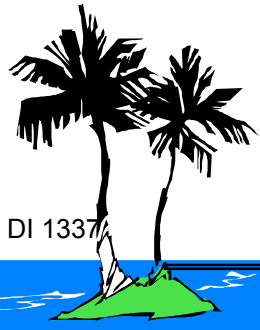
# 1998 Di USERS GROUP


# Introduction to MQSeries

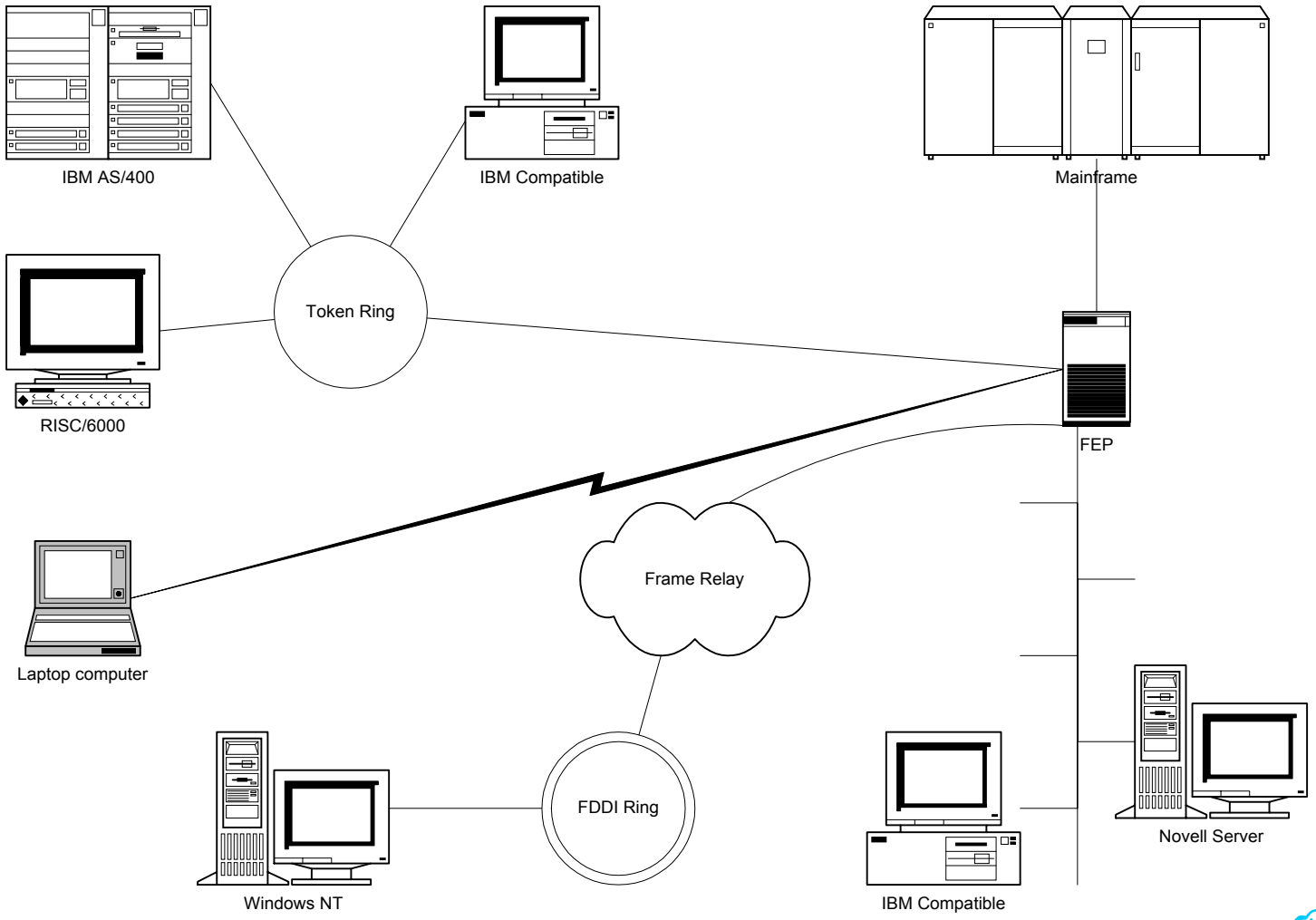
Patrick T. Verdugo
Certified MQSeries Engineer
MessageQuest™, Inc.



# IBM Trademarks

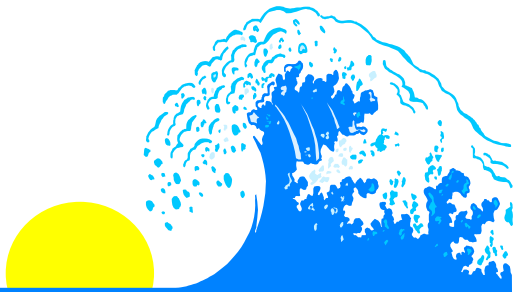
- IBM
- AIX
- OS/2
- OS/400
- VTAM
- MVS/ESA
- CICS/ESA
- CICS/MVS
- CICS/400
- SNA
- MQSERIES
- MQ

# Today's Enterprise



# Business Demands for Messaging and Queuing

- Multiple Data Sources
- A single request can involve multiple transactions on different systems
- Deferred Processing
- Separation of Business Topology from Processing Topology



# Beginning of Messaging

- Messaging and Queuing is not new to the Computer Industry, it has been around since the 1960's
- IMS is an example of a Queuing System
- CICS Transient Data and Temp Storage Queues
- TCAM is an example of an early Messaging and Queuing System
- Messaging and Queuing facilities have not been generally available to application programmers (Until now …)

# Messaging Concepts

- **Synchronous Messaging** (time-dependent) is a communications technique (most prevalent in the computer industry today)
  - Synchronous example: Telephone Call
- **Asynchronous Messaging** (time-independent) communications technique
  - Asynchronous example: E-mail

# Messaging Concepts
# Program to Program Messaging

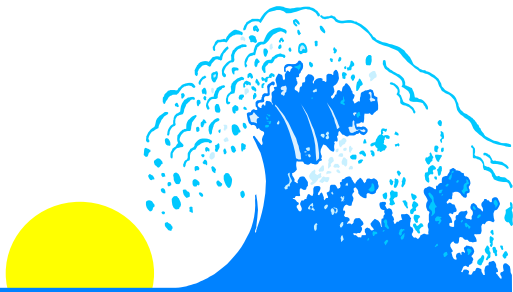
- **Synchronous Methods:**
  - **CPI-C** Common Programming Interface for Communications
  - **RPC** Remote Procedure Call
- **Asynchronous Method:**
  - **MQI** Message Queue Interface

# Messaging Concepts - CPI-C

- Both CPI-C and APPC offer program to program communications over a private logical connection
- The connection is reserved exclusively for the duration of the conversation
- The connection can only be used by the two programs communicating
- It is Synchronous, monolithic in structure and connection-dependent

# Messaging Concepts - RPC

- Supported by STUB calls in both the caller and the callee
- Using a call return mechanism over a synchronous connection
- A conversation is maintained
- Client / Server message flow model
- It is Synchronous, connection-dependent, but more complex (Client / Server model)



# Messaging Concepts - MQI

- Asynchronous programs fill and empty queues to exchange messages
- Programs are never logically connected
- Programs are indirectly associated by one or more application queues
- Simple in nature

# Messaging Summary

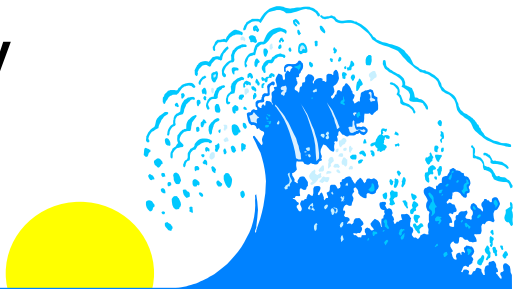
- CPI-C, RPC and MQI are companion interfaces
- They can coexist
- They do not compete and are not substitutes for one another
- A single program could use all three interfaces

# Queuing Concepts

- Queuing is a no-connection communication choice
- Queuing is a time adaption technique, used for saving information until the intended recipient is ready to accept the information
- Entities are indirectly communicating, each operating at its own preferred speed (buffering mechanism)

# Queuing Concepts

- A message queue is a storage area for saving messages in an orderly manner for later retrieval
- Queues eliminate timing dependencies
- Advantages of queues
  - reduce the number of connections (network expense reduction)
  - Concurrent processing, sharing of messages
  - Load balancing and load distribution
  - Queues and programs can be administered separately

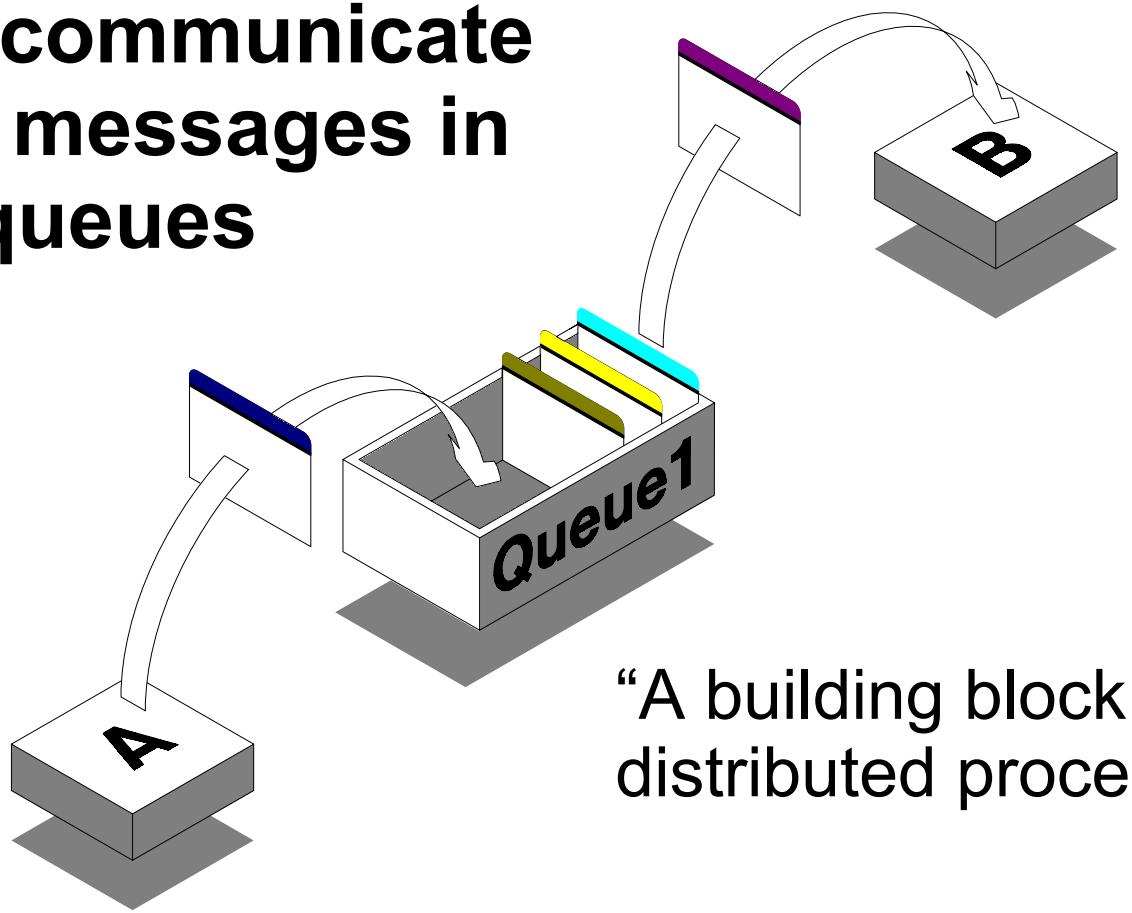


# Queuing Concepts

- Asynchronous real-world examples
- A telephone answering machine
- A letter sent by Mail
- An E-mail
- Queues can be *private or shared*
- Queues can be *local or remote*
- Queues can be *dynamic or static*

# Messaging and Queuing

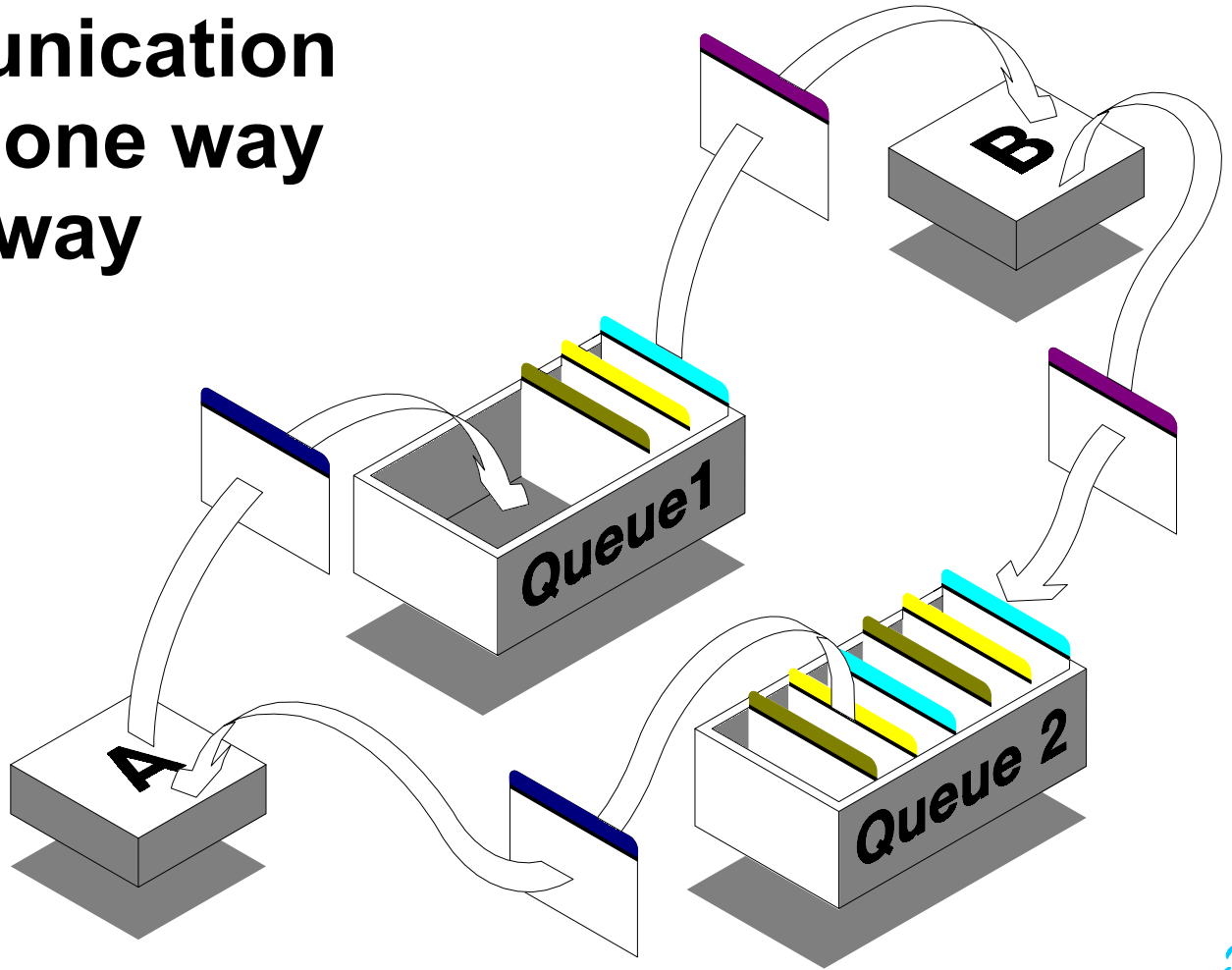
- **Programs communicate by putting messages in message queues**




"A building block for distributed processing"

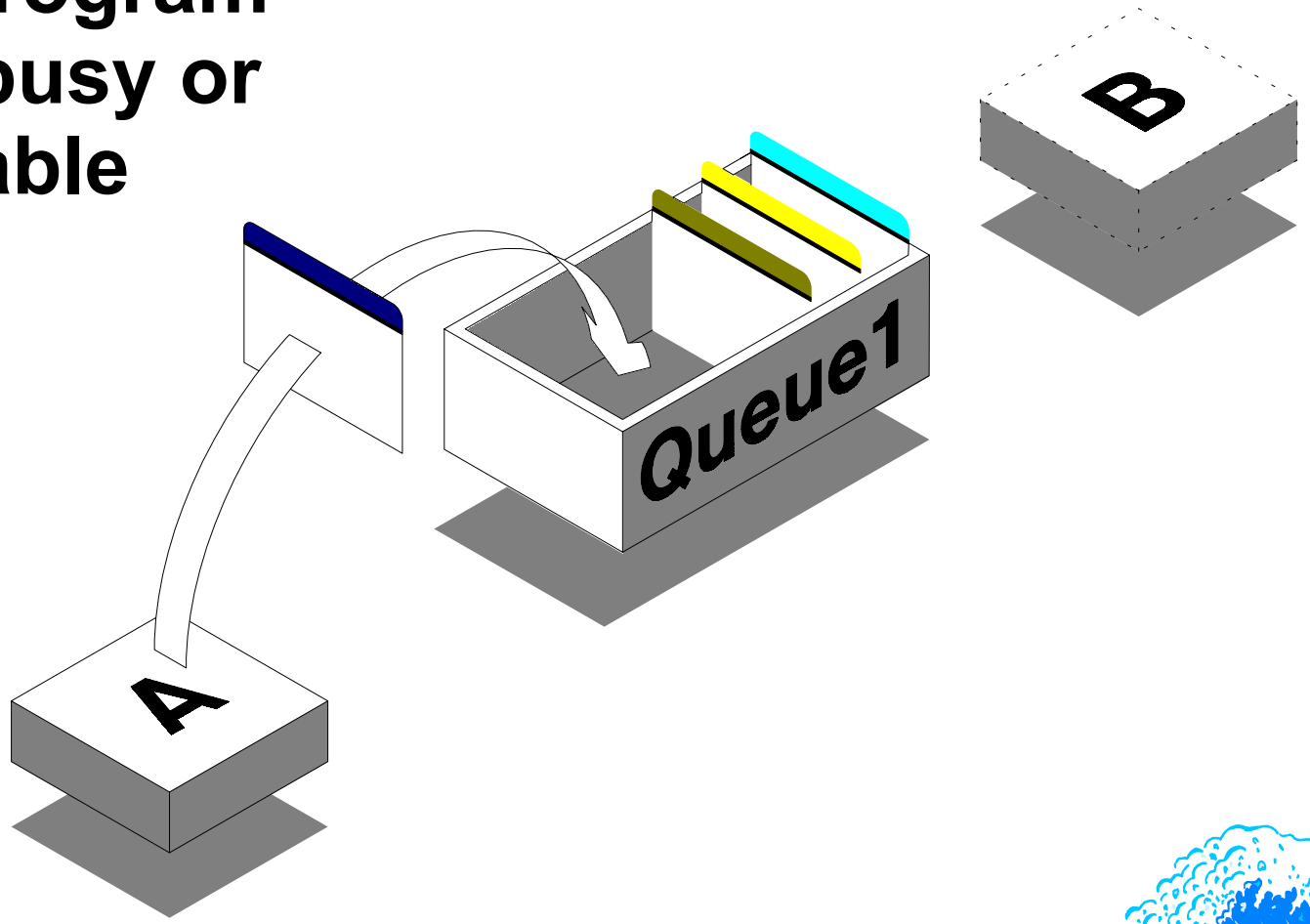
# Messaging and Queuing

- Communication can be one way or two way



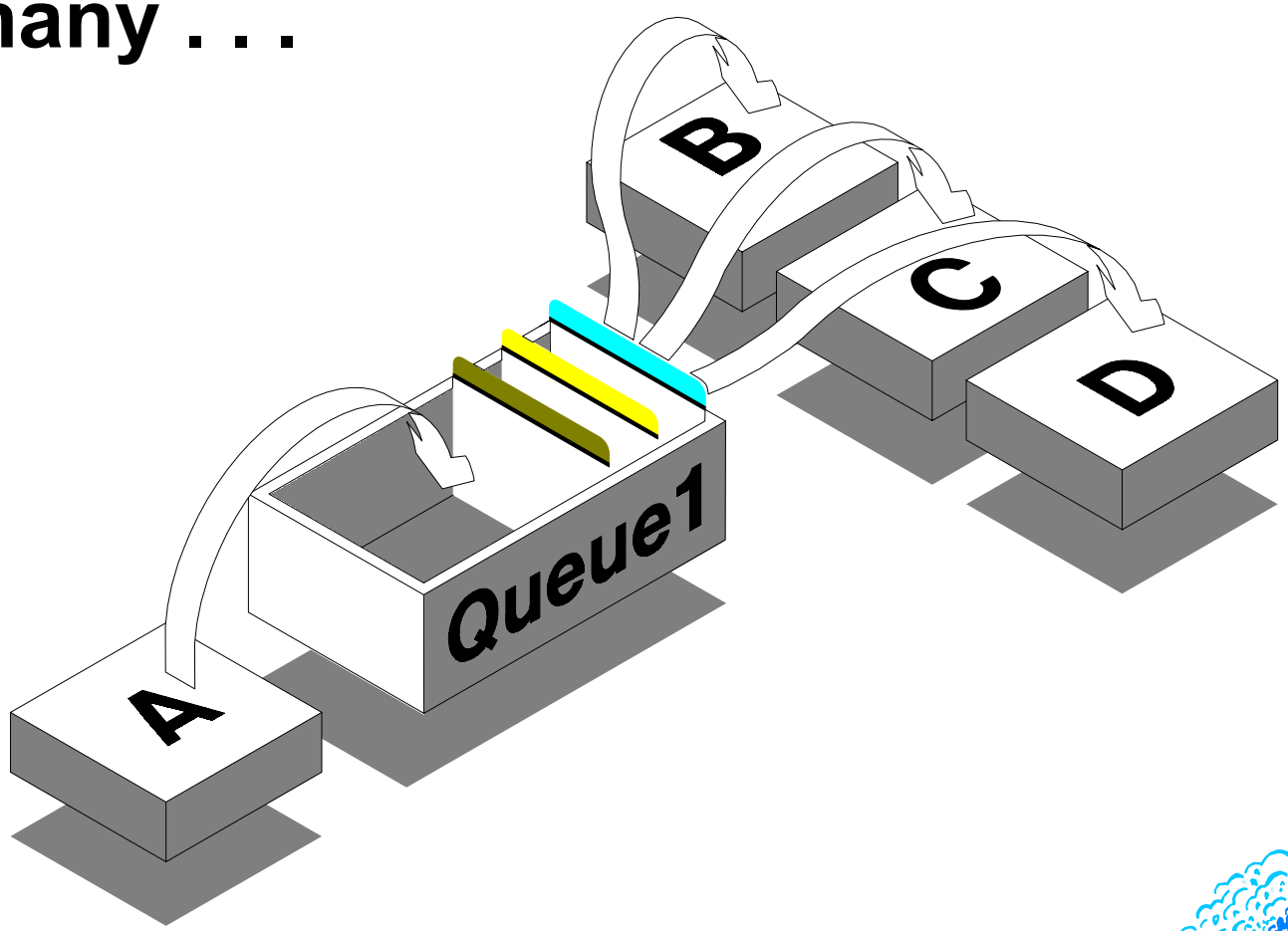
# Messaging and Queuing

- Either program can be busy or unavailable



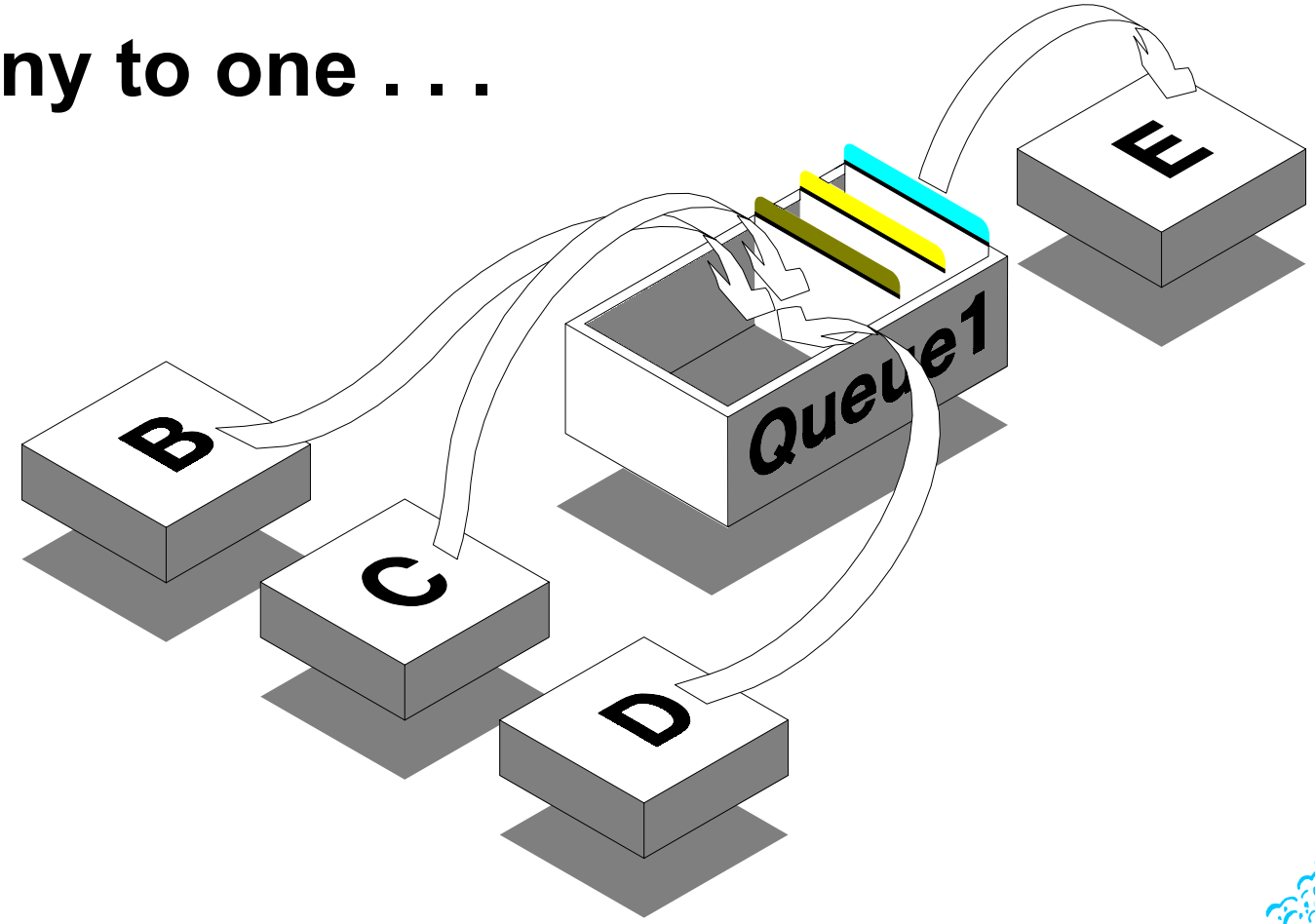
# Messaging and Queuing

- One to many . . .



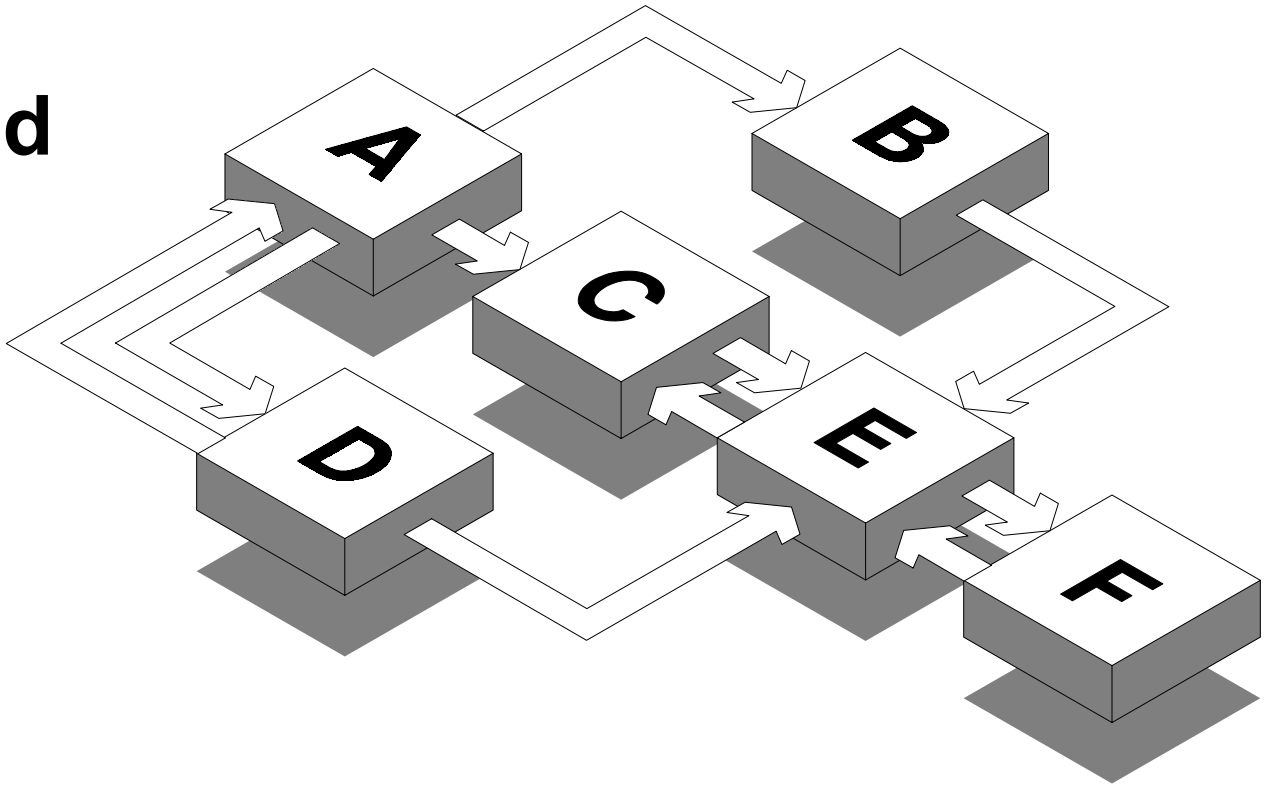
# Messaging and Queuing

- Or many to one . . .



# Messaging and Queuing

- **Which can be combined**




There are no constraints on application development

# What is MQSeries?

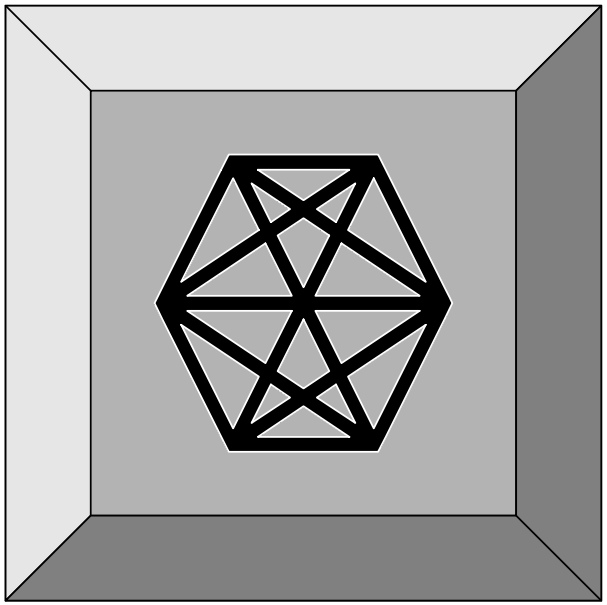
**IBM Products that enable business applications to exchange information, across different platforms, by sending and receiving data as messages . . .**

# The Unique Value of MQSeries

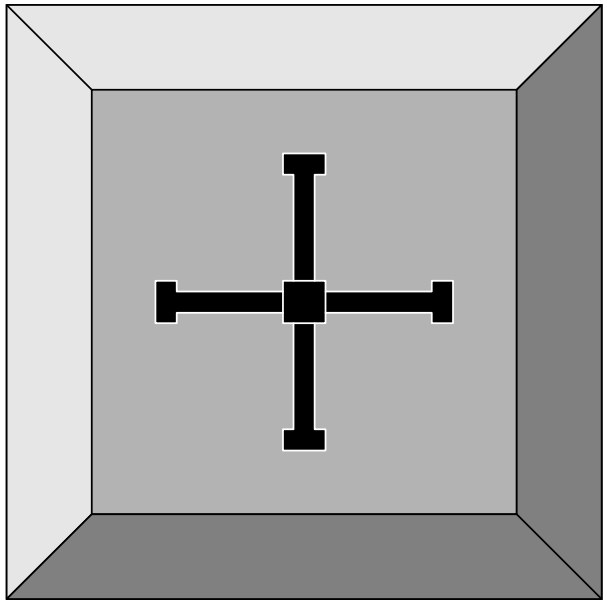
- One interface across many platforms (allows programs to be ported easily)
- Shield developers from network protocols
- Asynchronous API
- Programming benefits
  - Allows for speed differences
  - Not dependent on availability
  - Queues can be shared for input and output

# The Unique Value of MQSeries

- **Less networking by application**



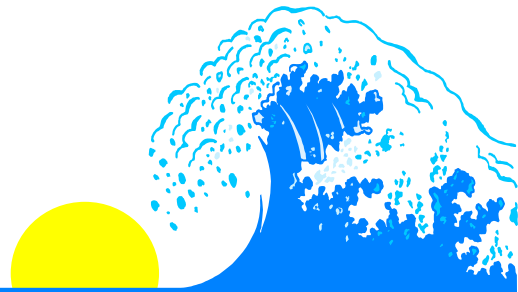
MESH NETWORK



STAR NETWORK

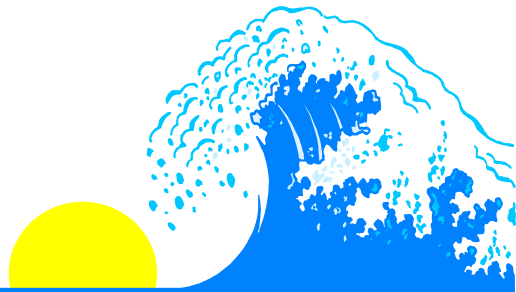
# MQSeries Server Platforms

- MVS/ESA
- Tandem NSK
- OS/400
- DEC Open VMS VAX
- DEC Open VMS AXP
- DYNIX/ptx
- SINIX +DC/Osx
- AIX
- AT&T GIS (NCR)
- HP-UX
- SunOS
- Sun Solaris
- OS/2
- Windows
- Windows NT



# MQSeries Hosts and Clients

- HOSTS
  - VSE/ESA
  - SCO UNIX
  - UnixWare
  - Windows
- Clients
  - Dec Open VMS VAX
  - Dec Open VMS AXP
  - DYNIX/ptx
  - SINIX + DC/Osx
  - AIX
  - AT&T GIS (NCR)
  - HP-UX
  - SunOS
  - Sun Solaris
  - SCO Unix
  - OS/2
  - Windows
  - Windows NT
  - MAC
  - JAVA



# MQSeries Client Connections

- Protocol Support depends on the MQSeries Implementation
- TCP/IP
- NETBIOS
- LU6.2
- IPX (Novell Networking)
- DECNet

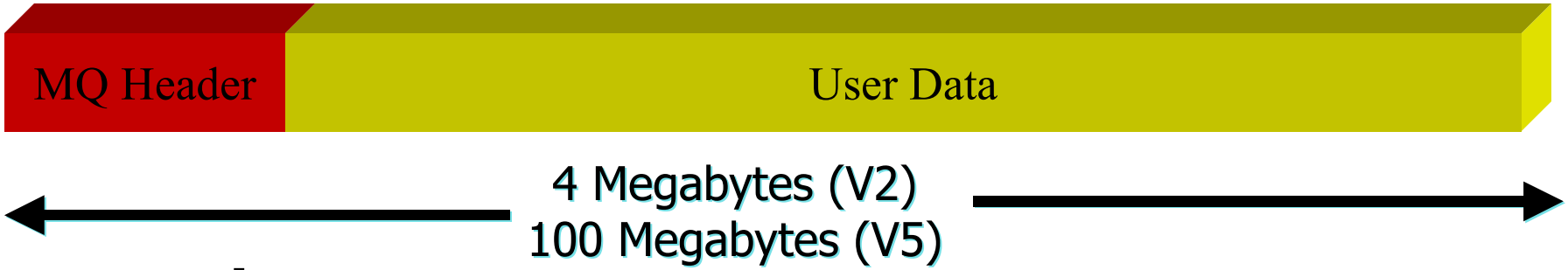
# Message Driven Processing

- Application design style
- The Application is divided into separate discrete functional blocks
- Inputs and outputs being interchanged by messages, which are put on queues
- Programs can start executing as a result of one or more messages arriving on a queue
- Leads to quicker development as compared to other design styles

# “What is a Message”

- **Message Type**
  - *Datagram* - no reply is required
  - *Request* - used when the sender of the message requires a reply from the receiver
  - *Reply* - used when the receiver of a request message sends a reply to that message
  - *Report* - used to inform an application about events that relate to another message
    - COA - confirmation on arrival
    - COD - confirmation on delivery
    - Expiration
    - Exception

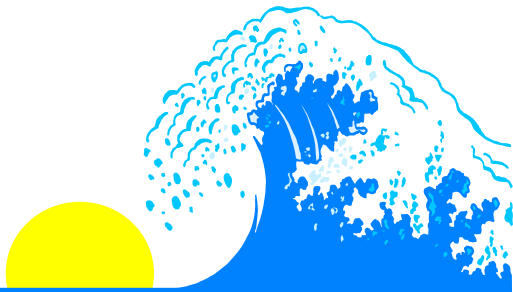
# “Attributes of a Message”




- Length
  - 4 MB on Version 2 Queue Managers
  - 100 MB on Version 5 Queue Managers
- Header
  - Defined by MQSeries
    - User may elect to use Message Id, Correlation Id, Priority and Persistence
  - Data
    - User defined, may contain any type of data (character data, image files, attachments, binary data, etc.)

# MQSeries Objects

- Queue managers
- Queues
- Namelists
- Process definitions
- Channels



# Queue Manager

**A Queue Manager provides the messaging and queuing services to application programs**

- **Some environments only support one Queue Manager (example: AS/400)**
- **In MVS/ESA applications are connected to a queue manager through the Batch adapter, CICS adapter, or IMS adapter**

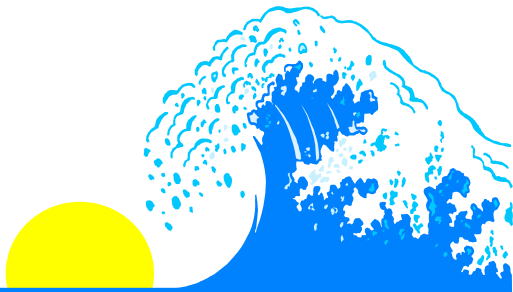
# Queues

- A place to store messages
- Each Queue has attributes
  - get enabled
  - put enabled
  - exclusive or shared
  - maximum number of messages (depth capacity)
  - maximum message size that can be put on the queue

- Physical implementation is not visible to the application program:
  - Main Storage Buffers
  - File(s) on Disk or other permanent storage
  - Both

# Queues

- Queues are owned by a Queue Manager, not by Application Programs
- Queue Definitions are held by Queue Managers
- Types:
  - Local
  - Remote
  - Alias
  - Reply-to
  - Model

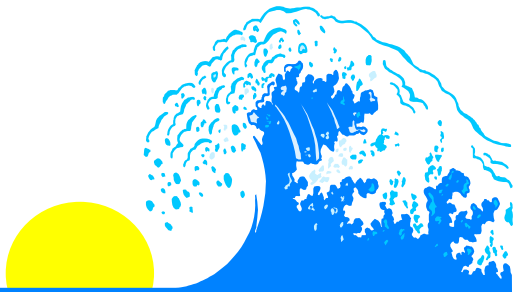


# Local Queue

- Resides on the same queue manager as the connected process
- May be opened as input, output or both
- May be used for one of the following uses:
  - *Normal*
  - *Transmission*
  - *Initiation*
- Optionally supports Trigger Processing

# Remote Queue

- Is not a “real” queue, is a definition only
- Represents a local queue on another Queue Manager
- Messages for it will be staged on a Transmission queue
- Can only be opened for output



# Transmission Queue

- A local queue with a special usage
- Minimum of one for each remote queue manager
- Default name of the remote queue manager
- Not intended to be accessed directly by application programs
- Supported by Message Channel Agent for transport

# Initiation Queue

- Local queue
- Enables automatic initiation of a program to process messages
- Messages are placed on it by the Queue Manager

# Alias Queue

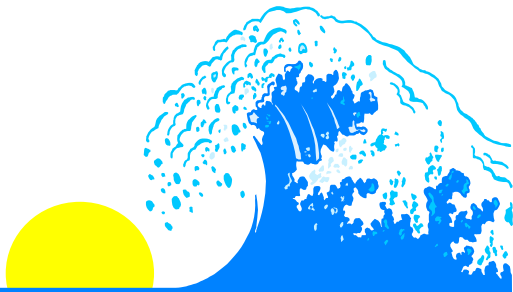
- Points to another queue
- Multiple aliases can exist for the same queue
- Maintained by the Queue Manager, it allows the user to use queues for different purposes or in different ways:
  - different default attributes
  - different security
  - different usage (input / output)

# Reply-to Queue

- Used to collect replies from requests or report messages
- Identified by ReplyToQ field in the message descriptor (MQMD)
- The name has two parts (may be resolved by the Queue Manager)
  - local queue name
  - queue manager name

# Model or Dynamic Queue

- A model queue when opened, becomes a dynamic queue
- Two kinds of dynamic queues are possible:
  - *Temporary* - deleted when closed
  - *Permanent* - lasts until specifically deleted
- Names of dynamic queues are generated according to a requested pattern



# Dead Letter Queue

- A repository for all non deliverable messages
- Always a local queue
- The Queue Manager adds a prefix (MQDLH) to explain who / why
- Design considerations:
  - Persistence
  - User is responsible for the procedure to clean up the Dead Letter Queue
  - Some implementations include a Dead Letter Handler

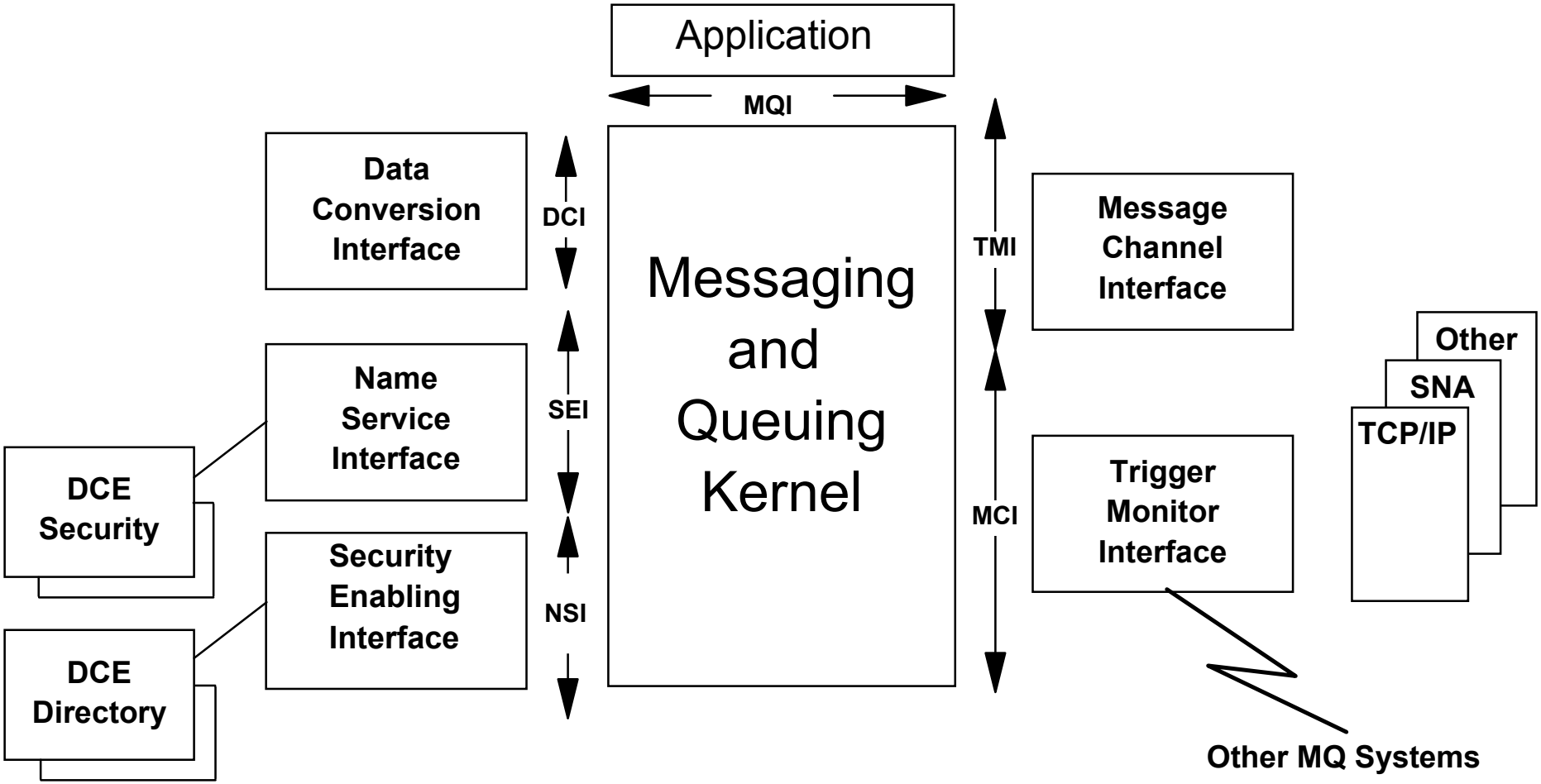
# Messages arrive in the DLQ ...

- The target queue is full
- The target queue does not exist
- The queue is PUT inhibited
- The sender is not authorized
- The logs are full (cannot log)

# Process Definition

- A process definition object defines an application to be started by a trigger monitor
- It includes the application ID, the application type, and application specific data

# MQSeries Framework



# Planning

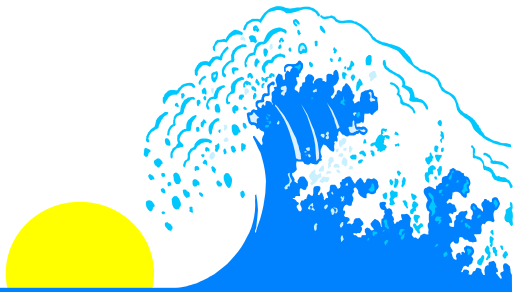
- What do I have to plan for?
  - Naming Convention
  - MQSeries Topology
  - Use of Name Resolution
  - Dynamic Routing
  - High Availability
  - Fail-over
  - Performance
  - Backup and Recovery
  - Installation
  - Administration

# Planning

- For system administrators, messages have two important attributes
  - Persistence: Survives an MQSeries restart
  - Priority: impacts the order in which a message is retrieved from a queue
- Priority can also impact the way that trigger events are generated

# Planning

- Network Topology Review
- MQSeries Routing and Name Resolution Strategy
- MQSeries Enterprise Naming Strategy
  - Queue Naming Convention
  - Message Channels
  - Client / Server



# Planning

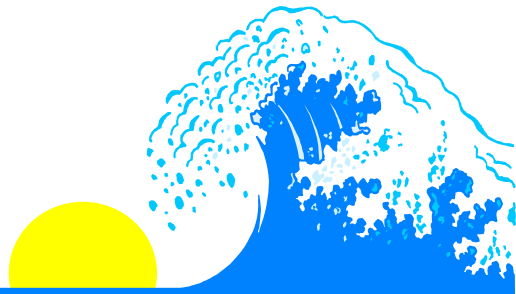
- Trigger Processing
- The ability to start a process based on the arrival of a message
- Three Types
  - FIRST
  - EVERY
  - DEPTH
- Trigger Monitors
  - Special programs that monitor Initiation Queues

# MQSeries Benefits

- Messages are sent once, and only once
- Messages can survive a Network Failure
- Messages can survive an O/S Failure
- Data flow is automatically resumed after a Failure
- Log Data preserves the messages

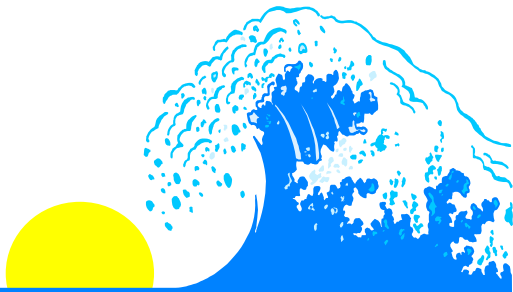
# MQSeries Benefits

- **MQSeries uses asynchronous messaging which eliminates the dependency on:**
  - **the receiving program**
  - **the network**
  - **the receiving system**
- **The sending process always completes its unit of work**



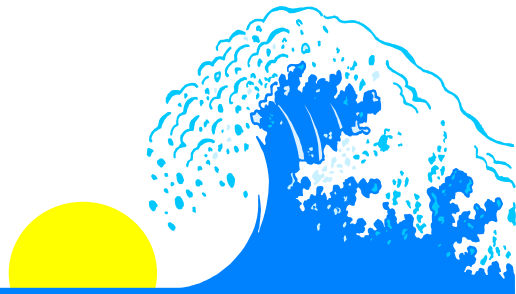
# MQSeries Benefits

- MQSeries abides by the restrictions imposed by the local and target system security
- MQSeries provides exit points to enhance security (encryption, authentication, non-repudiation)



# MQSeries Benefits

- MQSeries provides exit points to use Data Compression
- MQSeries performs Data Conversion from one platform to another
- EBCDIC to ASCII, ASCII to EBCDIC
- User defined conversion (foreign code pages)



# Error Recovery

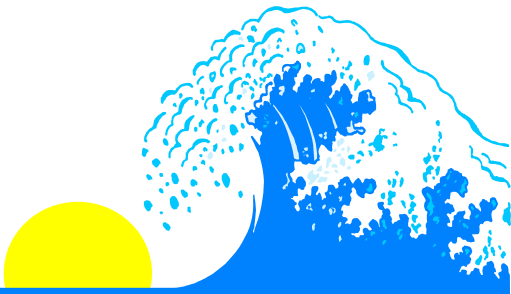
- If a remote queuing error or session error occurs, error messages are sent to the local system operator or console

- If any error occurs while sending or receiving a message, the transaction is terminated, and error messages are sent to both the local and remote system consoles

# Recovering from Errors

- If a message cannot be put on a remote queue, the message is written to the dead-letter queue and a report is sent back to the message sender only if requested

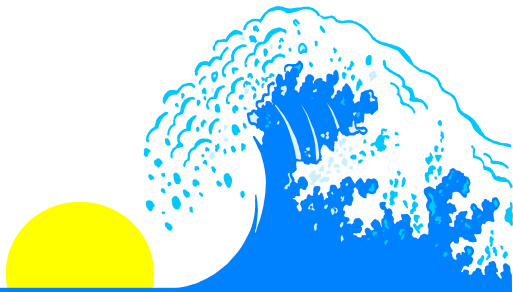
- It is important to ensure that there is a dead-letter queue defined for each queue manager



# How Triggers Work

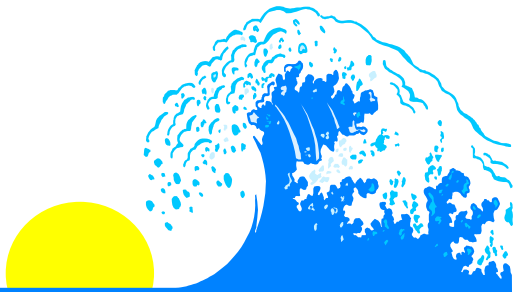
- When a message is put onto a message queue, a trigger is generated and the Queue Manager is notified

- The Queue Manager then writes a trigger message (containing some user defined data) to the initiation queue



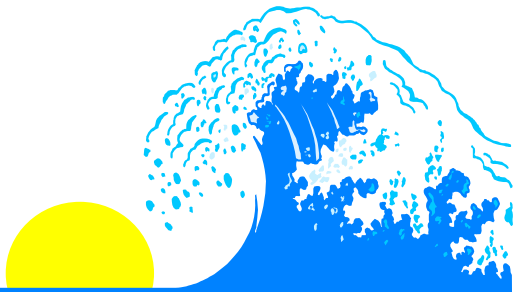
# Trigger Monitors

- Any process can be triggered
- Functionally equivalent to the MVS triggers
- Trigger Monitors operate the same
- Initiation Queues must be defined
- Process Definitions indicate what process to start

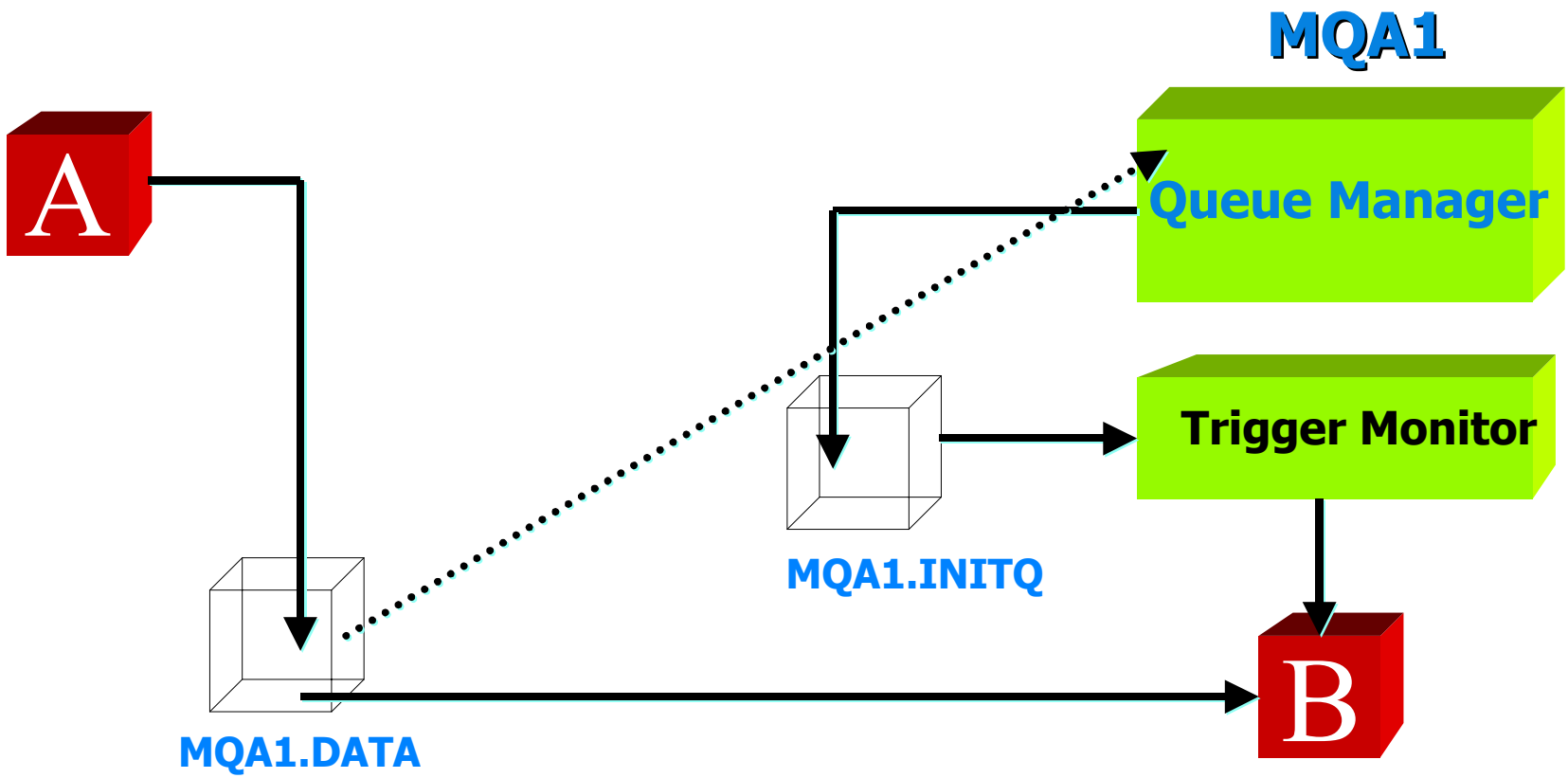


# Trigger Types

- There are three types of triggers:
  - First
    - The arrival of the first message satisfies the trigger condition
- Every
  - The arrival of each message causes a trigger
- Depth
  - When  a depth is reached it causes a trigger and resets the trigger



# The Trigger Process

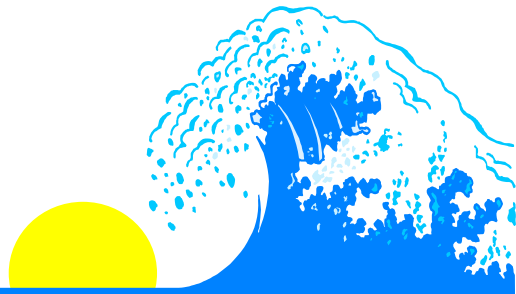


# Trigger Type First

- As soon as the first message arrives on the application queue, it satisfies the trigger condition, and the corresponding process is started
- The triggered application runs until there are no more messages to process
- Example: Airline Schedule Boards that receive updates sporadically

# Trigger Type Every

- *Every* time a message arrives on the application queue, it satisfies the trigger condition and the process is started
- The application runs until the single message is processed
- *Example*: a Server environment where requests come from multiple locations on an unpredictable basis



# Trigger Type Depth

- Messages arrive on an application queue and accumulate until they reach the depth threshold, this satisfies the trigger, the process is started and the trigger is reset
- The application runs until the messages are all processed (a logical unit of work) and sets the trigger condition on again prior to ending
- Example: Mortgage Application

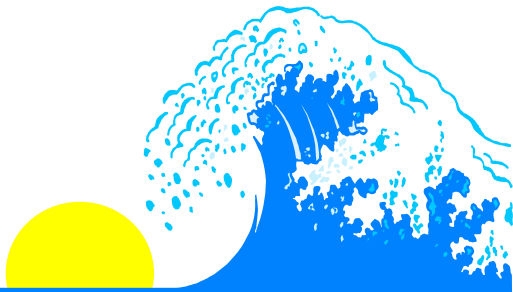
# Trigger Monitors

- A Trigger Monitor is a program that monitors an initiation queue
- It retrieves messages and processes them according to information in the trigger message
- Triggered Channels: the corresponding Message Channel Agent (MCA) is started to retrieve messages from the transmission Queue

# Why Choose MQSeries

# Why choose MQSeries

- Safe Investment - strategic to IBM
- Multi-Platform - over 25 IBM and non-IBM platforms supported
- Risk Free - world class support from IBM
- Customized - integrates with the Internet, Lotus Notes, SAP, TXseries, CICS, etc.
- De-Facto Leader and Standard - greater than 50% of the Market share

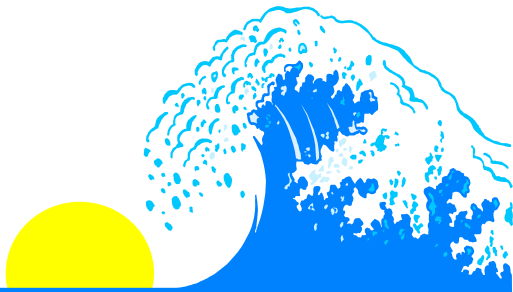


# Why choose MQSeries

- Assured Delivery - messages are delivered once and only once
- CME $2 trillion of options in a single day
- Running since 1995, have not lost a single message
- Recognized - 4 Industry Awards
- Dynamic and Modern - 200 developers adding new features and function (JAVA, OO, etc.)

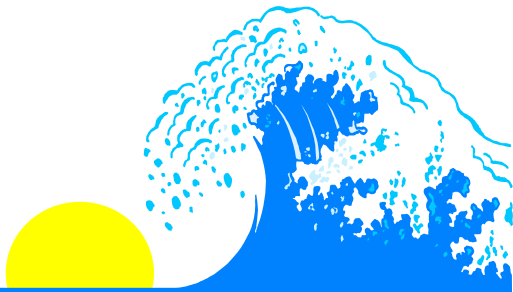
# Why choose MQSeries

- **Scalable**
  - **US Customs 23M messages per day soon to be 100+M messages per day**
- **Endorsed - frequent and positive coverage by all major IT Consultant Groups and IT Publishers**
- **Tried & Tested - In production for 4+ years**



# Why choose MQSeries

- **Well Supported - an industry of valued partners have invested in MQSeries providing:**
  - **Ports**
  - **Applications**
  - **Tools**
  - **Training**
  - **Consultancy**



# Why choose MQSeries

- **Cross Enterprise - the majority of MQSeries sales are for distributed servers, demonstrating MQSeries is not just a Mainframe product**
- **Well Connected - over 3,000 customers to-date including the majority of Fortune 100**
- **Productive**
  - **"Anticipated development effort of seven months was actually achieved in seven weeks after receiving MQSeries Software"**

    *- Barclay's Bank*

# Why choose MQSeries

- Managed - supported by several System Management vendors
  - BMC
  - Candle Corporation
  - Boole & Babbage
  - Technology Investments, Inc.
- Pedigree - developed at Hursley Lab, home of transaction systems for over a quarter of a century