SYSTEM TUXEDO

By
Channu Kambalyal
Sushil Consultants Inc.
www.sushilconsultants.com

BRIEF HISTORY

- TUXEDO - Transactions in Unix Extended for Distributed Operations
- Construction started in 1983 as a UNIX System (UNITS) with in Bell Labs, AT & T.
- Initially code named DUX. TUX and DUX were later combined and Tom Bishop gave the name "TUXEDO".
- In 1989, UNITS project was transferred to USL division of AT & T.
- In 1993, it was acquired by Novel Inc.,
- In 1996 its distribution and development was transferred to BEA Systems Inc.

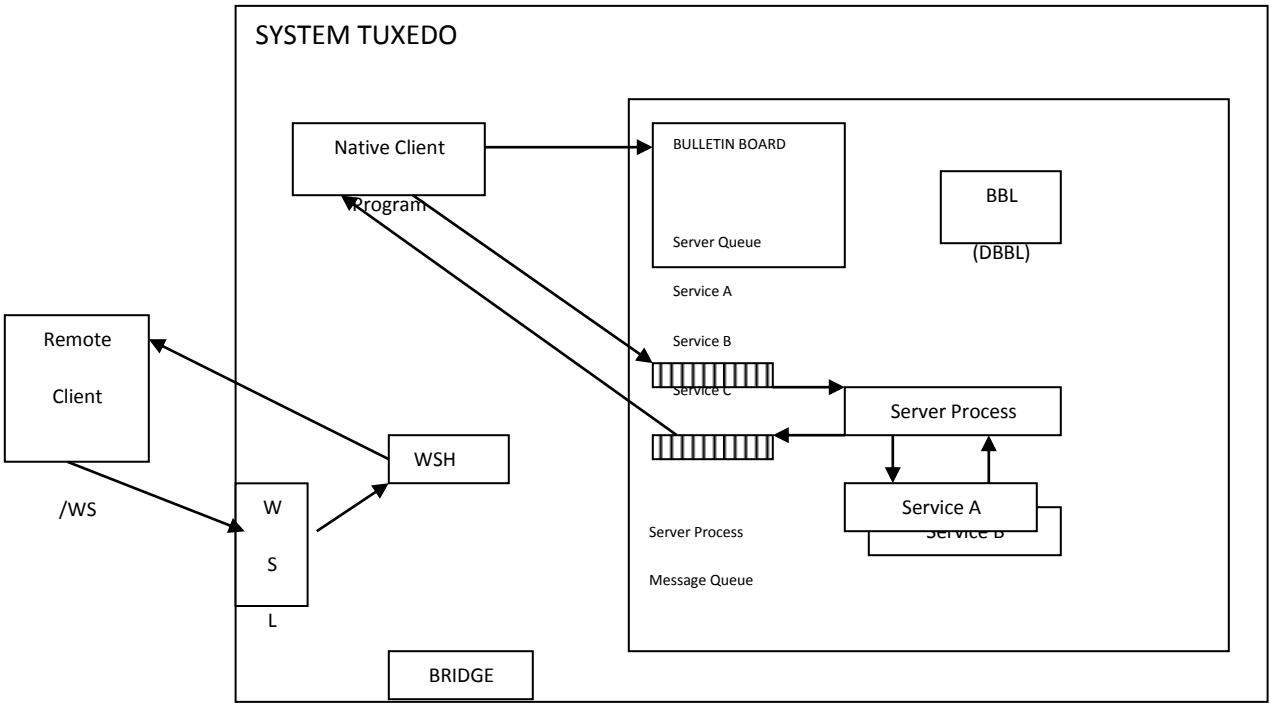
FEATURES OF TUXEDO

- Communication types - Synchronous, Asynchronous, Conversational, Unsolicited Notifications
- Typed Buffers
- Transaction Management - Global Transactions - Two Phase Commit
- /WS - Remote Clients
- /QUEUE - Persistent Queues (Also called Reliable Queues)
- Data Dependent Routing (DDR)
- Event Broker
- Programmed Administration

BULLETIN BOARD

- Collection of data structures in shared memory used by Tuxedo to monitor a running application.
- Contains information about application servers, services, clients, transactions, etc.,.
- Part of address space for client server programs during initialization.
- Provides a name serving function, mapping service names to physical location addresses.
- Routes client requests to service routines based on service names.
- Automatically started during application start-up.
- Uses TUXCONFIG file.
- Propagated to all machines and synchronized by administrative processes.

(See - System Tuxedo Diagram below)



OVERVIEW OF SYSTEM TUXEDO

TYPED BUFFERS

- Message buffers used to pass application data between client and server processes.
- All typed buffers allocated, populated, freed using **ATMI functions**.
- Typed buffers have self-describing in headers (memory header).
- Commonly used buffers are:
    - STRING - null terminated string data.
    - FML (Field Manipulation Language) - self-described data - can be used for Data Dependent Routing.
    - VIEW - C structure with sub-types - can be used for Data Dependent Routing.
    - CARRAY - Used for array of bits and raw buffer.
    - Others - X_OCTECT, X_C_TYPE, X_COMMON
- Provision for custom defined buffers.

ATMI

- Application to Transaction Monitor Interface
- Some commonly used APIs:
    - tpinit() - joins client application to System Tuxedo; uses TPINIT structure
    - tpalloc(), tprealloc() - allocate space for message buffer with additional space for ATMI header
    - tpcall() - used to make synchronous call to a Tuxedo Service
        - Internally uses tpacall() with TPGETANY flag!
    - tpacall(), tpgetrply - used to make asynchronous call to a Tuxedo Service
    - tpterm() - used to disconnect a client application
    - tpsvrinit() - allows for one time initialization for a server
    - tpreturn() - used to send reply back to requester

    - tpforward() - used to forward a clients request to some other service
        - Has combined properties of tpreturn() and tpacall() with TPNORPLY flag.
    - tpopen() - opens connection to database
    - tpbegin() - starts a transaction
    - tpcommit() - commits a transaction
    - tpabort() - aborts a transaction
    - tpsuspend() - suspends a transaction
    - tpresume() - resumes a transaction
- ATMI Flags
    - Most ATMI functions have a flag parameter to control operation of a particular function and can be combined using | (OR) operator
    - Some commonly used flags are:

TPSIGSTRT - call is re-issued if processing is interrupted by signal
TPNOBLOCK - causes to return immediately with TPEBLOCK error without invoking service if there is blocking error
TPNOTIME - prevents time-out. (NOTE: TPNOBLOCK and TPNOTIME are mutually exclusive)
TPNOTRAN - prevents the requested application from being included in current transaction

BASIC ADMINISTRATION

- buildclient - Utility to compile and link native client code
- buildserver - Utility to compile server code
- UBBCONFIG file - Every Tuxedo application is described with a single file called Units (also called Universal) Bulletin Board configuration. This is an ASCII file compiled into a binary format file (usually named tuxconfig) using tmloadcf command.
- UBB file has 7 sections as follows:

*RESOURCES
Has info about IPCKEY, MAXACCESSORS, MAXSERVERS, MAXSERVICES, DOMAINID, MASTER, MAXGTT, MODEL, MAXDRT, MAXRFT, MAXRTDATA.

*MACHINES
Has info about LMID, MAXWSCLIENTS

*GROUPS
Has info about server groups and associated resource managers.

*NETWORK
Network Information - NADDR, NSLADDR

*SERVERS
Has info about servers boot order and associated groups.

*SERVICES
List of available services.

*ROUTING
Has Data Dependent Routing criteria

(NOTE: For a sample of Configuration File see UBBCONFIG file)

- Commonly used administrative utilities:
    - tmloadcf Uses UBBCONFIG file to create TUXCONFIG file
    - tmunloadcf Gives asci version of TUXCONFIG. (Note: Use this utility to see default values!)
    - tmboot To startup Tuxedo Applications
    - tmshutdown To shutdown Tuxedo Applications
    - tmadmin Tool to administer Tuxedo System
    - tagent Tool to administer Tuxedo System remotely.

ENVIRONMENT VARIABLES

- Commonly used Environment Variables:
    - TUXDIR, TUXCONFIG, APPDIR, PATH, SHLIB_PATH (Only HP-UX), TLOGDEVICE
    - FIELDTBLS, FLDTBLDIR, FIELDTBLS32, FLDTBLDIR32
    - VIEWFILES32, VIEWDIR32

FML BUFFERS

- Divides message into field values
- Allows flexibility in defining message data
- Minimizes resource allocation for message buffer
- Supports Data dependent Routing
- Field Table Files
    - Has fields that may be placed in FML Buffer
- Some commonly used FML functions:
    - fadd(), fadd32()
    - fchg(), fchg32()
    - fget(), fget32()
    - foccur(), foccur32()
    - falloc(), frealloc(), ffree()
    - fneeded(), fneeded32()
- mkfldhdr32 - FML utility to compile field table file to obtain a header file
- UD32 (Units Debugger/Driver) - A debugging tool used to test Tuxedo Services using FML buffer.

VIEW BUFFERS

- View File
    - Used to describe the fields that appear in a C structure or COBOL record
    - Must have extension .v
    - May be used at compile time or at execution time.
- FML VIEW Function
    - fvstof(), fvstof32() - functions to convert VIEW Buffer to FML Buffer
    - fvftos(), fvstof32() - functions to convert FML Buffer to VIEW Buffer
- viewc and view32
    - Utilities to compile *.v view file to generate *.h file and *.V (binary) file.

APPLICATION ERROR HANDLING

- ULOG File - Userlog File created each day Tuxedo.

- Some Error Handling functions:
    - userlog()
    - tperrno()
    - tpstrerror32()

/WORKSTATION (/WS)

- /WS (also called System Workstation) defines an environment for for remote workstation clients to access Tuxedo Services through a Handler Process. /WS has 2 servers:
    - WSL - Workstation Listener - Initial Contact point for workstation clients.
    - WSH - A process that is a native client on behalf of remote clients.
- /WS Functions
    - tuxputenv()
    - tuxgetenv()
    - tuxreadenv()

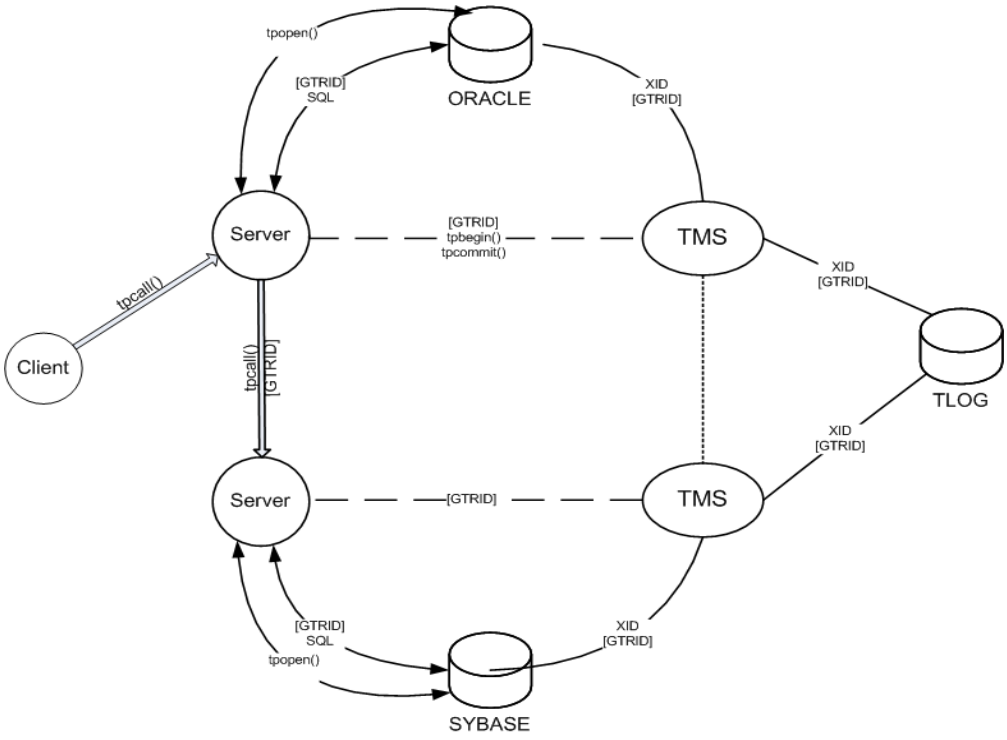
ADMINISTRATIVE SERVERS

- BBL - Bulletin Board Liaison. A server process that is started when Tuxedo application is started.
- DBBL - Distinguished Bulletin Board Liaison. A BBL that plays the role of DBBL, activated when a networked Tuxedo application is started.
- BRIDGE - A server process that is activated when a networked Tuxedo application is started to transfer messages from local messages from local machine to servers on other machines.

TRANSACTIONS

- Transaction Manager (TM) - Transaction Manager plays the role of co-ordinator and directs the Resource Managers to perform the transaction activity.
- Resource manager (RM) - Resource manager provides applications to shared resources. Resources can be databases, reliable queues, fax machines, money machines, etc.
- XA Interface - XA Interface is a set of API calls between RMs and TMs.
- Two Phase Commit - 2PC Algorithm is used to ensure ACID properties of a Tuxedo Transaction.
- AUTOTRAN - In Tuxedo system Transaction Mode may be started using AUTOTRAN = Y in SERVICES section of UBBCONFIG or using ATMI transaction functions.
- ATMI Transaction Functions:
    - tpbegin()
    - tpcommit()
    - tpabort()
    - tpsuspend()
    - tpresume()

Note: See following diagram for an illustration on how transactions are handled by /T.



TRANSACTIONS IN TUXEDO

COMMUNICATION CALLS

- Synchronous call:
  - tpcall (char * servicename, char * sendbuffer, long ilen, char * receivebuffer, long * olen, long flags)
- Asynchronous call:
  - tpacall(char * servicename, char * sendbuffer, long * ilen, long flags)
  - tpgetreply(int * calldescripter, char * outbuffer, long *olen, long flags)

SYSTEM QUEUE - /Q

- A sub-system of /T for application clients and servers to communicate with other components using persistent / reliable queues.
- XA Resource Manager: /Q is treated as a Resource Manager.
- /Q ATMIs used:
  - tpenqueue()
  - tpdedequeue()
- /Q Servers - TMS_QM, TMQUEUE, TMQFORWARD.
- Configuring /Q - qmadmin utility may be used to create the queue objects - qdevices, qspaces and queues.
- TPQCTL Structure - Both tpenqueue() and tpdequeue() pass a pointer to TPQCTL structure. This structure has info like dequeue time, priority, msgid, corrid, replyqueue, failurequeue, cltid, etc.,.

CONVERSATIONAL COMMUNICATION

- A client establishes a connection with a service, sends one or more requests for processing and waits for one or more process responses to be returned.

- Configuration - Conversational servers are identified in UBBCONFIG by setting the CONV parameter to Y in SERVERS section and MAXCONV value RESOURCES section.
- ATMIs used:
    - tpconnect() - return a Connection Descriptor.
    - tpsend()
    - tpreceive()
    - tpdisconnect()

UNSOLICITED NOTIFICATIONS

- Used to send unsolicited messages to one or more clients.
- ATMIs used:
    - tpsetunsol(message handling function)
    - tpchkunsol()
    - tpnotify()
    - tpbroadcast()

EVENT BROKER

- A sub-system of Tuxedo that receives event messages. Filters and distributes to the subscribers.
- Event Broker has 2 servers:
    - TMSYSEVT
    - TMUSREVT
- ATMIs used:
    - tppost()
    - tpsubscribe()

DATA DEPENDENT ROUTING

- DDR allows client request to a service routine based on data stored in the input buffer.
- Routing Criteria for DDR is defined in ROUTING section in the UBBCONFIG file.

SECURITY

- Tuxedo provides several five (5) levels of security.
- Security Levels and Configuration:
    - Application Level - Configured by setting APP_PW value for SECURITY parameter in RESOURCES section.
    - User Authentication- SECURITY = USERAUTH.
    - Access Control Lists (ACLs) - SECURITY = ACL
    - Mandatory ACLs - SECURITY = MANDATORY_ACL.
    - Link-Level Encryption - Mostly used over Tuxedo network.

LOAD BALANCING AND PRIORITY

- Tuxedo load balancing algorithm is based on the categories - Type Validation, Data Dependent Routing, Load Balancing using Client Centric, Under Utilized queue syndrome and round robin.
- Configuration is made in the RESOURCES section by setting LDBAL = Y and for each service in the SERVICES section. Default value is 50.
- Tuxedo also allows setting Priority for each service.

- Priority is set for services in SERVICES section by setting PRIO=(a number between 1 - 100).

AUTOMATED ADMINISTRATION

- Automated administration may be done using System Tuxedo provided ".TMIB" service, and using the GET and SET operation parameters and using configuration classes and run-time classes.

WEB GUI

- Tuxedo provides a web based administration utility called WEBGUI.

REFERENCE

Oracle Documentation on TUXEDO and JOLT