEY DOCKET NUMBER:	359157
NAME OF SUBMITTER:	Oleh Hereliuk

Total Attachments: 15
source=359157#page1.tif
source=359157#page2.tif
source=359157#page3.tif
source=359157#page4.tif
source=359157#page5.tif
source=359157#page6.tif
source=359157#page7.tif
source=359157#page8.tif
source=359157#page9.tif
source=359157#page10.tif
source=359157#page11.tif
source=359157#page12.tif
source=359157#page13.tif
source=359157#page14.tif
source=359157#page15.tif

ISSUED PATENTS

Description	Registration/ Application Number	Registration/ Application Date
Intelligent Serial I/O Subsystem	4,972,368	Nov. 20, 1990
Programmable Connector	4,972,470	Nov. 20, 1990
Two Level Fiber Optic Communication from Three-Valve Electronic Signal Source	5,272,558	Dec. 21, 1993
Automatic Gain Control Device for Transmitting Video Signals between 2 locations	5,410,363	April 25, 1995
System for Extending Length of a Connection to a USB Peripheral	6,571,305	May 27, 2003
Switch Node for Connecting a Keyboard Video Mouse to Selected Servers in a Interconnected Switch Node Network	6,615,215	Sept. 2, 2003
Switch Node for Connecting a Keyboard Video Mouse to Selected Servers in a Interconnected Switch Node Network	6,615,272	Sept. 2, 2003
Compact Serial To Ethernet Conversion Port	6,881,096	April 19, 2005
System for Extending Length of a Connection to a USB Peripheral	6,898,660	May 24, 2005
System for Extending Length of a Connection to a USB Peripheral	6,922,748	July 26, 2005

PENDING PATENTS

Description	Registration/ Application Number	Registration/ Application Date
System and Method for Debugging Software Applications on Remote Devices	10/791,109	March 2, 2004
Method and System For Program Transformation in Managed Runtime Environments Based Upon Flow Sensitive Local Type Constraint Analysis	10/791,110	March 2, 2004
Secure Data Transfer Using an Embedded System	10/896,088	July 21, 2004
Secure COM Port Redirector Overview	10/929,858	Aug. 30, 2004
Method and System for Program Transformation	10/931,539	Sept. 1, 2004
In-Band Firewall for an Embedded System	10/909,981	Sept. 3, 2004
Serial-To-Ethernet Conversion Port	11/060,664	Feb. 17, 2005
Data Syndication Apparatus and Methods	11/075,266	March 7, 2005
Wireless Communication Port	11/084,342	March 17, 2005
Communication Protocol Converter & Method of Protocol Conversion	10/712,084	June 14, 2005
Sister Application to Data Communication Controller & Methods	09/780,985	Nov. 15, 2004
Remote Management Module and Methods	11/031,643	-----
Method and System for Dynamic Distributed Object-Oriented Environment	10/229,251	Aug. 27, 2002

INTELLECTUAL PROPERTY SECURITY AGREEMENT

This Intellectual Property Security Agreement is entered into as of May 17, 2006 by and between SILICON VALLEY BANK ("Secured Party") and LANTRONIX, INC. ("Grantor").

RECITALS

A. Secured Party and Grantor are entering into that certain Loan and Security Agreement of even date herewith (as the same may be amended, modified or supplemented from time to time, the "Loan Agreement"; capitalized terms used herein which are not defined, have the meanings set forth in the Loan Agreement). Secured Party and Grantor are entering into that certain Loan and Security Agreement (Exim Program) of even date herewith (as the same may be amended, modified or supplemented from time to time, the "Exim Loan Agreement"; capitalized terms used herein which are not defined, have the meanings set forth in the Exim Loan Agreement).

B. Pursuant to the terms of the Loan Agreement and the Exim Loan Agreement, Grantor has granted to Secured Party a security interest in all of Grantor's right, title and interest, whether presently existing or hereafter acquired, in, to all Intellectual Property and all other Collateral.

NOW, THEREFORE, as collateral security for the payment and performance when due of all of the Obligations, Grantor hereby grants, represents, warrants, covenants and agrees as follows:

AGREEMENT

1. Grant of Security Interest. To secure all of the Obligations, Grantor grants and pledges to Secured Party a security interest in all of Grantor's right, title and interest in, to and under its Intellectual Property (as defined in the Loan Agreement and Exim Loan Agreement, respectively), including without limitation the following:

(a) All of present and future United States registered copyrights and copyright registrations, including, without limitation, the registered copyrights, maskworks, software, computer programs and other works of authorship subject to United States copyright protection listed in Exhibit A-1 to this Agreement (and including all of the exclusive rights afforded a copyright registrant in the United States under 17 U.S.C. §106 and any exclusive rights which may in the future arise by act of Congress or otherwise) and all present and future applications for copyright registrations (including applications for copyright registrations of derivative works and compilations) (collectively, the "Registered Copyrights"), and any and all royalties, payments, and other amounts payable to Grantor in connection with the Registered Copyrights, together with all renewals and extensions of the Registered Copyrights, the right to recover for all past, present, and future infringements of the Registered Copyrights, and all computer programs, computer databases, computer program flow diagrams, source codes, object codes and all tangible property embodying or incorporating the Registered Copyrights, and all other rights of every kind whatsoever accruing thereunder or pertaining thereto.

(b) All present and future copyrights, maskworks, software, computer programs and other works of authorship subject to (or capable of becoming subject to) United States copyright protection which are not registered in the United States Copyright Office (the "Unregistered Copyrights"), whether now owned or hereafter acquired, including without limitation the Unregistered Copyrights listed in Exhibit A-2 to this Agreement, and any and all royalties, payments, and other amounts payable to Grantor in connection with the Unregistered Copyrights, together with all renewals and extensions of the Unregistered Copyrights, the right to recover for all past, present, and future infringements of the Unregistered Copyrights, and all computer programs, computer databases, computer program flow diagrams, source codes, object codes and all tangible property embodying or incorporating the Unregistered Copyrights, and all other rights of every kind whatsoever accruing thereunder or pertaining thereto. The Registered Copyrights and the Unregistered Copyrights collectively are referred to herein as the "Copyrights."

(c) All right, title and interest in and to any and all present and future license agreements with respect to the Copyrights.

(d) All present and future accounts, accounts receivable, royalties, and other rights to payment arising from, in connection with or relating to the Copyrights.

(e) All patents, patent applications and like protections including, without limitation, improvements, divisions, continuations, renewals, reissues, extensions and continuations-in-part of the same, including without limitation the patents and patent applications set forth on Exhibit B attached hereto (collectively, the "Patents");

(f) All trademark and servicemark rights, whether registered or not, applications to register and registrations of the same and like protections, and the entire goodwill of the business of Grantor connected with and symbolized by such trademarks, including without limitation those set forth on Exhibit C attached hereto (collectively, the "Trademarks");

(g) Any and all claims for damages by way of past, present and future infringements of any of the rights included above, with the right, but not the obligation, to sue for and collect such damages for said use or infringement of the rights identified above;

(h) All licenses or other rights to use any of the Copyrights, Patents or Trademarks, and all license fees and royalties arising from such use to the extent permitted by such license or rights;

(i) All amendments, extensions, renewals and extensions of any of the Copyrights, Trademarks or Patents; and

(j) All proceeds and products of the foregoing, including without limitation all payments under insurance or any indemnity or warranty payable in respect of any of the foregoing, and all license royalties and proceeds of infringement suits, and all rights corresponding to the foregoing throughout the world and all re-issues, divisions continuations, renewals, extensions and continuations-in-part of the foregoing.

2. Loan Agreement. This security interest is granted in conjunction with the security interest granted to Secured Party under the Loan Agreement and Exim Loan Agreement, respectively. The rights and remedies of Secured Party with respect to the security interest granted hereby are in addition to those set forth in the Loan Agreement and Exim Loan Agreement, respectively, and the other Loan Documents, and those which are now or hereafter available to Secured Party as a matter of law or equity. Each right, power and remedy of Secured Party provided for herein or in the Loan Agreement, Exim Loan Agreement or any of the other Loan Documents, or now or hereafter existing at law or in equity shall be cumulative and concurrent and shall be in addition to every right, power or remedy provided for herein and the exercise by Secured Party of any one or more of the rights, powers or remedies provided for in this Agreement, the Loan Agreement, the Exim Loan Agreement or any of the other Loan Documents, or now or hereafter existing at law or in equity, shall not preclude the simultaneous or later exercise by any person, including Secured Party, of any or all other rights, powers or remedies.

3. Covenants and Warranties. Grantor represents, warrants, covenants and agrees as follows:

(a) Grantor has no present maskworks, software, computer programs and other works of authorship registered with the United States Copyright Office except as disclosed on Exhibit A-1 hereto.

(b) Grantor shall undertake all reasonable measures to cause its employees, agents and independent contractors to assign to Grantor all rights of authorship to any copyrighted material in which Grantor has or may subsequently acquire any right or interest.

(c) Grantor shall promptly advise Secured Party of any Trademark, Patent or Copyright not specified in this Agreement, which is hereafter acquired by Grantor.

(d) Grantor shall not register any maskworks, software, computer programs or other works of authorship subject to United States copyright protection with the United States Copyright Office without first complying with the following: (i) providing Secured Party with at least 15 days prior written notice thereof, (ii) providing Secured Party with a copy of the application for any such registration and (iii) executing and filing such other instruments, and taking such further actions as Secured Party may reasonably request from time to time to perfect or continue the perfection of Secured Party's interest in the Collateral, including without limitation the filing with the United States Copyright Office, simultaneously with the filing by Grantor of the application for any such registration, of a copy of this Agreement or a Supplement hereto in form acceptable to Secured Party identifying the maskworks, software, computer programs or other works of authorship being registered and confirming the grant of a security interest therein in favor of Secured Party.

4. General. If any action relating to this Agreement is brought by either party hereto against the other party, the prevailing party shall be entitled to recover reasonable attorneys fees, costs and disbursements. This Agreement may be amended only by a written instrument signed by both parties hereto. To the extent that any provision of this Agreement conflicts with any provision of the Loan Agreement and/or the Exim Loan Agreement, the provision giving Secured

Party greater rights or remedies shall govern, it being understood that the purpose of this Agreement is to add to, and not detract from, the rights granted to Secured Party under the Loan Agreement and Exim Loan Agreement, respectively. This Agreement, the Loan Agreement, the Exim Loan Agreement and the other Loan Documents comprise the entire agreement of the parties with respect to the matters addressed in this Agreement. This Agreement shall be governed by the laws of the State of California, without regard for choice of law provisions. Grantor and Secured Party consent to the nonexclusive jurisdiction of any state or federal court located in Santa Clara County, California.

5. <u>WAIVER OF RIGHT TO JURY TRIAL</u>. SECURED PARTY AND GRANTOR EACH HEREBY WAIVE THE RIGHT TO TRIAL BY JURY IN ANY ACTION OR PROCEEDING BASED UPON, ARISING OUT OF, OR IN ANY WAY RELATING TO: (I) THIS AGREEMENT; OR (II) ANY OTHER PRESENT OR FUTURE INSTRUMENT OR AGREEMENT BETWEEN SECURED PARTY AND GRANTOR; OR (III) ANY CONDUCT, ACTS OR OMISSIONS OF SECURED PARTY OR GRANTOR OR ANY OF THEIR DIRECTORS, OFFICERS, EMPLOYEES, AGENTS, ATTORNEYS OR ANY OTHER PERSONS AFFILIATED WITH SECURED PARTY OR GRANTOR; IN EACH OF THE FOREGOING CASES, WHETHER SOUNDING IN CONTRACT OR TORT OR OTHERWISE.

IN WITNESS WHEREOF, the parties have cause this Intellectual Property Security Agreement to be duly executed by its officers thereunto duly authorized as of the first date written above.

Address of Grantor:

15353 Barranca Parkway
Irvine, CA 92618

DIRECTOR OF FINANCE
[illegible] R. [illegible]

Grantor:

Lantronix, Inc.

By: J W Kerrigan
Title: CFO
Name: JAMES W. KERRIGAN

Address of Secured Party:

3003 Tasman Drive
Santa Clara, California 95054

Secured Party:

SILICON VALLEY BANK

By: ______________________
Title: Vice President

EXHIBIT A-1

REGISTERED COPYRIGHTS

(including copyrights that are the subject of an application for registration)

Description	Registration/ Application Number	Registration/ Application Date
EPS & ETS printer/terminal servers	TX-4-201-719	January 29, 1996

EXHIBIT A-2

UNREGISTERED COPYRIGHTS

None

EXHIBIT B

ISSUED PATENTS

Description	Registration/ Application Number	Registration/ Application Date
Intelligent Serial I/O Subsystem	4,972,368	Nov. 20, 1990
Programmable Connector	4,972,470	Nov. 20, 1990
Two Level Fiber Optic Communication from Three-Valve Electronic Signal Source	5,272,558	Dec. 21, 1993
Automatic Gain Control Device for Transmitting Video Signals between 2 locations	5,410,363	April 25, 1995
System for Extending Length of a Connection to a USB Peripheral	6,571,305	May 27, 2003
Switch Node for Connecting a Keyboard Video Mouse to Selected Servers in a Interconnected Switch Node Network	6,615,215	Sept. 2, 2003
Switch Node for Connecting a Keyboard Video Mouse to Selected Servers in a Interconnected Switch Node Network	6,615,272	Sept. 2, 2003
Compact Serial To Ethernet Conversion Port	6,881,096	April 19, 2005
System for Extending Length of a Connection to a USB Peripheral	6,898,660	May 24, 2005
System for Extending Length of a Connection to a USB Peripheral	6,922,748	July 26, 2005

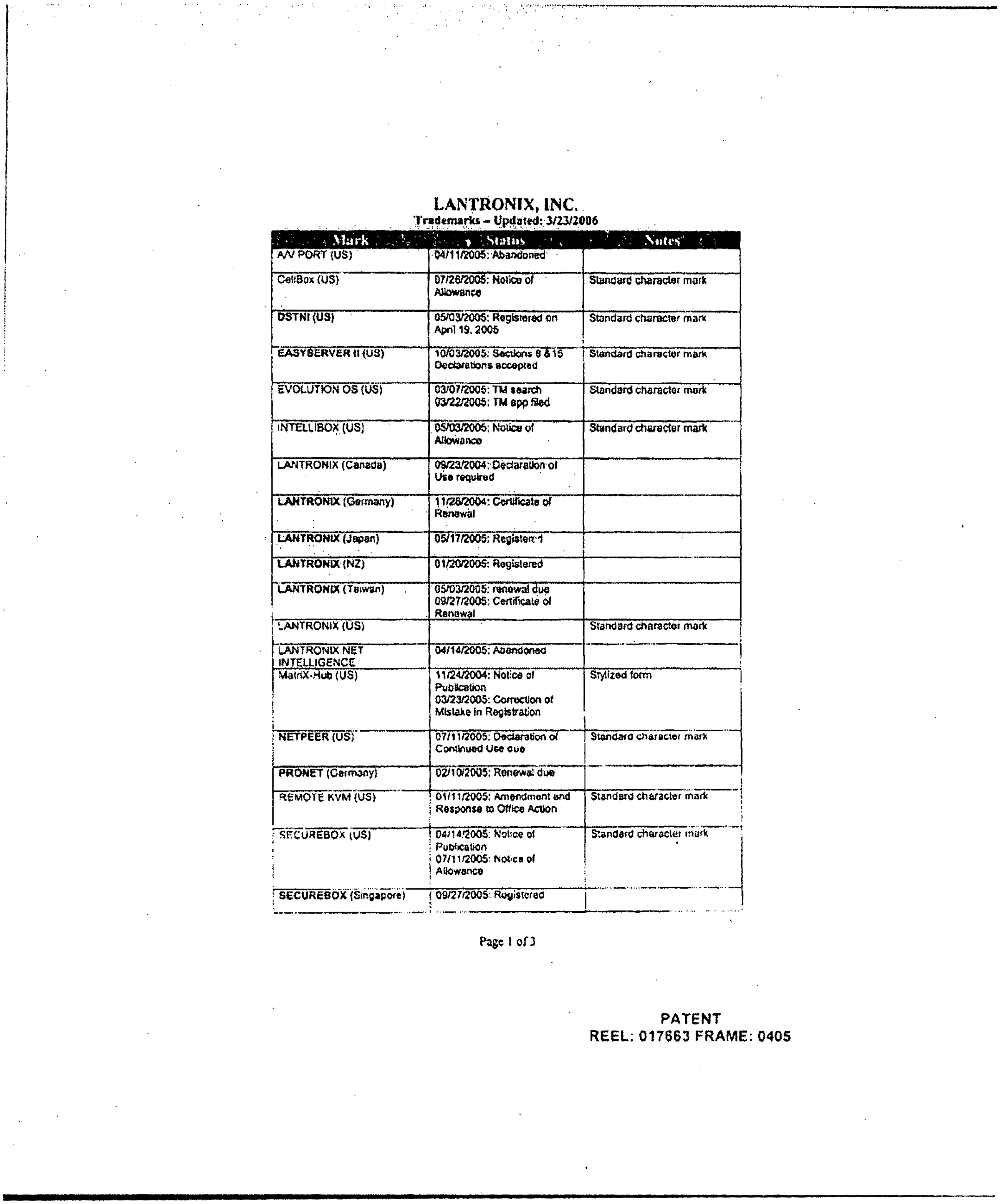
PENDING PATENTS

Description	Registration/ Application Number	Registration/ Application Date
System and Method for Debugging Software Applications on Remote Devices	10/791,109	March 2, 2004
Method and System For Program Transformation in Managed Runtime Environments Based Upon Flow Sensitive Local Type Constraint Analysis	10/791,110	March 2, 2004
Secure Data Transfer Using an Embedded System	10/896,088	July 21, 2004
Secure COM Port Redirector Overview	10/929,858	Aug. 30, 2004
Method and System for Program Transformation	10/931,539	Sept. 1, 2004
In-Band Firewall for an Embedded System	10/909,981	Sept. 3, 2004
Serial-To-Ethernet Conversion Port	11/060,664	Feb. 17, 2005
Data Syndication Apparatus and Methods	11/075,266	March 7, 2005
Wireless Communication Port	11/084,342	March 17, 2005
Communication Protocol Converter & Method of Protocol Conversion	10/712,084	June 14, 2005
Sister Application to Data Communication Controller & Methods	09/780,985	Nov. 15, 2004
Remote Management Module and Methods	11/031,643	-----
Method and System for Dynamic Distributed Object-Oriented Environment	10/229,251	Aug. 27, 2002

EXHIBIT C

TRADEMARKS

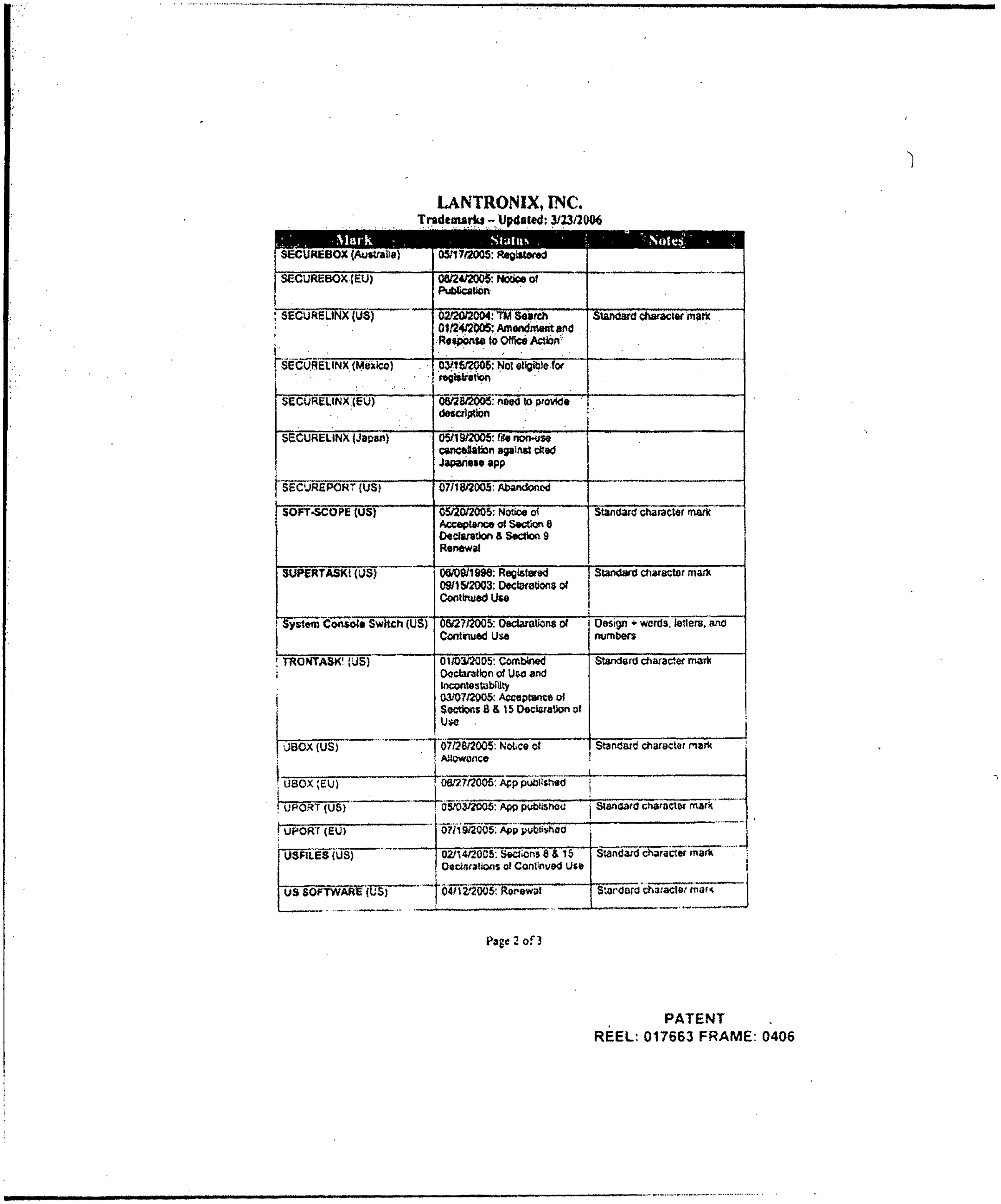
Description	Registration/ Application Number	Registration/ Application Date
SEE ATTACHED		



LANTRONIX, INC.

Trademarks – Updated: 3/23/2006

Mark	Status	Notes
A/V PORT (US)	04/11/2005: Abandoned	
CellBox (US)	07/28/2005: Notice of Allowance	Standard character mark
DSTNI (US)	05/03/2005: Registered on April 19, 2005	Standard character mark
EASYSERVER II (US)	10/03/2005: Sections 8 & 15 Declarations accepted	Standard character mark
EVOLUTION OS (US)	03/07/2005: TM search 03/22/2005: TM app filed	Standard character mark
INTELLIBOX (US)	05/03/2005: Notice of Allowance	Standard character mark
LANTRONIX (Canada)	09/23/2004: Declaration of Use required	
LANTRONIX (Germany)	11/26/2004: Certificate of Renewal	
LANTRONIX (Japan)	05/17/2005: Registen 1	
LANTRONIX (NZ)	01/20/2005: Registered	
LANTRONIX (Taiwan)	05/03/2005: renewal due 09/27/2005: Certificate of Renewal	
LANTRONIX (US)		Standard character mark
LANTRONIX NET INTELLIGENCE	04/14/2005: Abandoned	
MatriX-Hub (US)	11/24/2004: Notice of Publication 03/23/2005: Correction of Mistake in Registration	Stylized form
NETPEER (US)	07/11/2005: Declaration of Continued Use due	Standard character mark
PRONET (Germany)	02/10/2005: Renewal due	
REMOTE KVM (US)	01/11/2005: Amendment and Response to Office Action	Standard character mark
SECUREBOX (US)	04/14/2005: Notice of Publication 07/11/2005: Notice of Allowance	Standard character mark
SECUREBOX (Singapore)	09/27/2005: Registered	



LANTRONIX, INC.

Trademarks – Updated: 3/23/2006

Mark	Status	Notes
SECUREBOX (Australia)	05/17/2005: Registered	
SECUREBOX (EU)	08/24/2005: Notice of Publication	
SECURELINX (US)	02/20/2004: TM Search 01/24/2005: Amendment and Response to Office Action	Standard character mark
SECURELINX (Mexico)	03/15/2005: Not eligible for registration	
SECURELINX (EU)	06/28/2005: need to provide description	
SECURELINX (Japan)	05/19/2005: file non-use cancellation against cited Japanese app	
SECUREPORT (US)	07/18/2005: Abandoned	
SOFT-SCOPE (US)	05/20/2005: Notice of Acceptance of Section 8 Declaration & Section 9 Renewal	Standard character mark
SUPERTASK! (US)	06/09/1998: Registered 09/15/2003: Declarations of Continued Use	Standard character mark
System Console Switch (US)	06/27/2005: Declarations of Continued Use	Design + words, letters, and numbers
TRONTASK! (US)	01/03/2005: Combined Declaration of Use and Incontestability 03/07/2005: Acceptance of Sections 8 & 15 Declaration of Use	Standard character mark
UBOX (US)	07/28/2005: Notice of Allowance	Standard character mark
UBOX (EU)	06/27/2005: App published	
UPORT (US)	05/03/2005: App published	Standard character mark
UPORT (EU)	07/19/2005: App published	
USFILES (US)	02/14/2005: Sections 8 & 15 Declarations of Continued Use	Standard character mark
US SOFTWARE (US)	04/12/2005: Renewal	Standard character mark

LANTRONIX, INC.

Trademarks – Updated: 3/23/2006

Mark	Status	Notes
Viewtaski	10/03/2005: registered on April 2, 1996 (acquired from USSW) – Affidavit of Continued Use (Section 8) required.	
VDE/200 (US)	05/03/2005: Sections 8 & 9 Declarations of Renewal	Standard charactor mark
WIBOX (US)	07/26/2005: Notice of Allowance	Standard character mark
WIBOX (Australia)	05/17/2005: Registered	
WIBOX (Japan)	05/17/2005: Registered	
WIBOX (EU)	06/20/2005: App published	
WIPORT (Australia)	09/27/2005: Registered	
WIPORT (EU)	07/11/2005: App published	
WIPORT (US)		Standard charactor mark
WIPORT (Mexico)	05/17/2005: Registered	
WIPORT (Singapore)	07/19/2005: App published	
XPORT (US)	05/19/2005: Notice of Publication	Standard character mark
XPORT (EU)	06/24/2005: App published	
XPORT (Mexico)	05/17/2005: Registered	
XPORT AR (US)	03/08/2005: App filed	Standard character mark
XPORT AR (Australia)	07/19/2005: App published	

EXHIBIT D

US006615215B1

(12) **United States Patent**
Petty

(10) **Patent No.: US 6,615,215 B1**
(45) **Date of Patent: Sep. 2, 2003**

(54) **METHOD FOR GRADUATED LOAD SENSITIVE TASK DISPATCHING IN COMPUTING SYSTEM**

(75) Inventor: **W. Clinton Petty**, Reston, VA (US)

(73) Assignee: **CommerceQuest Inc.**, Tampa, FL (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/527,247**

(22) Filed: **Mar. 17, 2000**

(51) **Int. Cl.**^{7} **G06F 17/30**
(52) **U.S. Cl.** **707/101**
(58) **Field of Search** 707/100, 101; 709/102, 226, 231, 201, 313, 314, 319; 372/232, 412; 370/252; 375/214

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,918,687 A	*	4/1990	Bustini et al.	370/235
4,945,548 A	*	7/1990	Iannarone et al.	375/214
6,023,722 A	*	2/2000	Colyer	709/201
6,167,032 A	*	12/2000	Allison et al.	370/252

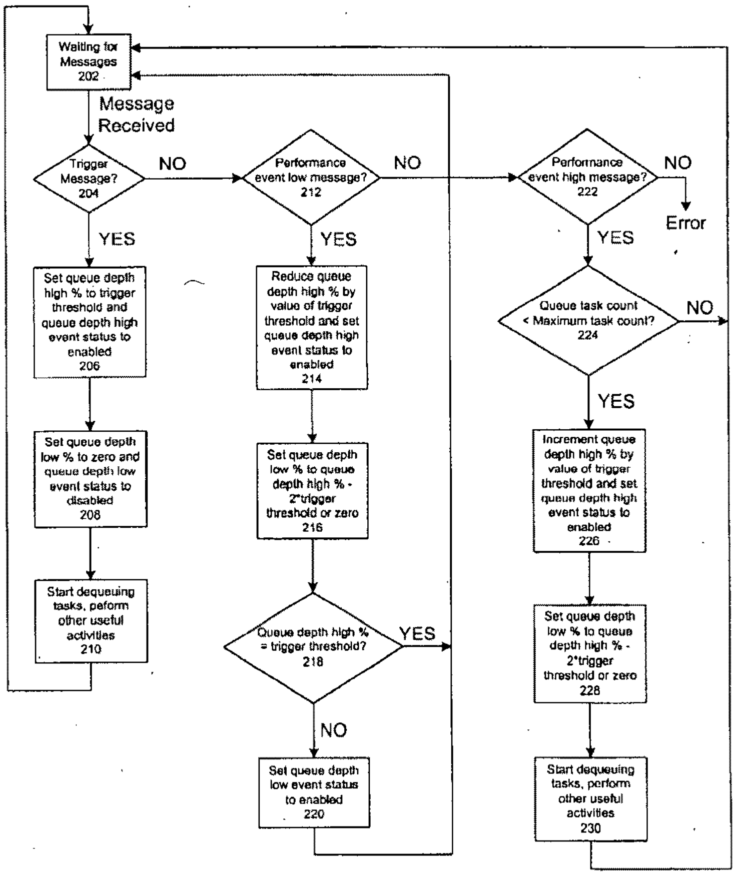
* cited by examiner

Primary Examiner—Diane D. Mizrahi
Assistant Examiner—Apu M Mofiz
(74) *Attorney, Agent, or Firm*—Foley & Lardner

(57) **ABSTRACT**

A system and method for a dynamic response to trigger messages and high and low performance event messages supports a graduated dispatching of tasks for processing messages in a queue. As messages are enqueued, queue attributes are altered, including an override of the standard flip-flop behavior of the queue depth high and low event statuses. The alteration of the queue attributes creates a bracketing at the point at which the last event occurred with a pair of "tripwires." The tripwires are continuously kept on either side of the queue depth at which the last performance event was generated. When the depth changes enough to cause the depth to cross one of the tripwires, the tripwires are moved, and dispatching of tasks for processing messages in the queue may be executed or any other logic deemed useful that is sensitive to queue depths.

25 Claims, 3 Drawing Sheets


Waiting for Messages 202
Message Received
Trigger Message? 204
NO
YES
Set queue depth high % to trigger threshold and queue depth high event status to enabled 206
Set queue depth low % to zero and queue depth low event status to disabled 208
Start dequeuing tasks, perform other useful activities 210
Performance event low message? 212
NO
YES
Reduce queue depth high % by value of trigger threshold and set queue depth high event status to enabled 214
Set queue depth low % to queue depth high % - 2*trigger threshold or zero 216
Queue depth high % = trigger threshold? 218
YES
NO
Set queue depth low event status to enabled 220
Performance event high message? 222
NO
Error
YES
Queue task count < Maximum task count? 224
NO
YES
Increment queue depth high % by value of trigger threshold and set queue depth high event status to enabled 226
Set queue depth low % to queue depth high % - 2*trigger threshold or zero 228
Start dequeuing tasks, perform other useful activities 230

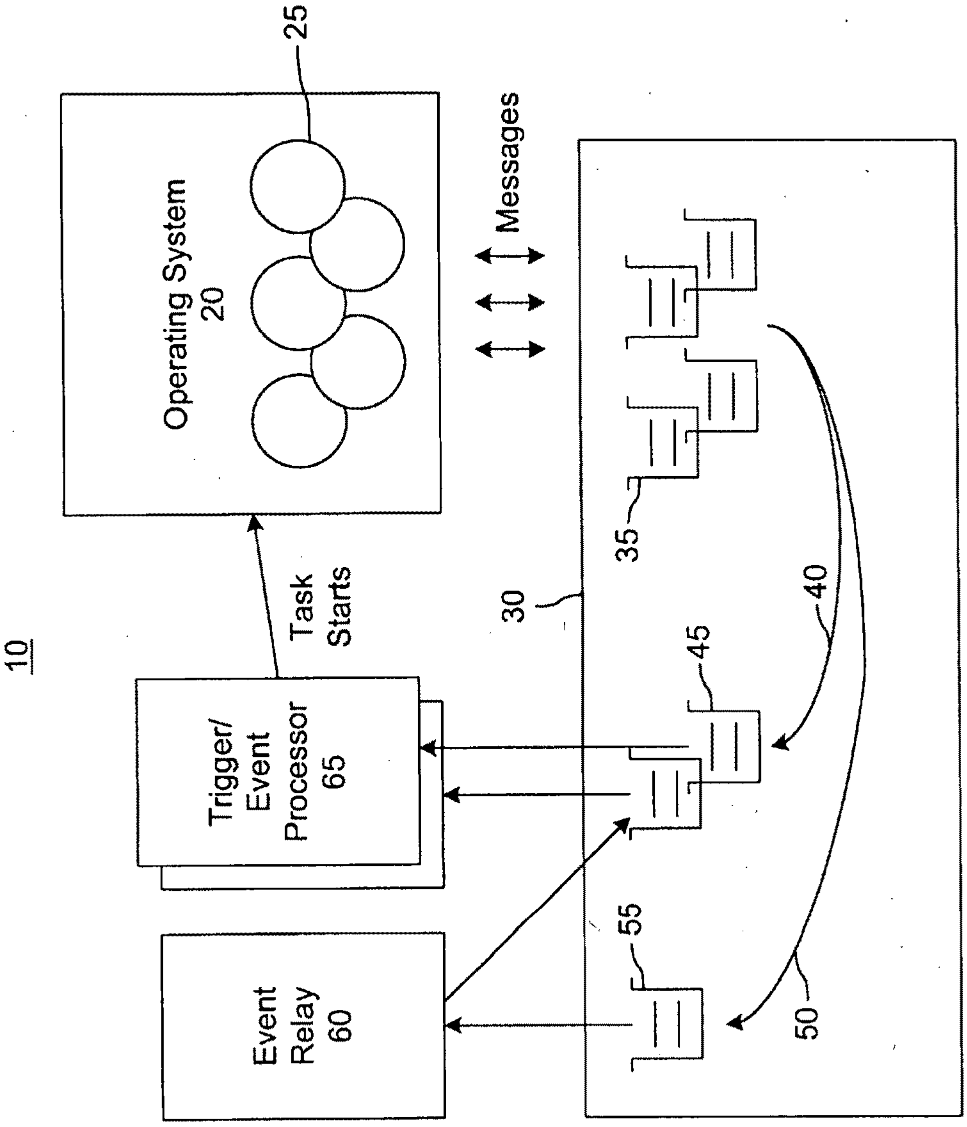

10
Operating System
20
25
Messages
Task
Starts
Trigger/
Event
Processor
65
Event
Relay
60
30
35
40
45
55
50

FIG. 1

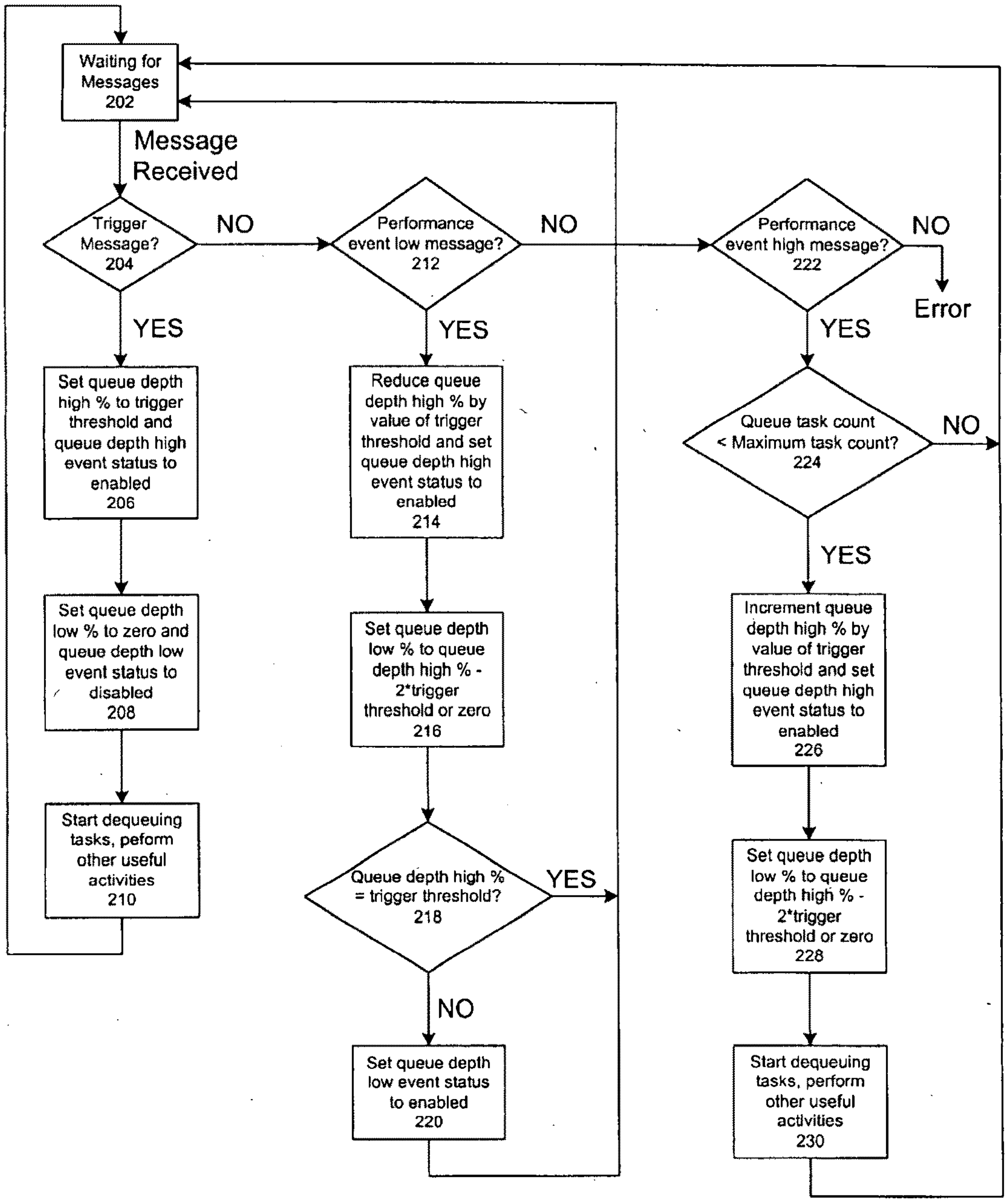

Waiting for Messages 202
Message Received
Trigger Message? 204
NO
YES
Set queue depth high % to trigger threshold and queue depth high event status to enabled 206
Set queue depth low % to zero and queue depth low event status to disabled 208
Start dequeuing tasks, peform other useful activities 210
Performance event low message? 212
NO
YES
Reduce queue depth high % by value of trigger threshold and set queue depth high event status to enabled 214
Set queue depth low % to queue depth high % - 2*trigger threshold or zero 216
Queue depth high % = trigger threshold? 218
YES
NO
Set queue depth low event status to enabled 220
Performance event high message? 222
NO
Error
YES
Queue task count < Maximum task count? 224
NO
YES
Increment queue depth high % by value of trigger threshold and set queue depth high event status to enabled 226
Set queue depth low % to queue depth high % - 2*trigger threshold or zero 228
Start dequeuing tasks, perform other useful activities 230

FIG. 2

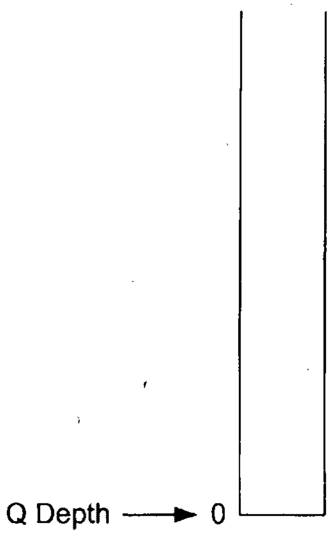


FIG. 3A

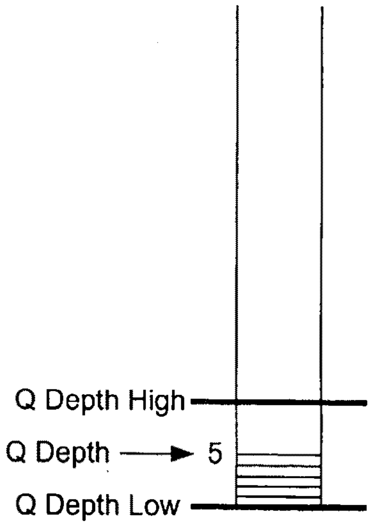


FIG. 3B

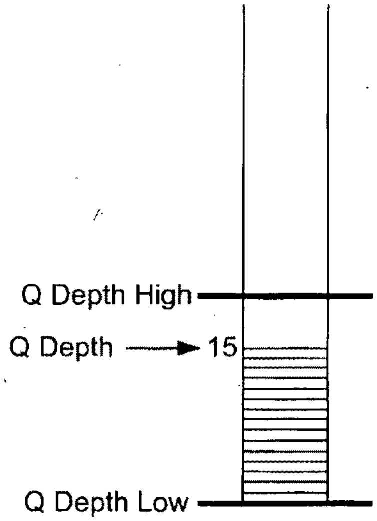


FIG. 3C

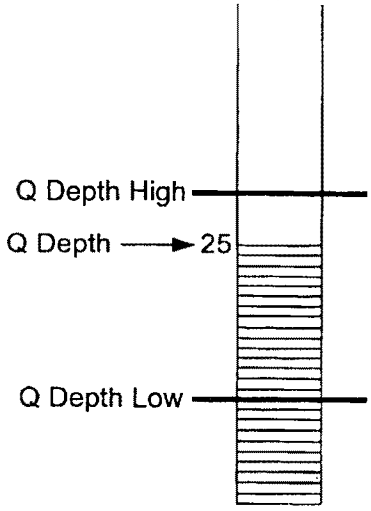


FIG. 3D

METHOD FOR GRADUATED LOAD SENSITIVE TASK DISPATCHING IN COMPUTING SYSTEM

This application is based on provisional application Ser. No. 60/125,021 filed Mar. 18, 1999.

FIELD OF THE INVENTION

The invention relates generally to the field of message processing, and more specifically, to a system and method for adjusting the processing of messages according to changes in queue depth.

BACKGROUND OF THE INVENTION

MQSeries™, which is a product of the International Business Machines Corp. (IBM), performs message queuing and transport within and among a large variety of computer system types and networks. This product enables a computer program to place (enqueue) a data message into a named queue, the data message generally being picked up by another computer program. It is also possible that the queue designated as the target of the enqueue actually resides in a different physical computer. In this case, MQSeries accepts the enqueue request into the local system and then asynchronously transmits the message via a network to the remote system in which the queue is actually found. The message may then be dequeued by a program on that system. As described above, MQSeries enables different computer programs to exchange information via messages on queues and to maintain ignorance of where the recipient of a sent message is or the type of network implemented between the host machines.

In addition to this primary functionality, there are several ancillary features that aid in the use and management of MQSeries. Among these ancillary features are the performance event message feature and the trigger feature.

The performance event message feature is designed to report a "significant" change in the depth of a queue, where the depth of the queue corresponds to the number of messages in the queue. What constitutes a significant change is determined by a combination of settings, described below, which may be specified by a system administrator in MQSeries. The settings are attributes of all "local" queues, which are queues that actually hold or can hold messages.

A first of these attributes of a queue is a maximum depth attribute. This attribute sets the maximum number of messages that a queue can hold. After the maximum is reached, no more enqueues, i.e. placement of messages, will be accepted into the queue. Another attribute of each queue is a queue depth high percentage attribute. This attribute sets a percentage of the maximum depth value that is recognized as a high threshold. If the depth of the queue equals or exceeds the number of messages represented by this value, the queue is a candidate for generation of a performance event-high message.

The value of a queue depth high event status attribute determines whether to check if the high threshold has been reached. This attribute is a flag value that can be set to enabled or disabled. If the flag is enabled, then a performance event-high message will be generated whenever the number of messages in the queue reaches or exceeds the high threshold, which corresponds to the percentage of the maximum depth set by the queue depth high percentage. If the status is set to disabled, however, the performance event-high message will be suppressed. Anytime a performance event-high message is generated, the queue depth high event status attribute is automatically set to disabled, and a queue depth low event status attribute, described below, is set to enabled.

Like the high threshold attributes, MQSeries also maintains attributes for a low threshold. Among the low threshold attributes is a queue depth low percentage attribute, which sets a low threshold as a percentage of the maximum depth value described above. If the depth of a queue is equal to or less than the low threshold, the queue is a candidate for generation of a performance event-low message.

Each queue also has a queue depth low event status attribute. This attribute represents a flag value that can be set to enabled or disabled. If the flag is enabled, then a performance event-low will be generated whenever the number of messages in the queue is less than or equal to the low threshold, which corresponds to the percentage of the maximum depth set by the queue depth low percentage. If the status is set to disabled, the event will be suppressed. Anytime a performance event-low message is generated, the queue depth low event status attribute is automatically set to disabled, and the queue depth high event status attribute is set to enabled.

In addition to these local queue attributes, MQSeries implements special facilities for the delivery of events to a monitoring computer program. Part of this implementation is a performance event queue. The performance event queue is a specially named local queue into which MQSeries places performance event messages, such as the performance event-low and performance event-high messages. There is only one performance event queue in each executing instance of MQSeries. If a program wishes to receive the messages placed in the performance event queue, the program opens the queue and waits for the messages to appear using the same application programming interfaces (APIs) provided by MQSeries for use in dequeuing messages from any of its local queues. The performance event messages are enqueued synchronously to the performance event queue with the messaging event, such as an enqueue or dequeue on some queue that caused the performance event message to be generated.

A typical use of the above features can be exemplified with a queue defined with the following attributes: maximum depth set to one thousand; queue depth high percentage set to ten; and queue depth high event status set to enabled. With maximum depth set to one thousand and queue depth high percentage set to 10, the high threshold is 100. In addition, queue depth low percentage is set to one, and queue depth low event status is set to disabled. With queue depth low percentage set to 1, the low threshold is 10.

As messages first begin to arrive in the queue, no performance event is recognized. Although the queue is a candidate for a performance event-low each time an enqueue occurs and the resulting queue depth is ten or less, the event is suppressed because the queue depth low event status is disabled. When the queue's depth reaches one hundred, the queue becomes a candidate for generation of a performance event-high message. This performance event message will be generated because the queue depth high event status is enabled. At the same time the performance event message is generated, the queue depth high event status will be set to disabled, and the queue depth low event status will be set to enabled. If over time the depth of the queue drops back to ten or less, then the performance event-low message will be generated, the queue depth low event status will be disabled, and the queue depth high event status will be re-enabled.

As demonstrated by the above example, the queue depth high event status and the queue depth low event status are designed to operate in a flip-flop manner, such that when one is enabled, the other is disabled and vise versa. This flip-flop manner supports a model often used in many computer system management products, in which various managed resources are seen as being either in an alertable state or a normal state. If the resource was included as an item in a management display panel, the visual representation of the resource may, for example, change its color to red when in an alertable state, and then back to green when the state returns to normal. When the alertable state is displayed, computer operations procedures may be invoked as a response to the state with the intent of returning the resource to a normal state.

While this behavior is well suited to a systems management model for resource monitoring, it is not well suited for a task dispatching monitoring model. MQSeries provides a trigger feature for task dispatching. There is no coordination, however, between the performance event message feature and the trigger feature in the MQSeries product.

With respect to the trigger feature, MQSeries employs a set of facilities for starting computer programs when queues become active, such as when messages have arrived in a queue, with the intent that the computer programs can dequeue and process the messages that have arrived. An instance of such a computer program in execution is referred to herein as a "task." The following are the key elements of the trigger feature.

A first of these key elements is a trigger message. The trigger message is a specially formatted message generated by MQSeries, which indicates that a local queue has become "active." The trigger message is defined with attributes indicating that a task is supposed to startup and process the messages arriving in the local queue.

An initiation queue is an otherwise ordinary local queue, which is designated by the system administrator as being the target of MQSeries generated trigger messages related to one or more local queues. The initiation queue is analogous to the performance event queue discussed above. Any local queue supported by triggering facilities must be related to a respective initiation queue so that MQSeries will know where to deliver any trigger messages that are generated in conjunction with that local queue. A trigger Monitor is a computer program that monitors information generated by MQSeries for use in dispatching tasks to process messages arriving in queues.

MQSeries supports three types of triggering. A first trigger means that a trigger message should be generated when the queue goes from empty to non-empty if no tasks are currently executing in association with the queue. A depth trigger means that a trigger message should be generated when the queue reaches a specific depth if no tasks are currently executing in association with the queue. An every trigger means that a trigger message should be generated every time a new message is enqueued to the queue.

Both the first and depth trigger types respond to the queue reaching a certain predefined condition to trigger the generation of a single trigger message. Assuming a task is started in response to the generated trigger message, no further trigger messages will be generated because the presence of even one task dequeuing messages from the queue will suppress further trigger message generation based upon the first and depth trigger types.

The every trigger type causes a trigger message to be generated every time a new message is enqueued to the queue. Even if an existing task immediately dequeues a newly arrived message, a trigger message will still be created by the every trigger when the message arrives. These three trigger types have a static behavior. As a result, the number of tasks started in association with a queue cannot be related intelligently to the current enqueue/dequeue rates experienced by the queue.

SUMMARY OF THE INVENTION

Briefly, consistent with the present invention, a method for detecting and reacting to changes in depth of one or more queues which store messages processed by tasks executing in a computer system sets a high threshold of a depth of the queue to a first value and detects when the depth of the queue equals or exceeds the high threshold. The high threshold is raised by a predetermined increment each time the depth of the queue equals or exceeds the high threshold.

In another aspect of the invention, a method for detecting and reacting to changes in depth of one or more queues which store messages processed by tasks executing in a computer system stores messages processed by tasks executing in a computer system starts at least one task for processing one or more messages stored in a queue. A high threshold of a depth of the queue is set to a first value. At least one additional task for processing the messages in the queue is started if the depth of the queue equals or exceeds the high threshold set to the first value.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram of a task dispatching system consistent with the present invention.

FIG. **2** is a block diagram flowchart of a task dispatching process consistent with the present invention.

FIGS. **3A–3D** are graphical representations of a queue

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The present invention will be described in the context of a specific embodiment, but the invention is not intended to be so limited.

A system and method consistent with the present invention enables a dynamic response to trigger messages and high and low performance event messages and supports a graduated dispatching of tasks for processing messages in a queue. As messages are enqueued, queue attributes are altered, including an override of the standard flip-flop behavior of the queue depth high and low event statuses. If a standard flip-flop toggling of the status of the high and low performance events is used, a system cannot dispatch tasks in a graduated manner. Given that a queue's depth is seen to move in steps, with the number and size of these steps under administrative control and with monitoring logic able to re-evaluate the queue's dispatching needs at each step, it is possible to alter queue attributes to create a bracketing at the point at which the last event occurred with a pair of "tripwires." The tripwires are continuously kept on either side of the queue depth at which the last performance event was generated. When the depth changes enough to cause the depth to cross one of the tripwires, the tripwires are moved, and dispatching or any other useful logic may be executed.

FIG. **1** is a block diagram of a task dispatching system **10** consistent with the present invention. As shown in FIG. **1**, the task dispatching system **10** includes an operating system **20** in which one or more processing tasks **25** are executing. The processing tasks **25** are computer programs that dequeue and process the messages that have arrived in the queues where each instance of such a computer program in execution is referred to as a task. The processing tasks 25 dequeue and process messages found on the queues for which they are associated. It is also possible that the processing tasks generate messages to be transferred to other tasks and programs executing in operating system 20 or in an operating system outside of the task dispatching system 10. The generated messages, if any, are passed to a message transfer system 30. The message transfer system 30 is preferably implemented using MQSeries.

The messages transferred to the message transfer system 30 are placed into one of a plurality of queues 35. As messages are placed into the queues 35, or arrive from other systems, various event messages may be generated. These event messages include trigger messages 40, which are stored in one of a plurality of initiation queues 45, and performance event messages 50, which are stored in a performance event queue 55. In general, there are more queues 35 than initiation queues 45, such that each initiation queue preferably services a plurality of queues 35. The performance event queue 55 passes the performance event messages 50 to an event relay 60. The event relay 60 takes the performance event messages 50 from the performance event queue 55 and transfers them to the initiation queue 45 corresponding to the queue 35 generating the performance event message 50. The trigger messages, as well as the performance event messages passed by the event relay 60, that are stored in the initiation queues 45 are passed to a respective one of a plurality of trigger event processors 65. Each instance of an initiation queue 45 corresponds with a respective one of the trigger event processors 65.

The trigger event processors 65, in response to the event messages received from the initiation queues 45, determine whether or not a task should be generated to process the messages in the queue 35 that generated the event message. To make this determination, the trigger event processors 65 look at attributes associated with the queue 35. Depending upon these attributes, the trigger event processors 65 may issue a task start instruction to have one or more additional tasks assigned to process the messages in the queue 35.

The following is an example of an environment in which the task dispatching system 10 may be implemented. In this example, a program supports a webpage where a user can enter a query. When the user submits the query, the program supporting the web page takes the data transmitted from the browser, and builds a message, such as an MQSeries message. This message is then enqueued on the machine running the web-server and transferred to the machine in which the program supporting the query against the desired data is deployed. A processing task 25 that can process the desired query, if not already running, is started in response to a trigger message. The processing task 25 dequeues the message and sends a response message back to the still-waiting web-server program via some other queue 35. As the volume of these queries rises, an increased number of processing tasks 35 for dequeuing and processing the queries can be started in proportion to the volume, increasing overall throughput.

In a system and method consistent with the present invention, the monitoring of the depth of the queues 35 allows for a graduated response to traffic loads of the queues. As events are generated according to the monitoring of the queue depth, additional tasks may be started, for example, as the depth of the queue rises through successive incremental increases in depth. If the depth starts to fall or a maximum number of tasks allowed for the queue is reached, then the starting of further tasks may be suppressed. This graduated response can be implemented, for example, in a program that receives trigger messages and performance event messages, as discussed above, which are used to intelligently dispatch tasks in a manner consistent with the needs of the queues involved.

In addition to the attributes and elements of the performance event message feature and the trigger feature discussed above, a system and method consistent with the present invention defines the following attributes relative to a queue 35 being monitored. To avoid having a task generated for every enqueue of a message in a queue 35, it is preferable that the first and/or depth trigger type is used.

A first of these additional attributes is a task trigger increment. When a trigger event processor 65 detects that additional tasks are needed to process the incoming messages, the value of the task trigger increment determines how many tasks will be started. In a simple approach, the value may be a static number of tasks. However, a more sophisticated heuristic approach in which the enqueue/dequeue rates are analyzed may be used to determine the value of the task trigger increment so as to optimize tasking. This analysis for determining an optimum tasking is possible because the performance event messages include enqueue/dequeue counts for a specified interval of time.

A trigger threshold indicates a change in queue depth that should be considered as significant to the trigger event processor 65. The value of the trigger threshold is expressed as a percentage of the maximum queue depth attribute of the queue. For example, if a queue's maximum queue depth is one thousand, then a value of one for this attribute would cause the trigger event processor 65 to re-evaluate the prpcessing of the queue whenever its depth passes a multiple of ten messages, which equals 1 percentage of 1000 messages.

Maximum tasks sets a limit on the number of tasks that may be used to process the messages of a queue. When the number of tasks running against a queue is equal to this parameter, then the trigger event processor 65 will not attempt to start any additional tasks, regardless of further depth increases. Like the task trigger increment, the value of maximum tasks may be a static number, but more sophisticated heuristic approaches may also be used.

FIG. 2 is a flow diagram of a task dispatching process consistent with the present invention. As shown in FIG. 2, the process starts in an idle state where the task dispatching process is waiting for a trigger message or a performance event message (step 202). When a message is received, it is first determined whether the message is a trigger message (step 204). This determination is preferably made by the trigger event processor 65. The trigger message may have been generated based upon a first trigger type in response to a message being stored in an empty queue.

If the message is a trigger message, the trigger event processor 65 sets the queue depth high percentage to the trigger threshold and sets the queue depth high event status to enabled (step 206). The trigger event processor 65 also sets the queue depth low percentage to zero and sets the queue depth low event status to disabled (step 208). Finally, the trigger event processor starts a number of tasks for processing the queue 35 which generated the trigger message (step 210). The number of tasks started corresponds to the task trigger increment, using either a simple static value, or a heuristically determined value.

If the message is not a trigger message, the trigger event processor 65 determines whether the message is a performance event low message (step 212). If the message is determined to be a performance event low message, the trigger event processor reduces the queue depth high percentage by subtracting the value of the trigger threshold from the current value of the queue depth high percentage attribute and the queue depth high event status is left as "enabled" because this event has no effect on it (step **214**). The reduction in the queue depth high percentage lowers the level of this attribute by one increment of the trigger threshold. It is preferable for the queue depth high percentage to be maintained at a value no lower than the value of trigger threshold.

At the same time, the trigger event processor **65** sets the queue depth low percentage to a value that is at least two units of the trigger threshold below the current queue depth high percentage, but not less than zero (step **216**). The trigger event processor then determines if the queue depth high percentage is equal to the trigger threshold (step **218**). If they are equal, then processing returns to the idle state of waiting for a new message (step **202**). However, if they are not equal, then the queue depth low event status is set to enabled (step **220**). Normally, the queue depth low event status is set to disabled automatically in response to a performance event low message. Having both the queue depth low event status and queue depth high event status enabled at the same time overrides the normal inverse relationship between their settings. In general, no tasks are started as a result of a performance event low message because this message indicates that the number of tasks currently executing are dequeuing messages faster than new ones are being enqueued, making additional tasks unnecessary.

If the message is neither a trigger message nor a performance event low message, the trigger event processor **65** checks whether the received message is a performance event high message (step **222**). If not, then there is an error. If the message is a performance event high message, the following steps are performed. First, the trigger event processor **65** determines if the number of tasks currently running to dequeue messages from the queue that generated the performance event high message is less than the maximum tasks attribute for the queue (step **224**). This maximum tasks attribute may be statically defined attribute or a heuristically derived one.

If it is determined that the number of tasks is not less than the value indicated by the maximum tasks, then no alterations are made to the queue attributes and no new tasks are started. Instead, processing returns to the idle state of waiting for a new message (step **202**). In addition to returning to the idle state, an alert may be generated to a systems management facility to notify operations staff that message enqueue rates are exceeding dequeue rates, and the trigger event processor **65** is no longer able to start additional tasks.

If the number of currently running tasks is less than the maximum tasks value, the trigger event processor **65** increments the queue depth high percentage by adding the value of the trigger threshold to the current value of the queue depth high percentage attribute and the queue depth high event status is set to enabled (step **226**). This increase in the queue depth high percentage, in effect, raises the value of the attribute by one increment of the trigger threshold value. In addition, keeping the setting of the queue depth high event status as enabled overrides the normal behavior of disabling the queue depth high event status after receiving a performance event high message. As a result, the generation of a performance event high message is enabled for a higher threshold.

The trigger event processor also sets the queue depth low percentage to a value that is at least two units of the trigger threshold below queue depth high percentage, but not less than zero (step **228**). The queue depth low event status is set to enabled automatically by the reception of the performance event high message. In addition to the alterations to the queue attributes, the trigger event processor **65** starts a number of tasks for processing the messages from the queue **35** which generated the trigger message (step **230**). The number of tasks started corresponds to the task trigger increment, using either a simple static value, or a heuristically determined value.

FIGS. **3A–3D** show a graphical representation of a queue in a task dispatching process consistent with the present invention. For purposes of illustration, it is assumed that the maximum depth of the queue is 1000 and the trigger threshold is 1%, which equals 10 messages. FIG. **3A** shows an idle queue having no messages. In this state, the performance event message attributes, including the queue depth high percentage, the queue depth low percentage, the queue depth high event status and the queue depth low event status are undefined.

FIG. **3B** shows the status of the queue after five messages have been enqueued in a queue. A trigger message is generated based upon the first trigger type in response to the first of the newly arrived messages. The trigger message causes the undefined performance event message attributes to be set. The queue depth high percentage is set to the trigger threshold of one percent (i.e., ten messages), the queue depth high event status is enabled, the queue depth low percentage is set to zero, and the queue depth low event status is disabled. In addition, the trigger event processor **65** starts a number of tasks for processing the messages present in the queue. The number of tasks started corresponds to the task trigger increment.

FIG. **3C** shows the status of the queue after the number of messages stored in the queue has risen to fifteen. Since the number of stored messages has risen above ten, which corresponds to the queue depth high percentage, a performance event high message is generated. In response to the performance event high message, the queue depth high percentage is incremented by the value of the trigger threshold from one percent to two percent, and the queue depth high event status is enabled. At the same time, the queue depth low percentage is set to the higher of zero and a value equal to the queue depth high percentage minus two times the trigger threshold. Since the value of the queue depth high percentage has been set to two percent and two times the trigger threshold is also two percent, the queue depth low percentage remains at zero. In addition, the queue depth low event status is automatically enabled in response to the performance event high message. The trigger event processor **65** also starts a number of tasks corresponding to the task trigger increment, as long as the number of currently running tasks is less than the maximum tasks attribute.

FIG. **3D** shows the status of the queue after the number of messages stored in the queue has risen to twenty-five (25). Since the number of stored messages has risen above two percent (i.e., twenty messages), which corresponds to the queue depth high percentage, a performance event high message is generated. In response to the performance event high message, the queue depth high percentage is incremented by the value of the trigger threshold from two percent to three percent, and the queue depth high event status is enabled. At the same time, the queue depth low percentage is set to the higher of zero and a value equal to the queue depth high percentage minus two times the trigger threshold. Since the value of the queue depth high percentage has been set to three percent and two times the trigger threshold is two percent, the queue depth low percentage is set to one percent (i.e., ten messages). In addition, the queue depth low event status is automatically enabled in response to the performance event high message. The trigger event processor **65** again starts a number of tasks corresponding to the task trigger increment, as long as the number of currently running tasks is less than the maximum tasks attribute.

If the depth of the queue then dropped below ten, a performance event low message would be generated. In response to the performance event low message, the queue depth high percentage would be reduced by the value of the trigger threshold from three percent to two percent. The queue depth high event status would be enabled automatically in response to the performance event low message. At the same time, the queue depth low percentage would be set to the higher of zero and a value equal to the queue depth high percentage minus two times the trigger threshold. Since the value of the queue depth high percentage has been set to two percent and two times the trigger threshold is also two percent, the queue depth low percentage would be set to zero. In addition, since the queue depth high percentage is greater than the trigger threshold, the queue depth low event status would be enabled. No new tasks are started, nor are any tasks discontinued. However, it would be possible to have the task dispatching process remove tasks in response to performance event low messages.

As demonstrated by the queue example described in FIGS. **3A–3D**, by moving the event levels (tripwires) up and down with increases and decreases in depth, the system is kept sensitive to further changes in depth since the levels stay close to one trigger threshold of the current depth. As a result, a material increase or decrease in queue depth can be detected promptly.

A further variation on the responses to the queue depth changes may be to use a flexible trigger threshold which is adjusted based on, for example, the current depth of the queue as a percent of the maximum depth. With this approach, the trigger threshold may be smallest when the queue depth is low, and then increase as the queue depth increases, since at higher queue depths there should be more total tasks running, making the threshold larger where depth changes are significant.

The foregoing description of a preferred embodiment of the invention has been presented for purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed, and modifications and variations are possible in light of the above teachings or may be acquired from practice of the invention. The embodiment was chosen and described in order to explain the principles of the invention and a practical application to enable one skilled in the art to utilize the invention in various embodiments and with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the claims appended hereto, and their equivalents.

What is claimed is:

1. A method for detecting and reacting to changes in depth of one or more queues which store messages processed by tasks executing in a computer system, comprising:

setting a high threshold of a depth of the queue to a first value;

setting a low threshold of a depth of the queue to a second value lower than the first value;

detecting when the depth of the queue equals or exceeds the high threshold;

raising the high threshold by a predetermined increment each time the depth of the queue equals or exceeds the high threshold; and

selectively adjusting the low threshold when the depth of the queue equals or exceeds the high threshold.

2. A method according to claim **1**, further comprising:

starting at least one task for processing one or more messages stored in the queue each time the depth of the queue equals or exceeds the high threshold.

3. A method according to claim **2**, further comprising the steps of

reducing the value of the high threshold if the depth of the queue is equal to or less than the value of the low threshold; and

reducing the value of the low threshold if the depth of the queue is equal to or less than the value of the low threshold.

4. A method according to claim **1**, further comprising:

starting at least one task for processing one or more messages stored in the queue each time the depth of the queue equals or exceeds the high threshold if the number of tasks currently processing the messages in the queue is less than a predetermined amount.

5. A method according to claim **1**, further comprising stopping at least one task for processing one or more messages stored in the queue if the depth of the queue is equal to or less than the value of the low threshold.

6. A method according to claim **1**, wherein the low threshold is raised when the depth of the queue equals or exceeds the high threshold and the high threshold is higher than a predetermined value.

7. A method for detecting and reacting to changes in depth of one or more queues which store messages processed by tasks executing in a computer system, comprising:

starting at least one task for processing one or more messages stored in a queue;

setting a high threshold of a depth of the queue to a first value;

setting a low threshold of a depth of the queue to a second value lower than the first value;

starting at least one additional task for processing the messages in the queue if the depth of the queue equals or exceeds the high threshold set to the first value; and

stopping at least one task for processing one or more messages stored in the queue if the depth of the queue is equal to or less than the value of the low threshold.

8. A method according to claim **7**, further comprising:

raising the value of the high threshold if the depth of the queue equals or exceeds the high threshold set to the first value.

9. A method according to claim **8**, further comprising:

starting at least one additional task for processing the messages in the queue if the depth of the queue equals or exceeds the raised value of the high threshold.

10. A method according to claim **8**, further comprising:

starting at least one additional task for processing the messages in the queue if the depth of the queue equals or exceeds the raised value of the high threshold and the number of tasks currently processing the messages in the queue is less than a predetermined amount.

11. A method according to claim **7**, further comprising:

setting the high threshold to a third value lower than the first value if the depth of the queue is equal to or less than the low threshold set to the second value; and

setting the low threshold to a fourth value lower than the second value if the depth of the queue is equal to or less than the value of the low threshold.

12. A method according to clam **1**, further comprising stopping at least one task for processing one or more messages stored in the queue if the depth of the queue is equal to or less than the value of the low threshold.

13. A method according to claim 7, wherein the low threshold is raised when the depth of the queue equals or exceeds the high threshold and the high threshold is higher than a predetermined value.

14. A computer system for detecting and reacting `to changes in depth of one or more queues which store messages processed by tasks executing in the computer system, comprising:

means for setting a high threshold of a depth of the queue to a first value;

means setting a low threshold of a depth of the queue to a second value lower than the first value;

means for detecting when the depth of the queue equals or exceeds the high threshold;

means for raising the high threshold by a predetermined increment each time the depth of the queue equals or exceeds the high threshold; and

means for selectively adjusting the low threshold when the depth of the queue equals or exceeds the high threshold.

15. A computer system according to claim 14, further comprising:

means for starting at least one task for processing one or more messages stored in the queue each time the depth of the queue equals or exceeds the high threshold.

16. A computer system according to claim 14, further comprising the step of:

means for starting at least one task for processing one or more messages stored in the queue each time the depth of the queue equals or exceeds the high threshold if the number of tasks currently processing the messages in the queue is less than a predetermined amount.

17. A computer system according to claim 14, further comprising:

means for reducing the value of the high threshold if the depth of the queue is equal to or less than the value of the low threshold; and

means for reducing the value of the low threshold if the depth of the queue is equal to or less than the value of the low threshold.

18. A computer system according to claim 14, further comprising means for stopping at least one task for processing one or more messages stored in the queue if the depth of the queue is equal to or less than the value of the low threshold.

19. A computer system according to claim 14, further comprising means for raising the low threshold when the depth of the queue equals or exceeds the high threshold and the high threshold is higher than a predetermined value.

20. A computer program stored on a computer readable medium for detecting and reacting to changes in depth of one or more queues which store messages processed by tasks executing in a computer system, the computer program configured to:

set a high threshold of a depth of the queue to a first value;

set a low threshold of a depth of the queue to a second value lower than the first value;

detect when the depth of the queue equals or exceeds the high threshold;

raise the high threshold by a predetermined increment each time the depth of the queue equals or exceeds the high threshold; and

selectively adjust the low threshold when depth of the queue equals or exceeds the high threshold.

21. A computer program according to claim 20, further configured to:

start at least one task for processing one or more messages stored in the queue each time the depth of the queue equals or exceeds the high threshold.

22. A computer program according to claim 20, further configured to:

start at least one task for processing one or more messages stored in the queue each time the depth of the queue equals or exceeds the high threshold if the number of tasks currently processing the messages in the queue is less than a predetermined amount.

23. A computer program according to claim 20, further configured to:

reduce the value of the high threshold if the depth of the queue is equal to or less than the value of the low threshold; and

reduce the value of the low threshold if the depth of the queue is equal to or less than the value of the low threshold.

24. A computer program according to claim 20, further configured to:

stop at least one task for processing one or more messages stored in the queue if the depth of the queue is equal to or less than the value of the low threshold.

25. A computer program according to claim 20, further configured to:

raise the low threshold when the depth of the queue equals or exceeds the high threshold and the high threshold is higher than a predetermined value.

* * * * *

EXHIBIT E

SENDER: *COMPLETE THIS SECTION*

- Complete items 1, 2, and 3. Also complete item 4 if Restricted Delivery is desired.
- Print your name and address on the reverse so that we can return the card to you.
- Attach this card to the back of the mailpiece, or on the front if space permits.

1. Article Addressed to:

Oleh Hereliuk
CBCInnovis dba Federal Research
1023 Fifteenth Street, NW, Suite 401
Washington, DC 20005

OPEN1660 3/14/13

2. Article Number
(*Transfer from service* 7009 2820 0003 6668 4937

COMPLETE THIS SECTION ON DELIVERY

A. Signature
X ☐ Agent
☐ Addressee

B. Received by (*Printed Name*) C. Date of Delivery

D. Is delivery address different from item 1? ☐ Yes
If YES, enter delivery address below: ☐ No

3. Service Type
☐ Certified Mail ☐ Express Mail
☐ Registered ☐ Return Receipt for Merchandise
☐ Insured Mail ☐ C.O.D.

4. Restricted Delivery? (*Extra Fee*) ☐ Yes

PS Form 3811, February 2004 Domestic Return Receipt 102595-02-M-1540

Sprinkle
IP LAW GROUP
1301 W. 25th Street, Suite 408
Austin, Texas 78705

7009 2820 0003 6668

Oleh Hereliuk
CBCInnovis dba Federal Research
1023 Fifteenth Street, NW, Suite 401
Washington, DC 20005

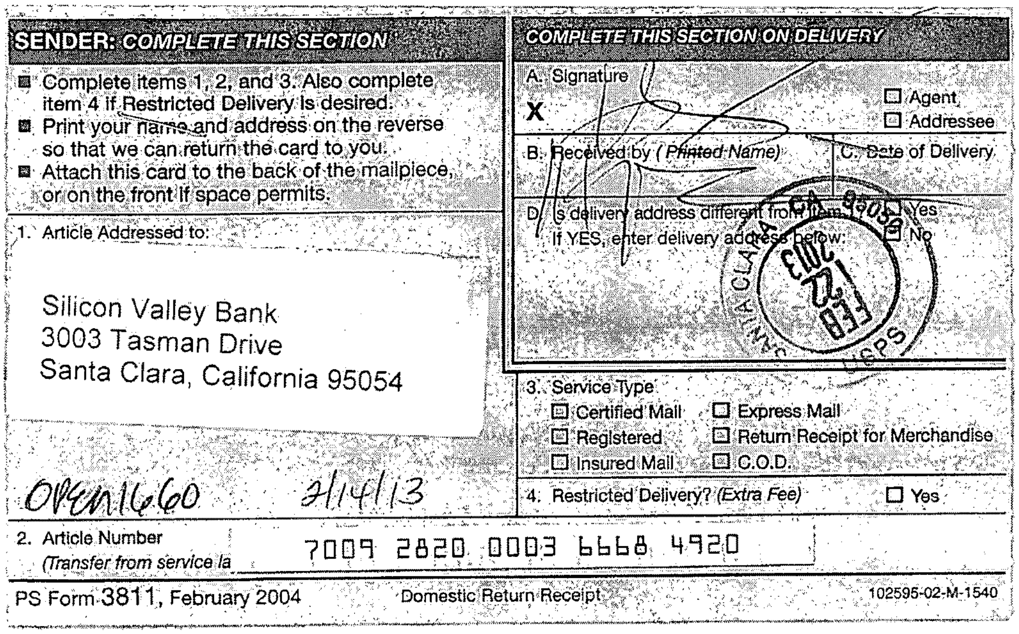
Foe

S42805.108
$6.77 0
US POSTAGE
FIRST-CLASS
062S0005335647
78705

Attorney Docket No.
OPEN1660

Serial No. 09/527,247
Customer No. 109422

EXHIBIT F



SENDER: *COMPLETE THIS SECTION*

- Complete items 1, 2, and 3. Also complete item 4 if Restricted Delivery is desired.
- Print your name and address on the reverse so that we can return the card to you.
- Attach this card to the back of the mailpiece, or on the front if space permits.

1. Article Addressed to:

Silicon Valley Bank
3003 Tasman Drive
Santa Clara, California 95054

OPEN1660 2/14/13

COMPLETE THIS SECTION ON DELIVERY

A. Signature
X □ Agent □ Addressee

B. Received by (*Printed Name*) C. Date of Delivery

D. Is delivery address different from item 1? □ Yes
If YES, enter delivery address below: ☒ No

SANTA CLARA CA 95050
FEB 22 2013
USPS

3. Service Type
☒ Certified Mail □ Express Mail
□ Registered □ Return Receipt for Merchandise
□ Insured Mail □ C.O.D.

4. Restricted Delivery? (*Extra Fee*) □ Yes

2. Article Number
(*Transfer from service la*
7009 2820 0003 6668 4920

PS Form 3811, February 2004 Domestic Return Receipt 102595-02-M-1540