RMDS White Paper Series

Core RMDS Distribution Architecture

Version 1.1

27 June 2001

REUTERS

Contents

Contents

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to familiarize readers with the fundamental concepts, protocols and components of Reuters Market Data System (RMDS) as they pertain to the distribution of real-time market data in the trading floor environment. RMDS is a highly scalable system that distributes real-time market information from a variety of market data sources to a large number of client applications.

Readers interested in just the high level description of the system need only read sections 1-5.

1.2 Scope

The Reuters Market Data System, taken in its entirety, consists of many components that provide a wide range of functionality across a number of data types and several distribution domains.

This paper only discusses a portion of RMDS—but a key portion. This is the "core distribution architecture," which integrates and distributes real-time price data, as well as unprocessed news and time-series data, in a LAN environment. The topics covered include:

- How data are requested, distributed, and updated in real time
- Strategies used to ensure data quality
- Resilience to failure at various points in the system
- Scaling of real-time services

There is a tremendous amount that this paper does not cover. Topics that are left for other papers include:

- Managing qualities of service (such as delaying and intervalizing traffic)
- Authorization (permissioning)
- Systems for transforming, filtering, and personalizing the unprocessed price, news, and historical data that come through the core distribution architecture
- System management
- Distribution outside of the LAN, such as Internet, wireless, and WAN

1.3 Glossary

API	Application Programming Interface
Broadcasting	Sending data or control messages to any and all interested parties on the network.
Broadcast feed	A datafeed that transmits in only one direction—from the feed provider to the customer site. All sites receive the same set of transmissions.
Cache	In-memory live database.
Cachelist	List of subjects/instruments held in memory at a particular moment in time.
Consumer	Any client application that subscribes to data. Also called a *subscriber*.

DTIC	Delay TIC
Fault tolerance	The ability of a system or component to continue to operate correctly despite hardware or network malfunction. The main fault tolerance mechanisms are called *N+1*, *hot standby*, and *source mirroring*.
Feed	A wire (or wire service) that carries market data to a customer site. See also *Producer*.
Feed handler	A server process that interfaces a datafeed to the Reuters Market Data Hub. This is one type of a source application.
Heartbeat	A status message sent periodically by a process to indicate that the process is alive.
Hot standby	A fault tolerance scheme employed by the RTIC in which one server process is active and the other stands ready. If a currently active process fails, the secondary process becomes the active process. See also *Fault tolerance*.
Interactive feed	A datafeed that sends data requested by clients.
Load balancing	A strategy to (according to some criteria) distribute work load across multiple source servers.
Market Data Hub	A combination of feed handlers and supporting infrastructure.
N+1	A fault tolerance scheme using multiple secondary servers (backups). Currently not implemented by the Market Data Hub.
Producer	A source of information, such as Reuters DataFeed, internal rates engine, or contributions spreadsheet. Also called a publisher. See also *Feed*.
Publishing	Making data available for general consumption.
REA	Reuters Enterprise Architecture. This is a technical reference architecture for financial eBusiness.
RFA	Reuters Foundation API
RMDS	Reuters Market Data System
RRCP	Reuters Reliable Control Protocol. This is a transport protocol that builds reliability on top of UDP/IP. It is used to send RRMP packets between components on Market Data Hub. It is similar in function to the RV protocol.
RRDP	Reuters Reliable Datagram Protocol. This protocol is used to communicate between the key Market Data Hub processes. It is made up of two protocol layers—RRCP and RRMP.
RRMP	Reuters Reliable Management Protocol. This is the market data semantics that hub components use to communicate with each other. It uses the RRCP layer for reliable end-to-end communications.
RTIC	TIC—RMDS Edition. This real-time in-memory cache stores the most current market information indexed by subject name. It uses RRDP to obtain market data.

RV	TIB/Rendezvous. This is a transport protocol that builds reliability on top of UDP/IP. It is used by the components on the Distribution Layer for reliable end-to-end communication.
SASS	Subject Addressed Subscription Service protocol used by the components on the Distribution Layer to support market data semantics.
Service Manager	An optional Market Data Hub component that implements advanced information resource management features. The Service Manager ensures that requests from the RTIC are directed to the most appropriate Source Distributor at the optimum rate.
Service	A meaningful set of functions implemented by one or more servers working together. A service is a virtual entity. Applications interact with the service as if it were a single entity, even though it may be implemented by several processes on more than one host computer.
SFC	SSL Foundation Classes
Snapshot	The value of an instrument at a particular moment in time.
Source Distributor	A Market Data Hub component that provides market data. It is responsible for responding to image requests, forwarding updates and (optional) caching.
Source application	An application that writes information to the network. Source applications connect to a Source Distributor in order to supply data from an external information source (e.g. datafeed) or internal information source (e.g. spreadsheet).
Source mirroring	A fault tolerance scheme employed by the Market Data Hub infrastructure using a primary and secondary source server to provide tick-by-tick data recovery.
STIC	Snapshot TIC
Subject	A name for identifying data objects. For example, the subject `EQ.IBM` might identify all pricing data about IBM stock, while `EQ.IBM.N` might identify price data from the New York Stock Exchange only. Also referred to as an instrument or item.
Subscribing	Requesting a stream of data by subject name.
TIB	The Information Bus—this is the TIB market data system
TIC	TIB Information Cache
TMF	TIC Message Filter
Watchlist	List of subjects/instruments subscribed to by one or more consumers at a particular moment in time. See also *cachelist*.

1.4 References

[1] *Reuters Market Data Systems and the Trading Solutions Architecture*

[2] *TIB/Rendezvous Concepts*

[3] *SFC Developer's Guide*

2 Reuters Market Data System Overview

Before discussing the core distribution architecture, it is useful to put it into context. The Reuters Market Data System can be conceptually broken down into several pieces:

- The Distribution Layer is based on TIB/Rendezvous (RV). This is the enterprise “backbone,” as it provides reliable multicast transport for all data and services within the enterprise—market data and otherwise. Components in this layer cache, filter, and personalize data.
- The Market Data Hub is a layer to support publishing applications that require special reliability features typically associated with market data. The most common example is datafeeds from vendors and exchanges—i.e., from the markets—but the Market Data Hub is often appropriate for internal applications as well.
- The Internet Finance Platform delivers streaming and snapshot data to browsers, pagers, cell phones, and other personal devices.
- Across all these domains is a single set of APIs—the next generation SFC—which incorporates the high-level models of today’s SFC, as well as low-level interfaces evolved from the TIB API suite.
- New value-added systems for news, time series, and analytics go well beyond what has been offered on TIB and Triarch to date.
- DACS is used to permission market data and potentially anything else that users view and do.
- TIB/Hawk manages systems and applications.
- TIB/Repository is for centralized configuration. **Note**: This capability will not be available for deployment in the near future.
- Kobra is a professional desktop integration environment, with a rich set of user-interface objects related to market data.
- PowerPlus Pro provides real time market data, reference data, and analytics into the Excel container.

This paper explains the architecture of the core distribution infrastructure of RMDS. This includes the Market Data Hub and core portions of the Distribution Layer

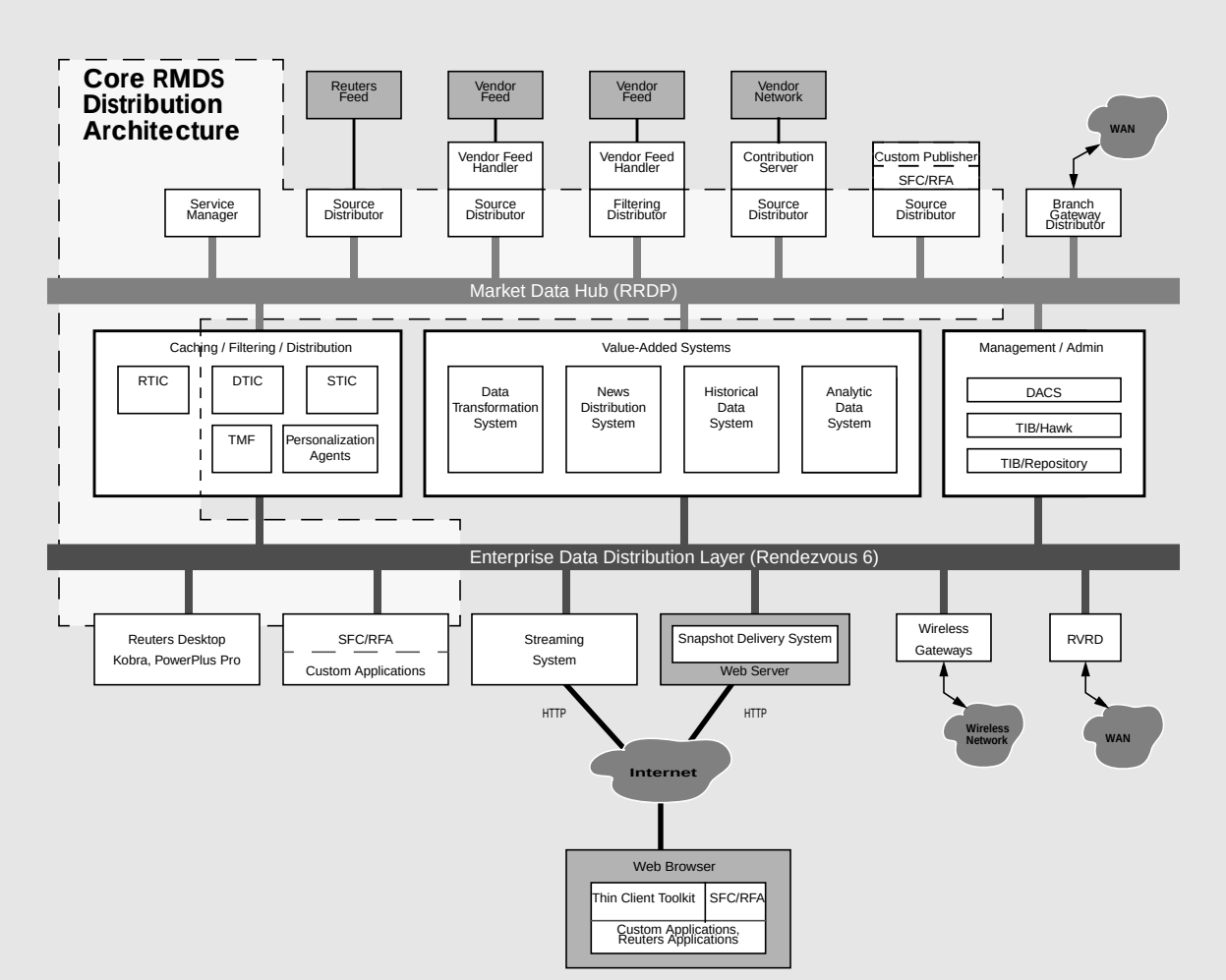

Figure 1: Conceptual View of RMDS

3 Core Market Data System Overview

The core RMDS distribution architecture facilitates the reliable flow of real-time market data from datafeed handlers and other programs supplying information through the core RMDS distribution architecture, to a range of analytical and display applications.

The core RMDS distribution architecture provides the underlying software components that manage the data flow both to and from the network. Applications, programmed with one of the editions of the SDK using the SFC/RFA, interface to one another through the core RMDS distribution architecture. SFC/RFA consumer applications can retrieve market data from all network sources, subject to being appropriately permissioned. Likewise, SFC/RFA source applications are able to publish market data and other types of data, also subject to being appropriately permissioned.

The SFC/RFA offers a set of well defined high-level and low-level capabilities (implemented as a class hierarchy) to enable rapid development of domain specific applications.

This section describes the basic features, components, and data flow within the core RMDS distribution architecture.

3.1 Product Features

The key features of the core RMDS distribution architecture are:

- Delivery of a wide variety of market information including real-time quotes, chains, time-and-sales, news stories and news headlines.
- Delivery of page-based information.
- Support for broadcast and interactive data publishers.
- Intelligent in-memory local data cache (through the use of RTIC).
- Intelligent datafeed management and load balancing.
- Scalable architecture that supports a very large number of users.
- Timely failure detection and notification.
- Support for resiliency and fault tolerance.
- Built on industry standards.
- LAN-based delivery.
- Support for IP multicast and IP broadcast.
- Flexible market data traffic management.
- Support for contributions to external information vendors.
- Support for access control (through the use of DACS).
- Manageability support (through the use of TIB/Hawk).
- Open/Extensible--easy integration of new feeds and client applications via the SFC/RFA.

3.2 A Typical Network

Depending on requirements, it is possible to create RMDS-based solutions which vary from a small set of components or just a few machines, to large and complex integrated solutions.

To better understand the interactions within the core RMDS distribution architecture, it is beneficial to look at a basic network able to support a small to large group of users. A recommended topology for the basic RMDS network employs separate LAN segments for the Market Data Hub and Distribution Layer in order to separate the market data traffic from the client traffic.

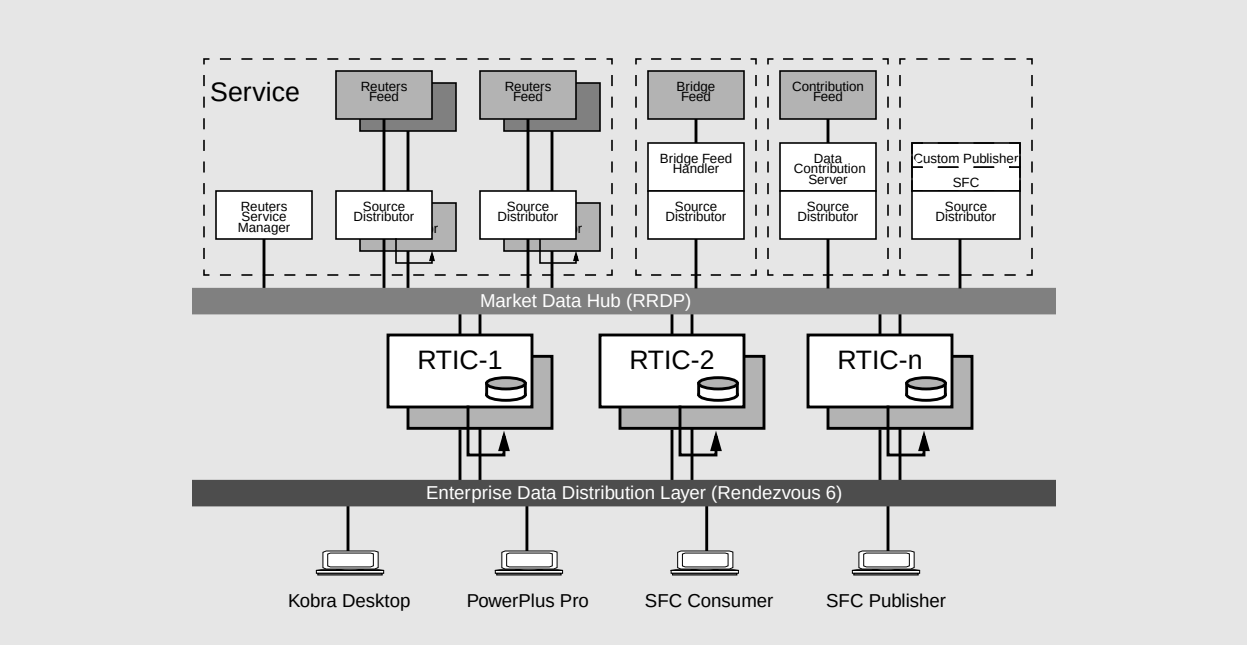

Figure 2: Typical RMDS Network Environment

3.3 Basic Data Flow

1. Applications make requests over RV by multicasting the subject name that is of interest.
2. An RTIC that is configured for the subject space pays attention to the request.
3. If the item is not in the RTIC cache, the RTIC obtains it from the Market Data Hub, with or without the help of a Service Manager. The Service Manager is an optional process that helps optimize a service that consists of many load balanced servers.
4. Once it obtains the image, the RTIC caches it and sends it over RV to the requester. All updates to that subject are then multicast.
5. Subsequent requests for that same subject are satisfied from the RTIC cache and updates are obtained from the same multicast stream. There is no point-to-point fanout.

The advantage of being able to use multicast is that where there are a large number of recipients for a message, the message only needs to be sent once. This offers a tremendous benefit in reducing the load on the sender of the message and the network when compared to sending messages point to point.

In Figure 2, the nodes represent data **producers**, **consumers**, and **information cache** applications. These components function as follows:

- **Producers on the Market Data Hub**—or source servers/publishers—are applications that supply information. These sources typically interact with a vendor's datafeeds, but also may distribute proprietary value-added data. Each of these source applications is built with the SFC/RFA or the legacy SSL "C" API. The SFC/RFA communicates with the Source Distributor, which encapsulates source independent functions like data caching, source mirroring, load balancing, preemption, support for interactive and broadcast feeds, etc. in order to simplify development of feed handlers. Developers of feed handlers are mainly concerned with handling of the vendor specific datafeed protocol details.

  Multiple sources can be managed by an optional infrastructure component called the Service Manager, which provides centralized request routing and service-wide intelligent recovery.

  When needed, the source mirroring capability (see section 6.7.6) can be configured to provide fault tolerant operations. Multiple pairs of mirrored servers can be deployed and load balanced within the same source service.

  In excess of 50 different source servers are available for RMDS, including handlers to access information from Reuters, Bridge, Knight-Ridder, as well as many direct stock exchange and Inter Dealer Broker (IDB) feeds. Also available are data publishing/contributing applications that work in conjunction with a Data Contribution Server (DCS) to contribute data to information vendors.

- **Producers on the Distribution Layer**—or local publishers—are applications that typically distribute proprietary value-added or internally available data. Each of these applications is built with the SFC/RFA. The SFC/RFA Library provides direct communication to the information cache (RTIC). Support is limited to broadcast type publishers. There is no direct support for load balancing, fault tolerance, and state management.

- **Consumers**—or subscribers—are applications that consume real-time market information for the purpose of computation or display. These processes are often display oriented applications that add value to the underlying raw data. Consumer applications are built with SFC/RFA.

  As part of the RMDS product line, Reuters provides two strategic professional desktop applications: Kobra and PowerPlus Pro.

  In addition, server-based applications can also act as consumers.

- **TIC—RMDS Edition (RTIC)** is a high-speed, in-memory data cache and intelligent distribution logic that protects publishing applications from the request traffic of subscriber applications, while accelerating response time for those subscribers. It consumes data provided by producers on the Market Data Hub or Distribution Layer, caches it and re-publishes to consumer applications upon request. Each RTIC instance is configured to support a unique set of subjects/data items. The RTIC is a key component of the core RMDS distribution architecture providing the means for integrating market data from the Market Data Hub with other data and services throughout the financial enterprise. The RTIC is able to support a wide variety of applications, from simple broadcast publishers on the Distribution Layer to extremely large feeds on the Market Data Hub. In addition, end users are able to utilize the RTIC cache to publish data for internal consumption by other users.

  The RTIC can handle all forms of data, such as quotes, chains, time-and-sales, news stories and headline streams, etc. Information such as news and time-series is only available in a basic form via the TIC. Most customers will prefer to access these data types via value added systems such as News Distribution System, etc., which are described in separate documents. Access to RTIC data is permissionable by DACS.

  The RTIC helps to protect the Market Data Hub by caching data locally. This enables requests from new consumers for data already cached to be satisfied without making additional requests on the Market Data Hub.

  RTICs are deployed in the fault tolerant configuration to provide resiliency.

4 System Design Goals

The core RMDS distribution architecture has been designed with several key goals in mind, including:

- Integration
- Openness
- Flexibility
- Efficient and Reliable Communication
- Scalability
- Optimum Use of Resources
- Failure Detection and Notification
- Resiliency and Fault-Tolerance
- Manageability
- Access Control

4.1 Integration

Trading floor environments demand open, extensible and flexible systems that can support a rapidly changing set of requirements. The architecture to support trading floors must address not just the market data delivery, but should also allow integration with other real-time decision support applications such as:

- Settlements
- Order routing
- Portfolio and risk management
- Analytics
- Distribution of internally generated information

It is clear that in order to meet such diverse requirements, a flexible and high-performance messaging platform capable of supporting enterprise-wide business processes is needed. The core RMDS distribution architecture employs TIBCO messaging technology—TIB/Rendezvous (RV)—to meet these requirements. RV provides a powerful and flexible platform—based on reliable multicast—for development and integration of distributed applications. RV is practically an industry standard.

4.2 Openness

Using common protocols, APIs and message formats, all RMDS components are able to easily exchange information with each other in real time. This also facilitates integration of custom applications developed by the clients.

SFC/RFA offers a set of well defined high level and low level capabilities (implemented as a class hierarchy) to enable rapid development of domain specific applications.

4.3 Flexibility

The system integrator has flexibility to decide what is the best way and place to deploy shared data and services across the entire network. Users are not aware whether their requests are satisfied locally or remotely from a site connected via a WAN.

The core RMDS distribution architecture is a fully distributed system built using open systems principles. It has been designed to make as few assumptions as possible about the environment in which it runs.

The use of industry standards TCP/IP and UPD/IP for communication between components allows great flexibility in terms of the physical network implementation.

4.4 Efficient and Reliable Communication

In a financial environment where the timeliness of information delivery is key, much attention must be paid to delivering data quickly. This includes optimizing the delivery mechanism to account for the possibility that a significant portion of the data will be of interest to multiple users or applications, and ensuring that the state of the information delivered, or due to be delivered, is well known and well managed.

The core RMDS distribution architecture ensures maximum efficiency and reliability in the distribution of data by employing reliable multicast/broadcast-based communication protocol on the Distribution Layer and reliable broadcast-based communication protocol on the Market Data Hub. These protocols utilize the power of IP multicasting and IP broadcasting in order to guarantee timely delivery of real-time information from information sources to potentially thousands of interested consumers. Multicast-based communication permits a high fan-out of data in an efficient manner. Only a single network packet is needed to deliver a single piece of data to multiple consumers.

4.5 Scalability

The system must be able to support a very large number of applications and users, and scale with the growth of the business. This should include the ability to operate across geographically disperse sites of the enterprise. The growth should be primarily achievable by the addition of additional hardware and software components without the need to upgrade the existing infrastructure components. Also, there should be no need to change the basic method of application integration.

The core RMDS distribution architecture employs, in addition to the highly scalable multicast/broadcast-based communication protocols, the old and proven strategy—”divide and conquer”—to achieve massive scalability.

- An information database can be distributed across multiple host machines.
- A client network can be segmented across multiple LAN segments.
- An information database can be replicated at each client network.
- Source services can be distributed across multiple feed handlers, each running on different host machine.
- Data traffic can be segmented across multiple IP multicast domains.

4.6 Optimum Use of Resources

The core RMDS distribution architecture employs various strategies to optimize use of communication, CPU and datafeed resources in order to minimize ownership cost and improve service provided to users.

4.6.1 Communication Resource Management

IP multicast/broadcast based data distribution, interest-based delivery of data and cache replication are the key strategies employed by the core RMDS distribution architecture to conserve bandwidth. Interest based delivery is implemented at subject and field levels, driven by user subscriptions and/or by static configuration. The cache replication keeps the request/response traffic on the network segment local to the requester while also accelerating response time.

Other methods used to minimize communication resource requirements include only expanding enumerated data types and handling ripple fields at the desktop. Intervalization of data can also be employed to reduce communication requirements.

4.6.2 CPU Resource Management

Interest-based data distribution as described in section 4.6.1 and data intervalization are the key strategies employed by the core RMDS distribution architecture to conserve CPU. Intervalization is the ability to limit the number of updates per time interval for a given instrument. Data stream intervalization can be deployed on the Market Data Hub to protect the whole market data system, and/or on the Distribution Layer to protect user workstations.

Data stream intervalization reduces rate of updates and can also be effectively applied to conserve communication resources.

4.6.3 Datafeed Resource Management

The core RMDS distribution architecture has been designed to effectively manage complex, limited and expensive data resources provided by vendor datafeeds. Where multiple source servers implement a single service (i.e., multiple feed handlers of the same type are installed), the system is designed to provide both resilience in case of failure and optimum usage in normal operating circumstances. Where the vendor feeds provide a complex set of constraints—such as cache limitations, data request (or throttle) limitations, and the management of mixed data delivery mechanisms—optimized management is key. The datafeeds can be of interactive or broadcast type.

Key design features for this optimization include:

- **Transparent to users**

  Multiple instances of a source server appear as a single service to the users and their applications.

- **Non-duplication of supply**

  Where multiple instances of a server are present, the system strives to ensure that any given item is supplied—or cached—by only a single server.

- **Full utilization of cache**

  All available server cache slots will be used before new item requests force item preemption.

- **Management of constrained resource situations**

  Where there are not enough resources to go around, managed item preemption is possible.

- **Optimum response timing**

  Intelligent placement of requests to ensure rapid response. If any source server is already supplying a given item, a new request for that item will automatically be fulfilled from that server. In the case when multiple items are requested at once and resources are available on multiple servers, the requests are spread across the servers; i.e. the requests are effectively grouped and different groups are sent in parallel to different servers.

- **Request throttling** (to avoid server overload)

  The Infrastructure includes facilities by which source servers can indicate that they are temporarily unable to service requests (or throttled) as well as indicate the maximum request queue size which they can accept.

- **Priority based preemption** (for environments where cache resources are limited)

  The system provides a powerful scheme to set relative priorities for items. These priorities are used to choose the optimum candidate, i.e. cached item, to preempt to make way for a new request. The priority scheme enables various features to be implemented, including locked items and unconditional preemption candidates.

- **Automatic re-request management**

  The system (rather than the user) manages request retries, effectively asserting control where there may be insufficient resources to go around, and gracefully manages server failure/failover.

- **Item blocking**

  Items can be blocked from cache so that they are effectively barred from use.

4.7 Failure Detection and Notification

Prompt detection of various failures and efficient notification is critical in the trading floor environment where the value of information is time-sensitive.

The core RMDS distribution architecture employs heartbeats to promptly and reliably detect failures of RMDS components. For efficient failure notification the core RMDS distribution architecture uses Group State messages which allow a source server failure to be reported using a single message.

4.8 Resiliency and Fault-Tolerance

Operations that are resilient and fault-tolerant are mandatory in a mission critical information distribution system. Fault tolerance is a system ability to provide un-interrupted service despite failure of 1 to n-1 components, where n is the total number of redundant components configured to provide the fault tolerant service. Resiliency is a system ability to minimize disruption of service in the event of system failure through the use of backup/redundant resources. However, the failure is not transparent to the affected users.

The fault tolerant operations are transparent to both the source applications and consumer applications. The net effect is that there is no single point of failure in the core RMDS distribution architecture network.

The core RMDS distribution architecture implements a series of measures to ensure that the resiliency and fault tolerance capabilities are handled as effectively as possible.

Resiliency and fault tolerance are provided by both the Source Distributors and the RTIC. Depending on the level of resiliency required, the Source Distributors can be configured in either load balanced or source mirrored modes.

In load balanced mode, multiple source servers implement a single service. If an individual source server fails, the system automatically recovers failed items—sharing the load equally on the remaining servers.

If the loss of even a single update is unacceptable, source mirroring can be employed. In this case if a source fails, a backup source will take over ensuring that no updates are lost.

Also the RTIC can be configured in a hot standby mode where, in the event of a failure, a secondary RTIC can take over.

4.9 Manageability

Keeping the enterprise scale system running without disruptions is mandatory in a mission-critical market data distribution system. System management plays an important role in achieving that objective.

All core RMDS distribution architecture components are deeply instrumented for publication of management data to enable custom TIB/Hawk-based management tools to monitor and control the behavior of all components.

4.10 Access Control

The market data distribution system must provide facility for controlling access—at various levels—of publishing and subscribing applications to the market data system. Without such control, wide open access to all available data may be cost prohibitive, and also be a security risk.

The core RMDS distribution architecture provides for data authorization enforcement via the Data Access Control System (DACS). RMDS system administrators use DACS to control access rights to information on the network and obtain usage reports. DACS is described in a separate document.

5 System Model

This section introduces the models and concepts on which the core RMDS distribution architecture is based. It treats the core distribution architecture as a “black box” when describing how data enters and exits the architecture.

5.1 Market Data Model

At its most abstract level, the Market Data Hub may be represented as a collection of services that provide data for consumption by application components. A service provides data items of a particular type. Items are retrieved by a name which is unique in the context of the service.

5.1.1 Services

Common characteristics of services are:

- Services are uniquely identified by name.
- Services provide data of a particular type and format, e.g., page-based, record-based, or application defined.
- Services have state and undergo state changes which may be indicated by events with explanatory textual information.
- Services may provide alerts in the form of broadcast messages, e.g., data items worthy of special notice.
- Services may be permissioned to control access to their data; that is, users may be authorized to access data from the service.
- Services provide access to instruments via interactive or broadcast feeds.

5.1.2 Data Items

Common characteristics of data items are:

- Items have data format and type. Standard types are pages or records. Other user-defined data types may also be used.
- Items may be in different states (e.g., OK, STALE, or CLOSED) and may undergo changes in state. The state coupled with the information provided by a data item is often referred to as a **data stream**.
- Items may have priority relative to other items.
- Specific groups of items (e.g. the items that come from one exchange or that comprise a Reuters product) may be permissioned to control access to data.

There are semantics associated with data items which are defined by the type of item and the particular service providing the data.

- **Page Items**
  Page items are supplied from source services that format their market data as pages (e.g., an IDB feed such as Cantor Fitzgerald). Page items are a row/column ordered set of cells comprised of character values and presentation information (such as character set, foreground color, highlight, etc.). Pages may have semantics that relate to vendor specific constructs (e.g., display modes).

- **Record Items**
  Records come from record-based source services (such as RDF) and contain a collection of fields (e.g., last trade price, current bid price) pertaining to a particular financial instrument. The content of that collection is not pre-defined and may grow or shrink over time. Higher level semantics relate to underlying market instruments (e.g., ticks associated with market movements) or to vendor specific constructs (e.g., Reuters IDN Chains). Updates are received as ticks (up or down), corrections, or close and verify. There are different types of fields and each field may be individually updated.
- **User Defined Data Items**
  Data items may have other formats, provided that both the services providing the data and the consumer applications receiving the data agree on the meaning of the data item.

5.2 Publisher/Subscriber Model

The core RMDS distribution architecture facilitates the reliable flow of real-time market from market data publishers to market data consumers. The communication between data publishers and data consumers is anonymous. The consumers don't know who and where the publishers are. The publishers don't know who and where the consumers are. The location transparency provides important benefits such as:

- De-couples the publishers and consumers
- Transparent relocation of services
- Transparent recovery from publisher failures
- Transparent upgrade of publishers, consumers and the core RMDS distribution architecture components

The core RMDS distribution architecture provides persistence of data published by interactive and broadcast market data producers through the use of caching. Caching has the important benefit of protecting the publishing applications from the request traffic of subscriber applications, while accelerating response time for those subscribers.

The publisher and consumer applications are developed on top of the SFC/RFA to gain access to the core RMDS distribution architecture. The SFC/RFA:

- Provides a single entry point to the system
- Enables rapid development of applications
- Supports full market data semantics—including well defined data stream states
- Provides support for access control to market data
- Shields applications from future changes to the underlying components of the core RMDS distribution architecture

Refer to reference [3] for a detailed and up-to-date description of SFC/RFA facilities.

The core RMDS distribution architecture provides two separate underlying service layers to support SFC/RFA-based publishers—the Subject Addressed Subscription Service (SASS) on the Distribution Layer, and the Reliability Layer on the Market Data Hub. These layers offer different levels of service and will be discussed in the sections which follow. It should be noted that the SFC/RFA makes these differences transparent to the publisher applications.

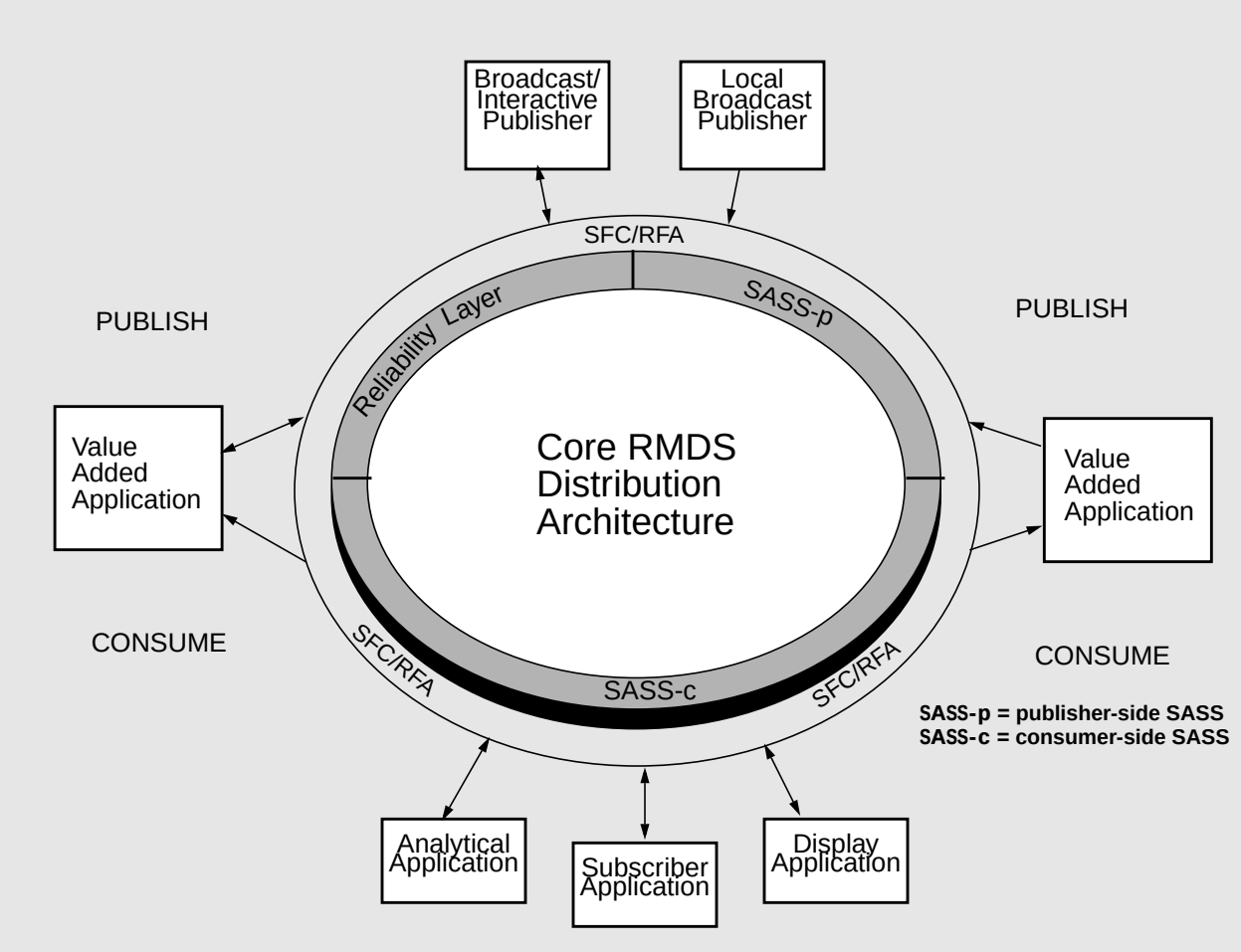

Figure 3: SFC/RFA-based Publisher/Subscriber Model

5.2.1 Contribution Model

A particular use of the publishing model is for contributions. The core RMDS distribution architecture supports contribution of data to external information vendors via the contribution applications/handlers using the semantics of Service and Instrument Name. Both the contributing application as well as contribution handler are SFC/RFA based applications. The contribution handler is required to respond with a positive or negative contribute acknowledgment which may also contain data. If the contribution service consists of multiple contribution handlers/server, the data packet is routed as follows, depending on a service-specific configuration setting:

- to the server indicating the lowest load
- round robin, to the next server
- to all servers and wait for acknowledgment from all
- to all servers and wait for acknowledgment from any

Contributions are anonymous; the contributing applications don't know which contribution server(s) will process the contributions, and the contribution servers are not aware of the identity of the contributing application. If the contribution handler needs to know the identity of the contributing application, an

application level protocol must be implemented on top of the contribution facility. Contribution data is not cached in the core RMDS distribution architecture. Contributions are subject to access control.

The comprehensive description of contribution facilities provided by the core RMDS distribution architecture is covered in a separate document.

6 Key Architectural Features

This chapter describes the key features of the core RMDS distribution architecture pertaining primarily to the following capabilities:

- Market data persistence via in-memory caching
- Datafeed load balancing
- Data quality assurance (DQA)--i.e. fault tolerance, source mirroring, failure detection, notification and auto-recovery
- Scalability
- Efficient and reliable communication

The next section provides an overview of the system functional layers and components, and summarizes important capabilities and restrictions of these layers. The subsequent sections provide a comprehensive description of all key capabilities.

6.1 Overview

Fast updating, time sensitive data has a unique set of requirements on a distribution system. To be able to satisfy these requirements the core RMDS distribution architecture relies on the following functional layers as illustrated on Figure 4:

- The Reliability Layer supporting both the interactive and broadcast publishers. The Market Data Hub is implemented based on the Reliability Layer. The layer is implemented by the RTIC, Source Distributor, Service Manager and the publisher-side of the SFC/RFA interface library.
- The publisher-side of the SFC/RFA/SASS-based Distribution Layer supporting broadcast publishers only. This layer is implemented by the RTIC and the producer-side of the SFC/RFA and SASS interface libraries. The layer is a lightweight publish facility but provides less services than the Reliability Layer.
- The consumer-side of the SFC/RFA/SASS Distribution Layer supporting market data distribution to the consumer applications. This layer is implemented by the RTIC and the consumer-side of the SFC/RFA.

The following sections list key capabilities of each layer.

6.1.1 Reliability Layer

Capabilities

- Use of RRMP protocol to provide full support for market data semantics including the source service concept. See section 7.4.1 for an overview of RRMP.
- Dynamic recognition of source server and source services.
- Optional data cache if redundant caches on the Distribution Layer and Market Data Hub are desired.
- Intelligent load balancing of broadcast and interactive datafeeds/publishers in order to share load across multiple datafeeds and to increase resiliency against datafeed failures.
- Interest-based access to data to conserve communication and CPU resources.
- Source mirroring for fault tolerant operation of publishers (interactive and broadcast).

- Use of RRCP transport protocol to support reliable broadcast communication for efficiency and scalability.
- Support for congestion avoidance to maximize performance and minimize probability of communication failures.
- Full DQA support.
- Support for permissioning of publications.
- Support for source application supplied DACS locks to enable content based permissioning.

6.1.2 Producer-side of the Distribution Layer

Capabilities

- Use of SASS protocol to support full market data semantics except for the source service concept. See section 7.3.1 for an overview of SASS.
- Scalability can be achieved by partitioning an instrument database across multiple publisher applications.
- Use of RV protocol to support reliable multicast/broadcast communication for efficiency and scalability. See section 7.3.2 for an overview of RV.
- Support for permissioning of publications.

Capabilities Not Supported (Only supported by the Reliability Layer)

- No support for interactive publishers.
- No load balancing.
- No support for fault tolerance of publishers.
- No support for DQA.
- No support for source application supplied DACS locks to enable content based permissioning.

6.1.3 Consumer-side of the Distribution Layer

Capabilities

- Use of SASS protocol to provide full support for market data semantics including the source service state notifications. See section 7.3.1 for an overview of SASS.
- Support for market data persistence via the RTIC cache.
- Information database can be partitioned across multiple RTICs to achieve scalability and performance.
- Support for interest-based data distribution—driven by the consumer subscriptions—to conserve communication and CPU resources.
- Support for fault tolerant RTIC operations.
- Use of RV transport protocol to support reliable multicast/broadcast communication for efficiency and scalability. Also, RV generates failure notifications (Slow Consumer and Data Loss) to support DQA. See section 7.3.2 for an overview of RV.
- Full DQA support.

- Support for access control.

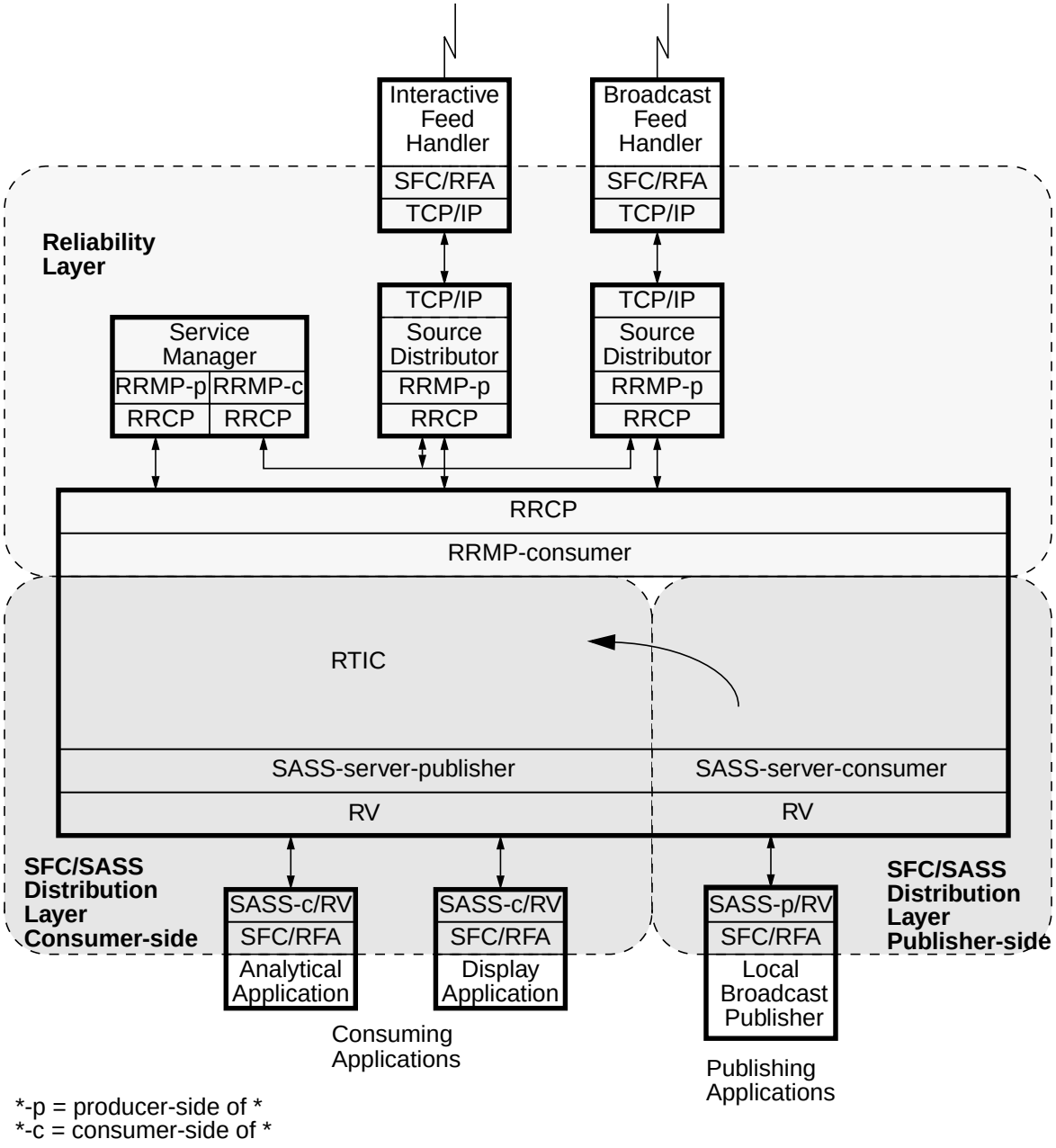

Figure 4: Software Layers and Process View of RMDS

6.2 Scalable High Performance Information Cache (RTIC)

The RTIC is at the heart of the core RMDS distribution architecture. The RTIC is a high-speed, in-memory data cache which protects publishing applications from the request traffic of subscriber applications, while accelerating response times for those subscribers. The RTIC is designed to be massively scalable and scales well regardless of the number of instruments or consumers it needs to support.

The information cache acts as a middleman between consumers and producers. It performs the following operations:

1. Listens to subscriptions from consumers.
2. Interacts with sources on the Market Data Hub to obtain market data.
3. Consumes market data from broadcast publishers on the Distribution Layer.
4. Stores a complete current value of each instrument published by the feed handler/data publisher. Subsequent updates from the datafeed are applied to the instrument data in cache.
5. Sends the complete up-to-date image of a instrument in response to the client subscription request. The response normally is sent point-to-point.
6. Broadcasts or multicasts updates—as they arrive from the feed handler—to the client network if there is at least one subscriber to the particular instrument.
7. Stops broadcasting updates when it determines that the instrument has no subscribers.

6.2.1 Data Partitioning

The RTIC data partitioning feature allows the RTIC environment to be configured so that a single information database can span multiple physical machines. Whether the RTIC database is supported by a single machine or multiple machines, the RTIC environment is completely transparent to the consumer and publisher applications. For the consumer applications, access to data is uniform through the use of subjects.

The RTIC processes provide a distributed RTIC database for the site and can be individually configured to optimize the trade-offs in cache size, update rate, and number of users.

The RTIC instrument universe can be partitioned by the service name, instrument name and exchange name into any number of unique non-overlapping instrument sets and assigned to one or more RTIC machines. Regular expressions are used to specify instrument names to allow even greater granularity of partitioning.

Figure 5 shows a partitioned RTIC environment. RTIC-1 and RTIC-2 process market data from the RDF source server, and RTIC-3 and RTIC-4 process market data from the ISFS source server.

RTIC-1 processes market data and subscriptions for items that come from exchanges A, B, C, and RTIC-2 processes market data and subscriptions for items that come from exchanges D, E, F.

RTIC-3 processes market data and subscriptions for items from any exchange and symbols that begin with A-G, and RTIC-4 processes market data and subscriptions for items from any exchange and symbols that begin with H-Z.

Note that all RTICs ignore subscription requests for instruments they are not configured to support. Subscription requests are broadcast or multicast and received by all RTICs. Only the designated RTIC will process the request.

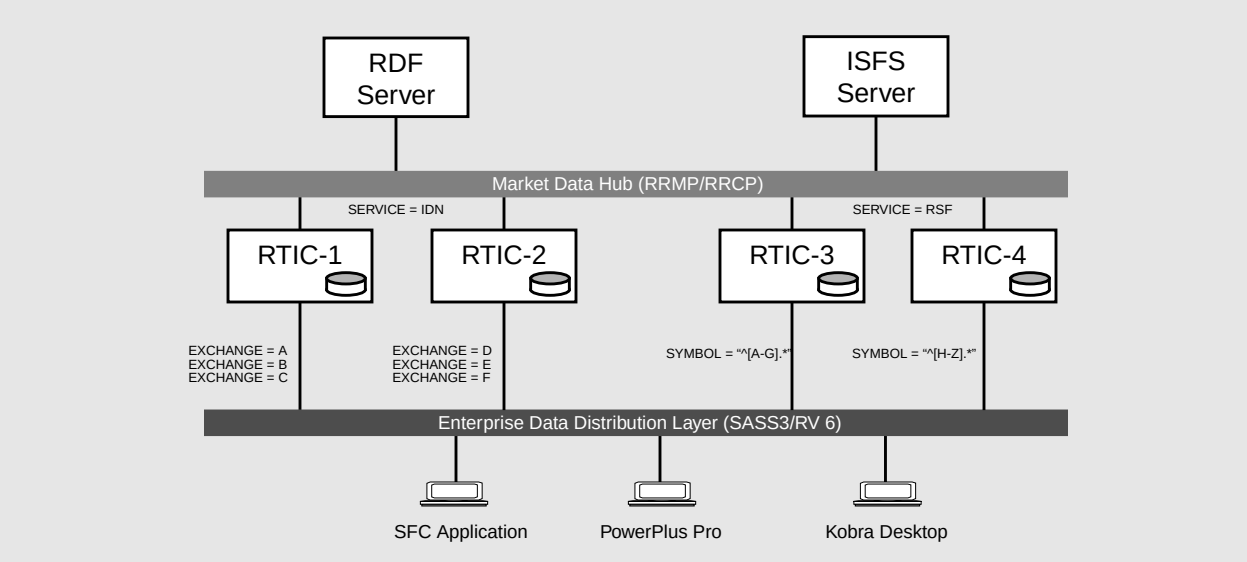

Figure 5: Partitioned RTIC Database Environment

6.3 Flexible Load Balancing

The Market Data Hub provides two load balancing strategies—distributed and centralized—for managing interactive and broadcast services that employ multiple sources/datafeeds. The distributed strategy allocates service management responsibilities to each Market Data Hub consumer—i.e. each RTIC. All RTICs make their service management decisions independent of each other. The centralized strategy assigns the service management responsibilities to a Service Manager which offers the advantages of centralized request routing.

The Market Data Hub can load balance servers that support the same set of instruments as well as servers with disjoined or overlapping sets of instruments.

Source servers can be dynamically added to or removed from a given service.

Load balancing utilizes the following factors when making request routing decisions:

- The **request limit** specifies the maximum number of outstanding requests (for new items) to a server. The request limit is advertised dynamically by the source server. This factor allows the Market Data Hub to optimize the retrieval time of items from the datafeed by not over-committing too many requests to any one server. A source server that responds faster than other source servers will receive more requests. A source application may set the request limit to zero, if it cannot accept any more requests for new items.
- The **load level** specifies the current load on each source server provided. The load level is defined as a single number and has no intrinsic meaning to the Market Data Hub. However, all source servers within a service must use the same algorithm to compute their load so that relative comparisons across multiple instances of a source application have significance. The load level can be generated by a source application or by the Source Distributor on behalf of the application. The formula used by the Source Distributor to calculate the current load is as follows:

  load = (currentWatchlistSize / maxCacheSize) * maxLoadLevel

  where the maxLoadLevel is configurable.The Source Distributor generates the load information periodically. The application is free to generate the load information whenever it chooses.

- To minimize the number of instrument preemptions, the **load level threshold for concurrent request** (LLTFCR) divides the source servers into two separate groups for load balancing purposes. The source servers are ordered within each group according to their current load level. For maximum performance, fresh requests are spread across all source servers below the LLTFCR. The objective here is to optimize item retrieval time and not worry about exact load balancing while the source servers are not yet close to being fully loaded. The least loaded source servers are still preferred over the more loaded servers when the limited number of instruments is pending retrieval. When all source servers have their current load level above LLTFCR, all requests are initially sent to the least loaded server. Only if the initial request fails, is it sent to the next least loaded server and so on.

6.4 Dynamic Discovery of Source Services

Source service availability and the state of each service are determined dynamically on the Market Data Hub. Consumer applications can query the core RMDS distribution architecture for the list of available source services and automatically receive updates (Service Up or Service Down) whenever the state of the service changes or new services are created.

6.5 Source Failure Notification

As datafeed capacity becomes increasingly larger and datafeed delivery mechanisms more complex, a market data infrastructure must be able to affect the state of a subset of the total number of active instruments in an efficient manner. The core RMDS distribution architecture assigns instruments to groups and reports their state via the Group State messages. Using the Group State, the core RMDS distribution architecture can change the state of any number of instruments with a single message. By default, the core RMDS distribution architecture automatically assigns a group ID to each server in a service. In the event that one server in the service fails, only a single Group State message is needed to affect the state of all instruments provided by that server. The SFC/RFA will fan out Group State messages and generate one Item State event for each instrument of the group.

6.6 Single Open

The basic idea behind the “single open” feature is that when a consumer application subscribes to an instrument, the instrument remains open until:

- the consumer application indicates that it’s no longer interested in the instrument, or
- a source application or the core RMDS distribution architecture drops—or closes—the instrument with a non-recoverable state indication.

The core RMDS distribution architecture, including the SFC/RFA, does everything possible to provide data for a valid instrument. It attempts to obtain data for an instrument for all recoverable conditions, including component failures and recoverable state events supplied by source applications. The feature is provided to free application developers from having to implement item re-request strategies, and to utilize system resources more efficiently by relying on the core RMDS distribution architecture to do the recovery.

6.7 Data Quality Assurance

The core RMDS distribution architecture provides for prompt detection of infrastructure component and communication failures, efficient failure notification, and high resiliency to various failures to assure data quality at all times.

In addition, the core RMDS distribution architecture offers various strategies for failure avoidance as described in section 6.11.

NOTE: There is no DQA support for source applications residing on the Distribution Layer—i.e. publishing via the SASS layer. Consumer application are not notified when a source application on the Distribution Layer fails.

6.7.1 Failure Detection and Notification

The core RMDS distribution architecture utilizes the heartbeat mechanism at various levels to detect component failures in a timely fashion and provide notification to downstream applications. The heartbeat mechanism is utilized to detect failures between the following components:

- Source Distributor and source applications on the Market Data Hub
- Source Distributors, RTICs and Service Manager
- RTICs and consumer applications
- Source Distributors in fault tolerant configuration
- RTICs in fault tolerant configuration

The core RMDS distribution architecture relies on RV and RRCP protocol implementations for notifications when data messages are lost and can't be recovered (Data Loss error), and when a consumer application is slow and causes inbound buffers to overflow (Slow Consumer error). These two errors indicate permanent loss of information and are considered fatal errors.

6.7.2 Data Failure and Recovery

An instrument or a group of instruments may fail due to preemption or to datafeed, source application, Source Distributor or RTIC failure. When this happens, the consumers are promptly notified that the instrument or the group of instruments are STALE via a Status message or a Group State message to speed up the notification and recovery.

The precise behavior depends on the cause of the instrument failure and is as follows.

When an RTIC (i.e. both the primary and secondary RTIC) fails and is later restored, the SFC/RFA layer tries to re-subscribe and re-establish the interrupted instruments (see also section 6.7.5 for description of RTIC failure and recovery scenarios). Consumer applications do not get involved in the recovery. Failed instruments are recovered at a controlled/configurable rate via new images.

When instruments fail due to preemption or to datafeed, source application or Source Distributor failure, the RTIC or the Service Manager tries to re-request the failed instruments from the remaining source servers of a given service. Consumer applications do not get involved in the recovery. Failed instruments are recovered at a controlled/configurable rate via new images.

If not all failed instruments can be recovered due to limited capacity of the source service, they are placed on the recovery queue and recovered by the infrastructure later.

6.7.3 Communication Failure and Recovery

Distribution of messages via RV or RRCP messaging layers is highly reliable but not guaranteed in order to achieve efficiency and scalability. This efficiency and scalability is required to distribute market data in real-time to a very large number of clients. These protocol are so called "optimistic" protocols utilizing "negative acknowledgments" (basically a retransmission request). Because of their optimistic nature, they are well suited to low-loss, high-bandwidth network environments such as LANs.

Communication failure and recovery on the Distribution Layer

The RV layer will notify the SFC/RFA/SASS layer when market data messages are lost and can't be recovered, and when an application is slow to process inbound messages. The SFC/RFA/SASS layer promptly notifies the application that all instruments are STALE.

Then the SFC/RFA layer tries to re-subscribe and re-establish the interrupted instruments/data streams. The application does not get involved in the recovery. Failed instruments are recovered at a controlled/ configurable rate via new images.

NOTE: The automatic failure recovery feature is configurable in the SFC/RFA. The automatic recovery should only be enabled on a properly sized and clean network.

Communication failure and recovery on the Market Data Hub

The RRCP layer will notify the RTIC when market data messages are lost and can't be recovered, and when the RTIC is slow to process inbound messages. The RTIC promptly notifies the consumer applications that a subset of (typical case) or all instruments are STALE via a Group State message to speed up the notification and recovery.

Then the RTIC tries to re-subscribe and re-establish the interrupted instruments. The consumer applications do not get involved in the recovery. Failed instruments are recovered at a controlled/ configurable rate via new images.

The Market Data Hub implements a simple and effective strategy for congestion avoidance in order to minimize the probability of losing messages. The RRCP layer generates protocol error reports and makes them available to RTICs, Source Distributors and Service Managers. The reports contain various error rates—point-to-point, retransmission requests, etc.—and enable Hub components to reduce the non real-time activities, such as subscription re-requests, in order to reduce the network traffic.

NOTE: The automatic failure recovery feature is configurable in the RTIC. The automatic recovery should only be enabled on a properly sized and clean network.

6.7.4 RTIC Fault Tolerance

Multiple RTICs can't be load balanced as each RTIC supports a unique set of instruments to provide scalability; therefore, the RTICs are deployed in a fault tolerant configuration to provide resiliency.

The RTIC provides fault tolerance based on the concept of hot standby. Two RTIC instances configured as a fault-tolerant pair (primary and secondary RTIC) subscribe to the same instruments on the Market Data Hub and maintain identical data caches. However, only the primary RTIC distributes data to the consumers.

Should the primary RTIC fail, the secondary automatically detects the failure, due to missing heartbeats, and takes over the data distribution as the new primary with limited impact on the active data streams. Note that the secondary RTIC doesn't buffer data updates and doesn't replay updates to ensure integrity of the active data streams. However, the RTIC guarantees the integrity of data by re-broadcasting the up-to-date images for instruments which received an update within the last N seconds (configurable interval) before the switchover.

The configuration of an RTIC, i.e. whether or not hot-standby is configured, is transparent to the source applications, Source Distributors and consumer applications.

Should both the primary and secondary RTICs fail, the SFC/RFA/SASS library linked with a consumer application will detect loss of the RTIC's heartbeats and notify the application.

If the primary and secondary RTICs are segmented (due to network segmentation) or time out each other, they will run as two individual separate RTICs. When the network or timeout problem is resolved, both RTICs will notice each other and one of the RTICs will assume role of the secondary.

6.7.5 RTIC Recovery from Failures

The RTIC failure recovery operations differ depending on whether the partner RTIC is running.

RTIC recovery when partner RTIC is running

When the old primary RTIC is restored and the partner RTIC is running as the new primary, it will assume the role of the secondary. The new secondary will synchronize its cache with the primary by listening to the REFRESH message traffic generated by the consumer applications. The SFC/RFA/SASS library periodically broadcasts REFRESH messages for each instrument to which the consumer application is actively subscribing. The RTIC uses REFRESH messages to issue subscription requests to the Market Data Hub and synchronize its cache with the primary. The secondary RTIC does not distribute any market data to the consumer applications.

There is an option to re-populate the cache of the secondary RTIC upon startup from an already running primary RTIC. Cache recovery is most useful when caching data from broadcast publishers on the Distribution Layer. Cache recovery should not be enabled for sources on the Market Data Hub. Cache recovery doesn't cause the RTIC to re-subscribe to instruments on the Market Data Hub.

RTIC recovery when partner RTIC is not running

When the old primary RTIC is restored and the partner RTIC is not running, it will assume the role of the primary and it will gradually restore all interrupted instrument data streams using the recovery method described above. The only difference is that as the instruments are recovered, the RTIC broadcasts initial images and subsequent updates to consumers.

6.7.6 Source Mirroring

Where disruption of an update stream is unacceptable, mirroring of source servers should be used. The source mirroring capability provides seamless tick-by-tick recovery (no loss/duplication of updates) by the healthy secondary server in case of failure of the primary server.

Both the primary and secondary server receive all requests from RTICs or a Service Manager and subscribe to the same items on their datafeeds, but only the primary server actively responds to the requests and supplies updates, while the secondary is passive and simply attempts to remain synchronized with the primary.

The secondary server monitors all data stream messages from the primary server and attempts to synchronize its watchlist, cachelist and update data streams with the primary. Any differences are noted and recorded by the secondary until they can be resolved.

The configuration and operation of a pair of mirrored servers is transparent to both source and consumer applications as well as to RTIC and Service Manager. They see and handle a pair of mirrored servers as a single logical server. Multiple pairs of mirrored servers can be deployed and load balanced within the same source service.

Should the primary server fail, the secondary takes over the synchronized update stream without loss of updates. The standby uses the recorded differences between the primary and secondary to resolve any data stream differences between itself, the RTICs and Service Manager.

Should both the primary and secondary source servers fail, consumers of the affected data streams are notified and the RTICs or Service Manager will be responsible for the recovery.

If the primary and secondary servers are segmented (due to network segmentation) or time out each other, they will run as two separate load-balanced source servers. In that case, manual intervention is required to restore fault tolerant operations.

Source mirroring works on a per-server basis in the Source Distributor, therefore different source applications that are each in a source mirroring configuration can mount to the same Source Distributor. Each server, though, should be run on a unique machine.

Application requirements

Any type of source application (i.e., supplying ANSI pages, Marketfeed records, or generic data items) can be configured to operate in the fault tolerant configuration, assuming the following conditions are met:

1. The secondary Source Distributor's ability to seamlessly take over the data streams is dependent on close timing of update delivery to each Source Distributor by each source application. This implies that the performance characteristics of the hardware and datafeeds utilized by each source application should be closely matched to minimize delays in the update stream. If the time difference between update streams from two source applications is larger than configurable maximum temporal time difference, then it may not be possible for the secondary server to synchronize itself with the primary server's data streams (i.e., tick-by-tick recovery may not be possible). However, the integrity of data is guaranteed even if two data streams can't be matched; the subset of items that received an update within the last N seconds (configurable interval) will be re-synchronized via new images.
2. Each datafeed/source application must deliver the same update stream for each item open on the datafeed; i.e., the order of update delivery and the update's data content must be the same for a given item. Both the data order and the data content are used for update data stream synchronization. Typically this requirement is met automatically due to the fact that both datafeeds are provided by the same vendor. However, if each datafeed originates at a different physical location—typical case to assure true redundancy—then the data streams may not be the same on both datafeeds all the time due to communication errors, data aggregation because of temporary lack of communication resources, etc. These differences will result in a subset of items temporarily out of sync until problems clear.
3. Requirement 2 above is relaxed for source applications providing Marketfeed data. Only a portion of the update message, as delimited by the first Record Separator character in the message and the last character of the message, is used during the matching. The header portion of the message (preceding the first Record Separator) is not used. **Note**: The data element of an update message contains a full Marketfeed update (i.e., it consists of the framing characters, the header, and the list of <FID number, FID data> pairs). The beginning of the field list is indicated by the first Record Separator character in the update data.
4. A source application must disconnect from the core RMDS distribution architecture upon datafeed failure and re-connect when the datafeed is restored.

6.7.7 Source Server Recovery from Failures

The source server failure recovery operations differ depending whether the partner source server is running or not.

Source server recovery when partner server is running

When the old primary source is restored and the partner source is running as the new primary, it will assume the role of the secondary. The new secondary will synchronize its cache and watchlist with the primary by requesting the download from the primary server. If the primary server fails before the synchronization process is complete, i.e. before the secondary server is fully operational, the secondary

server will restart itself by broadcasting Goodbye message to the network. Consumers will be notified about the failure and the RTICs or Service Manager will initiate recovery of the failed instruments.

Source server recovery when partner server is not running

When the old active server is restored and the partner server is not running, it will assume the role of the active and it will be load balanced.

6.8 Unified Item Priority, Preemption, and Item Locking

The Market Data Hub uses priority to effectively integrate and manage item requests, preemption, item locking, removal of unwatched items, and item recovery. The Market Data Hub defines priority as a tuple <class, count>, where class is a number from 0 to 10 representing the relative importance of the item, and count corresponds to the number of users/subscribers. The RTIC assigns a default priority class of 1 to all item requests made to the market data sources. The system administrator has the ability to assign a different priority class to each item.

Preemption is the mechanism used by the Market Data Hub to ration finite cache resources when user demand exceeds the number of available cache slots. For example, if a server's cache holds only 20 items, and there is a 21st request, the "least important" item (an item whose priority is less than or equal to that of the requested item) is identified by the infrastructure and deleted from the cache to create a slot for the new request. If the new request's priority is lower than the lowest class of any item already in cache, the request is placed in the recovery queue and is retried periodically based on a configurable time interval. The system administrator can exercise control over cache contents, preemption, and locked items to achieve a balance between speed of service and network traffic.

Certain items may be so important that they should always be in cache, effectively "locked" in cache and available for any user. The administrator is able to accomplish this by defining a priority class for an item on a particular service. The item priority class, if defined, overrides the default class value. The priority class of an item request and a number of configurable thresholds may be used together to bias system operation to achieve specific performance or efficiency objectives.

Three configuration variables are available to define thresholds for item locking, preempted item recovery, and removal of unwatched items.

- Items with a priority class equal to or greater than the **locked item threshold** are not eligible for preemption, even if they are not currently watched. The locked item threshold is used by the Source Distributor and the Service Manager when making preemption decisions.
- If an item is preempted and its priority class is equal to or greater than the **preemption recovery threshold**, then the Market Data Hub will automatically try to recover that item. Such items are placed in the preempted item recovery queue.
- The **remove unwatched item threshold** defines the priority class level where all unwatched items with lower priority are automatically deleted from the RTIC and/or Source Distributor cache to conserve datafeed bandwidth.

6.9 Source Distributor

The Source Distributor encapsulates datafeed independent functions such as source mirroring, load balancing, preemption, support for interactive and broadcast feeds, etc. in order to simplify development of feed handlers. Optionally, the Source Distributor is also capable of maintaining a cache of items so that new requests for currently serviced items can be satisfied without interaction with the source application. In this case the Source Distributor automatically applies any updates written by the application to the cache.

The Source Distributor can simultaneously support a configurable number of independent source applications for the same or different services. Source applications use the SFC/RFA Library to communicate with the Source Distributor infrastructure component.

Source applications typically run on the same node as the Source Distributor but they can also run on a remote node if performance requirements warrant the added expense.

6.10 Service Manager

The RTIC supports a fully distributed request routing algorithm. Each RTIC is responsible for routing item requests to one or more source servers after a poll of all server caches fails to locate the item. Although precautions are taken, a small amount of cache duplication is unavoidable due to the distributed nature of the decision making process. When a source server fails, each RTIC detects the failure, sends a Group STALE state indication to all consumer applications to affect the state of items supplied by the failed server, and then re-requests all failed items from the remaining servers using the normal request routing algorithm. The re-request activity generates some extra traffic on the network after a server failure.

To address the limitations of distributed request routing, a Service Manager can be deployed that offers the advantages of centralized request routing. Since the Service Manager is a separate process, it is not burdened by the update or image traffic from all source servers supporting a service. Being the central decision maker for the service, it provides:

- Efficient service wide request routing without cache duplication
- Priority based “source led” recovery
- Service wide preemption
- Stale item recovery

To address concerns about the Service Manager becoming a single point of failure, the RTIC has its own distributed request routing algorithm and the Source Distributor implements the local preemption algorithm. These algorithms are used if the Service Manager is not running or is detected as having failed.

An important feature of the Service Manager is that it can be stopped and started without disruption to established data streams and normal system operation. When the Service Manager is stopped, or if it fails, the RTICs and Source Distributors automatically take over their assigned functions. When restarted, it automatically downloads the necessary information from Source Distributors to restore full information resource management without human intervention.

A market data service will typically employ a Service Manager when multiple source servers exist for the service; however, a Service Manager is not mandatory in this case. If the Market Data Hub service consists of a single source server, the benefits provided by the Service Manager are reduced and, therefore, it is unlikely that a Service Manager will be used in this case.

When there are multiple market data services on the network, a Service Manager may be provided for some services but not for others.

6.10.1 Load Balancing

The Service Manager’s request routing algorithm can be configured to achieve system objectives of resiliency and efficiency. The Service Manager either knows where the item is cached or simply forwards the request to the “best” server. By contrast, the default distributed routing algorithm implemented in the RTIC is of linear time order complexity. The RTIC has a limited cache list, therefore it polls all servers to determine if a requested item is cached.

Efficient routing is performed via two queues, the High Priority Queue (HPQ) and the Low Priority Queue (LPQ). New requests are inserted into the HPQ in network priority order; therefore, requests with high priority are processed before low priority requests. After a source server failure, all items from the failed server are inserted into the LPQ, in priority order.

6.10.2 Source Led Recovery

When a source server fails, each RTIC detects the failure, sends a Group STALE state indication to all consumer applications to affect the state of items supplied by the failed server and then waits for the Service Manager to restore the interrupted data streams. Service Manager led recovery proceeds much more quickly and efficiently because high priority items are recovered first. Images are broadcast to all RTICs, refreshing all users of an item simultaneously, without the re-request traffic that would otherwise be generated by the RTICs. The recovery of failed items, assuming server capacity exists, completes in constant time order complexity, as a function of the number of RTICs.

6.10.3 Global Preemption

The Service Manager maintains a complete cachelist and watchlist for all source servers supporting the service. Therefore, it is capable of identifying and preempting the least important item across all servers. Preempted items are held in the preempted item retry queue, and the Service Manager attempts to recover them periodically.

The global preemption request routing can be enabled instead of the load balancing. Global preemption is applicable to the interactive services and only when each source server is capable of supplying the same set of items.

6.10.4 Request Flow Control

The Service Manager offers centralized service-wide request management to improve the average request response time for new items and provides ability to control the maximum per-service image retrieval rate.

6.10.5 Fault Tolerance

The Service Manager would appear to be a single point of failure, but the system is designed such that its failure is completely transparent to consumer applications. In the unlikely event of a Service Manager failure, the RTICs and Source Distributors take over responsibility for the functionality provided by the Service Manager. Since the Service Manager is involved only in the routing of requests, any outstanding requests that are lost are automatically recovered by the RTICs. After the Service Manager is restarted, a synchronization protocol is invoked in the background that downloads all information needed by the Service Manager to seamlessly resume its function.

6.10.6 Stale Item Recovery

The Service Manager implements a simple scheme to ensure that users will receive good data, even if both good data and stale data are available from a service with multiple source servers. If only stale data is available on the service, then the stale data (marked as STALE) will be delivered to the users.

If a good item becomes stale, then the Service Manager will periodically make an attempt to recover the stale item. The frequency of recovery attempts is configurable.

6.11 Traffic Management

The core RMDS distribution architecture provides two main strategies for maintaining data quality in the face of extreme message rates: traffic segmentation and traffic intervalization.

6.11.1 Traffic Segmentation

By design, the RMDS network may employ separate IP network segments for the Market Data Hub and Distribution Layer in order to separate the market data traffic from the client traffic and relieve infrastructure components of the burden of filtering out irrelevant traffic at the software level. RMDS provides the means for further segmentation of traffic on the Market Data Hub and Distribution Layer as described in the sections that follow.

Traffic segmentation on the Distribution Layer

The core RMDS distribution architecture is both very flexible and highly scalable, supporting the construction of RMDS networks spanning from one client node to thousands of nodes. It is sometimes beneficial and/or necessary to segregate client nodes and place them onto separate LAN segments as illustrated in Figure 6, or onto separate IP multicast domains as illustrated in Figure 7. Such configuration effectively creates a network consisting of multiple independent Distribution Layers and a single shared Market Data Hub. Organizations may consider such configuration due to the following benefits:

- ability to support practically unlimited number of user workstations
- segmentation of network traffic
- isolation of network problems on a particular segment so as not to affect other segments
- segregation of work groups
- segregation of low and high end-user workstations

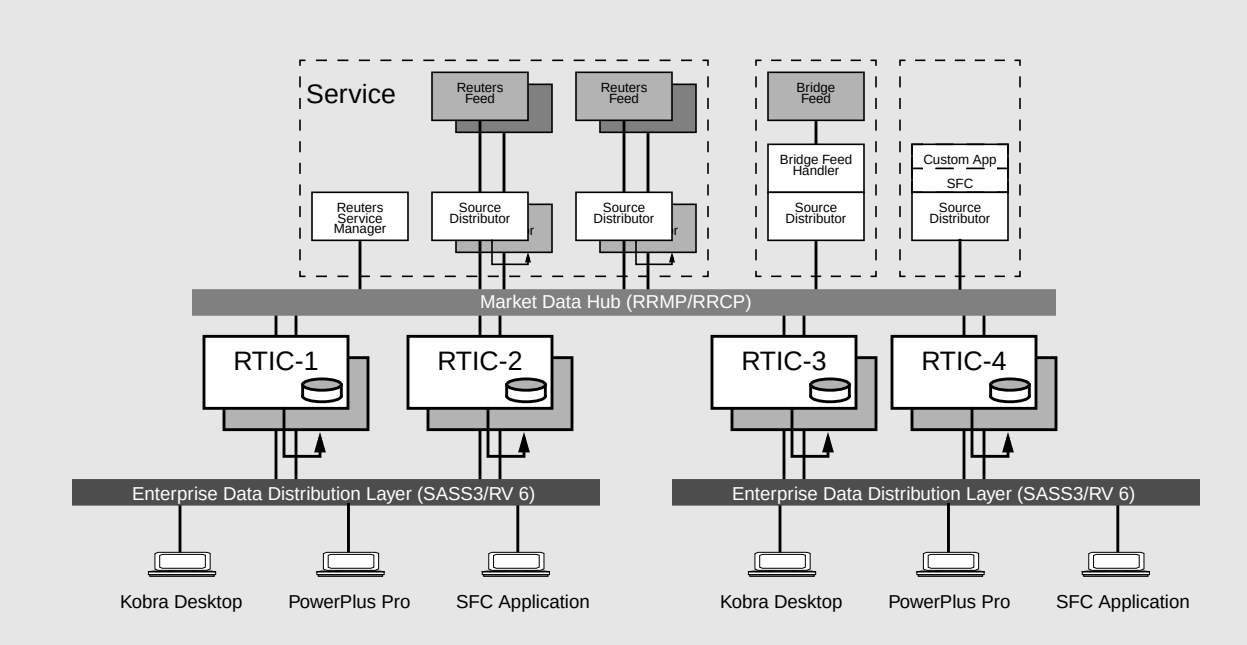

Figure 6: RMDS Segmented Network Environment

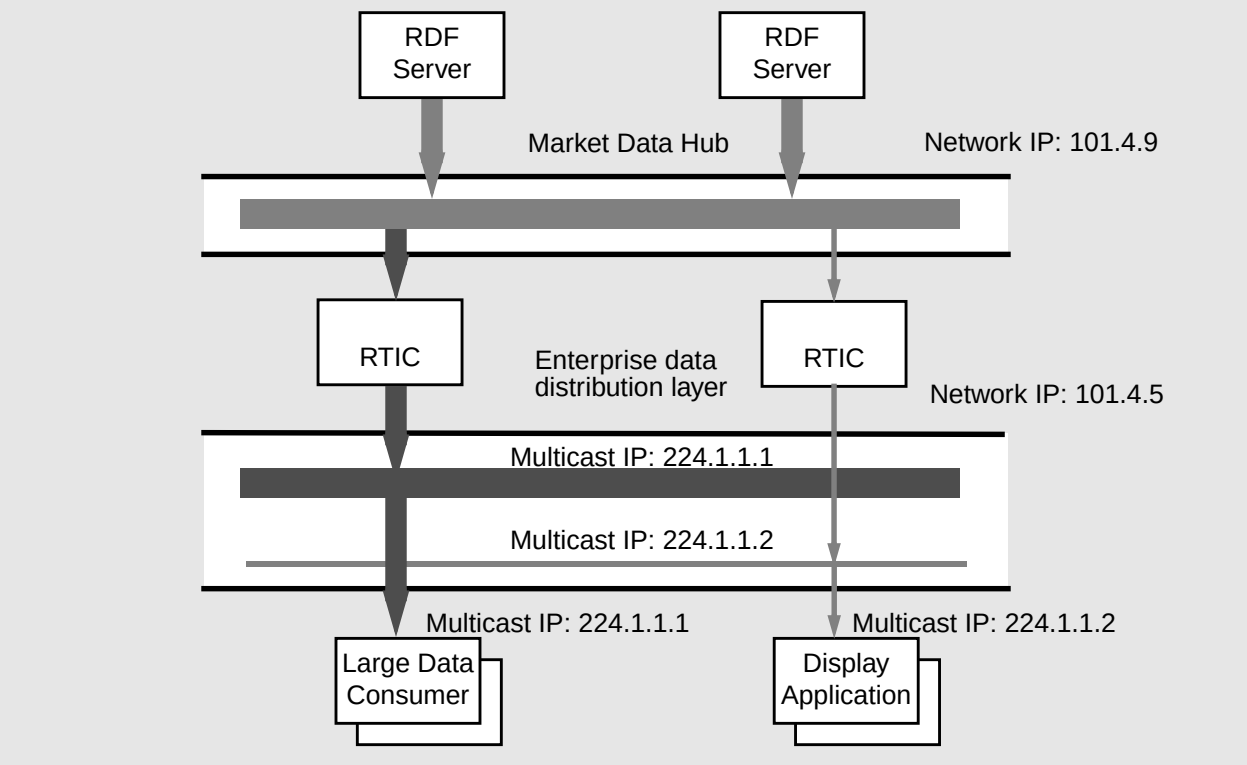

Figure 7: Using IP Multicast to Separate Slow and Fast Consumers on a Single IP network

Traffic segmentation on the Market Data Hub

The RTICs and the market data sources can be easily segregated and placed onto multiple physical or virtual LANs. This is possible since each RTIC is dedicated to provide a unique set of subjects, or instruments, and therefore it only needs access to a specific set of sources. All RTICs are still visible to all consumers on the Distribution Layer.

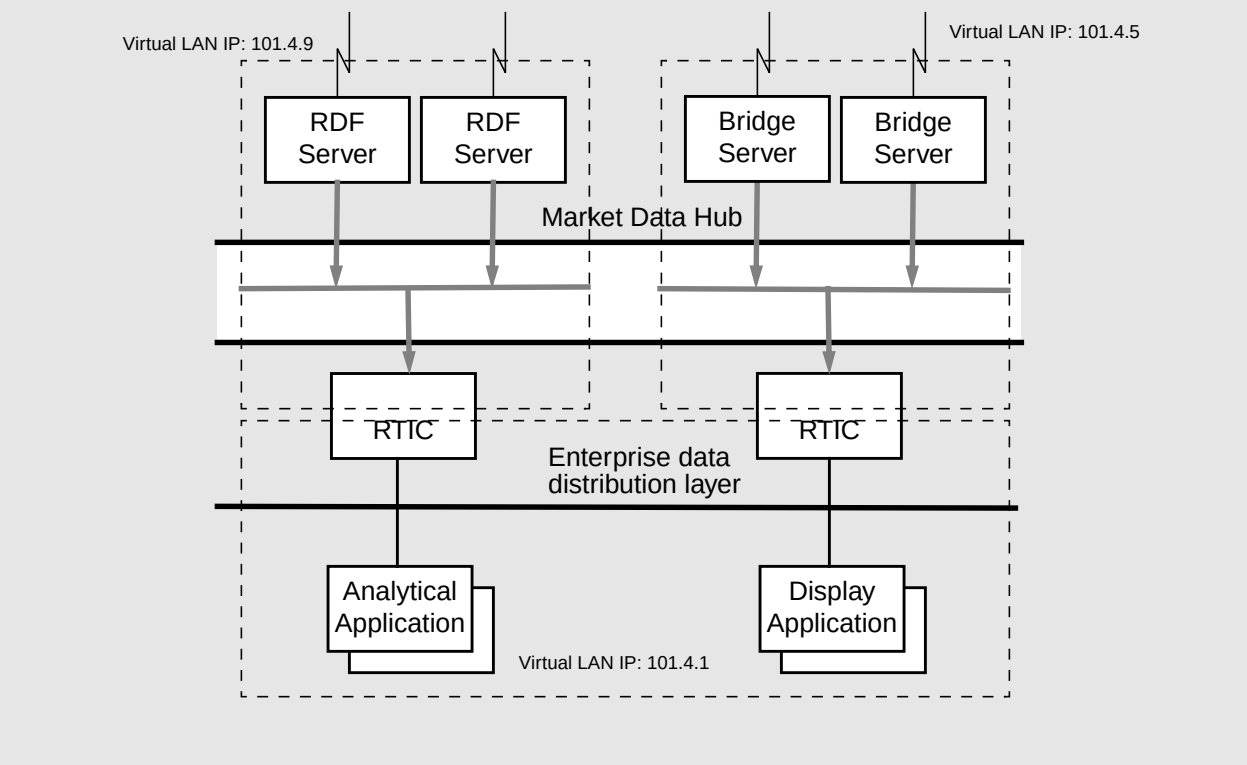

Figure 8: Using Virtual LANs to Segment Traffic on Market Data Hub

6.11.2 Traffic Intervalization

The core RMDS distribution architecture offers a mechanism to protect the low-end user workstations and slow user applications against current message rates and future increases in message rates such as those expected to follow from decimalization of U.S. exchanges.

The TIC Message Filter (TMF) component can be installed on the RV backbone to provide so called “intervalized” data to applications that do not require tick-by-tick data. The TMF is a client of the RTIC, subscribing to a real-time (i.e. tick-by-tick) data stream and sending out updates which contain the current value of any field that changed during the configured interval. The RTIC and each TMF are sending updates on separate IP multicast addresses. Multiple TMFs can be configured, each applying different intervalization. Depending on the requirements, an application can be configured to receive the real-time data stream or a particular intervalized data stream. The initial values of data are supplied by the associated RTIC. The capability is completely transparent to the client applications.

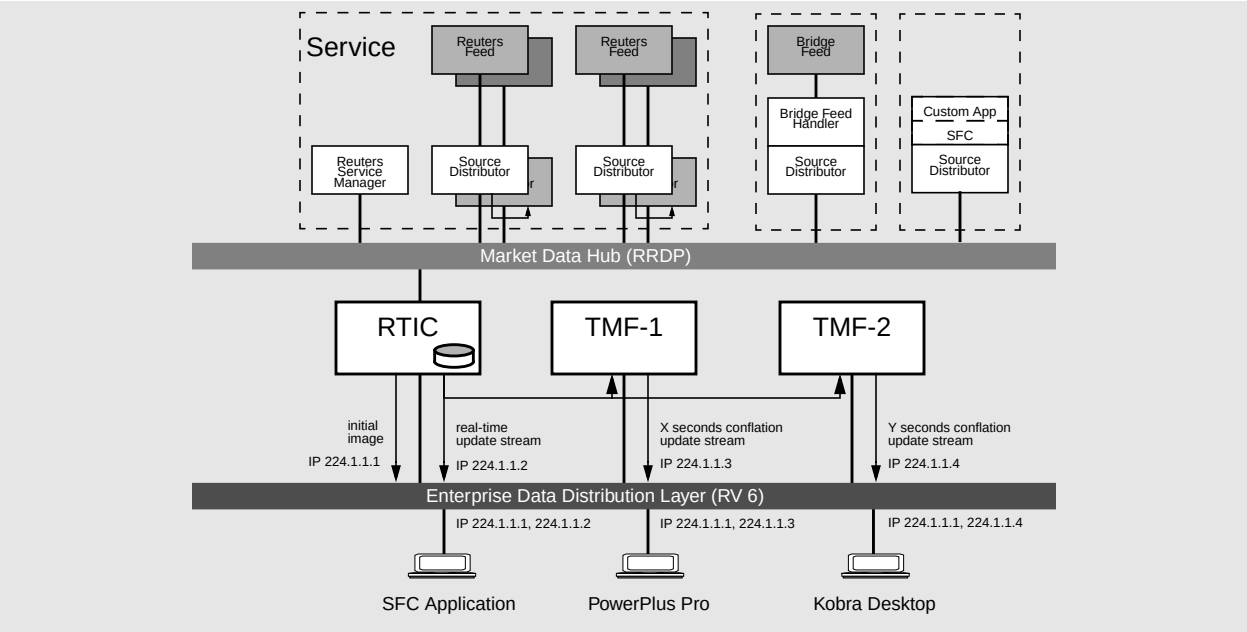

Figure 9: Using TMF on RV Backbone to Create Intervalized Data Streams

The market data traffic can also be intervalized at the Market Data Hub layer to protect the Market Data Hub, the RTICs and the user applications against extreme update rates. The Filtering Distributor can be deployed for the selected services on the Market Data Hub as illustrated in Figure 10.

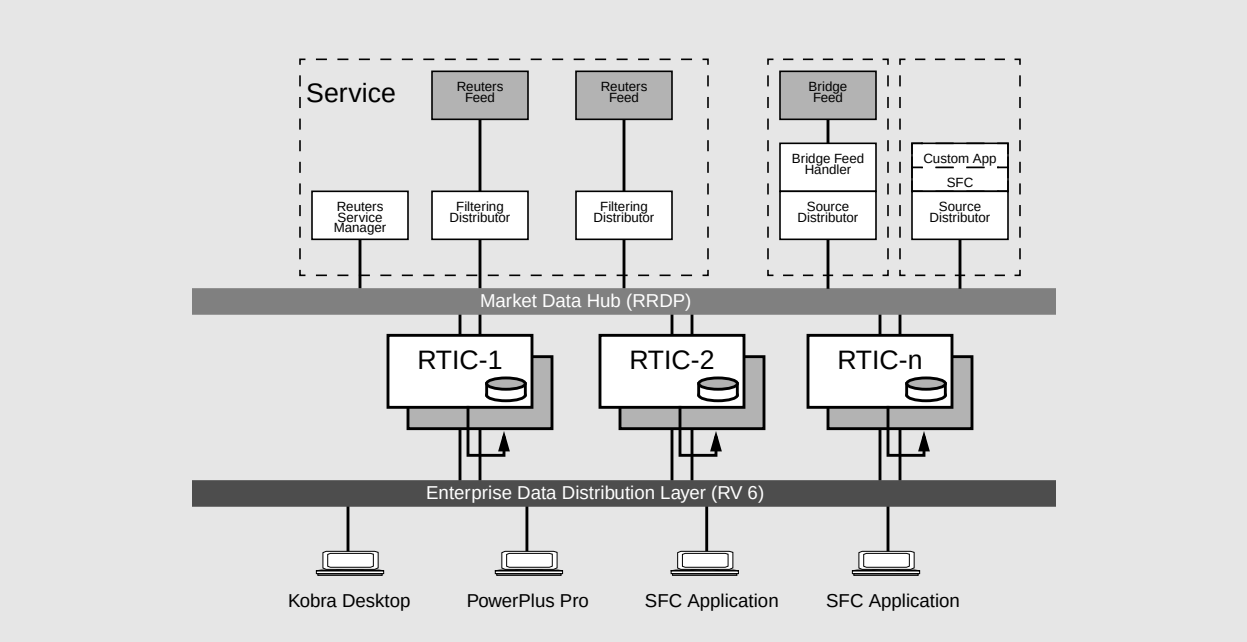

Figure 10: Using Filtering Distributor on RRDP Backbone to Create Intervalized Data Streams

7 System Operation

The core RMDS distribution architecture is made up of several processes which interoperate. The optional Service Manager process optimizes the system operation. Therefore, the topics of request routing, item preemption, and source failure recovery with respect to system operation are treated in the next sections both with and without the presence of the Service Manager.

7.1 System without Service Manager

The core RMDS distribution architecture for a source service that does not employ a Service Manager is illustrated in Figure 11. All RTIC requests are sent directly to the source servers. All source server responses and unsolicited messages are sent directly to the RTICs. The RTICs are responsible for caching instruments and providing initial values, routing requests and recovering failed items, while the source servers are responsible for item preemption.

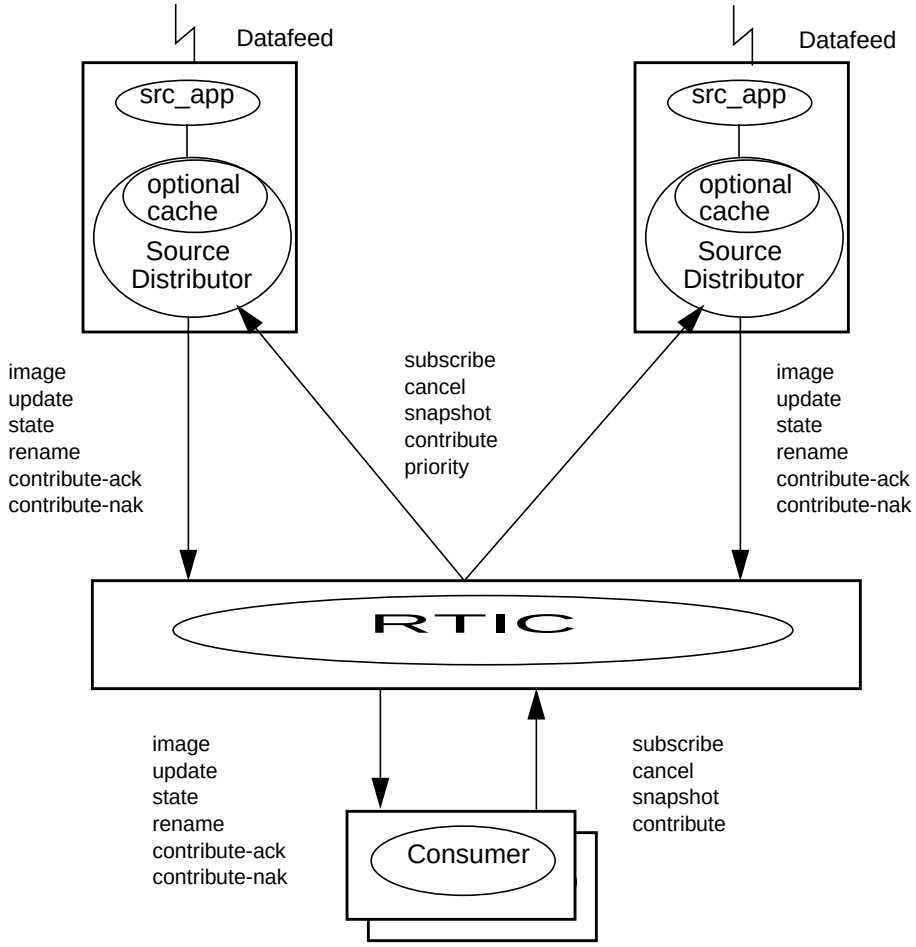

Figure 11: RMDS Architecture and Data Flows without Service Manager

7.1.1 Subscription Request Handling

The RTIC listens for subscription requests from consumers. It ignores requests for instruments for which it has not been configured to listen to.

If the RTIC database can supply the requested instrument, the RTIC replies with an image to the consumer in a point-to-point message over TIB/RV. Subsequent updates for that instrument are broadcast to the consumers as they are received.

If the requested instrument is not in the RTIC database, the RTIC makes an attempt to obtain the instrument from the source service on the Market Data Hub regardless of the datafeed type—interactive or broadcast. Note that all source services on the Market Data Hub appear as interactive to the RTIC. The next section explains the request routing strategy the RTIC implements to process subscription requests.

Subscription request routing

Request routing is performed by each RTIC independently of other RTICs on the network. The goal of RTIC request routing is to balance the item retrieval response time with the efficient use of limited and expensive datafeed resources.

The following is a high level description of the request routing operations:

1. A request from a consumer application for a instrument which is not cached on the RTIC causes the RTIC to poll all source servers in the service one by one. If polling fails to locate the instrument, step 2 is executed.
2. The RTIC selects a source server using the factors and strategy outlined in section 6.3 and forwards the request to that server.
3. If the selected server responds with an image, the RTIC caches the received data and broadcasts the image to all consumers. The subscription request handling is complete at this point.
4. If the selected server rejects the requests with a recoverable status, the RTIC applies step 2 to the remaining source servers.
5. If all source servers reject the request with a recoverable status, the request is placed on the recovery queue and recovered by the RTIC later.
6. If the request for an instrument is rejected by the selected source server with non-recoverable status, the RTIC broadcasts the status to all consumers and drops the instrument. If the Non-Recoverable-Polling feature is enabled for a particular source service, the instrument is dropped only when all source servers reject the request with non-recoverable status. The ability to search all source servers within a source service regardless of recoverable or non-recoverable rejections is useful when load balancing source servers supplying heterogeneous datasets, i.e. non-identical datasets.

The precise operation of the routing algorithm is a function of the values supplied to a number of configuration variables.

7.1.2 Preemption

If there is no spare capacity to hold a new item, then an existing item must be removed, or preempted, from the cache.

If preemption is enabled in the Market Data Hub for a particular market data service, each Source Distributor performs preemption. Preemption is limited in scope to a single source server.

Preemption applies only to interactive source services. By default, preemption is enabled in the Market Data Hub. It may be disabled on a per-service basis via a service-specific configuration parameter. If the

preemption capability is disabled in the infrastructure, a source application must implement preemption based upon a source-specific algorithm.

Preemption occurs in the Source Distributor when a new item is requested and the cachelist of the server is full. One of the cached items, the “best candidate” within a particular source server, is identified and removed from the cache to make room for the new item. The preempted item is placed on the RTIC’s preemption recovery queue.

The preemption algorithm is based solely on the network priority of the requested item relative to the priority of all watched items on a particular source server.

If the cache of a source server is full but the watchlist is not, one of the unwatched items will be removed from the cache to make room for the new item. Removal of an unwatched item is not considered as preemption.

7.1.3 Request Flow Control

The RTIC provides flow control of Subscribe requests between itself and the Source Distributor. The RTIC monitors the number of outstanding requests to each Source Distributor, and if the number of outstanding requests is at the limit, the RTIC suspends sending item requests to that Source Distributor until it receives one or more responses.

The Source Distributor provides flow control of Subscribe requests for new items between itself and the source applications. The Source Distributor monitors the number of outstanding requests to each source application it supports, and if the number of outstanding requests is at the limit, the Source Distributor suspends sending requests for new items to that application. If the source application sets its “new item request limit” size to zero, it must be prepared to process and reject all requests sent to it by the Source Distributor.

The RTIC does not provide dynamic flow control for close and priority requests. However, the RTIC does allow a maximum message rate to be defined for close and priority requests to allow source servers to keep up.

Close requests are generated when a consumer is no longer interested in a particular item, and priority requests are used to raise the priority of outstanding requests.

7.1.4 Request/Item Failure and Recovery

When the RTIC detects that a watched item has been closed on a source server with a recoverable state indication, such as in the case of preemption, it tries to re-request the failed item from the remaining source servers. Similarly, if the RTIC detects a source server failure, it tries to re-request the failed items from the remaining source servers. Consumer applications do not get involved in item recovery, thus minimizing the network traffic. However, consumer applications are notified that the data streams have been interrupted, via a STALE or Group STALE state indication, and that the system is trying to recover the failed items. Failed items are recovered at a controlled/configurable rate. Any failed items that cannot be recovered due to limited capacity of the market data service are placed on the recovery queue to be recovered by the RTIC later.

When an RTIC receives a consumer request for an item and the request cannot be satisfied immediately, it is placed on the recovery queue. The requesting application is notified that the request could not be satisfied and that the system will try to re-request the item later.

The RTIC does not get involved in the recovery of a pair of mirrored servers when the primary source server fails. The secondary server is responsible for the recovery in this case. However, if both primary and secondary source servers fail, the RTIC tries to recover the failed items (using the strategy described above) from the remaining servers.

7.1.5 System Summary without Service Manager

- Preemption is limited in scope to an individual source server and therefore service-wide preemption is not provided.
- Since all RTICs act independently, RTIC re-requests for failed instruments are not synchronized and therefore a source server may generate multiple image responses to recover a single instrument (assuming the instrument is watched by multiple RTICs).
- Failover/recovery performed by the RTIC is not based on item priority.
- Since all RTICs make their decisions independently, load balancing under heavy request traffic is less effective than when performed by the Service Manager.
- The RTIC maintains limited cachelists for the market data service(s) so it always employs polling to retrieve instruments that are not currently being watched on the RTIC, and that can add to the network traffic.
- Under certain race conditions, duplicate data streams and cache duplication for interactive type source servers may be created on the network.

7.2 System with Service Manager

The core RMDS distribution architecture for a market data service that employs a Service Manager is illustrated in Figure 12.

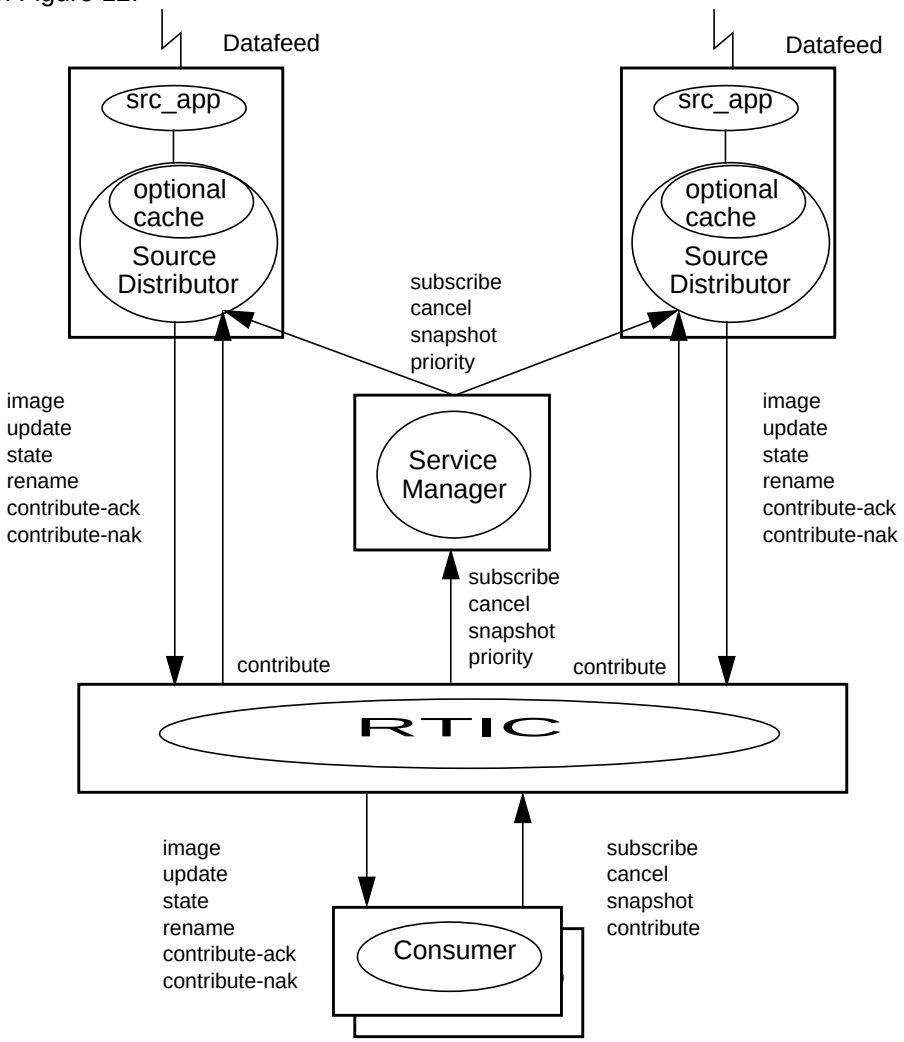

Figure 12: RMDS Architecture and Data Flows with Service Manager

One Service Manager utilizes knowledge of all servers in a service to provide optimized request routing o the Market Data Hub.

For each source server within a service, the Service Manager builds and maintains an item watchlist and a cachelist using RTIC requests and source responses. It utilizes the cachelist and watchlist to open and close data streams on the market data service, to efficiently locate cached items, and to provide effective request routing, item preemption, and failover.

All RTIC-originated Subscribe, Priority and Cancel requests are routed to an appropriate source server through the Service Manager associated with the target service. **Note**: Images are sent point-to-point when only a single RTIC subscribes to an instrument, otherwise they are sent via broadcast.

The Service Manager decides which of the source servers should satisfy the request and routes the request to that source server. The selected source server broadcasts the resulting data stream (image,

updates and state) directly to the RTICs and to the Service Manager; the data stream is not routed through the Service Manager on its way to the RTICs.

If global preemption is enabled and the source service is fully utilized, the Service Manager has the ability to preempt an existing cached item to satisfy the request. If a Service Manager is operational for a service, the preemption function in the Source Distributor is effectively disabled for that particular service.

7.2.1 Subscription Requests Handling

The RTIC handling of the subscription requests for items in the RTIC cache is the same regardless of whether the Service Manager is configured or not (see section 7.1.1 for details).

If the requested instrument is not in the RTIC database, the RTIC forwards the subscription request to the Service Manager to obtain the instrument from the source service on the Market Data Hub. The Service Manager provides two separate request routing strategies—load balancing and global preemption. The next section explains how the Service Manager selects a source server to satisfy the request.

Subscription request routing—load balancing

Load balancing attempts to ensure the best possible request response time and use of limited and expensive datafeed and system resources when multiple source servers are available for a service. Load balancing is most effective when all servers are capable of providing the same set of items.

The load balancing algorithm with and without the Service Manager is similar (see section 7.1.1) except for the following important difference:

> For each source server within a service, the Service Manager builds and maintains an item watchlist and a cachelist using RTIC requests and source responses. Once the Service Manager determines the location of the item (i.e. the item is in the Service Manager's cachelist), the request is forwarded to the respective source server without having to, potentially, search all source servers for the item.

Subscription request routing—global preemption

Global preemption applies only to interactive source services and can be disabled/enabled on a per-service basis via a service-specific configuration parameter.

The global preemption algorithm is based solely on the network priority (i.e. class and count) of the requested item relative to the priority of all watched items across all source servers of the service.

The following describes the request routing operations:

1. If the Service Manager knows the location of the item (i.e. the item is in the Service Manager's cachelist), the request is forwarded to the respective source server without having to, potentially, search all source servers for the item.
2. If the Service Manager does not have the item in the service cachelist, it considers the following factors in choosing a server to service the request:
    a. Cache utilization on each source server. Cache utilization is calculated based on the number of items in the cache and the maximum cache size. This factor enables the Service Manager to attempt to achieve equal cache/datafeed utilization across multiple source servers.
    b. The "best preemption candidate" across all source servers of the service. If the priority of the best preemption candidate is greater than the priority of the requested item, there is no server "eligible" to service the request and the request is placed on the recovery queue. This factor prevents the Service Manager from preempting an item which has a higher priority than the requested item.

If an eligible item is preempted, it is placed on the preemption recovery queue and recovered later by the Service Manager

Removal of an unwatched item in order to satisfy a request for a new item is not considered as preemption.

The precise operation of the routing algorithm is a function of the values supplied to a number of configuration variables.

7.2.2 Request/Item Failure and Recovery

When the Service Manager detects that a watched item has been closed on a source server with a recoverable state indication, such as in the case of preemption, it tries to re-request the failed item from the remaining source servers. Similarly, if the Service Manager detects a server failure, it tries to re-request the failed items from the remaining source servers. Consumer applications and RTICs do not get involved in item recovery, thus minimizing network traffic. However, consumer applications are notified that the data streams have been interrupted, via a STALE or group STALE state indication generated by the source infrastructure, and that the system is trying to recover the failed items. Failed items are recovered at a controlled/configurable rate and according to their priority on the network. Any failed items that cannot be recovered due to limited capacity of the market data service are placed on the low priority (recovery) queue to be recovered by the Service Manager later.

When the Service Manager receives an RTIC request for a new item, it is placed on the high priority queue according to the request network priority for immediate routing. If a new request is received by the Service Manager for an item already on the low priority queue, the item is moved to the high priority queue for immediate routing, subject to all routing rules.

The Service Manager does not get involved in the recovery of a pair of mirrored servers when the primary source server fails. The secondary server is responsible for the recovery in this case. However, if both primary and secondary source servers fail, the Service Manager tries to recover the failed items (using the strategy described above) from the remaining servers.

7.2.3 Service Manager Failure Processing

When the Service Manager fails, system operation is identical to the operation of a non-Service Manager system, i.e. service management reverts to distributed mode.

The system components detect that the Service Manager has failed when:

- the Service Manager broadcasts a Goodbye message upon shutdown, or
- no messages have been received from the Service Manager for a configurable amount of time.

The failure of the Service Manager is completely transparent to consumer applications. All active data streams remain un-affected and continue to receive updates. All outstanding requests which are pending images are re-issued automatically by the RTICs. Any Close or Priority messages sent within a configurable time interval prior to detection of the Service Manager failure are assumed to have been lost; therefore, the RTICs re-transmit them. The size of this time window (in seconds) depends upon how quickly a Service Manager failure can be detected by RTICs and Source Distributors. This simple algorithm ensures that the Source Distributor and RTIC watchlists remain synchronized after a Service Manager crash without a full download, saving time and network traffic.

7.2.4 Service Manager Restoration Processing

During the restoration process of the Service Manager, each source server in the service downloads its watchlist, cachelist, and state to the Service Manager. After the download completes, the Service Manager eliminates any duplicate data streams and cache duplication before starting to service requests from the RTICs. During the download phase, source servers continue to process consumer requests.

7.2.5 System Summary with Service Manager

The features of the Service Manager are:

- Support for high capacity, high performance, and high resiliency source services. Market data services can be partitioned and distributed across multiple physical nodes in order to achieve high performance and high resiliency. The request routing decisions, however, are centralized in order to achieve effective load balancing and preemption.
- Resiliency to a Service Manager failure. Service Manager failure does not affect active data streams. While the Service Manager is down, RTICs take over the responsibility for the request routing.
- No duplicate data streams on the network and no cache duplication for source servers of interactive type.
- Efficient retrieval of cached items. Once the location of an item is learned, the retrieval cost is fixed and is independent of the number of source servers in the service.
- Efficient failover/recovery of failed items. RTICs do not get involved in the recovery. The number of network packets is significantly reduced when items are watched by multiple RTICs. The Service Manager generates a single request per item and multiple RTICs are recovered via a single image per item. Recovery proceeds from high to low priority items.
- Efficient handling of concurrent requests for the same item from multiple RTICs. A source generates an individual response to each RTIC request. However, if multiple RTIC requests are pending at the source server for the same item, they are satisfied by a single image. The image is multicast (broadcast) to all targeted RTICs. The Service Manager reads the image header to maintain watchlist and cachelist synchronization.
- Subscribe, Cancel, and Priority request messages are routed through the Service Manager. The Service Manager processes these messages and forwards them to the appropriate source servers. Routing of requests through the Service Manager guarantees message synchronization between RTICs, Source Distributor and Service Manager.
- The market data generated by source servers is broadcast directly to RTICs, i.e. the data is not routed through the Service Manager.
- Contribution requests and responses are not routed through the Service Manager.

7.3 Communication on Distribution Layer

The components on the Distribution Layer communicate using SASS/RV protocol which is made up of two protocol layers:

- Subject Addressed Subscription Service
- TIB/Rendezvous

Typically application developers do not need to be aware of the services provided by SASS/RV as these are abstracted by the SFC/RFA. This also applies to the actual format of the messages such as Marketfeed.

7.3.1 SASS (Session Layer)

SASS is a session protocol provided on top of the RV transport that supports market data semantics. SASS supports the RV-style publish/subscribe model with subject-based addressing. SASS protocol provides a means to maintain reliable market data streams despite component failures or network errors. It was specifically designed to distribute real-time market data between a large number of consumer

applications and an in-memory cache such as the RTIC. SASS supports direct publish/subscribe communication between consumer applications as well as indirect publish/subscribe communication with an RTIC acting as middleman and providing persistence for market data.

7.3.2 TIB/Rendezvous (Transport Layer)

Underlying the Distribution Layer is a single high-performance messaging layer from TIBCO Software Inc.: TIB/Rendezvous (RV). RV is a patented real-time, reliable IP-based multicast distribution system using publish/subscribe with subject-based addressing that has been proven in demanding real-time environments such as the trading floor. RV's "publish once, subscribe anywhere" paradigm provides a very high level of scalability and gives applications complete location independence.

TIB/RV provides a foundation for the TIB/ActiveEnterprise suite of products such as TIB/RVTX for transactional support, TIB/RVCM for certified messaging, TIB/Hawk for monitoring and management of distributed applications, etc., and thus facilitates/enables integration of other real-time decision support applications within the core RMDS distribution architecture.

The RV reliable distribution protocol is an optimistic protocol utilizing "negative acknowledgments". Lost messages are automatically retransmitted and re-sequenced to ensure a reliable, ordered message stream. Users are notified when message(s) are lost and cannot be recovered. Use of subject-based addressing enables effective and efficient filtering of unwanted multicast/broadcast packets at the RV protocol level.

RV is implemented on top of UDP (part of TCP/IP) which makes it portable across platforms that support TCP/IP. UDP provides point-to-point and broadcast datagrams, but is unreliable and does not guarantee that datagrams will be delivered in the order they were sent.

RV adds value in that it provides the following services:

- Reliable point-to-point datagram stream
- Reliable, sequenced broadcast datagram stream
- Preservation of temporal order within each stream
- Preservation of message boundaries
- Message fragmentation and reassembly
- Automatic packet recovery
- Communication of error/failure notifications
- Subject-based routing/filtering
- IP multicast support

7.4 Communication on Market Data Hub

RTICs, Service Managers and Source Distributors communicate over a Market Data Hub backbone network using Reuters Reliable Datagram Protocol (RRDP) which is made up of two protocol layers:

- Reuters Reliable Management Protocol (RRMP)
- Reuters Reliable Control Protocol (RRCP)

The services provided by RRMP/RRCP are available to applications indirectly via the SFC/RFA layer.

7.4.1 RRMP (Session Layer)

RRMP is a Market Data Hub protocol that deals with data semantics. It was specifically designed to distribute real-time market data between a large number of Source Distributors and RTICs, and the Service Managers. RRMP handles images and updates for page and record based data, as well as generic data types.

In addition to market data support, RRMP also provides services that support the efficient distribution of system data on a Market Data Hub such as:

- error reporting and recovery
- information resource management
- dynamic network configuration and source service discovery

7.4.2 RRCP (Transport Layer)

On the Market Data Hub, a reliable broadcast communication is based on the Reuters Reliable Control Protocol (RRCP). The RRCP distribution protocol, an optimistic protocol utilizing “negative acknowledgments”, provides reliability and ordering guarantees similar to RV protocol. Lost messages are automatically retransmitted and re-sequenced to ensure a reliable, ordered message stream. Users of RRCP such as RTIC are notified when message(s) are lost and cannot be recovered. RRCP implements bitmap addressing to support effective and efficient filtering of unwanted broadcast packets at the RRCP protocol level. RRCP provides network error reports to enable applications to implement various strategies for congestion avoidance.

RRCP is implemented on top of UDP (part of TCP/IP) which makes it portable across platforms that support TCP/IP. UDP provides point-to-point and broadcast datagrams, but is unreliable and does not guarantee that datagrams will be delivered in the order they were sent.

RRCP adds value in that it provides the following services:

- Reliable point-to-point datagram stream
- Reliable, sequenced broadcast datagram stream (with bitmap filtering of unwanted messages)
- Preservation of temporal order within each stream
- Preservation of message boundaries
- Message fragmentation and reassembly
- Automatic detection of node presence and node reboots
- Automatic packet recovery
- Communication of error/failure notifications
- Error/failure reports (to support failure avoidance strategies)

The RRCP layer has no knowledge of market data semantics. In fact, with the exception of the bitmap used for filtering, the RRCP layer has no knowledge of the RRMP layer.