

I hereby certify under 37 CFR 1.8 that this correspondence is being deposited with the U.S. Postal Service with sufficient postage as Express Mail # EB 663163465US in an envelope addressed to:
Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450
ON 10-19-07. Cheryl E. Harman
Cheryl E. Harman
Tel: 888-376-0095

VOX-P002PRV

**TELECOMMUNICATION AND MULTIMEDIA MANAGEMENT
SYSTEM AND METHOD**

INVENTORS:

Thomas E. Katis
Jackson Hole, WY
Citizenship: USA

Matthew J. Ranney
Oakland, CA
Citizenship: USA

James T. Panttaja
4016 West Soda Rock Lane
Healdsburg, CA 95484
Citizenship: USA

Daniel B. Leifker
695 Castro Street No. 2
San Francisco, CA 94114
Citizenship: USA

Mary G. Panttaja
4016 West Soda Rock Lane
Healdsburg, CA 95484
Citizenship: USA

Assignee:

RebelVox, LLC
512 Second Street, First Floor
San Francisco, CA 94107

TELECOMMUNICATION AND MULTIMEDIA MANAGEMENT SYSTEM AND METHOD

1	INTRODUCTION	3
1.1	REBELVOX PRODUCT INTRODUCTION	3
1.2	MARKET CHALLENGES	5
1.3	PRODUCT CONCEPT	6
1.3.1	<i>What RebelVox Delivers</i>	7
1.3.2	<i>The Keys to Creating the Value Potential</i>	8
1.3.3	<i>Why It Can Be Done Today</i>	8
1.3.4	<i>Additional Solutions in the RebelVox Repertory</i>	9
2	REBELVOX MARKET OVERVIEW	10
2.1	TARGET MARKET DESCRIPTION	10
2.2	TARGET CUSTOMER DESCRIPTION	11
3	REBELVOX FEATURE/FUNCTION OVERVIEW	18
3.1	APPLICATION FUNCTIONALITY	18
3.2	ALL FEATURES AND FUNCTIONS BY APPLICATION	18
3.2.1	<i>RebelVox Multiple Conversation Management System (MCMS)</i>	18
3.2.2	<i>Features and Functions</i>	18
3.2.3	<i>Performance Attributes of MCMS</i>	26
3.2.4	<i>RebelVox Multiple Conversation Management System - Concurrent (MCMS-C)</i>	26
3.2.5	<i>RebelVox Store and Stream</i>	27
3.2.6	<i>RebelVox Distributed Services Architecture</i>	28
4	REBELVOX ARCHITECTURE OVERVIEW	29
4.1	HIGH LEVEL ARCHITECTURE	29
4.2	REBELVOX COMPONENT ENUMERATION	32
4.2.1	<i>RebelVox Component Product Requirements</i>	33
5	REBELVOX USER INTERFACE	58
5.1	INTERFACE COMPONENTS	58
5.2	INTERFACE FEATURES AND BEHAVIORS	59
6	PRODUCT INFRASTRUCTURE	61
6.1	1.1 HOW REBELVOX RELATES TO TELECOMMUNICATION PROTOCOLS	61
6.2	1.1.1 NETWORK COMMUNICATION	61
7	INDEPENDENT SOLUTIONS	67
8	REBELVOX GLOSSARY OF TERMS	68
8.1	SYSTEM COMPONENTS	68
8.2	MEDIA	68
8.3	MEDIA MANAGEMENT	69
8.4	PEOPLE	70
8.5	CONVERSATION MANAGEMENT	70
8.6	SOME SYSTEM CONCEPTS: MORE DEPTH AND DETAIL	71
9	REBELVOX PROCESS DIAGRAMS	80

1 INTRODUCTION

1.1 *RebelVox Product Introduction*

RebelVox is delivering entirely new modes of voice communications to the market. Beyond phone calls, conference calls, and voice mail, RebelVox delivers an essentially new way of communicating and deriving lasting value from your voice communications. We call it “voxing”.

How does it work?

RebelVox supports new modes of engaging in conversations and/or managing multiple simultaneous conversations using a variety of media types (voice, video, text, location, data, etc.). Users can engage in conversations by sending short voice messages to designated recipients at the click of a button (or a voice command, or an application selection, or any other user interaction modality). Depending on preferences, priorities, and their attention, the recipient might hear the message in real time, or they might simply be notified that a message is ready for retrieval. Users could transition from a series of back and forth transmissions to a synchronous, full duplex conversation (similar to today’s phone calls) and then back again. Recipients don’t need to wait for a message to finish being left before listening to it. They can listen to the message while the sender is still speaking, listen at a faster rate of speed and actually catch up to the speaker to join into a live conversation.

Users of the system can control their interactions with their most important conversations in real time without missing any messages from their other conversations. Users can be notified of urgent or high priority communications from a conversation that they are not currently listening to.

All messages from all conversations are stored for retrieval and can be reviewed at any time. The system is designed to make the sending of messages and their retrieval easier, faster, better organized, and to allow an easy transition from sending messages to a full conversation and back.

Organizations will be able to manage media communications as an asset, as the system will provide the ability to archive, search, catalog, organize, report, and use other management protocols normally applied to textual communication assets.

Is this VoIP?

While one mode of media transmission will be voice over IP, RebelVox is not merely VoIP. Current VoIP systems emulate traditional circuit-switched telephone calls, merely shifting the transmission protocol. All of the different configurations of this type of call, one-to-one live calls, multi-party conference calls, and voice mail are very simplistic methods evolved from using connected voice circuits. Current computational power and communication potential enables us to do much more. We are able to evolve these modes of communication.

Is this what is referred to as Unified Communications?

There has been some talk of a concept of Unified Communications, or “UC.” The various definitions revolve around a concept of tying together many of the major current forms of communication, such as

telephone, voicemail, email, fax, instant messaging, video, conferencing, file sharing, etc. with a common platform or user interface.

In a system of integrated components, you still have the individual components. You have not dramatically improved how the individual components actually serve us. They are just tied together with an application.

No, it's Voxing using the RebelVox system.

The RebelVox system is a new communication application that has attributes of current systems, but it is not just an assembly of them. For example, a system of voice messaging that can be live or recorded has attributes of voicemail and of a phone call, but it is not the same as the two, separately, or even as the two combined. If two individuals are using the RebelVox system to send recorded voice messages back and forth at a rate that they merge into a live, synchronous, full-duplex voice conversation that is indistinguishable from a telephone call from a user standpoint, it is clearly not voicemail, but is it just a telephone call? No. It is different in many ways.

When you "vox" someone or a group:

- The conversation is still discrete recorded voice messages.
- You can stop listening live and revert to sending voice messages and then merge back into a live call again.
- The conversation can be archived.
- You can divert your attention and not miss anything.
- You can pause the conversation for seconds, minutes, hours, or even days and pick right up again where you left off.
- You can rejoin a conversation at its live point, or you can rejoin it where you left off and "catch up to live" without missing any messages.
- No dedicated circuit is required for the conversation.
- It is not a PSTN call.
- It is not a VOIP call.
- To initiate a "conversation" you can simply begin transmitting to someone else or a group. If the person on the other end notices that they are getting a message, they can begin listening at the beginning and reply at any time.

Rather than an assembly of discreet technologies, the RebelVox system is a new communication application and protocol that has attributes and features of many disparate existing forms of communication and adds new ones. The RebelVox system is also missing many attributes of current systems. Fortunately, these are mostly attributes that people will be happy to leave behind.

- The need to wait while a circuit is established to initiate a call.
- The need to wait while the phone rings to begin a conversation.
- The need to wait for a circuit to be established, the phone to ring (or not) and to listen to greetings and a prompt in order to leave a voice message.
- The need to wade through a set of disjointed, unorganized, and unrelated messages in your voicemail box.

The RebelVox system can map to your organization, your internal processes and groups through a hierarchical top-down structure based on an organizational chart, process flow document, unit command and

control diagram or other plan. Additionally, the RebelVox system could be used to organically map your organization – by groups and by processes. Want to see how your organization really works? Want to see the real process flows? The RebelVox system can plot the real process and communication flows of an organization or project, with audit, control and search functions possible.

1.2 Market Challenges

The current state of voice and media communications suffers from inertia in the evolution of our understanding of what can be done and what should be done. While we need unification of our communication assets, actions, and behaviors, we also require a revolution in the modes of voice communication that we use. It is now possible to *re-invent voice* communications, not just unify the staid methods currently in use.

Voice Communications

- The use and management of voice communications has not kept pace with the utility of text-based communications. Using voice is not as flexible as text and we do not have the same breadth of management tool functionality for voice that we do for documents and text.
- Current systems are based on relatively simplistic usage patterns: real time live (mostly unrecorded) calls and disjointed voice messages (that are typically deleted as they are heard). These forms of voice do not capture the real power that we can achieve with voice communications and they do not take advantage of network, bandwidth, storage, or speed potentials that are now available in our communication infrastructure.
- The ongoing unleashing of communication bandwidth enables us to provide substantially more management capability for voice communications and messaging, but it will not happen without re-inventing how we think about voice communications.

Media Conversations As An Asset

- For the most part, we do not have an integrated way to manage voice as an organizational and corporate asset. We generally do not record and persistently store even the fact of and simple details about calls that are critical to our processes. And most of us do not record the payload of those conversations in any manageable form. Much of our potential communication assets are merely rendered into oblivion, gone as quickly as the words are spoken.
- For those industries where words can be binding and critical to decision making, we have few tools for meeting any regulatory or contract management requirements with our voice conversations. And we have no ability to track verbal messages and conversations with our other communication assets that relate to any given business process.
- We typically don't verify conversations and voice communications in areas of high responsibility, for example, in critical care units in hospitals where consultations may take place via voice or video and a

structured history of those communications organized along with sensor readouts and test results can create a complete traceable picture of the treatment, prognosis, and responsibilities over time.

- In ordinary business and personal systems, we have little ability to store voice, much less, organize and systematically manage and retrieve it. While this is available for high-end technical systems (trading desks and call centers, for example), the functionality provided by recording voice conversations is not generally available, but would be a core part of the RebelVox system.
- In some jurisdictions the recording of phone calls is prohibited or tightly regulated. It is unclear exactly how the RebelVox system will be treated legally in each of these areas, but the RebelVox system will have the ability to modify its data capture depending on who the RebelVox user is connected to. When a RebelVox user is on a call with a person on a normal phone (or VOIP phone, mobile phone, etc) the RebelVox system can handle the regulatory requirements in a number of ways. These ways include:
 - Not recording the conversation at all
 - Recording only the meta-data about the time and circumstances of the call
 - Recording the whole call, but restricting access to the recording (in the event of a court order, etc)
 - Recording the call but deleting the media at the end of the call
 - Recording the call but deleting only the media from the non-RebelVox user

Inadequacy of Tactical Communications

- Tactical communications suffer from a dependence on real time and analog methods of communication which do not support many functions you need in critical activities:
 - Guaranteed delivery of urgent messages
 - Ability to hear and manage multiple conversations at the same time
 - Ability to get perfect messages (or “good enough” messages) even if bandwidth and network quality are an issue
 - Ability to prioritize messages, individuals, groups, or channels
 - Ability to time-shift less important messages and manage replay and recovery of conversations
 - Ability to control duration and frequency of radio transmissions while maintaining ability to perfectly deliver all messages to all recipients.
- It takes more than being on the same channel, it truly requires a comprehensive voice management system that can deliver all of the features.

1.3 Product Concept

The RebelVox Communications System revolutionizes voice communication by providing, in addition to standard calls, conference calls, and voice messaging, the:

- Instant communication of push to talk radio,

-
- Immediacy and richness of a telephone call,
 - Archival properties of email and documents,
 - Threaded context of instant messaging,
 - Time-shifting management tools of systems like Tivo or other Digital Video Recorders (DVRs),
 - Multiple message management properties of email.

This is an entirely new approach to voice communications and supports new methods for engaging in conversations and managing media as an integral part of a unified communications solution and an overall information management strategy.

In this introduction to the RebelVox product concepts we will summarize:

- What RebelVox delivers
- The keys to the value potential
- Why it can be done today
- Additional solutions in the RebelVox repertory

1.3.1 What RebelVox Delivers

The RebelVox system revolutionizes communications as a media management system for voice, video, text, location, data, and other media that can be deployed in software and/or hardware using existing communication systems (Internet, mobile phone networks, private networks, military networks, police networks, etc.) or other wired or wireless communication methodologies.

The RebelVox system provides the following array of functionality:

1. A **conversation management system** for all types of voice (also other audio, video, or other media) communications including one-to-one calls, conference calls, and messages delivering everything with high levels of currency (low levels of latency – as good as the network will allow) with quality that is “good enough” and the eventual delivery of “perfect” copies, while also optimizing for network communication constraints. Conversations can be in “real time” or time-shifted (listened to some time after received) providing the ability to pause, rewind, replay, and catch-up-to-live, as well as shift seamlessly between the two states---“TiVo” every call or conference call in real time or listen time shifted. The system knows the status of all other participants represented by their degree of presence: network status (online or offline), their intention (to listen to a message live or time-shifted), and their attention (are they actually listening now).
2. A **multiple conversation management** system enabling the concurrent participation in and organization of multiple conversations simultaneously as well as queued and prioritized live calls and messages, where a conversation is an ongoing set of messages happening in real time or in time-shifted modes.
3. A **media management system** that enables users to prioritize, archive, tag, search, and recover all conversations and related media turning voice into a practical media of record--functionality similar to

Apple's "Spotlight" or Google's toolbar for voice and media. The incorporation of Voice Recognition could allow easier search and management of messages and conversations.

4. A **distributed service architecture**, which enables high levels of distributed functionality between RebelVox systems, devices, and gateways across networks including standard radio, PSTN, and other networks.

These key systems provide exceptional value for users of the system, including:

- a. Increased productivity from ability to participate in and manage multiple conversations,
- b. Increased capability and productivity from flexible communications management,
- c. Increased capability to manage and meet regulatory requirements surround critical communications,
- d. Increased capability to effectively manage active response to emergency and tactical events.

1.3.2 The Keys to Creating the Value Potential

The value of the RebelVox system derives from modifying the method of human interaction with voice telephony through a new application that expands the capabilities of devices through the unique use of specific processing principles applied across the board to voice, video, and other media for the first time:

1. **The ability to store voice data at the encoding device and at the receiving devices and systems** (instead of only rendering or transmitting it), which enables a whole host of new features and functions for the management, delivery, and use of voice. These features include, but are not limited to, the ability to time-shift the rendering of messages, the ability to recover from packet transmission failures, the ability to adjust for network bandwidth issues, the ability to catalog and manage voice data, and the ability for users to send media communications and work with received media communications whether or not they are on the network. The persistence of the storage of the voice data will be configurable. (The devices or system could store voice data permanently. It could be archived or backed up on a periodic basis. It could be discarded periodically. It could even be deleted from the encoding device after the data has been successfully delivered to the receiving device.
2. **Rapid streaming of media packets** combined with providing "real time" delivery, "good enough" delivery, and guaranteed delivery of a perfect copy.
3. **Cooperative data transmission protocols** that enable the optimization of media delivery based on the intentions of the receiver in addition to the quality of the network.
4. The addition of **semantically meaningful and configurable meta-data creating an information context** surrounding voice messages which make them highly manageable and reusable.
5. A highly **configurable user interface** that enables the system to be deployed in a wide variety of technical configurations including radio, mobile phone systems, PSTN, Internet applications, and others.

1.3.3 Why It Can Be Done Today

The strategic advances in RebelVox are made possible at this time by several factors in the marketplace:

-
1. **Continual increases in available bandwidth and improvements in connectivity.** The Internet is now available in your office, in your home, at the key places you go (coffee shops, airports), and on the street (with mobile technology).
 2. **Rapid increases in processing power capacity (at lower power consumption).** This makes it possible to provide routing and other capabilities on general-purpose computers.
 3. **Increases in efficiency of storage capacity.** General purpose servers can server as large data farms with substantial reductions in the cost of storage,
 4. **Evolution of communication protocols.** We have a rich set of options for communication over a variety of communications networks.
 5. **Evolution of powerful mobile devices.** The evolution of mobile phones into mobile computing devices with significant processing power, memory, battery life, large screens, and flexible user interfaces.

1.3.4 Additional Solutions in the RebelVox Repertory

Within the RebelVox architecture there are several general purpose processes that apply to other media management challenges which will be discussed in this document, including:

- **RebelVox Store and Stream.** A sender-receiver cooperative protocol for enabling delivery of “good enough” media with very low latency while also providing capture and archiving of “perfect” media images.
- **RebelVox Payload Optimization.** A sender-receiver cooperative protocol for optimizing packet contents, size, frequency, and quality used by the sender based on information known about the receiver’s state and intentions with regard to the payload.
- **RebelVox Flexible Call-Participation Management.** A call participant protocol that supports, in any conversation including conference calls, media viewing functionality including: pause, fast-forward, catch-up-to-live (listen more quickly with several options, including but not limited to: removal of silence, playing faster, removing hesitations), jump to head (go the “live” portion of the call”), replay from any point.
- **RebelVox Corporate Voice Messaging and Conference Calling Application.** This system, which could be delivered as software or on an appliance, would support all of the user and voice asset management of features of the core RebelVox MCMS-C, delivered as an application. It could include multiple call management, conference calls, voice messaging, and standard live calls, which would all have the RebelVox standard features of Flexible Call-Participation Management (pause, fast-forward, catch-up-to-live, jump to head, replay from any point), message tagging, prioritization, and archiving.

2 REBELVOX MARKET OVERVIEW

2.1 Target Market Description

The functionality delivered by RebelVox is profound in its ability to change the way voice is used. In those organizational and social functions where voice is an integral part of the process of engagement (social communications, doing business, communicating urgency, solving joint tactical problems), RebelVox will have a significant role in revolutionizing how we work.

RebelVox provides significant functionality and value to the following markets and application spaces (including, but not limited to):

1. **Business Communications** where voice is a method of executing business processes that can be improved by flexibility, call management, conversation management, retrieval, and tagging (e.g., summaries, actions items, contractual terms).
 - a. Integrated Messaging: Conference calling, voice messaging, live calls, transitions between messaging and live calls
 - b. Distributed Meetings
 - c. Structured Conferencing (e.g., simultaneous briefings)
 - d. Regulatory controls over voice communications:
 - i. Project oversight
 - ii. Contract management
 - iii. Government contractors
2. **Public and Social Networks** where flexible voice communications can enhance relationships by allowing voice to be as flexible as text in engaging in conversations.
 - a. Dynamic conversation management
 - b. Voice over IP systems
3. **Media Industries** where the need to transmit large volumes of data can be enhanced by the RebelVox cooperative transmission protocols.
 - a. Broadcasting
 - b. Music and video streaming
 - c. File sharing and distribution
 - d. Peer to peer file transfers
4. **Regulatory Applications** where the ability to archive, tag, and search organized and structured conversations allows increased management control and verifiability.
 - a. Hospitals and medical communities
 - b. Flight control systems
5. **Legal and Compliance**
 - a. Discovery
 - b. Tagging of Non-Discoverable
 - c. Tagging of "covered under NDA"
 - d. Tools for compliance initiatives in organizations
6. **Tactical Communications** where control, message management (replay, pause, catch up to live), archiving, guaranteed delivery, and sophisticated routing can enable a more robust and safe response to tactical situations.

-
- a. First responders (Police, Fire, Paramedic)
 - b. Military
 - c. Private Security

2.2 Target Customer Description

Here are some examples of communication challenges that are addressed with RebelVox.

The Weekly Sales Meeting

Monday morning at 9 AM is the scheduled sales meeting, but the sales team is never in one place: some folks are in the office, some are on the road (metaphorically or actually in a moving car), and some are holed up in airports or hotels. In the conference room where a few have gathered, the remote team starts to call in. For various reasons, of course, they are all not available at 9 AM and the straggling begins. Each person gives a brief status review on the state of his or her projects; the stragglers miss many of them. The sales manager was on a phone call and missed the first half of the briefings, and as she comes in, she's not sure what to do about that. It takes 43 minutes to get through the briefings and get everyone on line, and then they can start the discussion of what they need to get done this week--the current goals and tasks that are the intent of the meeting. People who have been in the room since before the meeting started are beginning to get a little bored and step out for coffee. The sales manager has to step out and when she returns, the meeting has finished. She's not sure what the conclusion was.

Information is lost as the reports are missed and cannot be retrieved by anyone. (The current state of recording such a conversation would allow the whole mélange to be heard after the fact---without the ability to track individual messages and not in time to participate in the follow-on conversation with the needed information.) Few people are really up-to-date and the planning conversation isn't optimal. The sales manager suggests that Task A really needs to get done, having missed the message that it was completed.

Using a RebelVox communication protocol called Simultaneous Briefing, the team can solve the problem seamlessly. In one form of RebelVox structured conferencing, the Simultaneous Briefing (a type of standing conference call) is scheduled for 9 AM. Everyone one is to call into the conversation before 8 AM (even the night before) and leave his or her status briefing. Between 8 AM and 9 AM, the team can call in individually and listen to all of the status briefs from wherever they are, even using a "fast play" option to make it go more quickly. (The system could also provide transcribed versions of the messages.) At 9 AM, the available folks join the conversation and begin work; anyone who is late can continue to "catch up to live" until they seamlessly transition into the live conversation, having never missed a word. The manager can take a few minutes to catch up on all the briefs before she joins the conversation and is fully up-to-date when she engages her team. After her interruption, she is able to listen to the rest of the meeting in fast play mode or skip to a "summary" message left by the team at the end of the call, skipping all the messages in between. And if she is unsure of a point, she can replay the messages in total, or select them by participant.

Executive Communications

An executive comes out of an important client meeting and, while traveling to the airport, wants to communicate five different critical messages to different groups of people (20 people in total) that include

his secretary, his CEO, two of his teams, and another key manager working on the project. In today's world, he would have limited options including:

- Place twenty different phone calls to all recipients, one at a time, each call resulting in either a live conversation or a voicemail.
- Take the time to coordinate five conference calls with the required parties, if indeed that is even possible.
- Set up groups for delivering group voice messages to the teams, if that solution is available.
- Take the time to type out emails to the different groups, assuming he is carrying the appropriate communications gear.

He doesn't know when they will get the messages and the members of the team may not know who else has received which messages. If they use voice, there will be no link or thread between the messages as the conversation moves forward. (In fact, the messages will likely disappear from the conversation context, as they are deleted one by one.) The sense of a conversation will be lost and when the executive lands, he will not join an ongoing conversation that is a meaningful sequence of messages, but instead he'll get a set of disjoint messages from the various managers and groups.

Using the RebelVox system he could pull out his RebelVox enabled phone or device and make five transmissions to his established groups, starting five new conversations. He would not need to wait to see if anyone was available. He would not need to coordinate a meeting or conference call. He could select the individual or group he wanted to pass a message to, press a button and transmit the message. Individuals receiving the message could listen to it when they are ready and respond either with their own transmission or with a quick text message. Every message shared would be a part of the whole conversation and available to all participants whenever they were able to participate. Each of the five groups could continue to evolve the conversations around this initial message. As the executive lands, he will proceed to listen to, scan, or search for summary tags in the conversation and can catch up on everything quickly. And now he can participate in any or all of these based on his time and preferences.

A Conference Call

Peter and Paul work for Company A and are negotiating a deal with Company B. Representatives of both Companies are together on a conference call negotiating terms. Normally, Peter and Paul are in the same physical location and will frequently press the mute button to make comments and ask questions that they don't want the other parties to hear. However, if they are each calling in to a conference call number from different locations, their options are limited to:

- Calling each other on another phone (mobile phone, etc.) to have a side conversation.
- Using their Internet-connected computers and having an IM conversation while on the conference call.

Using the RebelVox system you aren't stuck in one "call". A "conference call" is just a conversation that you can participate in among others. You can be engaged in a full-duplex, synchronous conversation with the whole group, while at the same time periodically pressing a button to transmit messages to just one of the parties, or to a totally unrelated party. If Peter transmits a message to Paul, Paul won't be interrupted from his conversation. He will see that he has a message from Peter, and he can choose to listen to it (or

read it if it is a text message) when there is a lull in the conversation. Paul can speed up Peter's voice to minimize the time he is away from the primary conversation. On returning from Peter's message, Paul may speed up the primary conversation until he is caught up to what is currently being said.

At the conclusion of the negotiations, the participants can agree to record a conclusion and one or more messages can be tagged as summary, conclusion, action items, or contract. Any time they want to review what was decided in the meeting, any of the participants can pull up the appropriate message by searching with the right tag.

A Fire Department

At a large fire there will be multiple teams attacking different parts of the problem. Each team may have a particular task at hand:

- Rescuing anyone that is trapped.
- Fighting the fire itself.
- Protecting the surrounding area and evacuating people.
- Giving medical aid to victims.
- Controlling bystanders

There will be a leadership team coordinating and overseeing the activities of the different groups including the arrival of medical helicopters, ambulances, and other services.

If all of the firefighters and paramedics in the different groups are on the same channel then that channel becomes crowded and chaotic. Transmissions get "stepped on" when more than one person is transmitting at a time, and everyone becomes less efficient. For teams that are involved in dangerous activities, such as those inside a burning building, their priority is to communicate instantly and effectively with their nearby teammates. Having to share their channel with non-critical communications coordinating actions outside the building is not only inefficient but dangerous. A possible solution is to give different teams their own channels. This fixes the problem of overcrowded channels, but it creates a new problem of putting up barriers to communication between the different teams. Should the rescue team be on the same channel as the fire fighting team or the paramedic team? What channel does the Fire Chief listen to? Does he have a different radio for each team? Can he listen to them all simultaneously? What if he is in the middle of one conversation with the helicopter while someone fighting the fire inside the building is trying to have an urgent conversation on a different channel?

With a RebelVox solution everyone can be organized into multiple different groups, with each group having its own priority level. For example, all of the firefighters inside the building can have their own priority group. The leader of that team can be giving updates to the Fire Chief without the whole team hearing, or he can broadcast updates to everyone. Any urgent transmissions by his team members inside the building ("Look out, the floor is going to collapse!") override his non-essential side conversations and are played immediately. The push to talk buttons could be designed to have multiple levels of priority (e.g. push it in half way for a normal priority transmission and all the way for an urgent or emergency one).

If the Fire Chief wants to know what is going on in the building but it isn't an emergency, he can transmit a message to the team leader at normal priority (below the priority of his team members in the building. The

team leader will hear or see that he has a transmission from the chief and can listen to it. He can start listening to it whenever he wants, including before the Chief has finished transmitting.

If, on the other hand, the Fire Chief has an urgent message (e.g. "The building is going to collapse. Everybody get out!"), he can transmit at the highest priority level to everyone at once and it will override every other conversation.

One problem that radios can have is bad reception, especially in buildings and urban settings. This can be helped by having repeaters spread throughout cities and buildings, or even the construction of a mesh network, but it is still possible for there to be dead spots without reception. With the RebelVox system if someone was in a dead spot when a critical message was transmitted they would get that message (and all others that they missed) when they came back into coverage. The messages would be prioritized, so they could receive the most important ones first. In areas of marginal coverage, the RebelVox system would delay messages slightly until enough good data was available to be intelligible to the listener. Current systems simply deliver intermittent static, requiring all channel users to hear the request for and delivery of a repeat transmission, even if they received it perfectly the first time.

A Special Forces Team Leader

There are many different deployments of voice radios in tactical environments for Military, Police, Fire and other user groups. For the purpose of exploring the complexity and limitations of current tactical voice systems we will discuss the challenges of a Special Forces team leader in making effective communication during tactical operations. An SF team will have a primary and at least one alternate frequency (or channel) for their own internal use. If one frequency becomes jammed or otherwise ineffective, somehow everyone needs to figure out that they need to switch to the alternate frequency.

Higher units (Company, Battalion, Group, Task Force, Joint Command, etc.) will each have their own frequencies / channels. Each will likely have multiple ways of communicating using voice, including line of sight (LOS) frequencies, HF (High Frequency) and Satcom (Satellite Communication). Depending on whom the SF team leader wants to talk to, he will have all of the various frequencies and channels programmed in his radio(s) with the appropriate crypto fills, and he will have familiarized himself with all of the call signs.

The different options (LOS, HF, SINCGARS, Satcom, etc.) may have technical and practical differences, but they have enough similarities in their use for voice communication that we will simplify and refer to all of them as simply different channels.

An SF team will also have other support units outside its chain of command. These might include Close Air Support (CAS), MEDEVAC, Quick Reaction Forces (QRF), Artillery, etc. A team will also likely have other military units in the area that it needs to coordinate with. These might include other SF teams, Infantry units, Armor units, Mortar teams, Fire Base Tactical Operation Centers (TOCs), Civil Affairs units, Psyops units, etc.

Obviously, not everyone can share one big channel. In current tactical voice communications, on any one channel there can only be one person speaking (transmitting) at a time, and an unlimited number of people listening. In some instances a Satcom channel for a higher unit or Task Force becomes monopolized by a group (e.g. a particular battle that has priority) and all others are asked to use alternate channels.

Here are some illustrations of the critical problems that a SF team leader has in any field situation:

-
- The military assigns a vast number of different frequencies or channels to different units to improve the efficiency of communication within those groups. This results in a Balkanization of networks. It becomes very difficult to manage communication between and amongst different groups. Someone needs to leave the channel they were on to contact someone on another channel. That person cannot hear anything about what is happening on their own channel until they return to it. This **Balkanization of networks** is a critical problem.
 - There is no guarantee the person they are trying to reach will be on the channel they have switched to, or that they will be listening. Far worse than Schrödinger's cat, the uncertainty in military communication networks is that there is no guarantee that if someone isn't on their assigned primary frequency that they will be on their alternate, or on any other given channel. There are so many possible channels that an individual could choose that sometimes you cannot reach someone and you have **no ability to "ring" a specific person**.
 - It is also extremely difficult, and frequently impossible, to speak to only your intended audience. Often a message from one person to another person has to go out over a channel that dozens or hundreds are listening to. There is obviously no privacy, and there is a burden on the channel. This is caused by a **lack of granularity in the ability to control the members of any given conversation**.
 - If any one person on a radio net accidentally keys their microphone (e.g. by leaning against a car door, sitting on their radio, or some other careless act) they can make the entire channel unusable for everyone else. This problem is referred to as a **Hot Mic** and critically disables a radio network.
 - Everyone must go through the identification process on each call and must acknowledge receipt of each message. There is no built-in identification and authentication for voice communication. Each message begins with "you this is me." If my call sign is Smokey26A (pronounced "smokey two six alpha") and yours is Bighorn55 (pronounced "bighorn five five") then each time I contact you I have to say "bighorn five five this is smokey two six alpha over." You would reply "smokey two six alpha this is bighorn five five go ahead over." I would then deliver my message as "bighorn five five this is smokey two six alpha we are at checkpoint alpha and continuing mission over." If all went well the next transmission would be "smokey two six alpha this is bighorn five five roger, at checkpoint alpha, out." There is no ability for me to just select the contact I want and transmit "we are at checkpoint alpha and continuing mission" and have the system automatically find the person I want the message to find, identify who is sending it, and give me a receipt that the person has received it. These problems come from a system that **wastes time on identifiers, has an inability to leave a message, and no verification of receipt of a message**.
 - If more than one person is trying to reach you simultaneously there are a number of issues. The person that actually contacts you is the one that knows which channel you are on, not the one that has the highest priority message. If both people trying to contact you at the same time are on the correct channel, the one with the more powerful signal will "win" and drown out the less powerful transmission. You can only receive one message at a time, but no one can leave you a message, even if you are in the middle of another conversation. The system has **no tools for setting priority, causes you to miss messages that are not heard in real time and has no ability to manage more than one conversation at a time**.
 - If distance, terrain, buildings, interference, weather, or other factors are degrading the signal performance of the voice transmission, at some point the transmission becomes unintelligible. Many transmissions are radio checks. Many transmissions include the words "say again, over."

Retransmissions are often equally indecipherable. This is caused because there is a **minimum signal integrity to enable a live voice transmission** and there are **no other options than live calls**.

This is not a complete set of the issues of current tactical voice systems. Rather, it is a representative sample of the real problems our military and other government organizations face with their current communication systems. The RebelVox system addresses each of these problems.

The RebelVox system allows users to configure their devices to talk to any number and specificity of contacts at the touch of a button (addressing granularity). There is no need to carve people up into different frequencies of channels, because the system will include and exclude exactly whom you want it to (addressing Balkanization of networks).

The system offers full duplex conversations, so a “hot mic” doesn’t cancel out the ability for others to communicate. At worst, it adds some additional background noise to the channel. A net operator or controller could adjust down the gain of the offending party, or someone could “ring” him to let him know of his mistake. At any time someone could chose to “ring” someone with a variety of tones, vibrations, or other signals to request a live conversation, notify them of an action or event, or to communicate something else entirely.

Instead of “ringing” someone, users could simply select their recipient and transmit their message. If the recipient is listening to that channel they will hear it live. If they are speaking to someone else, depending on their presets and the levels of priority a few different things could happen. The message could be automatically recorded and the recipient notified. Alternatively, the recipient could be interrupted and the messaged played live if it is of a sufficiently high priority and urgency to warrant the interruption. If it was transmitted into a full duplex channel that both were a part of, then you could hear everyone talking as if it was on an open conference call. These settings and priorities can be configured by the users or by someone in the unit responsible for configuring the communication network.

All users are identified by the system, so verbal identifications could be abbreviated or eliminated altogether. All messages have notification of receipt and nothing is ever lost. Any messages that aren’t played live to the recipient are recorded and the recipient is notified.

If the signal quality gets so bad that the voice transmission becomes unintelligible live, the RebelVox system will keep automatically attempting to fix the data that makes up the transmission until the message is understandable. Users can opt to move into a “messaging mode” that will get the messages through as fast as the signal quality will support.

Not every message in the military has life or death urgency, but some do. The RebelVox system helps soldiers, sailors, airmen and Marines manage the full spectrum of their communications needs with the correct prioritization, management and control to be as efficient and effective as possible.

Tactical Communication Vulnerability

In tactical applications, radio transmitters become vulnerable in fields of operations where others are using radio direction finding to locate transmissions sources. The ability to control how often and how long a transmitter is broadcasting can be critical and this is managed by sending transmissions grouped into radio bursts. RebelVox’s ability to store all voice messages in a queue for periodic transmission bursts can be an effective security solution in these situations.

Getting Your Daily News

Sally wants to download the evening news to her iPhone so that she can watch it on the subway in the morning. The download is available at 5PM EST when she is still at her desk working. She finally gets it configured to automatically download when it is available, but discovers that when the download is underway, her computer becomes unusable. Her system tries to download it as quickly as possible assuming she is trying to watch it at that moment. It hogs all the computational and network resources. So she has to shut it down and finally has to cancel the download request. Her download either all comes at once when it is available (and she has to stop using her computer), or, she just can't have it.

A RebelVox cooperative data transmission would be configured to know how she wanted to process that download: when it should happen, what kind of priority it should have. It can know her intentions about the download and her current attention (she's working on something else) and optimize the sending protocols to meet her expectations on when it should arrive, how fast it should arrive, and how much resource to spend on it.

3 REBELVOX FEATURE/FUNCTION OVERVIEW

3.1 Application Functionality

The RebelVox Communication System provides the following functionality. This list is not exclusive, but an example of the features which will be included. The feature/functions sets are classified into the following application areas:

- RebelVox Multiple Conversation Management System (MCMS)
- RebelVox Multiple Conversation Management System - Concurrent (MCMS-C; uses MCMS functionality with additional features)
- RebelVox Store and Stream (SAS; media and payload transport)
- RebelVox Distributed Service Architecture (DSA)

3.2 All Features and Functions by Application

3.2.1 RebelVox Multiple Conversation Management System (MCMS)

RebelVox supports new modes of engaging in conversations and/or managing multiple simultaneous conversations using a variety of media and data types (voice, video, text, location, data, etc.). Users can engage in conversations by sending short voice messages to designated recipients at the click of a button. Depending on preferences and priorities, the recipient might hear the message in real time, or they might simply be notified that a message is ready for retrieval. Users could transition from a series of back and forth transmissions to a synchronous, full duplex conversation (similar to today's phone calls) and then back to voice messaging again.

Users of the system can control their interactions with their most important conversations in real time without missing any messages from their other conversations. Users can be notified of urgent or high priority communications from a conversation that they are not currently listening to.

All messages from all conversations may be stored for retrieval and can be reviewed at any time. The system is designed to make the sending of messages and their retrieval easier, faster, better organized, and to allow an easy transition from sending messages to a full conversation and back.

Organizations will be able to manage media communications as an asset, as the system will provide the ability to archive, search, catalog, organize, report, and use other management protocols normally applied to textual communication assets.

MCMS is a telephony application and consists of a distributed application architecture including a user interface, client API, client databases, network connectivity, server APIs, and server databases.

3.2.2 Features and Functions

Through a single proprietary architecture, the RebelVox Multiple Conversation Management System will provide the following functionality to users of the system:

1. The system will manage media (conversations and messages) in the following ways:

a. Conversation Management

- i. The system will transmit media (e.g., voice, video, and other media) between participants.
- ii. Media messages are organized into conversations, which are “threaded” together by a user-defined topic or attribute.
- iii. All media that is sent or received by the device is recorded for time-shifting (listening to a message some time after it was received), conversation management, and archival purposes. Media is available for listening or viewing while it is being received.
 1. Messages or conversation can be optionally marked for transience, that is, they will not be recorded beyond the requirements for immediate rendering.
- iv. Messages will be identified by participant (speaker) and the system will, by default, remix messages that were delivered at the same time, though a user could optionally listen to the transmissions separately.
- v. Each conversation is allowed to send media to a receiving device at any time, and the receiving device associates these messages with conversations appropriately, regardless of the current actions of the receiving user.
- vi. All conversations are fundamentally asynchronous. However, if two parties are actively engaged in a given conversation and their delay between transmissions is small, then their experience will be one of a synchronous conversation, as in current telephone or VoIP conversations. If either party delays their participation for whatever reason, the conversation may drift towards an asynchronous experience. When the network is weak or problematic, the worst-case solution in RebelVox is that the participants will be signaled that the live call has been shifted into message mode, and they can continue to leave messages. This model is similar to the way Instant Messaging (IM) conversations take place.
 1. A conversation can be optionally tagged to stay asynchronous (the system will not render messages as in a functioning real time conversation) with two options:
 - a. Only take messages
 - b. Only play messages.
 2. It could then be switched to a “normal” RebelVox conversation that will flow between synchronous and asynchronous naturally.
- vii. Messages may be time-shifted to allow a user to “catch up” on a conversation if they missed some messages or were involved in another conversation. Older messages may be replayed at any time at potentially higher speeds. Users may easily skip forward and backward through their messages and within individual messages. The catch up process may be configured on a message-priority basis to potentially skip messages below a certain threshold. This time-shifting process eliminates users having to broadcast a request to their entire group to have a message repeated.
- viii. When using the system for a Standing Conversation (a sequence of linked messages), the phone can give the user the choice to “Reply To” instead of requiring the user to “call” the other participants.
- ix. Each Conversation has a variable number of Contacts associated with it.

-
- x. The system can merge two conversations, which is how you can reach out to someone and then pull them into an existing conversation.
- b. Conversation Sharing**
- i. Conversations will be shared among the participants.
 - ii. Participants can be added to an active or archived call, which would enable the added participant to retrieve a previously completed call.
 - iii. The RebelVox Rich Audio Transcript Format (RATF) will enable users of the system to extract and send an “image” of a complete conversation including all the meta-data about the conversation, any tagging, and the payload. The RebelVox RTF Renderer will enable anyone to view and listen to a RebelVox conversation.
- c. Communication Management**
- i. Devices on the network typically communicate with a Server, but they may communicate directly (peer to peer) with other Devices under certain circumstances.
 - ii. A Device may also be an established Server for other Devices, typically in a vehicle-based or fixed-site Device with extra power and storage.
 - iii. MCMS will leverage the RebelVox distributed service architecture to enable conversations across: proprietary domains, communication device types (e.g., phones and radios), and disjoints in communication protocols.
 - iv. A device will have a configurable burst transmission capability, which will periodically send a set of queued transmissions as rapidly as the network can handle.
- d. Rendering**
- i. The system will render media appropriately for the device and media type.
 - ii. The system will mix messages as appropriate to creation times (overlapping messages).
 - iii. The system will be able to render transcriptions or translations of audio media knowing who recorded every portion of a conversation.
 - iv. The system will be able to adjust rendering according to user-defined criteria including playing faster, removing silence, and removing hesitation particles.
 - v. The application will have the ability to apply independent gain control to individual senders in a multi-party conversation.
 - vi. The time-shifting capability of the system enables the rendering process to do a better job of squelch control.
- e. Messaging and Signaling**
- i. Special messages or audible tones may be played before listening to or displaying messages depending on whether they are live or recorded, who they are from, priority, and other factors.
 - ii. The network can have the ability to signal the presence or absence of individuals on the network, as well as the ability to notify if users are no longer actively listening on a given Conversation.
 - iii. One user may “ring” or signal another to get their attention when they are paying attention to another conversation. This feature may be used or disabled.
 - iv. Messages may be left for people not on the network to be picked up the next time the individual signs on. This process will likely happen automatically when a message is being sent, but some sort of audible or visible feedback will alert the sender that this

message may not be received immediately or confirm that it has been received by the recipients device, and, if desired, that it has been heard.

- v. Individuals on a conference call or tactical network who have shifted their attention away from the device can be signaled or “rung” to notify them that their attention is needed back on the call. This indication may convey multiple levels of urgency. These levels can include priority, as well as whether an acknowledgement is required.

f. Presence and Status

- i. The system will have a cooperative architecture for sharing presence and status data between all the users.
- ii. The system will enable users to monitor the status of other user’s intentions, attention, and their Time Shift Delay on any given conversation (how far back they are from listening to the conversation live).
- iii. There will be privacy controls concerning availability of presence and status data.
- iv. It will be a configurable option to override the availability of presence and status information.
 - 1. One option will be to BCC (blind copy) someone into a conversation where other participants do not know that they are on the call. The BCC participant will be in listen-only mode.
- v. The system will deliver us messages that match our usage and intentions and will deliver to others messages that match their usage and intentions.
- vi. The user will indicate his status by his choices to listen or not to listen to conversations and the system will infer other status values from his behaviors and the systems knowledge of the processing of messages.
- vii. A non-RebelVox user can opt out of being recorded as part of the conversation.
- viii. The system can be configured to not record messages for some classes of users (or endpoints, e.g., PSTN) if required. They will be stored only until rendered.
- ix. Presence and status will be used to optimize network traffic and bandwidth.

g. Permissions and Roles

- i. The system will support a hierarchy of permissions based on users roles in the system.
- ii. Permissions will be tied to actions such as the ability to define groups, conversations, to end conversations, adding users, etc.
- iii. Permission to use key attributes may be controls, e.g., the “importance” attribute which causes messages to override priorities.

h. Integration

- i. The system will be constructed of functional components which can be together or independently integrated with other services, such as Media Messaging Systems, Unified Communication Systems, simple messaging systems, social networks, and general communication tools.

i. System Management

- i. The system will have management controls for administration of the system as well as system tracking and usage analysis.

2. The system will enable participation in a variety of conversation types:

- a. Live Conversations (users are speaking to each other in “real time”, considered synchronous)

-
- b. Standing Conversations (users are exchanging messages as in instant messaging systems, considered asynchronous)
 - c. A seamless transition through the continuum from Standing Conversations to Live Conversations and back
 - d. Conference Calls (scheduled or ad-hoc multi-party conversations, often synchronous)
 - e. Standing Calls (regularly scheduled multi-party conversations)
 - f. Structure Conversations (e.g., Simultaneous Briefings -- everyone leaves their weekly briefing for the group by 8AM; everyone listens to the other's briefings by 9AM; at 9AM everyone joins a live call)
3. **The RebelVox User Interface API (UI API) for the MCMS application** is designed to be highly configurable and will provide very flexible interfaces (on flexible devices such as web-based applications) and very configured interfaces (with few options) for devices that are constrained (such as traditional tactical radios). The configuration will be handled by a middle tier of functionality that will enable user interfaces that are graphics-based, signal-based (tones), radio-based, and others. The MCMS application will have a variety of instantiations, including, but not limited to programmable phones, proprietary devices (small computers and or devices), personal computers, tactical radios, and web applications. Some features of the MCMS application suite will likely be supported on particular platforms (e.g., complex reporting and archive management may only be delivered with a computer desktop or web application). Highly constrained user devices such as tactical radios, will be configured using special configuration applications that will run on general purpose computers which can download configurations to the radios. MCMS will enable users to manage conversations in the following ways:
- a. Define and Manage User System Data**
 - i. Define conversations according to participants, topics, times, and other tags.
 - ii. Manage contacts and groups of contacts (potential conversation participants).
 - 1. Organization and designation of groups
 - 2. The rules and hierarchies of groups
 - 3. Permission structures around the creating of groups and conversations.
 - iii. Manage default prioritization of contacts, groups, conversations, and messages.
 - iv. Manage importance of messages enabling users to send a message marked "urgent".
 - b. Manage Conversations and Calls**
 - i. Initiate calls with one or more contacts.
 - 1. Transmission control: voice activated, Push To Talk (PTT), PTT lock, and other options
 - 2. The system will send emails to invited participants in a pre-schedule conversation.
 - a. The system will be able to manage invitees' responses and present to the call participants.
 - ii. Manage the state of participation in calls, including the ability to determine which calls should be delivered and/or rendered and with what priority.

-
- iii. View indication of which individuals are transmitting on a group / conference call.
 - iv. Receive data on incoming calls and emergency calls and notifications.
 - v. Send messages with multiple media types: voice, pictures, video, text, location, or other sensor data, including primary data and attached media.
 - vi. Replay messages and/or conversations including the ability to pause, rewind, go-back N seconds, fast forward, play more quickly (remove silence, play fast), catch up to live (play more quickly and seamlessly transition to the head of the conversation).
 - vii. Mark a conversation as completed, disabling the ability to append more messages to the conversation.
 - viii. Sort and organize messages by:
 - 1. Conversation
 - 2. Sender
 - 3. Receiver
 - 4. Priority
 - 5. Time
 - 6. Group
 - 7. Tags
 - 8. Etc.
 - ix. Scan transcriptions of conversations and messages provided by voice recognition software.
 - x. Listen to translations of conversations and messages provided by voice recognition and/or language translation software.
 - xi. Pre-record messages and or responses – e.g. a customer service rep or others may want to save time by pre-recording or otherwise preparing messages that they believe they will need in the future. These pre-recorded messages should include field substitution, allowing canned messages to have (say) a latitude and longitude or other sensor data inserted. This should include voice synthesis for delivery, as well as delivery as part of the data stream. On the receiving side, this data should be extractable.
 - xii. Listen to “read-only” channels which could be a broadcast of a talk, a pre-recorded media stream (a book), or a live radio broadcast with the ability to set the priority to “background”.
 - xiii. Associate RebelVox conversations with collaborative media management, including but not limited to:
 - 1. Presentations
 - 2. Videos
 - 3. Collaborative whiteboards
 - 4. Instant messaging
 - 5. Etc.

c. Message Tagging and Meta Data

- i. Type or input a subject or other tag or title before sending an initial or subsequent message or the ability to add the same after a message has been sent.
- ii. Tag messages with general purpose tags:

-
1. There are two tagging attribute classes:
 - a. User Defined Tags
 - i. These attributes are strings associated with a conversation, a message, or a position within a message.
 - b. System Defined Tags
 - i. These have actions or semantics in the system associated with them.
 - ii. Some tags may have inherent permission structures indicating whether information is private or protected in some fashion.
 - iii. Some examples of system defined attributes are:
 1. Importance – an attribute that overrides priorities and causes an emergency message to play
 2. Summary – this message summarizes previous content – there would be a user interface mechanism to skip to these entries
 3. Private – Can't be played on a speaker – headset only
 4. Acknowledgment Required – the system will manage recording and displaying acknowledgements
 5. Action Item
 6. Conclusions
 7. Transactions
 - iv. The summary tag is an example of an extended tag and the features that can be built into the system. The ability to define a message as a summary (conclusion, follow-up, summary, action items, comments, NDA covered, patentable) could support the following functionality:
 1. Catch up to live choice might show summaries and conclusions instead of the conversation.
 2. The tag may be attached to a specific point in time in the conversation.
 3. If you listen to a summary, everything before the summary is marked as read.
 4. Subscriber could register intent to only listen to summaries. System might only forward summary. One could forward permissions to summaries and not to the regular messages.
 5. Users can require acknowledgement just on summaries.
 6. Uses of summary will include: closing a business deal, rebuttal, billing reference, annotation, discoverable.
 7. Future User Interface may enable a summary to have a beginning and end and continue through several messages.

-
- 2. One tagging user interface would be a configurable “tagging palette” which will display most often used tags; the user would click on a specific tag during a message (originating or playing) to tag that message (or that moment in a message) with a tag.
 - iii. Filter messages of a conversation to only those with a specific tag.
 - iv. Future tagging features may include:
 - 1. Ability to set after the fact annotations.
 - 2. Ability to set a back reference to a previous conversation (or a point in time).
 - 3. Ability to send forward links to calls and messages with time markers
 - v. The system will support the integration of instant messaging along with voice messaging.
 - 1. Instant messages can be, if desired, associated with conversations and messages.
- d. Moving Messages in the System**
- i. Forward conversations and messages and allocate permissions.
 - 1. Forwarded messages may be read-only.
 - ii. Add a participant to a closed conversation, which causes it to migrate to their home server.
 - iii. Establish read-only content, which may have a configurable option to not send “radio” channels back to the home server.
- e. Presence and Status**
- i. Control presence, intention, and attention information.
 - ii. View the presence, intention, and attention information of other participants in the conversations.
 - iii. Tones may be used to indicate that others have left a conversation.
- f. Search tools:**
- i. Use analysis tools for legal, regulatory management, or other reporting.
 - ii. Search and tag existing messages and conversations after the fact.
 - iii. Create custom reusable tags.
- g. Manage Existing Conversations**
- i. Delete, archive, or purge (remove from client, but not from Home Server) conversations.
 - ii. Retrieve conversations and/or messages with an array of user-defined tags and attributes; e.g., “Action Item” or “Summary Message”.
- h. Rendering Controls**
- i. Apply independent gain control on a per sender basis.
 - ii. Set “catch-up” or replay mode to determine style of “fast” rendering for Catch Up to Live or listen more quickly.
 - iii. Replay will enable a user to choose a spot to begin the replay
 - iv. On accelerometer-based phones, the conversation can be paused when the phone is moved away from the user’s head (assuming no headset or earphones).

i. Data Management

- i. When re-synchronizing their client systems with data on their home server, the system will:
 1. Get all the meta data including priority
 2. Get the data in priority order
4. The MCMS system will enable organizations to manage their voice (and other media communications) in the following ways:
 - a. Permanently maintain an archive of messages organized by conversations in their home server.
 - i. The client device will enable purging but not archiving.
 - b. Transcribe or translate messages and conversations.
 - c. Configure the archiving of media to meet any regulatory requirements.
 - d. Retrieve conversations or portions of conversations by any of the system or user-defined attributes or tags enabling an organization to build references and search libraries of information and coordinate voice assets with other communication assets in a system that reflects business processes and requirements.
 - e. Maintain authentication and security controls over voice communications.

3.2.3 Performance Attributes of MCMS

The RebelVox Store and Stream will support the following performance attributes:

1. Manage message transmission as fundamentally “full-duplex” enabling any party to send a message at any time, even while another party is also sending a message. This eliminates the “one transmitter at a time” problem with traditional radio systems.
2. A configurable and highly parameterized user interface API that will enable a wide range of user device configurations including interfaces with highly flexible computer applications, general purpose phones, dedicated RebelVox devices, and very constrained “radio switch” devices.
3. Every device is technically capable of talking to all of the other devices on the network.
4. The network could be the Internet, but it could also be any number of other types of network including but not limited to a cell phone network, a private network, radio bands, etc.

3.2.4 RebelVox Multiple Conversation Management System - Concurrent (MCMS-C)

MCMS-C is an extension of MCMS and provides additional functionality:

1. The ability for a user to manage participation in multiple concurrent conversations in the manner of instant messaging where each conversation can have it’s own behavioral attributes and priorities.

-
- a. One or more conversations can be delivered in “real time” (that is, as fast as possible) according to priorities set by the user.
 - b. The user can shift priorities between conversations.
 - c. Conversations that are chosen for time shifting or which become time shifted through prioritization can be rendered with “catch up” options (play more quickly, remove silence) including catch up to live which transitions into the head of the live conversation.
2. Prioritization of incoming and outgoing messages through resolution of priorities for contacts, conversations, and messages.
 - a. Hierarchies of priorities will be analyzed to choose the most important message to be delivered or rendered next.
 - b. Sent messages may be delivered asynchronously if the recipient is currently listening to a higher priority conversation or message, is unavailable, or network conditions are poor at the time.
 - c. If the priorities and preferences are appropriate, a message will be heard or displayed in real time by the recipient.
 - d. If the priorities or preferences are set to defer the message, the message will be available for listening or viewing at a later time.

3.2.5 RebelVox Store and Stream

The RebelVox Store and Stream will support the following performance attributes:

1. The system will be designed from the ground up with a cryptographically sound architecture (multi-party end-to-end authentication and encryption):
 - i. Messages are secure through the network from sender’s device to receiver’s device.
 - ii. Non-alterable signatures on messages will ensure authenticity.
 - iii. If devices are lost, they may be completely erased or disabled remotely.
 - iv. Network will implement a hierarchy of trust – both for tactical nets to exclude unauthorized users and also to give network operators the ability to control message spam and other undesirable activities.
2. The system will manage message transmission as fundamentally “full-duplex” enabling any party to send a message at any time, even while another party is also sending a message. This eliminates the “one transmitter at a time” problem with traditional radio systems.
3. Through the proprietary Store and Stream Protocols, MCMS-C will provide both:
 - b. Timely delivery of “good enough” media,
 - c. Eventual delivery of “perfect” media,
 - d. Optimization of delivery speed and bandwidth usage to meet the user’s performance requirements while managing network quality and capacity (RebelVox Payload Optimization),
 - e. Persistence of media enabling the ability to pause, replay, fast-forward, and catch-up-to-live, retrieve archived media, and maintain history.

-
5. The RebelVox system will bridge the performance gap between TCP (which always delivers a perfect copy no matter how much latency is involved) and UDP (where data is delivered quickly with no guarantees of completeness) by adapting delivery of data (any media) that is “good enough” to meet the user’s moment by moment specifications (known by defined intentions and attention) with respect to timeliness and quality; to optimize usage of bandwidth even when network quality is poor; and to always eventually deliver a perfect copy of the data.
 6. The system will also adapt and optimize media delivery (voice, video, images, etc.) when network quality is poor by progressively delivering “low quality” to “high quality” media streams and progressively rendering data as it is able to be delivered.
 7. The system will enable users to leave a message (or continue a message) for other parties whether or not they have network connectivity of any kind (whether it is totally unavailable or of poor quality). In the case of poor quality, the message will be transmitted as fast as available bandwidth can be effectively used (slow delivery). In the case of no network, the message will be queued for transmission as soon as network is available (time-shifted delivery).
 8. The Store and Stream Protocol will coordinate the start and end of conversations with the appropriate client applications.
 9. One way to implement the communication portions of the MCMS system could be with a Jabber architecture.

3.2.6 RebelVox Distributed Services Architecture

- The RebelVox Distributed Services Architecture will support many system configurations including:
 - Many users over IP using one to few Home Servers, which we can call a RebelVox domain.
 - Many users using separate Home Servers with routing provided by RebelVox Access Servers.
 - RebelVox Gateways Servers, which will enable communications between RebelVox domains that do not have identical communication or security protocols.
 - RebelVox Gateway Clients, which will enable communications between a RebelVox domain and non-RebelVox devices.
- The RebelVox network will interface with alternate networks including legacy communication radios, the PSTN, and Internet allowing full two-way conversations between users any network and RebelVox users. Users on the legacy side would be limited to the capabilities of their existing equipment. However, RebelVox users would enjoy all of the benefits of the RebelVox system for messages originated on a legacy net.
- The RebelVox network may also be interfaced with the Public Switched Telephone Network. This would allow a seamless transition from existing PSTN-based devices like mobile phones to the RebelVox system.
- The RebelVox network will also interoperate with VoIP systems.

4 REBELVOX ARCHITECTURE OVERVIEW

4.1 High Level Architecture

The RebelVox Architecture has three key systems (plus a set of other standard services) that interact to deliver the suite of features and functions required for all of the RebelVox applications. While many architectural structures could be used, the following is one possible breakdown of these systems into components from which the systems can be constructed:

1. **MCMS-C Application** (including the MCMS Application features) which is comprised of the following:
 - a. MCMS-C database
 - b. MCMS-C Middleware API
 - i. UI Application Support
 - ii. Database Services
 - iii. MCMS-C Messaging and Signaling Services
 - iv. MCMS-C Services (for Store and Stream)
 - c. MCMS-C User Interfaces (optionally built by partners for specific platforms)
2. **Store and Stream RebelVox Protocols (RVP)** which manage delivery, receipt, and optimization of payload
 - a. Payload Management Services
 - i. PIMB
 - ii. PIMB Reader
 1. PIMB Reader Transmit (PRT)
 2. PIMB Reader Render (PRR)
 - iii. PIMB Writer
 1. PIMB Writer Net Receive (PWN)
 2. PIMB Writer Encoder Receive (PWE)
 - b. Communication Services
 - i. Transmit Methods: platform/protocol specific
 - ii. Receive Methods: platform/protocol specific
 - iii. Payload Quality and Content Manager (PQM)
 - c. Quality Services
 - i. Data Quality Manager

-
- ii. Network Quality Manager
 - iii. Packet Loss Compensation and Interpolator (PLC)
3. Other Services
- a. Media Services
 - i. Encoding Methods (data receiving)
 - ii. Rendering Methods (data sending)
 - b. Routing Services
 - c. Security and Encryption Services
 - d. Archiving/Retrieving Services
4. **RebelVox Distributed Services Architecture** which enables the construction from RebelVox components of a variety of services including:
- a. Clients – user interfaces, client middleware, and devices
 - b. Home Servers – permanent storage of payload and system meta-data, and archiving
 - c. Gateway Client– gateways to non-RebelVox systems including VoIP, PSTN, radios, etc.
 - d. Gateway Server– gateways between service domains including transcoding
 - e. Directory Server – directory to RebelVox users and their Home Servers and other routing services
 - f. Access Servers – distributed routing servers

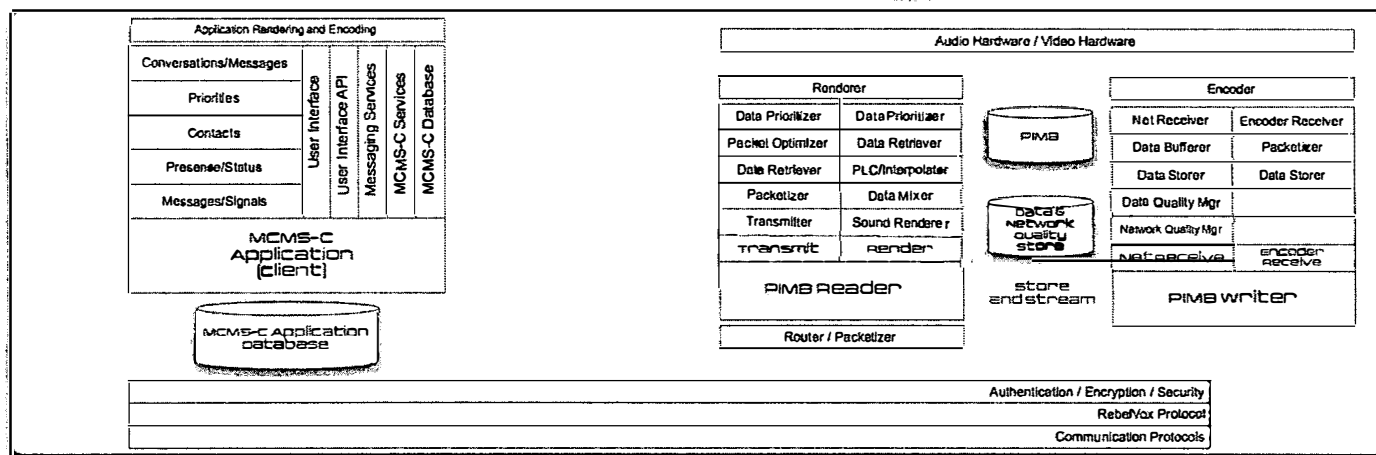
The RebelVox systems comprise a telephony application and as such will be independent of any particular communication or network protocol and will be able to run over IP (VoIP) and standard telephone protocols (PSTN, cellular networks) with a simple substitution layer for the transmit and receive functionality.

RebelVox Architecture

one possible solution for the RebelVox design

Devices running over IP

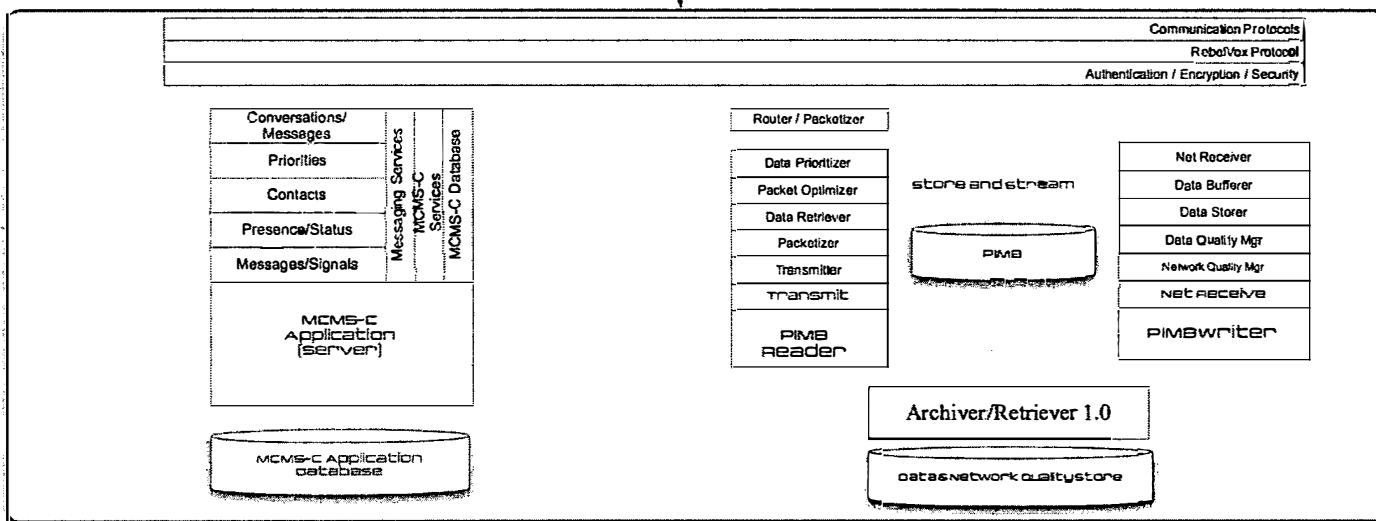
Dedicated RebelVox Devices



client



Home server



4.2 RebelVox Component Enumeration

REBELVOX PRODUCT REQUIREMENTS DOCUMENT

ERROR! BOOKMARK NOT DEFINED.

PR 4.2.1.1	MCMS-C APPLICATION	33
PR 4.2.1.1.1	MCMS-C DATABASE	33
PR 4.2.1.1.2	MCMS-C MIDDLEWARE API	34
PR 4.2.1.1.2.1	UI Application Support	34
PR 4.2.1.1.2.2	Database Services	35
PR 4.2.1.1.2.3	MCMS-C Messaging and Signaling Services	35
PR 4.2.1.1.2.4	MCMS-C Services (for Store and Stream)	36
PR 4.2.1.1.3	MCMS-C USER INTERFACES	36
PR 4.2.1.2	STORE AND STREAM REBELVOX PROTOCOLS (RVP)	36
PR 4.2.1.2.1	PAYLOAD MANAGEMENT SERVICES	37
PR 4.2.1.2.1.1	PIMB	38
PR 4.2.1.2.1.2	PIMB Reader	38
PR 4.2.1.2.1.3	PIMB Writer	41
PR 4.2.1.2.2	COMMUNICATION SERVICES	42
PR 4.2.1.2.2.1	Transmit Methods	42
PR 4.2.1.2.2.2	Receive Methods	42
PR 4.2.1.2.2.3	Payload Quality And Content Management (PQM) for Network Optimization	43
PR 4.2.1.2.3	QUALITY SERVICES	46
PR 4.2.1.2.3.1	Data Quality Manager (DQM)	47
PR 4.2.1.2.3.2	Network Quality Manager (NQM)	47
PR 4.2.1.2.3.3	Packet Loss Compensation and Interpolator (PLC)	47
PR 4.2.1.3	OTHER SERVICES	48
PR 4.2.1.3.1	MEDIA SERVICES	48
PR 4.2.1.3.1.1	Encoding Methods (data receiving)	48
PR 4.2.1.3.1.2	Rendering Methods (data sending)	48
PR 4.2.1.3.2	ROUTING SERVICES	48
PR 4.2.1.3.3	ARCHIVING/RETRIEVING SERVICES	49
PR 4.2.1.3.4	SECURITY AND ENCRYPTION SERVICES	50
PR 4.2.1.4	DISTRIBUTED SERVICES ARCHITECTURE	51
PR 4.2.1.4.1	CLIENTS	54
PR 4.2.1.4.2	HOME SERVERS	54
PR 4.2.1.4.3	GATEWAY CLIENT	55
PR 4.2.1.4.4	GATEWAY SERVER	56
PR 4.2.1.4.5	DIRECTORY SERVER	56
PR 4.2.1.4.6	ACCESS SERVER	56

4.2.1 RebelVox Component Product Requirements

Requirement	Description
PR Identifier	PR 4.2.1.1 MCMS-C Application
Priority	
Description	The Multiple Conversation Management System (MCMS-C) application includes the processing and data to manage all of the following: 1. Context and user data concerning media payload, 2. User interaction with conversations, 3. Presence and status data about participants, 4. Interaction with the conversations and messages, 5. Meta-data services for the Store and Stream functionality. (It does not include the processing or transmission of the payloads themselves, which is managed by the Store and Stream systems.) The application can be constructed with a multi-tier architecture including: • MCMS-C Database • MCMS-C Middleware API • UI Application Support – services for the user application • Database Services- services for the user application • MCMS-C Service (services for Store and Stream functionality) • MCMS-C User Interfaces (optionally built by partners for specific platforms)

Requirement	Description
PR Identifier	PR 4.2.1.1.1 MCMS-C Database
Priority	
Description	The MCMS-C database will support four instantiations: the client device, the Gateway Client, the Access Server, and the Home Server. While they will be essentially the same, there may be differences in meta data content and in the degree of “multi-tenancy”. (Client databases on client devices will hold only that user’s data; client databases on Gateway Clients will hold the client data for many clients in place of having it on their devices; Home Server databases will hold the data of all users who use this server as their home.) The MCMS-C database will store and manage the persistent meta data for the following entities in the system: • Contacts and Participants • Conversations and Messages (live and stored) • Default Priorities • Network routing and network statistics • Home Server In addition, the MCMS-C database will manage the moment-to-moment operational data of a user’s conversations, presence, and status, as well as that of all participants conversing with this user. • Conversations and Messages

- State of playback
- State of position in messages and conversation
- State of priorities and catch-up selections for each conversation
- Presence and status of all participants
- Network and data quality metrics for optimizing communications

Requirement	Description
PR Identifier	<i>PR 4.2.1.1.2 MCMS-C Middleware API</i>
Priority	
Description	The MCMS-C User Interface API is a critical architectural layer in the RebelVox system. The highly configurable RebelVox User Interface (RUI) API will isolate actual user interface implementations from the complexities of managing the application and provide a layer from which to construct both highly flexible and highly constrained user interfaces. This enables the RebelVox architecture to support user interfaces as varied as Flash-based windows applications, cellular phone devices, PSTN devices driven with tones, and physical radio communication interfaces. The RUI API will be a suite of behavior methods and a suite of configuration parameters enabling the deployment of custom interfaces by partners and licensees. It will support the following general purpose classes of methods: • UI Application Support • Database Services • MCMS-C Messaging and Signaling Services • MCMS-C Services (for Store and Stream)

Requirement	Description
PR Identifier	PR 4.2.1.1.2.1 UI Application Support
Priority	
Description	UI Application Support will be comprised of methods and functions that will support the operation of a user interface including basic create, read, update, and delete functionality and user interface data retrievals from the system. They will support an array of user screens and use cases that are required to interface with the system. The list of screens and interactions supported (not an exclusive enumeration) will enable the user to: • Log in • Manage Conversations ○ Create conversations ○ Join conversations ○ Monitor conversations ○ Manage listening status ○ Manage priorities ○ Review past conversations ○ Request archived conversations

- Manage Contacts
 - Discover contacts
 - Add contacts to conversations

See Section 5 in this document for further examples of user interface attributes.

Requirement	Description
PR Identifier	PR 4.2.1.1.2.2 Database Services
Priority	
Description	
Database services will include any and all methods required to access the database as necessary.	

Requirement	Description
PR Identifier	PR 4.2.1.1.2.3 MCMS-C Messaging and Signaling Services
Priority	
Description	There are messages, data, and signals that will require communications between all client applications and their Home Servers and any Access Servers that they may be using. MCMS-C Messaging and Signaling Services will provide these communications enabling the system to track and administer the state of the users and their communications. Parts of the shared meta data are critical to the operation of the RebelVox Payload and Quality Management protocol, which optimizes sender transmissions based on status of the receiver's intentions and attention. Following are a few examples (selected use cases) of the kind of coordination that will use messaging and signaling services: • To coordinate status of message transmission: ○ The server keeps track of what it has sent to a client ○ The client checks for missing portions of messages ○ The server sends and client maintains a "high water mark" to notice whether an entire message is missing (may be a sequence number per participant per conversation created by the "generating" client) • When joining a live conversation from the status of being disconnected, the client application has to get appropriate time stamps from originating client and/or server as it hits the network and the goes to the head (most current part) of the message. • When the communication network gets marginal for one participant in a live call, he (and other participants) will be notified that he is no longer on the live call. • Client will acknowledge receipt of key messages.

Requirement	Description
PR Identifier	PR 4.2.1.1.2.4 MCMS-C Services (for Store and Stream)
Priority	
Description	MCMS-C Services for Store and Stream will provide a suite of methods through which the meta data set by the application (user) can be accessed and used by the Store and Stream Methods, and through which metrics (data and network quality), acknowledgements, and activity can be recorded from the Store and Stream systems for maintaining the necessary status data for the user application. Some of the functionality support will include: • Maintaining the status of: ○ Message transmissions ○ Message acknowledgements ○ Network quality ○ Data quality ○ • User data concerning: ○ Participants and their various status states ○ Conversations ○ Messages and message time index pointers

Requirement	Description
PR Identifier	<i>PR 4.2.1.1.3 MCMS-C User Interfaces</i>
Priority	
Description	MCMS-C User Interfaces will be varied and highly configurable. They will vary from highly flexible windows interfaces, telephony devices, and highly constrained radio interfaces. RebelVox will be building the API that will enable RebelVox and other partners to build and deploy user interfaces across this wide range of platform architectures. This variety of user interfaces will be controlled by configuration of the system. Interfaces that support a high level of flexibility will have fewer things pre-configured in their system. Systems for radios, with their very constrained interfaces, will have highly pre-configured systems where options are limited and controls are streamlined.

Requirement	Description
PR Identifier	PR 4.2.1.2 Store and Stream RebelVox Protocols (RVP)
Priority	
Description	The Store and Stream RebelVox Protocols will support all facets of payload transmission, storage, and optimization.

It will use MCMS-C meta data to make decisions about what to transmit and how to transmit, and it will monitor packet transmission success and network quality while it is doing its work. It will run, in slightly different versions, on clients and servers.

The key advantages of Store and Stream are:

- It will allow RebelVox to render useful messages very quickly, while capturing the media persistently, enabling the system to store a perfect image of the sent message for any purpose.
- Through the sender/receiver coordination protocol, the system will be able to optimize the sending protocol based on the receivers attributes, enabling optimization effectiveness not otherwise available.

There are four key aspects of the Store and Stream Protocols:

- A specialized message storage structure that combines the functionality of a jitter buffer with the functionality of permanent storage
- A set of methods that receive messages and write to the storage structure
- A set of methods that read from the storage structure and transmit or render data
- Monitoring functions that create metrics about data and network quality

It includes the following components:

- Payload Management Services
 - PIMB (Persistent Infinite Message Store)
 - PIMB Reader
 - PIMB Reader Transmit
 - PIMB Reader Render
 - PIMB Writer
 - PIMB Writer Net Receiver
 - PIMB Writer Encoder Receiver
- Communication Services
 - Transmit Methods: platform/protocol specific
 - Receive Methods: platform/protocol specific
 - Payload Quality and Content Manager (PQM)
- Quality Services
 - Data Quality Manager/Network Quality Manager
 - Packet Loss Compensation and Interpolator (PLC)

Requirement	Description
PR Identifier	<i>PR 4.2.1.2.1 Payload Management Services</i>
Priority	
Description	

Requirement	Description
PR Identifier	PR 4.2.1.2.1.1 PIMB
Priority	
Description	The RebelVox PIMB is a set of data structures that have the following properties: • Persistence – The data in it is virtually available forever (with the ability to maintain a variety of retention rates). Archived data is never more than a request away. It will provide the persistence of file system storage and the random access of a database. • Infinite – The data available is infinite in scope through hidden requests to archived data. That is, a conversation can appear to be infinitely long. In the live system, the PIMB will function as an unconstrained message buffer in which the user can navigate forward and backward in time. • Buffered - It receives RebelVox message packets for “jitter” processing, rapid storage, scanning, and retrieval and stores them in a time-sequenced data structure. • It contains meta-about the data streams and message packets. The RebelVox PIMB is an evolution of an adaptive jitter buffer with the added attributes of persistence and a user selected rendering point. It will manage the “jitter” processing of incoming message packets, but instead of only rendering the media at a point in time, it will, in addition to rendering it as requested by the user, continue to update the buffer as need to create and maintain a perfect copy so that the system can always reproduce any message. Since, in the case of the RebelVox protocols, the message may not be rendered in real time, but time-shifted, the system will have often recaptured any missing packets before the message is rendered. Since the system does not merely render the data but stores it, additional functionality can be provided including, but not limited to: meta data about the messages (context, tagging), repair of the data, random access, translation and transcriptions, archiving, history maintenance, reconstructions, and optimized delivery. There are many possible implementations of the PIMB, which will be further developed. It could be implemented in a relational database of appropriate scope and capability. It is likely that this will be inefficient, requiring a purpose-built data store.

Requirement	Description
PR Identifier	PR 4.2.1.2.1.2 PIMB Reader
Priority	
Description	The PIMB Reader comprises the functions that read data from the <i>PIMB</i> structure. There are two basic processes that do this: • Reading the data that is ready to be transmitted to servers or other users through servers (on servers and clients.) • Reading the data that is ready to be rendered for the local client (only on clients.) There are some shared and some unique features in the two versions of the <i>PIMB Reader</i>. The details are covered in <i>PIMB Reader Transmit</i> and <i>PIMB Reader Render</i>.

Requirement	Description
PR Identifier	PR 4.2.1.2.1.2.1 PIMB Reader Transmit (PRT)
Priority	
Description	PIMB Reader Transmit (PRT) processes messages ready to be transmitted from a Client to the cloud (communication network) and from a Server into the cloud. Data Elements Required The information required is an array of data for queued target transmissions, which includes the following attributes: 1. User Data Per Conversation Per Other Participant a. Intention (to hear the data as soon as possible or to delay rendering; values – “now”, “time-shifted”, “catch-up-to-live”, “on hold”, “don’t bother”) b. Catch Up Mode (a configuration setting for this conversation determining how the user wants the system to “catch them up”: play faster, skip blanks, skip hesitations, play normal) c. TSD (Time Shift Delay; how far back is the Message Location Pointer from the Head (the most current part of a conversation) d. Importance (rating priority of message) 2. Payload Data a. Conversation Identifier b. Message Identifier c. Message Time Pointer (Time Index into Message) d. MTSD (the minimum time shift delay from the head of the message required by the system to guarantee usable quality; set by the <i>Data Quality Manager</i>) 3. Quality Data a. Metrics on the success of the network transmissions (Data Quality Metrics, Network Quality Metrics; see <i>DQM</i> and <i>NQM</i> documentation) Sequential Steps 1. Prioritize Data: Identify data to transmit: a. For conversations, choose the next message in the “priority” queue i. Determine if there is data ready to be transmitted b. Optimize the transmission by calling the <i>Payload Quality and Content Management Service</i> (PQM) 2. Retrieve the Data: Get the data records needed for identified transmissions 3. Create appropriate packets 4. Transmit the packets 5. Receive acknowledgement that the packets are being received 6. Record what was sent: The system will record that a message was sent (e.g., move the Message Time Pointer) Reference diagram is Section 9 Diagram 1.1.1.2.

Requirement	Description
PR Identifier	PR 4.2.1.2.1.2.2 PIMB Reader Render (PRR)
Priority	
Description	The PIMB Reader Render (PRR) processes messages ready to be rendered to the user's device (convert media for human consumption at a point in time.) Data Elements Required The information required is an array of data for queued target transmissions, which includes the following attributes: 1. User Data Per Conversation a. Intention (to hear the data as soon as possible or to delay rendering; values – “now”, “time-shifted”, “catch-up-to-live”, “on hold”, “don’t bother”) b. CatchUp Mode (a configuration setting for this conversation determining how the user wants the system to “catch them up”: play faster, skip blanks, play normal) c. TSD (Time Shift Delay; how far back is the Message Location Pointer from the Head (the most current part of a conversation) d. Importance (rating priority of message) 2. Payload Data a. Conversation Identifier b. Message Identifier c. Message Time Pointer (Time Index into Message) d. MTSD (the minimum time shift delay from the head of the message required by the system to guarantee usable quality; set by the Data Quality Manager) Sequential Steps 1. Prioritize Data: Identify data to render: a. For conversations choose the next message in the “priority” queue i. Determine if there is data to be rendered outside of its MTSD (minimum time shift delay, data of appropriate quality to render usefully) ii. Determine if there are multiple message streams in this conversation for the same time slice b. Determine how much data to render depending on the user interest, intention, TSD c. Determine how to render based on user's Catch Up to Live Intentions (speed, strip out silence, etc.) 2. Retrieve the Data: Get the data records needed for rendering 3. Do Packet Loss/Compensation and Interpolation: Process each of the message streams with a state of the art packet loss compensation and interpolation algorithms to manage any existing gaps in the packet sequence. 4. Mix the Data: Mix the message streams as needed with options to mix streams (according to time overlap), hear single streams (one speaker), or some subset. 5. Manage the Sound Stream: Generate the sound stream to the rendering device. 6. Record what was rendered: The system will record that a message was sent and can optionally record a separate data structure of the sound stream that was rendered. NOTES: Other rendering solutions:

- The system may implement a Render/Transmit algorithm, which would provide optimization for continuing with the current message and minimizing context switches. The system will make decisions based on message readiness (quantity and quality) and chatter load (the density of context switching in the user's message queue).
- The system may render tones to indicate context switching.
- In the same way that the system mixes streams with the same time slice on the same conversation, the system would be able to mix multiple conversation sources for any given point in time.

Reference diagram is Section 9 Diagram 1.1.3.1.

Requirement	Description
PR Identifier	PR 4.2.1.2.1.3 PIMB Writer
Priority	
Description	The PIMB Writer comprises the functions that write data to the PIMB structure. There are two basic processes that do this: • Receiving and writing the data from a sound capturing device (runs on Clients) • Receiving and writing data that is coming over the network (runs on the Clients and Servers) There are some shared and some unique features in the two versions of the PIMB Writer. The details are covered in PIMB Writer Net Receiver and PIMB Writer Encoder Receiver.

Requirement	Description
PR Identifier	PR 4.2.1.2.1.3.1 PIMB Writer Net Receive (PWN)
Priority	
Description	PIMB Writer Net Receiver (PWN) receives data from the network (from Home Servers, Access Servers, or other Clients with server functionality.) Sequential Steps • Receive the RebelVox Packet (RVP) from a RebelVox service (self describing packets) • Manage an adaptive jitter buffer process and store the messages in time-sequenced order in the PIMB transforming the packets into a time-sequenced store of media data. • Execute <i>Data and Network Quality Management</i> process which notes and requests missing packets. Notes: • PWN may have multi-threaded communications with the network receive process.

Reference diagram is Section 9 Diagram 1.1.3.2.

Requirement	Description
PR Identifier	PR 4.2.1.2.1.3.2 PIMB Writer Encoder Receive (PWE)
Priority	
Description	
PIMB Writer Encoder Receiver (PWE) receives data from the client's voice encoding process.	
Sequential Steps	
• Receive signals from the device's encoding services. • Create RebelVox Indexed Media Payload with appropriate identifiers with the received data. • Store the messages in order in the PIMB transforming the packets into a time-sequenced store of media data.	
Reference diagram is Section 9 Diagram 1.1.1.1.	

Requirement	Description
PR Identifier	<i>PR 4.2.1.2.2 Communication Services</i>
Priority	
Description	
The RebelVox Protocol will use standard and appropriate communication services for each deployment platform.	

Requirement	Description
PR Identifier	PR 4.2.1.2.2.1 Transmit Methods
Priority	
Description	
Transmit methods will be platform and protocol specific.	

Requirement	Description
PR Identifier	PR 4.2.1.2.2.2 Receive Methods
Priority	
Description	
Receive methods will be platform and protocol specific.	

Requirement	Description
PR Identifier	PR 4.2.1.2.2.3 Payload Quality And Content Management (PQM) for Network Optimization
Priority	
Description	

Summary

The task of the PQM is the timely delivery of “good enough” data quality for urgent messages and “perfect” delivery of all messages over time while minimizing real time bandwidth. This is a dynamic trade-off between more efficient bandwidth usage and currency (keeping the media up-to-the-moment), which the system enables by maintaining information about the intentions of the user with respect to currency and the nature of the network quality and rate. The RebelVox system accomplishes this with a cooperative management protocol between the communication pairs including information about the receiver’s requirements and network quality (packet loss). The PQM uses this information to dynamically adjust the constitution of packet contents (ratio of payload to headers; content of payload) and payload quality (reducing percentage delivered by re-packetizing) according to the control metrics maintained by other processes in the system.

There are four key goals of the optimization:

1. Timely delivery of an adequate copy of time sensitive data (“soon enough and good enough”);
2. Optimize all available bandwidth;
 - a. Use more if available;
 - b. Minimizing overall bandwidth consumed when bandwidth or network quality is degraded or constrained;
3. Ultimately delivering a perfect copy.
4. Maintain the ability for the *sender* to adapt transmission frequency, payload contents, and payload size as semantic context and network conditions of the *receiver* change.

The system has two key processes that enable the dynamic management of the needed information:

1. Manage a set of metrics (Network Quality Ratings) that determine an optimal bit rate for each sender/receiver pair.
2. Manage a shared semantic context that models the receiver’s usage intent with respect to the data transmission.

Here are some of the ways that one can optimize network transmissions when a system cooperatively manages the meta data that surrounds that transmission. The basics techniques are each applied as is meaningful (see logical breakdown):

1. Build packets that meet the optimal bit rate, and also optimize the content and size of the payload with one or more of the following options:
 - a. Send payload as fast as possible (e.g., 20 milliseconds every 20 milliseconds)
 - b. Increase payload in a packet to optimize the ratio of payload to header and use bandwidth more efficiently.
 - c. Once next-required media has been identified, if there is extra bandwidth, bundle other payloads

- going to the same destination with the needed payload.
- d. Reduce payload quality of time-domain data by repacketizing with a lower percentage of the data.
- e. Force a conversation out of live mode and into message mode.
- f. Curtail the transmission of non-time-sensitive messages

Description

Data Elements

The payload quality management process will use several quality management ratings (managed elsewhere) as well as system and user meta-data to optimize network communications.

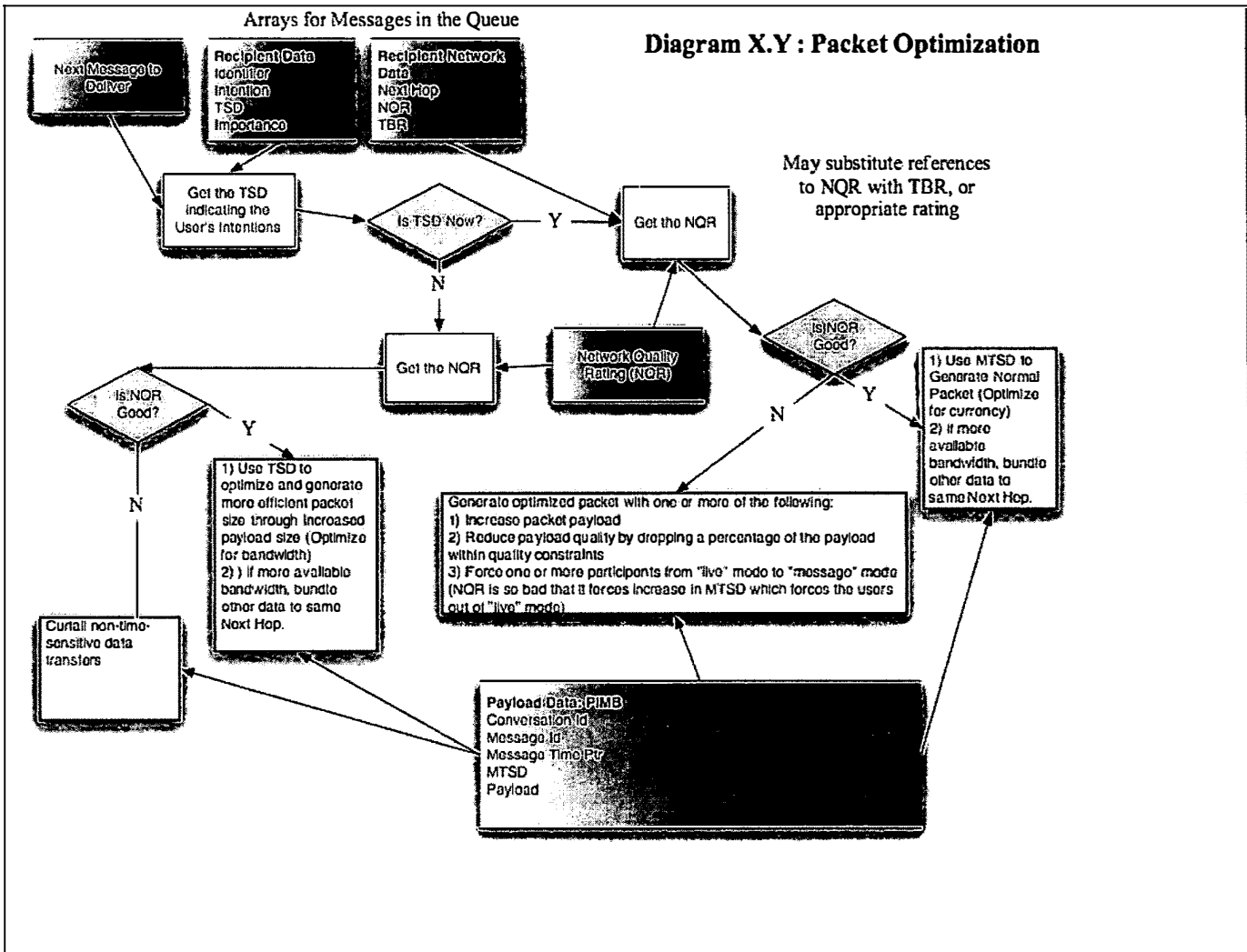
The information required is a collection of data for queued target transmissions, which includes the following attributes:

3. Recipient Data
 - b. Identifier (transmission target)
 - c. Intention (to hear the data as soon as possible or to delay rendering; values – “now”, “time-shifted”, “catch-up-to-live”, “on hold”, “don’t bother”)
 - d. TSD (Time Shift Delay; how far back is the Message Location Pointer from the Head (the most current part of a conversation)
 - e. Importance (rating of priority of message [to be updated as priority documentation is wrapped up])
4. Recipient’s Network Data
 - f. Next Hop (the direct communication target on route to the recipient; the identity of the “pair” to which the NQR data applies)
 - g. NQR (Network Quality Rating established by the NQM [not sure we need this in addition to TBR, needs review])
 - h. TBR (Target Bit Rate established by the NQM)
5. Payload Data
 - i. Conversation Identifier
 - j. Message Identifier
 - k. Message Time Pointer (Time Index into Message)
 - l. MTSD (the minimum time shift delay from the head of the message required by the system to guarantee usable quality; set by the Data Quality Manager)

Logical Processing

1. This process is invoked with the identity of the next payload to be transmitted.
2. Using the meta data enumerated above, evaluate as follows:
 - a. For time sensitive payloads (the user’s intention is to listen to the media in real time):
 - i. If NQM is Good and TBR is High (we can meet currency requirements)
 1. The process will package and send RebelVox packets of the Standard Packet Size (SPS, relatively small packets configurable according to optimal network bandwidth) outside of the MTSD. For live media the target should be to send 20 milliseconds of payload every 20 milliseconds (20in20).
 2. If TBR is higher than necessary to support the 20in20 target, the process will identify and prioritize payloads to the same Next Hop (by Intention, Priority,

-
- Importance, etc.) and select enough to bundle into a packet that will meet the TBR.
- ii. If network quality is poor (NQR is Bad and/or TBR too low to meet the 20in20 target) the process will use some or all of the following techniques to adjust the payload efficiencies, improving our use of the available bandwidth:
 - 1. Calculate a larger time-slice of payload than SPS thereby creating efficiencies in network traffic by improving the ratio of payload to the packet header data. (Trading currency for TBR.)
 - 2. Reduce payload quality by dropping a percentage of the payload being used within constraints to keep the payload “useful”. The rest of the data will be sent when the network bandwidth is available. (Trading “perfect” for TBR.)
 - 3. Force the call out of “live” mode for the participants who have poor bandwidth connectivity. This may drop everyone on the call into a message mode. (Abandoning currency.)
 - b. For non-time sensitive payloads (due to a user’s intention to listen to the data with a Time Shift Delay (TSD)):
 - i. If NQR is good and/or TBR is High:
 - 1. The process will calculate a larger time-slice than SPS thereby creating efficiencies in network traffic by improving the ratio of payload to the packet header data. (Currency is not important.)
 - 2. If more bandwidth is available, the process will identify and prioritize payloads to the same Next Hop (by Intention, Priority, Importance, etc.) and select enough to bundle into a packet that will meet the TBR.
 - ii. If NQR is not good:
 - 1. The process will preserve bandwidth by not sending any non-time-sensitive payloads.



Requirement	Description
PR Identifier	<i>PR 4.2.1.2.3 Quality Services</i>
Priority	
Description	There are two quality management services: • Data Quality Manager • Network Quality Manager Reference diagram is Section 8 Diagram 1.1.3.2.3.

Requirement	Description
PR Identifier	PR 4.2.1.2.3.1 Data Quality Manager (DQM)
Priority	
Description	The Data Quality Manager looks for missing packets and maintains metrics on data quality: - Observe data packets as stored in the PIMB - Note missing packets perhaps by monitoring a checksum. - Note flawed packets. - Request missing packets from the sending RebelVox service. - Update metrics on receive data quality. Reference diagram is Section 9 Diagram 1.1.3.2.3.1.

Requirement	Description
PR Identifier	PR 4.2.1.2.3.2 Network Quality Manager (NQM)
Priority	
Description	The Network Quality Manager will measure network quality and maintain the necessary data and metrics to optimize outgoing data transmissions. - Maintain the following network data: - For each transmission target: - The “next-hop” identifier (the direct transmission network service target) - The Maximum Available Bit Rate to that target endpoint (MABR, same use as TBR) - The transmission target’s (target participant’s) TSD, MTSD, Intention, and NQMM (network quality manager metrics) Reference diagram is Section 9 Diagram 1.1.3.2.3.2.

Requirement	Description
PR Identifier	PR 4.2.1.2.3.3 Packet Loss Compensation and Interpolator (PLC)
Priority	
Description	The RebelVox Protocol will use the most appropriate algorithm according to the codec and platform selected.

Requirement	Description
PR Identifier	PR 4.2.1.3 Other Services
Priority	
Description	
This includes other services needed to manage payload delivery.	

Requirement	Description
PR Identifier	<i>PR 4.2.1.3.1 Media Services</i>
Priority	
Description	
The RebelVox Protocol will use the most appropriate media services for the devices deployed.	

Requirement	Description
PR Identifier	PR 4.2.1.3.1.1 Encoding Methods (data receiving)
Priority	
Description	
Encoding methods will be device and platform specific.	
Notes: Threshold for voice recording should be very sensitive.	

Requirement	Description
PR Identifier	PR 4.2.1.3.1.2 Rendering Methods (data sending)
Priority	
Description	
Rendering methods will be device and platform specific.	

Requirement	Description
PR Identifier	<i>PR 4.2.1.3.2 Routing Services</i>
Priority	
Description	
Description The first iteration of the routing protocol will be an IP-based architecture, which will support 1 to few servers deployed in 1 to few locations connecting to a large array of clients that communicate solely with their Home	

Server. In the Internet case, the router will not have to choose where to send packets.

- Future Routing architectures will support federated or meshed architectures of services deployed in peer-to-peer and hierarchical structures where many clients will provide routing services in addition to that provided by dedicated servers. (See *Distributed Server Architecture*.)

Functionality

- Route messages to appropriate servers and clients based on:
 - The Network Quality Metrics for each route as determined by the *Network Quality Manager*; re-routing when necessary.
 - Establish new routes as necessary to improve performance and/or quality.

Requirement	Description
PR Identifier	<i>PR 4.2.1.3.3 Archiving/Retrieving Services</i>
Priority	
Description	

The Archiving/Retrieving Services will provide methods to purge data from client storage, archive data from Home Server storage, and move data to secondary storage to support the following capabilities:

- A configurable level of persistence of the media providing the possibility of infinite time-shifting or selected long-term storage.
- Conversation reconstructions:
 - Aligning multiple conversations along timelines
 - Forensics/history
 - Regulatory management and compliance
 - After-action reviews; post mortems
- Data transformations
 - Voice recognition and transcription
 - Translations
- Subscriptions (saving access to conversations)
- Create a searchable information resource of voice messaging through tagging and transcription

Special Functionality

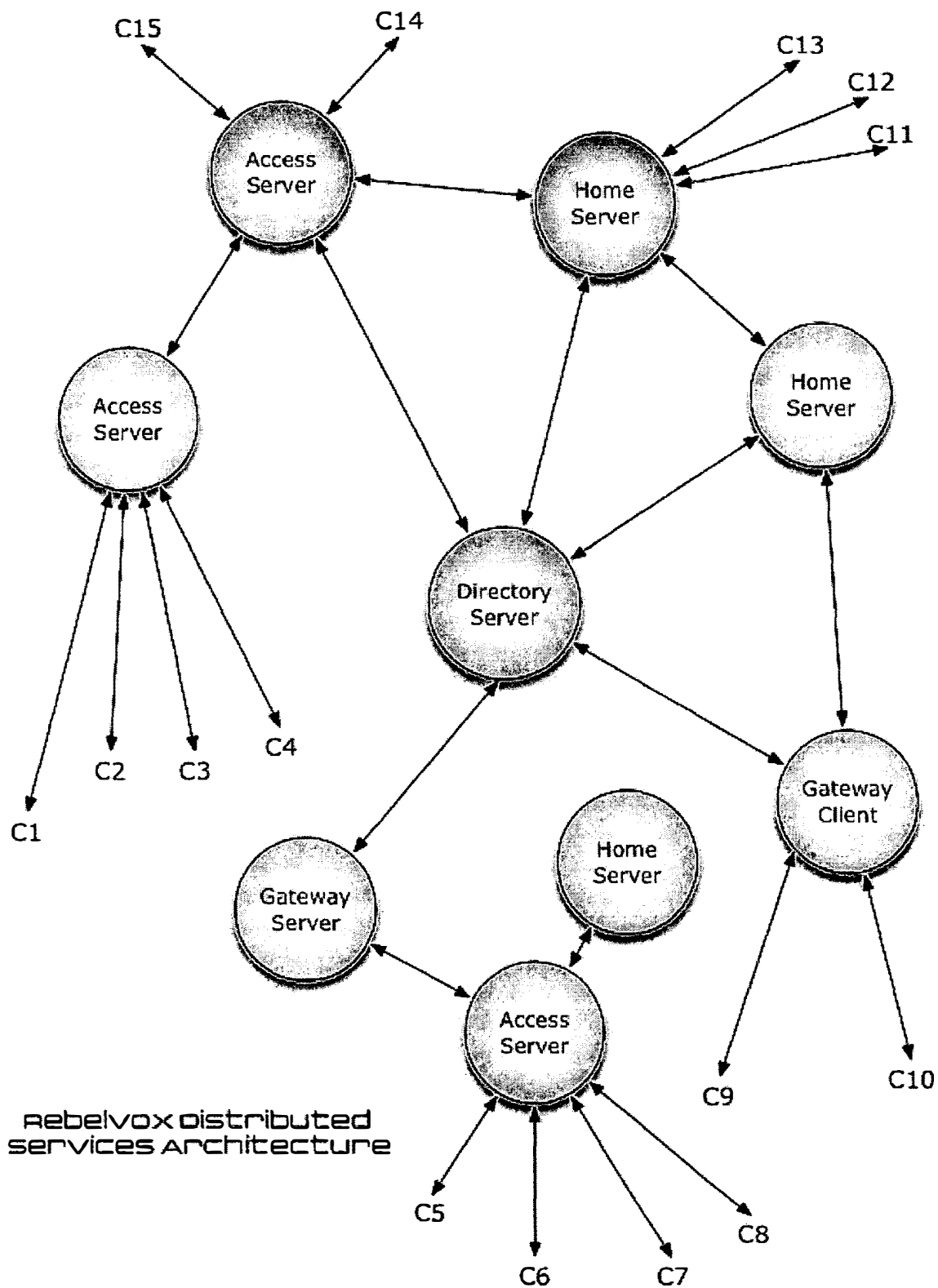
- The system will enable and control reasonable duplicate storage of conversations (multiple users, distributed home servers, etc.)
- The system will enable Home Server migration to the user's closest persistent storage.

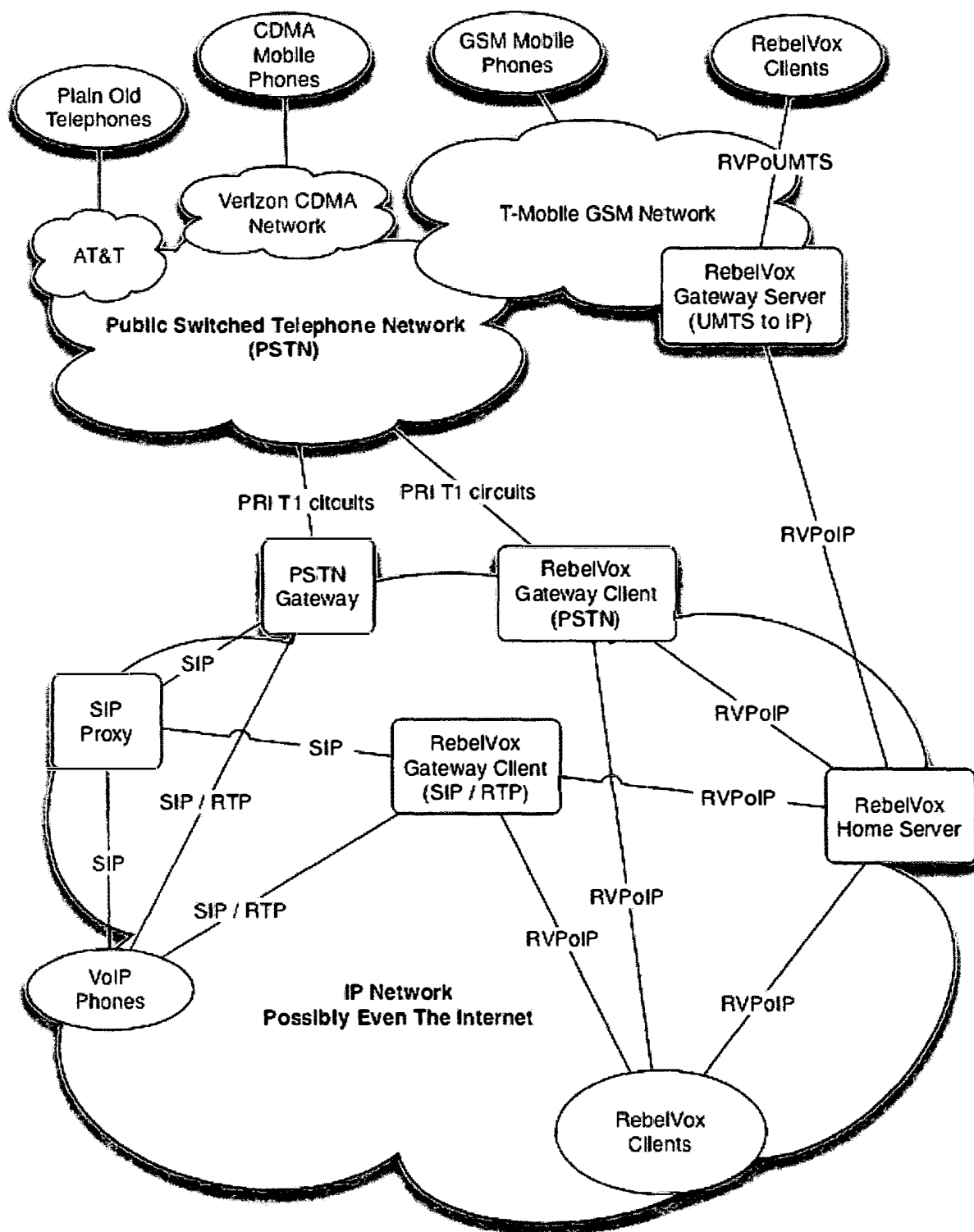
Example archive configurations:

- For the Home Servers: 30 day rolling purge;
- For the Client device: 3 days on the phone
- For a PC based application: – the choice to save or purge with a default pattern of managing purging as per DVR;
- Implementation of “Don't Save” options for each conversation where saving the voice transmission is a violation of law, privacy regulations, security measures, or the data is unnecessary

Requirement	Description
PR Identifier	<i>PR 4.2.1.3.4 Security and Encryption Services</i>
Priority	
Description	The RebelVox system supports a robust security framework that delivers end-to-end encryption, authentication, and non-repudiation. The specific security features that are employed by the system will be configurable based on the requirements of the deployment environment. A Public Key Infrastructure (PKI) will be employed to issue certificates to all clients and all servers. Using public key cryptographic techniques, senders will encrypt all data such that only the intended recipients can decrypt it. The originator and recipient of the message perform the cryptographic operations. Any other network elements that the message travels through will not be able to recover the message contents. This means that the RebelVox system can operate over an un-trusted and otherwise insecure network while still preserving the privacy and authenticity of messages. In many cases it is beneficial for the RebelVox Home Server to have access to the message contents, to do voice recognition for example. In this case, the Home Server will be added to the list of recipients by the senders when participating in a conversation. Access Servers do not generally have access to the unencrypted message contents. A similar situation arises when the users do not need or want their messages to be stored on the Home Server in an unencrypted form. In this case, a message encrypted for the ultimate recipient can be encapsulated within another message, which is encrypted for the Server. The Server (Access or Home) would then have access to enough of the message metadata to manage the transmission to the message's recipients, but the message payloads themselves would not be visible to the Server. Use of public key cryptography provides a natural mechanism for "signing" messages such that their originator can be verified. This digital signature provides all parties in a conversation a degree of non-repudiation on a per-message basis. Thus, RebelVox users can be reasonably certain that messages from other RebelVox users were indeed sent by that user. At the very least, the messages can be said to have originated from that user's device. A robust and scalable authentication scheme also follows naturally from the PKI. A hierarchy of trust is maintained, simplifying the management of large networks. For example, an international corporate network with headquarters in Dubai might have authorized their New York office to add new users in the United States and their Paris office to add new users in the European Union. The New York office could then delegate that authority further down to their San Francisco office for West Coast users. This spreads out the task of adding new users, but links all users of the system with a common trust system. When a Paris user gets a message from a West Coast user, the Paris user would see that the West Coast user is trusted by the San Francisco office, which is trusted by the New York office, which is trusted by the Dubai office. The Paris office trusts the Dubai office, and the Paris user trusts the Paris office. By extension then, the Paris user has some amount of trust of the West Coast user.

Requirement	Description
PR Identifier	PR 4.2.1.4 Distributed Services Architecture
Priority	
Description	The RebelVox Distributed Server Architecture provides for the evolution of the platform from a simple IP platform (single server, many clients) to a distributed meshed network of servers of different types include the possibility of client devices that can provide services. Each component of the architecture is constructed from a particular set of the RebelVox functional components. The components of the architecture are as follows: ○ Clients ○ Home Servers ○ Gateway Clients ○ Gateway Servers ○ Access Servers ○ Directory Servers





Requirement	Description
PR Identifier	<i>PR 4.2.1.4.1 Clients</i>
Priority	
Description	RebelVox Clients are used by RebelVox users. They are: ○ Devices (e.g., phones, radios) that run RebelVox MCMS or MCMS-C, or ○ The MCMS or MCMS-C applications running on general-purpose computers. ○ An application built on the RV UI API. Clients do the following things: ○ Run the MCMS or MCMS-C application supporting the ability to engage in multiple conversation management. ○ Run appropriate portions of Store and Stream: ○ PIMB Reader Render ○ PIMB Reader Transmit ○ PIMB Writer Encoder Receive ○ PIMB Writer Net Receive ○ Capture and render media (voice in the base case). ○ Store media for transmission, rendering, and retrieval inside a PIMB. ○ Maintain the route to the most appropriate Access server which has the route to its Home Server and all other transmission targets ○ Transmit outgoing messages. ○ Receive incoming messages.

Requirement	Description
PR Identifier	<i>PR 4.2.1.4.2 Home Servers</i>
Priority	
Description	A RebelVox Home server lives on a network and does the following: • Receives messages from Clients. • Stores media for transmission and retrieval inside a PIMB. • Runs appropriate portions of Store and Stream: ○ PIMB Reader Transmit ○ PIMB Writer Net Receive • Provides RebelVox Routing (RV Routing (finds Home Servers, Access Servers, and Clients); servers are found using DHCP). • Archives data into permanent random access storage per its configuration parameters. • Manages Home Server migrations.

Requirement	Description
PR Identifier	<i>PR 4.2.1.4.3 Gateway Client</i>
Priority	
Description	RebelVox Gateway Clients provide intermediation between RebelVox Servers and user devices that are not running MCMS (or MCMS-C). They do the following things: ○ Run a multi-tenant MCMS or MCMS-C application for the participants who do not have RebelVox devices, supporting the ability to engage in multiple conversation management. ○ Run appropriate portions of Store and Stream: ○ PIMB Reader Render ○ PIMB Reader Transmit ○ PIMB Writer Encoder Receive ○ PIMB Writer Net Receive ○ Capture and render media (voice in the base case). This is a special case of voice capture and render depending on the network protocols required by the gateway clients. ○ Store media for transmission, rendering, and retrieval inside a PIMB. ○ Record the routes to needed Home Servers. ○ Transmit outgoing messages. ○ Receive incoming messages. ○ Knows the route to the Home Server for each client. ○ Identifies the appropriate Access Server. Some examples of uses of Gateway Client connectivity: • PSTN – enabling standard phone users to use RebelVox • Tactical Radio – Tactical radios will have a dedicated RebelVox solution, but enabling radio within an RebelVox IP-based system will use a gateway • Video servers The PSTN Stand-alone Gateway Client could provide the following behaviors for telephone and cellular customers who are unable to run the MCMS application: • Multiple conversation management • Multi-party conference calling • Standing conversations • Voice messaging • Message listening management: rewind, pause, catch up to live, skip to head • Optional web interface for call management: establishing calls, managing calls, managing archived calls

Requirement	Description
PR Identifier	<i>PR 4.2.1.4.4 Gateway Server</i>
Priority	
Description	RebelVox Gateway Servers bridge between RebelVox domains that have mapping or re-encapsulation requirements. They will run the following RebelVox functions: • Where domains have different packetization requirements, the system will re-packetize. • Where a domain has no native IP connectivity, it will establish VPN into the network providing unified mediating access through VPNs. • As required, special security and cryptography may be implemented in addition to standard RebelVox security. • As required, a Gateway Server may provide transcoding services to allow for network translation and/or codec translation.

Requirement	Description
PR Identifier	<i>PR 4.2.1.4.5 Directory Server</i>
Priority	
Description	RebelVox Directory Servers provide a mechanism for authorization of users, routing, and billing.

Requirement	Description
PR Identifier	<i>PR 4.2.1.4.6 Access Server</i>
Priority	
Description	RebelVox Access Servers provide a mesh-architecture to manage communications in highly distributed systems. It will proxy access to Home Servers and/or Directory Servers when communication to such is degraded or unavailable, and providing current optimal routing information. They will run the following RebelVox functions: • Receives messages from Clients. • Stores media for transmission and retrieval inside a PIMB. • Runs appropriate portions of Store and Stream: ○ PIMB Reader Transmit ○ PIMB Writer Net Receive • Provides Rebel Vox Routing (RV Routing (finds Home Servers, Access Servers, and Clients); servers are found using DHCP). • A mapping function, which will maintain the network location and most efficient path between clients and servers. • Directory service functionality for a local subset of users.

-
- Store and Stream functionality (PIMB and its services) for a local subset of users.

5 REBELVOX USER INTERFACE

5.1 Interface Components

The RebelVox User Interface API will support a wide range of behaviors. Any given user interface will configure or restrict some activities (e.g., available “channels” or conversations, fixed set of participants, ability to set up a conversation). The outline included here represents some of the user behaviors and information that will be available from the RV UIAPI.

List of API Use Cases (may be screens)

- Multiple Conversation Management – Home Screen
- Conversations
 - Overview
 - Participants Presence/Participation
 1. Intention
 2. Where relative to head
 3. Setting catch-up preference
 - Payload Meta data w/navigation
 - Setup
 - Title
 - Scheduling
 - Participants (summary)
 - Priority
- Contacts
 - Add contacts
 - Find contacts
 - Delete contacts
 - Group contacts
 - Prioritize contacts and groups
- Action Controls
 - Play
 - Rewind
 - Catch Up to Live
 - Pause
 - Change conversation
- Setup
 - Device
 - Network
 - UI
 - Etc.
- Login/Security Screen

5.2 Interface Features and Behaviors

The following is a set of information, actions, and behaviors that will be required. They are grouped here by “screen” as a convenient way of grouping them, though there are many ways that the system can implement them.

Information Useful on all Screens

- Battery Indicator
- Connectivity Indicator
- Clock
- Microphone/Transmitter Status
- Transmission Status/Sync Status
- Listening/Playing Status
- Priority Messages Needing Attention
- Missed Call Invitations/Messages
- Text Messages/Emails

Information for a Conversation Overview Screen

- Name of conversation or list of participants
- Status
 - Is Call “live”?
 - Live call
 - Standing Call with activity
 - Inactive
 - Am I at the head?
 - Am I actually listening?
 - Am I transmitting? Is my microphone on?
 - Priority
 - Duration
 - Length of a call?
 - Time behind the head?
 - Unlistened to/outstanding messages
 - Priority
 - Normal?
 - Participants presence/status
 - Meta Data with Navigation
 - Attributes designation and designators
 - Setup Conversation
 - Title
 - Scheduling
 - Participants
 - Full
 - Summary Only
 - Summary with full authorization

Information for Multiple Conversation Summaries

-
- Conversation name/identifier
 - Live/Active Standing/Inactive
 - Listening
 - At head
 - Time shifted
 - Transmitting/Microphone On
 - Priority/Attributes
 - Indication of What I've Missed

Navigation Features

- DVR Style back/forward/fast per conversation
- Message navigation per IM message scanning
- Time indicators
- Zooming
- Dragging conversations up and down changes the priority
- Hang Up All
- "Home" Button
- Isolate going through voicemail in priority order

6 PRODUCT INFRASTRUCTURE

6.1 1.1 How RebelVox Relates to Telecommunication Protocols

The RebelVox system can run on a variety of different communication platforms. The communications between RebelVox components will be layered over standard networking protocols like the Internet or fixed and mobile telephony networks.

Using proprietary techniques, the RebelVox protocol will provide substantial enhancements to performance and control beyond the capabilities of these standard protocols.

6.2 1.1.1 Network Communication

RebelVox is designed around the concept of moving many small units of information between different components in the system. The components of this system may be interconnected by nearly all contemporary networking technologies. The individual message units are called RebelVox Packets or RVPs. The RebelVox system has been designed and optimized for IPv4 networks such as the Internet, but other types of networks may be supported as well. For the purposes of this document, the term “IP” should be taken to mean “IPv4 or IPv6” (IPv4 and IPv6 are the current network layer protocols for the Internet).

A RebelVox Packet (RVP) is an 8-bit encoded message that may contain a variety of different types of information. RVPs vary in size, depending on their function and payload, but to the underlying network layers, all RVPs appear to be the same kind of data.

Much of the RebelVox system is designed to accommodate some sort of packet-based network. In these types of systems, a hierarchy of protocols is layered together, each lower-level encapsulating the entire packet above it. Every packet that is sent on the network carries a certain amount of “overhead” in its headers. An efficiency / timeliness tradeoff can thus be made by adjusting the size of the payload, and RebelVox is designed to make such a tradeoff.

On a network whose native mode is more of a “stream” or “circuit” based model, the RebelVox system will still layer its messages into a sequence of RVPs. Several of the packet construction parameters that the RV system uses may be adjusted to values that exploit the lack of per-packet overhead. The RVPs will be placed sequentially in the circuit, possibly with some sort of spacer or identifying code that helps the other end find the boundaries of each RVP.

The RebelVox architecture generally assumes that each packet will be delivered intact or not at all. Some provisions are made to ensure the integrity of each packet, but the underlying network should try to avoid delivering corrupted data. Nearly all popular networking technologies already follow this model, but we imagine that some future packet radio networks might deliver corrupted packets due to RF interference.

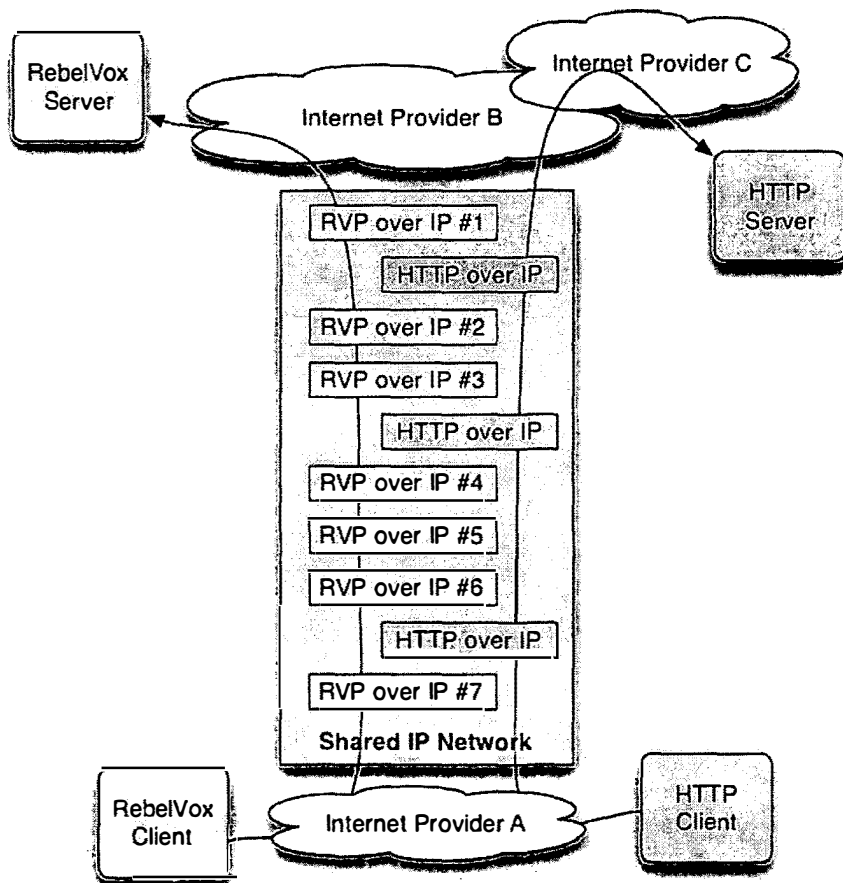
There are a few fundamental network operations that the RebelVox system requires:

- 1) Resolve RebelVox Address – for a given pre-named RebelVox node, we need some way to address messages so that the underlying network will deliver the RVP to the correct location. For IP networking, this address is an IP Address, a 32-bit number that uniquely identifies a host within the network. For other networking technologies, this could be some other type of identifier. IP networks use the Domain Name System (DNS) to resolve human-readable names into IP addresses, and the

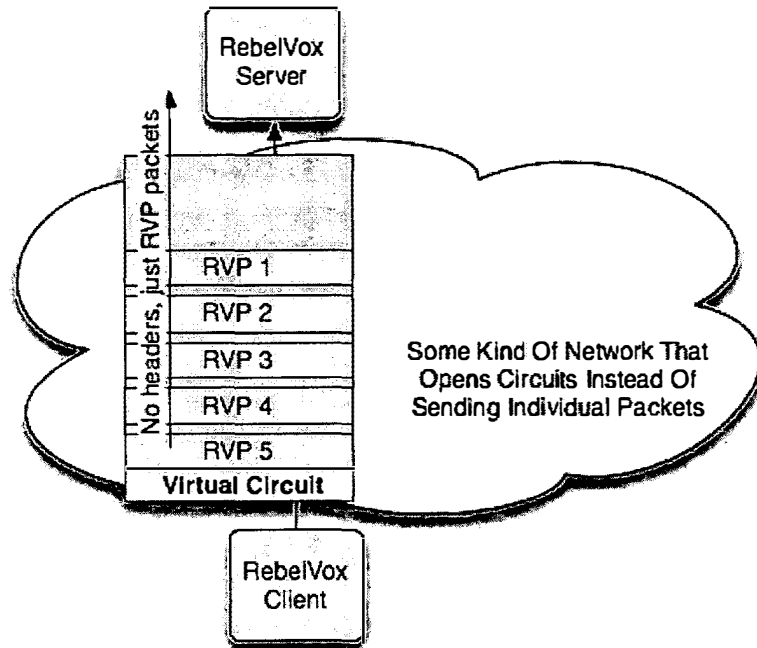
-
- Address Resolution Protocol (ARP) to resolve IP addresses into physical addresses. RebelVox will use both of those systems on an IP network, or analogous components on a different kind of network.
- 2) Send RebelVox Packet – Send a unit of data (packet) to a node on the network based on its pre-determined address. This packet may or may not be delivered to the remote party if the underlying network is unable to deliver it. For purely packet-based networks such as UDP/IP, this is a very straightforward operation. For a session-based network such as TCP/IP or any kind of circuit-switch network, sending a packet might require that a session or circuit be established first.
 - 3) Receive RebelVox Packet – Receive a packet from the network. For UDP/IP system, this is very straightforward. For session / circuit-based networks, this will require some sort of session to be established before any RVPs may be received.
 - 4) Manage MTU – determine the Maximum Transmission Unit, that is the maximum size of a single message that may be sent on the network. For packet-based networks, the underlying network imposes the MTU. For circuit-switched networks, the MTU is merely a tunable parameter for network efficiency and performance.

The general reason for using a packet network in the first place is that multiple different parties can share a single communications channel. Packets are sent sequentially based on various schemes that the routers enforce. In the case of RVP over IP, these RVP packets might need to share the overall path to their destination with other, non-RebelVox traffic.

The following diagram illustrates a RebelVox Client communicating with a RebelVox Server over an IP network that is shared by an unrelated HTTP client. The packets are interleaved by the network in ways that the RV system does not necessarily control.



RebelVox Packet communication over a shared Internet



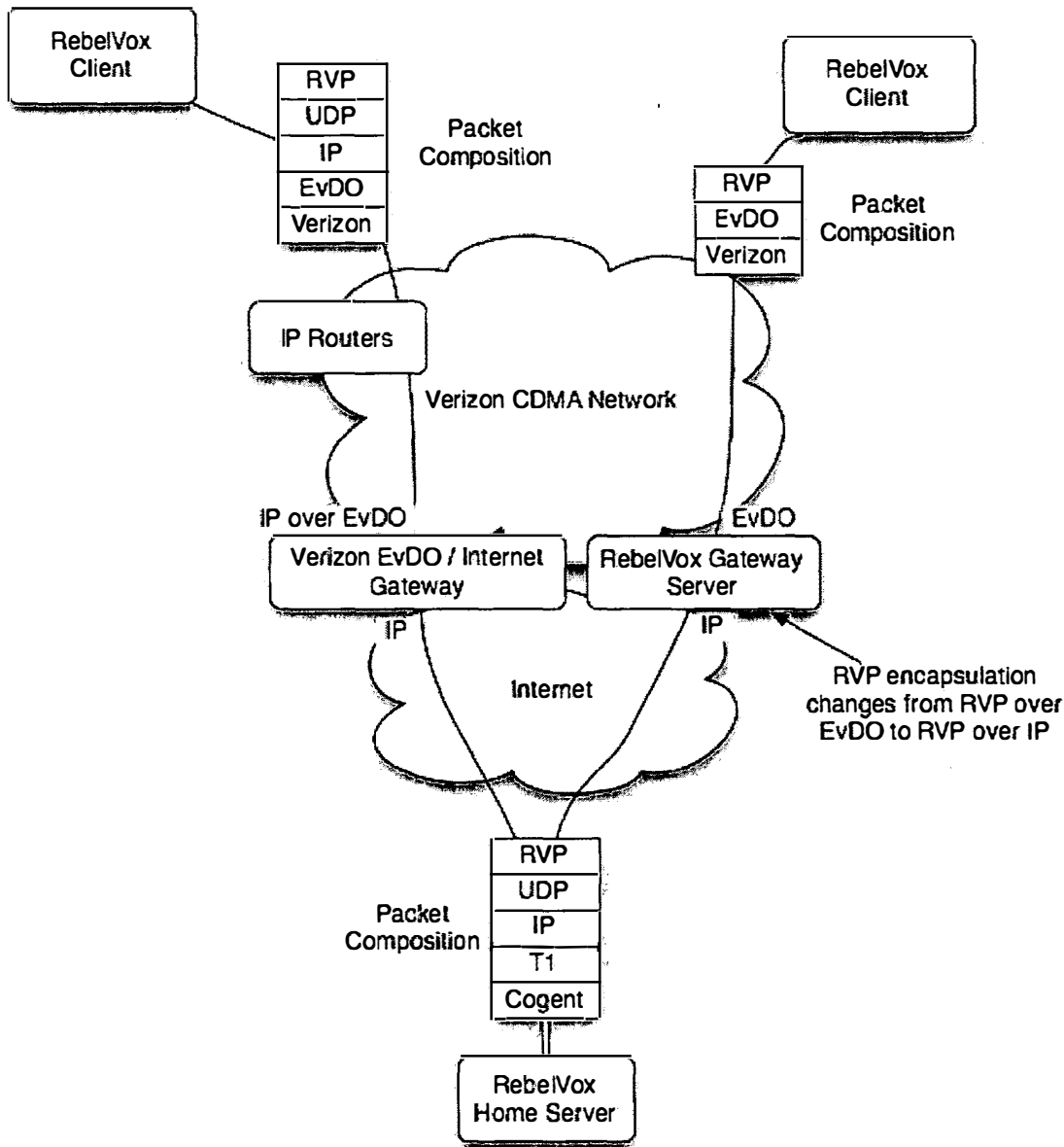
RebelVox Packet communication over a circuit-switched network

The underlying network technology will determine the maximum size of an RVP that may be transmitted efficiently. IP networks support packet fragmentation, but this comes at a substantial performance penalty. For IP over Ethernet networks, the local system has an MTU enforced by Ethernet itself to be 1500 bytes. For Gigabit Ethernet, this could be much larger. The largest IP packet that may be transmitted must be leave room for the Ethernet headers. The largest UDP packet must leave room for both IP and Ethernet headers. Thus, the largest RVP that may be generated for this network is the Ethernet MTU – IP header – UDP header = 1452 bytes. The RVP will have a header of its own, so the actual RVP media payload will be even less than 1452 bytes on an Ethernet network.

In a purely packet-based network, there are two potential values for MTU, the local link MTU and the path MTU. Determining the local link MTU yields the maximum size for an RVP to be efficiently sent out the local network interface. The path MTU yields the maximum size of an RVP that may be sent intact all the way to the remote node.

Even if a sending system is connected via Ethernet, the packet might still pass through various other systems with smaller MTUs. The smallest MTU on the path to the destination needs to be resolved and remembered by the sender somehow. In the IP world, there is a standard procedure for discovering this called "Path MTU Discovery". For other kinds of networks, the RV system will need an equivalent procedure.

Certain network paths might change the way in which the RVPs are encapsulated. For example, a RebelVox client connected via RVP over EvDO might have a path to a server connected by RVP over IP with a Gateway Server managing the interconnection. The Gateway Server uses a different encapsulation depending on which side of the network it is communicating with. The endpoints simply read and write RVP messages, which are unchanged by the Gateway Server.



RebelVox Mobile Network Data Formats

The RebelVox system assumes that the sender doesn't know if the packets will ever make their way to the destination, and that the network doesn't offer any clues either. This is the case for IP networks, but other types of networking technology might differ. If this were the case, the RV system could be improved to take advantage of this information. However, by building in the assumption that packets might be randomly dropped, RV can operate on any type of system that can deliver packets of data.

Of course, the RV system has to detect these packet drops and request retransmission when appropriate. These requests take the form of additional RVPs with a different internal type. An RV sender needs to hear back from the recipient server to adjust its sending rate or respond to retransmission requests. The

underlying network doesn't know the difference between any of the different "types" of RVP. RVPs are opaque segments of data as far as the encapsulating network protocol is concerned.

7 INDEPENDENT SOLUTIONS

There are other specific applications to which RebelVox solutions will be applied, some derivative of MCMS-C and some independent of media management.

Store and Stream. A sender-receiver cooperative protocol for enabling delivery of media with very low latency while also providing capture and archiving of “perfect” media images. The RebelVox system will bridge the performance gap between TCP (always delivers a perfect copy no matter how much latency is involved) and UDP (data is delivered quickly with no guarantees of completeness) by adaptive delivery of data that is “good enough” to meet the user’s moment by moment specifications (known by defined intentions and attention and their behavior) with respect to timeliness and quality; to optimize usage of bandwidth; and to always eventually deliver a perfect copy of the data. (See PR 4.2.1.2 *Store and Stream* and related sections of this document.)

Payload Optimization. A sender-receiver cooperative protocol for optimizing packet contents, size, frequency, and quality used by the sender based on information known about the receiver’s state and intentions with regard to the payload. (See PR 4.2.1.2.2.3 *Payload Quality And Content Management (PQM) for Network Optimization* in this document.)

Flexible Call-Participation Management. A call participant protocol that supports in any conversation, including conference calls, Tivo-like functionality including: pause, fast-forward, catch-up-to-live (listen more quickly with several options, including but not limited to: removal of silence, playing faster, removing hesitation particles), jump to head (go the “live” portion of the call), replay from any point, filter by tags or system attributes, filter by participant, etc.

Corporate Voice Messaging and Conference Calling System or Appliance. MCMS and MCMS-C can be deployed as an internal application (as software or on an appliance) and provide all or some of the functionality of MCMS and/or MCMS-C. (This would not necessarily need the following RebelVox functionality: security and encryption, the distributed architecture, advanced versions of network optimization, etc.)

8 REBELVOX GLOSSARY OF TERMS

8.1 System Components

- **Client.** A Client is the user application in the RebelVox architecture, which includes a user interface, persistent data storage, and RebelVox communication functionality. It may run on a general purpose computer, a portable computing device, a programmable phone, an analog phone, a radio (programmable or not), or other communication device. It will support encoding of media (capturing voice) and rendering of media, in addition to much of the other RebelVox architecture, which also runs on Home Servers. Users authenticate with the Client application, and the application manages all communications (messages and signals) and payload (media) transfers with Servers.
- **Gateway Client.** In the RebelVox Gateway Client, the client for many users runs on one general-purpose computer enabling RebelVox functionality on non-RebelVox devices. (See section PR4.3.1.34.) Clients that communicate through a gateway maybe be browser-based, applications on mobile devices, devices on radio networks, or simply PSTN-based phones.
- **Device.** A physical device that runs the Client application or interfaces with a Gateway Client. For the purposes of the Prototype, a User may only ever be actively logged in to a single device. Future versions will support simultaneous multi-device access.
- **Servers.** The infrastructure of the Network. Server nodes contain authentication information, Presence data, Media Archives, and anything else that might be necessary to facilitate communications between Devices. In addition to Gateway Clients (which are a type of server as well as a client), there are four types of servers.
 - Home Servers. (See section PR4.3.1.33.)
 - Gateway Servers. (See section PR4.3.1.35.)
 - Directory Servers. (See section PR4.3.1.36.)
 - Access Servers. (See section PR4.3.1.37.).

8.2 Media

- **Message.** An individual unit of communication from one Contact to another. Each message consists of some sort of Media. Messages are distinct units of communications (in any media) sent by a User. A message has attributes:
 - The sending User
 - The conversation it belongs to
 - Importance/Urgency – the sender can set a higher than normal level of importance on a message
 - The media payload (the actual media)
- **Media.** Audio, video, text, position, or other data that one Contact sends to another inside a Message.
- **Conversation.** A thread of Messages (identified, persistently stored, grouped, sequenced, and prioritized) between two or more Contacts on their devices. Users generally interact with their

device via Conversations, participating, reviewing, or adding to a Conversation as desired. When new Messages arrive, they either define a new Conversation, or they are added to an existing one.

- **Head of a Conversation.** The most recent moment of a conversation that has been encoded by the most recent speaker. It is where (when) you are positioned if you are listening “live” or where you “jump to” if you want to “jump to live.” Any listener is offset from the Head by the system enforced Minimum Time Shift Delay.
- **Multiple Conversation Management System-Concurrent.** RebelVox MCMS-C is an application built on RebelVox MCMS (Conversation Management System), which enables all of the feature functions of MCMS with the added feature of enabling users to manage and participate in multiple conversations at the same time through a system of priorities and time shifting. (See sections PR3.2.1 through PR3.2.4.)
- **Priority.** When users are managing multiple live conversations, priority is the mechanism through which the system determines what is to be rendered next. The user can set default priorities (conversations, contacts, or groups) for conversations he will receive. In the case of a conflict (more than one message is ready to be rendered), the system will resolve these priorities (with a few other factors) to determine what to render and what to time shift.
- **Importance (and other tags).** The system will support this special attribute of a message, which will enable a sender to cause a message to be rendered regardless of other priority settings (e.g., it is an urgent message). A high “importance” value will override other priorities. This feature is critical for tactical systems, though a system can be configured to use or disable this feature. A user will be able to apply other tags to messages and filter or organize messages by their tags.
- **Packet.** RebelVox will communicate over standard protocols (See section 6.1 Product Infrastructure) and will construct (or cause to be constructed) appropriate packets for the protocol in use. Each packet consists of header (meta data) and payload (media data).
- **Header or Packet Header.** The portion of a packet that describes the packet, the meta data concerning the payload, and/or its origination/destination.
- **RebelVox Packet.** A RebelVox packet is a proprietary packet that enables RebelVox to further refine and optimize delivery of messages, signals, and media packets. (See section 3.1.)
- **Media Payload (or Payload).** The actual media portion of network communication packets.

8.3 Media Management

- **Time Shift Delay (TSD).** The amount of time that rendering of a packet is delayed from when the packet arrives. A minimum delay (See Minimum Time Shift Delay) is introduced to compensate for jitter (variation in the expected arrival time or out of order packet arrival). Choosing to view a conversation time shifted (reviewing a message or Catching Up To Live, for instance) also can be viewed as using a TSD.
- **Minimum Time Shift Delay (MTSD).** The time shift delay enforced by the system to allow for jitter processing. This causes the system to delay rendering until an adequate number of the packets have arrived to create a usable media stream. The system may adjust over time using adaptive jitter buffer techniques.

-
- **Rendering.** Delivering a media stream to the speaker (display or sound generating device) for listening (viewing). Rendering includes mixing of multiple overlapping messages in a given conversation.
 - **Mixing.** Some messages in a Conversation are sent from multiple participants at the same time. They are received as separate message streams and the mixing process creates a single media stream from them aligned by time.
 - **Encoding.** The process of translating analog data (e.g., voice) into digital data.
 - **Adaptive Jitter Buffer.** Jitter buffers or de-jitter buffers are used to counter the jitter introduced by packet switched networks so that continuous rendering of audio (or video) transmitted over the network can be ensured. The data is stored in a buffer before rendering to allow a reasonably sized buffer of media to arrive. The media may be rendered before all the packets are received, trading off quality for currency. An adaptive jitter buffer is capable of dynamically changing its size to optimize the delay/quality tradeoff.
 - **Persistent Infinite Message Buffer (PIMB).** The RebelVox PIMB is a sophisticated adaptive jitter buffer with the additional attributes of potentially infinite persistence of the media, as it maintains all of the packets of a Message insuring the best quality of data available. (See section PR4.3.1.10.)
 - **Packet Loss Compensation (PLC).** During rendering of a media stream, the PLC component compensates for missing packets, interpolating the results to present the stream to a listener or viewer. Missing packets may be rendered as silence, or information from adjacent packets may be used to present some other sound or image.

8.4 People

- **User.** A person who is authorized to use the system.
- **Contact.** The record of a user of the system who may be contacted on their device. Users manage lists of contacts that either uniquely identify other users or describe multi-user groups called channels or groups. Users are themselves contacts. Users send Messages to other Users, and participate in conversations with other Users.
- **Group or Channel.** A type of Contact that may refer to and associate multiple Contacts. When Messages are sent to the Channel or Group, all Channel members will receive them. Contacts may join and leave certain channels based on the security settings of that Channel.
- **Participant.** A User who is identified as a target of a Conversation.

8.5 Conversation Management

- **Time Shifting.** To view a Message after it is created. There is actually always some amount of Time Shift caused by the time it takes to encode, send, receive, and decode messages. More time shift can be introduced by the requirements of delivery or by the choice of the user to listen to the message at a later time. In fact, because of Minimum Time Shift Delay (MTSD), there is always at least a small delay. Time Shifting is core to the MCMS-C system, allowing messages to be delivered:
 - On demand by rendering immediately or delaying as requested by the user, including retrieving old conversations from the archive for searching, reconstructions, etc;

-
- Delayed to accommodate other higher priority messages (or conversations) that need to be delivered first;
 - Delayed to assist in optimizing degraded network services and allow the conversation to continue;
 - Repeated if necessary for the message to be understood or reheard.
 - **Listening.** Listening in real time to an Active or Dormant Conversation.
 - **Reviewing.** Listening Time Shifted to an Active or Dormant Conversation.
 - **Intention.** A user-defined attribute capturing whether the user wants to listen to a conversation as soon as possible or delayed.
 - **Attention.** A user attribute capturing whether the user is listening to a given conversation at the moment.
 - **Catch Up To Live (CTL).** If a user is not at the Head of a conversation and wants to listen to all of the messages between where he is and the Head, he can use a feature called Catch Up To Live. CTL will use rendering modes (the Catch-Up Mode) to render the media more quickly (replay faster, remove silence, remove hesitation particles, etc.) though in an understandable form. The system will proceed until it is able to seamlessly flow into the live conversation. This is very useful in conference calls where users need to shift their attention occasionally.
 - **Catch Up Mode.** A user-configured mode to determine how the CTL process should do the catch-up (replay faster, remove silence, remove hesitation particles, or some combination).
 - **Jump To Live (JTL).** If a user is not at the Head of a conversation and does not want to listen to all of the messages between where he is and the Head, he can use a feature called Jump To Live. JTL will skip the unwanted messages and start playing the “live” conversation.

8.6 Some System Concepts: More Depth and Detail

Conversations

Conversations consist of Messages maintained in their original context and sequence. Any message sent either belongs to an existing conversation or begins a new conversation.

They are “organized” sets of messages, that is, messages organized by user-defined criteria. Here are some examples of “threads” which could hold the conversation together:

- The “group” of people who are participating (“alpha geeks”)
- The “channel” it is broadcast on possibly with date/time constraints
- A standing topic (weekly sales meeting)
- A “standing Conversation” which is considered a perhaps the default way to send messages to a given user group or contact

-
- An ad hoc topic (“planning the trip”)

The following lists are kinds of conversations that the system will support; others will evolve:

- Live Conversations (users are speaking to each other in “real time”, considered synchronous)
- Standing Conversations (users are exchanging messages as in instant messaging systems, considered asynchronous)
- A seamless transition from Standing Conversations to Live Conversations and back
- Conference Calls (scheduled or ad-hoc multi-party conversations, often synchronous)
- Standing Calls (regularly scheduled multi-party conversations)
- Structured Conversations (e.g., Simultaneous Briefings – a type of call where everyone leaves their weekly briefing for the group by 8AM; everyone listens to the other’s briefings by 9AM; at 9AM everyone joins a live call)

In a user interface (graphical UI, reporting, analysis) the messages can be viewed by any organization of the attributes we store, which can include:

- Defining attributes (group, participants, topic)
- Any other attributes:
 - Importance
 - Priorities
 - Particular groups or persons
 - Time of day; date
 - System defined tags
 - User defined tags

Conversation attributes will include, but not be limited to, the following:

- Name
- Participants
- Dates & Times (begin, end)
- Priority
- States:
 - Pending
 - Invitations have been sent out – waiting for acceptance
 - Not sure that this matters
 - Active
 - The Conversation is active, and one or more Participants are talking
 - Dormant
 - The Conversation is active, but no Participants are talking
 - Terminated

-
- Conversation has been terminated explicitly, or timed out based on time since created or time since last message

A Participant's Conversation

The relationship between a Participant and a Conversation has attributes. They may include, but not be limited to, the following:

- Priority – a Participant can set his priority for any given conversation.
- Status – The current relationship between a Participant and a Conversation
- States of participation in a conversation
 - Actively Participating
 - Actively listening to all real time communications (modulo priority)
 - Participating in more than one conversation at the same time
 - Time shifted
 - Catching Up to Live
 - Passively Participating (Receiving Urgent)
 - Not actively listening to real time communications
 - Possibly monitoring more than one conversation at the same time
 - Receiving all priority messages
 - Everything recorded on device
 - Standby (only distinction is that priority messages are not received; the distinction between Passively Participate and Standby may be more of a continuum – based on the priorities that the Participant specifies)
 - Not actively listening to real time communications
 - Possibly monitoring more than one conversation at the same time
 - Receiving all priority messages
 - Everything recorded on device
 - Will not receive priority messages
 - Ignoring
 - Invited but declined in some way
 - Not actively listening
 - Not being recorded on my device

Priority and Importance

There are several concepts that we have combined, in our earlier discussions, into the concept of priority. This discussion is to separate those ideas into distinct and implementable concepts.

From the Listener's Perspective

The first is called 'priority'. For any given conversation, there is a relative priority – from the listener's perspective. And it is the listener who will establish the relative priority as he listens to messages. One user interface that the system might use to represent the relative priority of the conversations is an ordered list, which the user can reorder:

Conversation A

Conversation B

Conversation C

In the above example, the user has indicated that messages on Conversation A have a higher priority than Conversations B and C. If there is ever a conflict – messages currently available on A and B for instance – then play the message on conversation A first.

If at a later time, Conversation C becomes more important to the user, he would change the order in the user interface:

Conversation C

Conversation A

Conversation B

For some users and/or some user interfaces, the order may be set for the user – and the user may have no ability to modify the order. For instance, on a tactical radio, there may be no way for a user to change the order of priorities.

This order of priority may be managed internally as a number between 1 and n. There may be other user interfaces that allow the user to explicitly set numbers in this range.

There could be default priority values for each of the following, with precedence determining which one wins.

- Groups
- Contacts (or users)
- Conversations.

These default priority ratings would set the initial priority. Some user interfaces will allow the user to adjust them dynamically.

High Importance or Emergency Messages---Set by the Sender

There are several choices on how to implement ‘emergency messages’ (This isn’t a perfect term – but conveys what we pictured as ‘highly important’ messages). In all cases the creator of the message indicates that it is an emergency message. This is one possible implementation:

- “Importance” is an attribute of a message. The sender indicates that this is an emergency message. This message will be played as the highest importance message and override any priority settings of

the listener. (Any conflicts among multiple emergency messages would be resolved based on the priority of the conversation).

Message Network Routing Solutions

The following takes an example and explores the process of sending individual messages (and in fact packets) of a conversation between all of the participants, including the Client, the Access Servers and the Home Servers. We have considered a number of ways to communicate between these participants, including sending each message to the Home Server of each the other participants, and having the Home Server have the responsibility of delivering the message to their client(s). With dispersed clients this can greatly increase network traffic and latency. The following example should make the problem clear, and suggests our current solution.

The example shows a network with two Access Servers (Access Server 0 and Access Server 1), several Home Servers (in the diagram Home Server 1, Home Server 2, through Home Server N), and five clients (C1, C2, C3, C4 and C5).

The two networks are connected by a low speed connection. There are three goals, which may conflict:

1. Minimize the traffic between the two networks. If possible, only send a single copy of a message from Network 1 to Network 2, or from Network 2 to Network 1.
2. Minimize the latency between two clients. If a message is being sent from C1 to C2 (or C5), and the recipient is actively listening, then the message should be delivered with minimum latency.
3. Keep the implementation simple.

Our initial solution, of having the clients send messages to the Home Server for each participant, and then have the Home Server deliver it to the clients, includes a lot of traffic between these two Networks. Here we describe an approach that minimizes that traffic, while maintain a low latency, and still being simple.

Access Server 1 and Access Server 2 will know that the only way to get from Network 1 to Network 2 will be to talk to each other. There may in fact be other network paths (it may actually all be a single network), but the other nodes will send all traffic that is to leave their 'network' to an Access Server.

When a client is turned on, it locates its Access Server. It may know the one (or small number) of Access Servers that it should try to connect with, or it may broadcast a message, and then identify an optimal Access Server that it will use as its contact point in to the network. It is possible that connectivity to a given Access Server will be lost at some point, in which case the client will again establish its Access Server.

Access Servers communicate with a directory server to identify the clients for which they are serving as the Access Server.

In the above example, C1, C2, C3 and C4 have identified Access Server 1 as their Access Server, and C5 has identified Access Server 2 as its Access Server.

Every Client has a designated Home Server (in our example we will assume that C1's Home Server is H1, C2's is H2, etc.).

When C1 initiates a conversation with C2, C3, C4, C5 and C6, it sends the request to Access Server 1 (the only server it can communicate with). Access Server 1 notes that it already has connectivity to C2, C3 and C4. It then requests from the Directory Server the status of C5 (online, talking to Access Server 2), and C6

(not online), and also requests the Home Server for all of the participants (C1, C2, C3, C4, C5, C6). It informs H1, H2, H3,...H6 that there is a conversation among these participants. (At this point, it can ignore C6 – which it cannot communicate with. If C6 comes online, it will be up to H6 to catch him up – and then connect him back in to the live conversation).

When C1 sends a message for this conversation, it is sent to Access Server 1. Access Server 1 then forwards the message to C2, C3, and C4. In addition, it sees that all of the other participants (C5, H1, H2...H6), have the same next hop – in this case, Access Server 2. It therefore sends a single copy to Access Server 2.

Access Server 2, knows which participants it is responsible for, and forwards the message to each of them. If they are not in this 'network', Access Server 2 will forward a copy to the next Access Server on the path to those recipients. It will again only send a single copy.

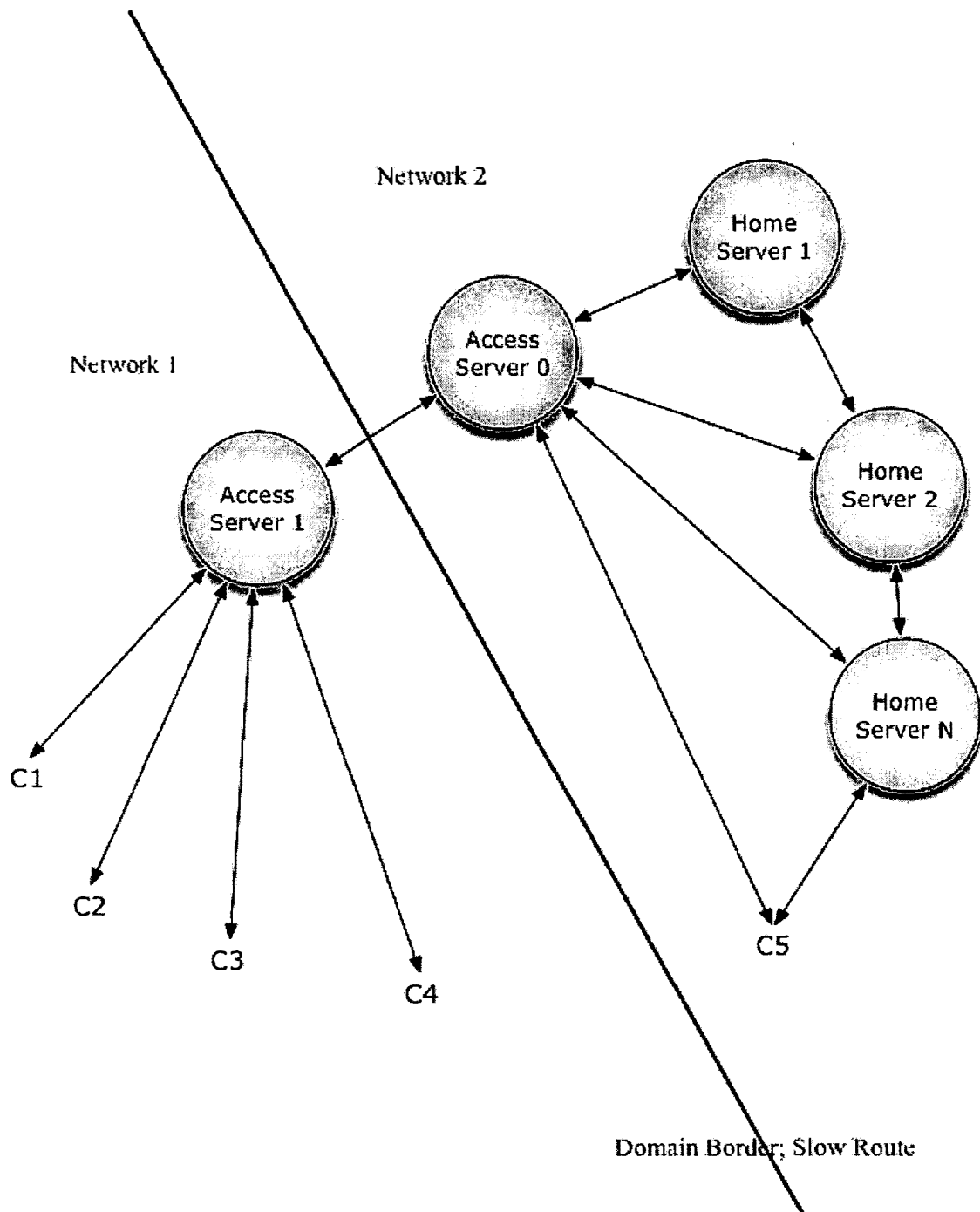
A given Access Server may issue a reinvoke to two Clients – allowing them to communicate directly with each other – in addition to sending the message to the Access Server (it still has to be sent to the Access Server to be forwarded to the Home Servers or to other Clients). This is an improvement in the latency of the message between these two clients. But it increases complexity on the client and the Access Server, and there are many variables that will influence whether it is better to do this or not. The total network traffic is actually the same (C1->Access Server 1, Access Server 1->C2, Access Server 1->C3) vs (C1->Access Server 1, C1->C2, C1->C3), however it is possible that C1, C2 and C3 are on a subnet with more available bandwidth. It is also possible that they are not, and that this would be a bad choice.

Our current conclusion is that we will only allow this reinvoke when:

1. There are only two participants (C1, C2)
2. They are sharing the same Access Server
3. The recipient is actively listening

In fact, an early implementation does not have to support this reinvoke.

Message Network Routing Solutions



RebelVox Packet Data Structures

The RebelVox Packet (RVP) is a structured message format designed to be flexible and efficient. Each RVP will be encapsulated inside another transport packet based on the underlying network technology. This is a preliminary view into some elements of the packet structure and how they might be encapsulated.

RVP

General structure of RVP packets is:

type	sub-type	length	payload
------	----------	--------	---------

This is likely encapsulated within another set of protocols, like this for example:

Ethernet	IP	UDP	type	sub-type	length	payload
----------	----	-----	------	----------	--------	---------

Types:

- 1 - Auth
- 2 - Signaling
- 3 - Media Payload
- 4 - Media Multiplex (one msg)
- 5 - Media Multiplex (multi-msg)

Type: 1 (auth)

Subtypes:

- 1 - DH key exchange
- 2 - Authentication

Type: 2 (signaling)

Subtypes:

- 1 - registration
- 2 - routing
- 3 - messaging setup
- 4 - network management

Type: 3 (Media Payload)

Subtypes (codec style):
Codec styles are negotiated at registration time. These represent the codecs and their various options.

Signalling Dialog between client A and access servers B and C:

A -> B may I use you as my access server? My codec style catalog is ABCDEF
 B -> A that would be fine, home server cost is Cbh, style catalog union is ABDEF
 A -> C may I use you as my access server? My codec style catalog is ABCDEF
 C -> A that would be fine, home server cost is Cch, style catalog union is ABCDEF
 A -> B can you reach X?
 B -> A yes, cost is Cbx (ex. 24)
 A -> C can you reach X?
 C -> A yes, cost is Ccx (ex. 101)
 A -> B I have a new message for X
 B -> A I am eager for your packets. Call this message IDax (ex. full message ID)

This allows a client to utilize multiple access servers to reach a given destination
 Verbose, human-readable payloads may be replaced by more compact representations..

Ethernet MAC A MAC B	IP IP A IP B	UDP A src prt B dst prt	RV Type 2 (sig)	Sub-Type 1 (reg)	length	May I use you as my access server? My style catalog is: ABCDEF
Ethernet MAC B MAC A	IP IP B IP A	UDP B src prt A dst prt	RV Type 2 (sig)	Sub-Type 1 (reg)	length	Sure. My Cost to reach your home server is 12, style catalog union ABDEF
Ethernet MAC A MAC B	IP IP A IP B	UDP A src prt B dst prt	RV Type 2 (sig)	Sub-Type 2 (route)	length	Can you reach host on X?
Ethernet MAC B MAC A	IP IP B IP A	UDP B src prt A dst prt	RV Type 2 (sig)	Sub-Type 2 (route)	length	Yes, my cost is 24
...						
Ethernet MAC A MAC B	IP IP A IP B	UDP A src prt B dst prt	RV Type 3 (media)	Sub-Type A (g.711 / 20ms)	length sequence number	media payload

Media Multiplex example:

Ethernet MAC src MAC dst	IP Protocol Options	UDP port src port dst	RV Type 4 (Mx)	message ID	total length	codec style	sequence number 1	media payload 1	sequence number 2	media payload 2
--------------------------------	---------------------------	-----------------------------	-------------------	---------------	-----------------	----------------	----------------------	--------------------	----------------------	--------------------

etc.

9 REBELVOX PROCESS DIAGRAMS

The following is a set of data flow diagrams (process diagrams) of the RebelVox system. Some of them have reference to sections of the text as necessary.

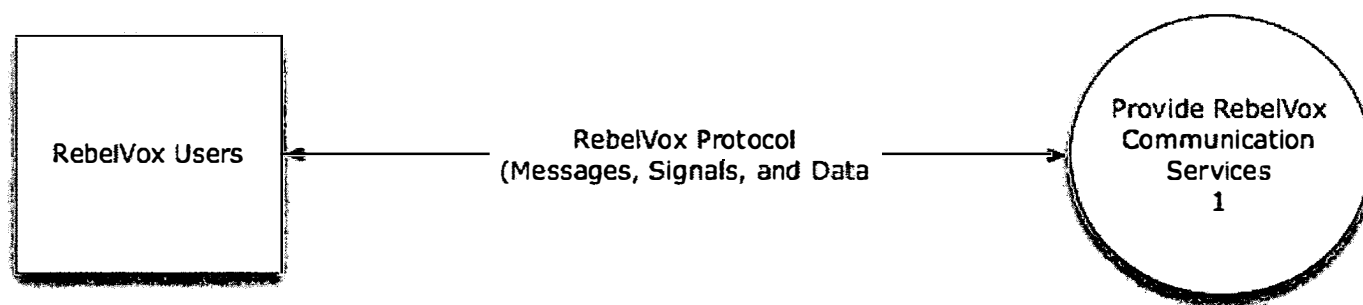


Diagram 0 : RebelVox Communication Services

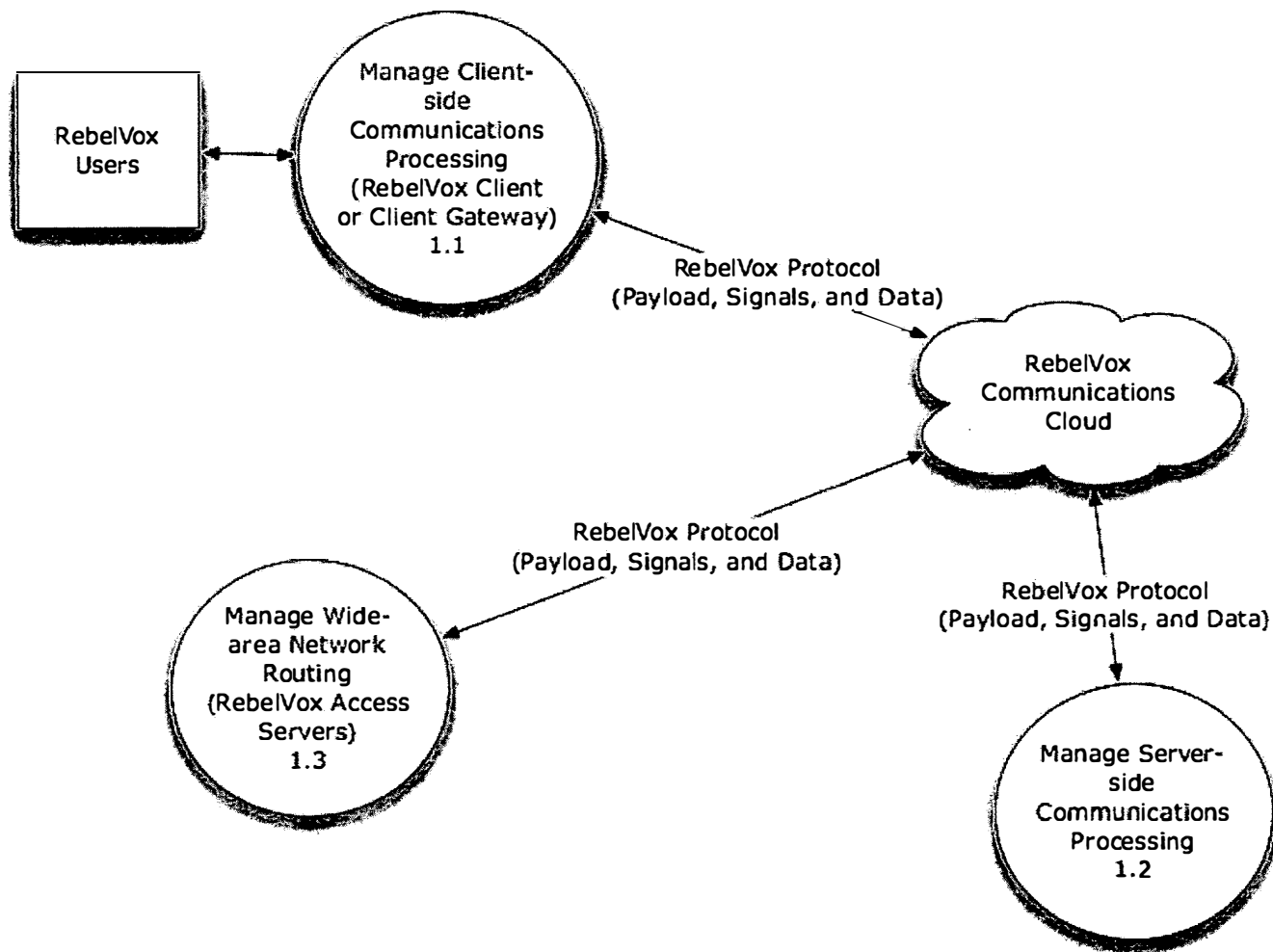


Diagram 1.0 : Provide RebelVox Communication Services

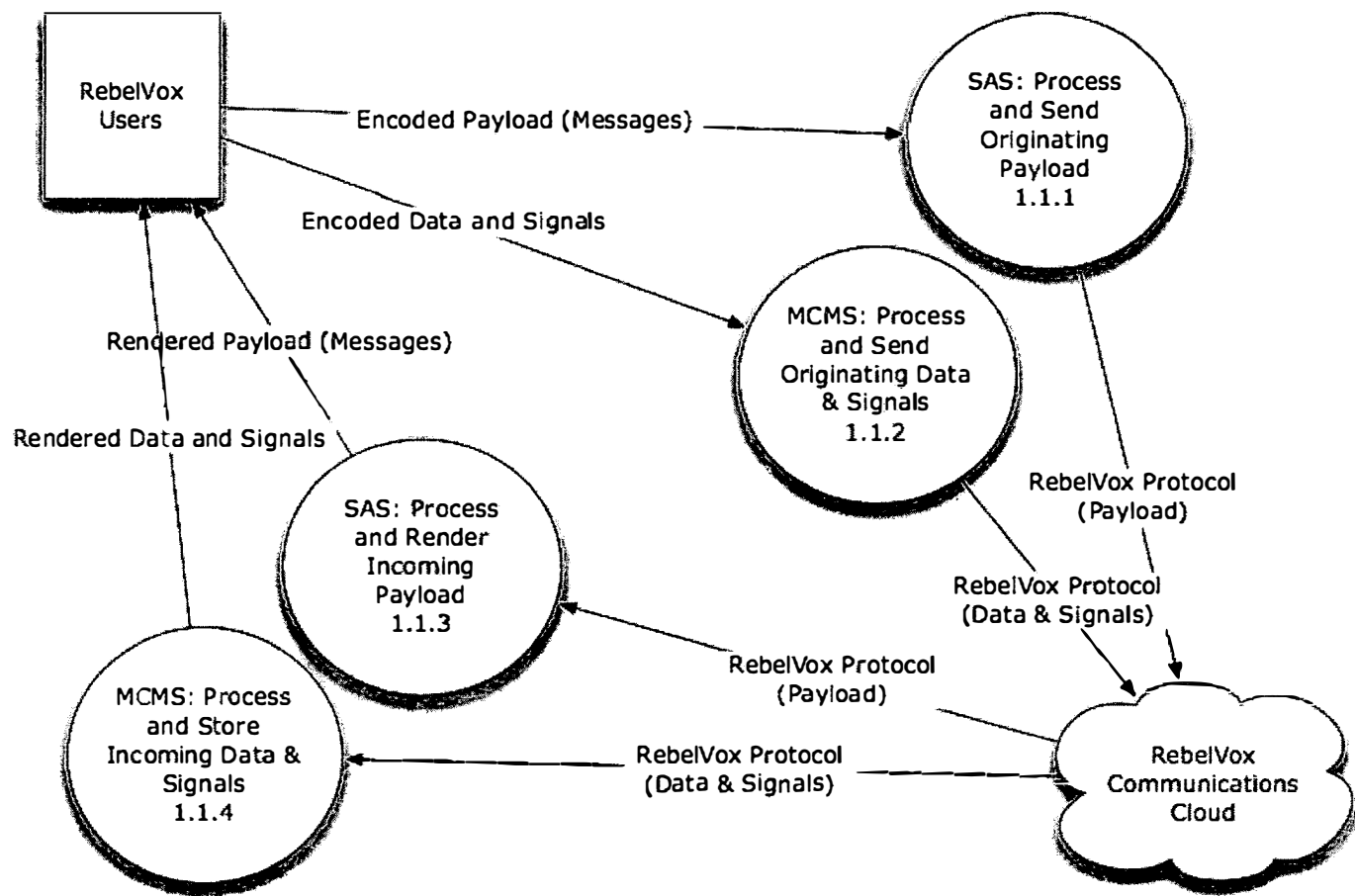


Diagram 1.1 : Manage Client-side Communications Processing

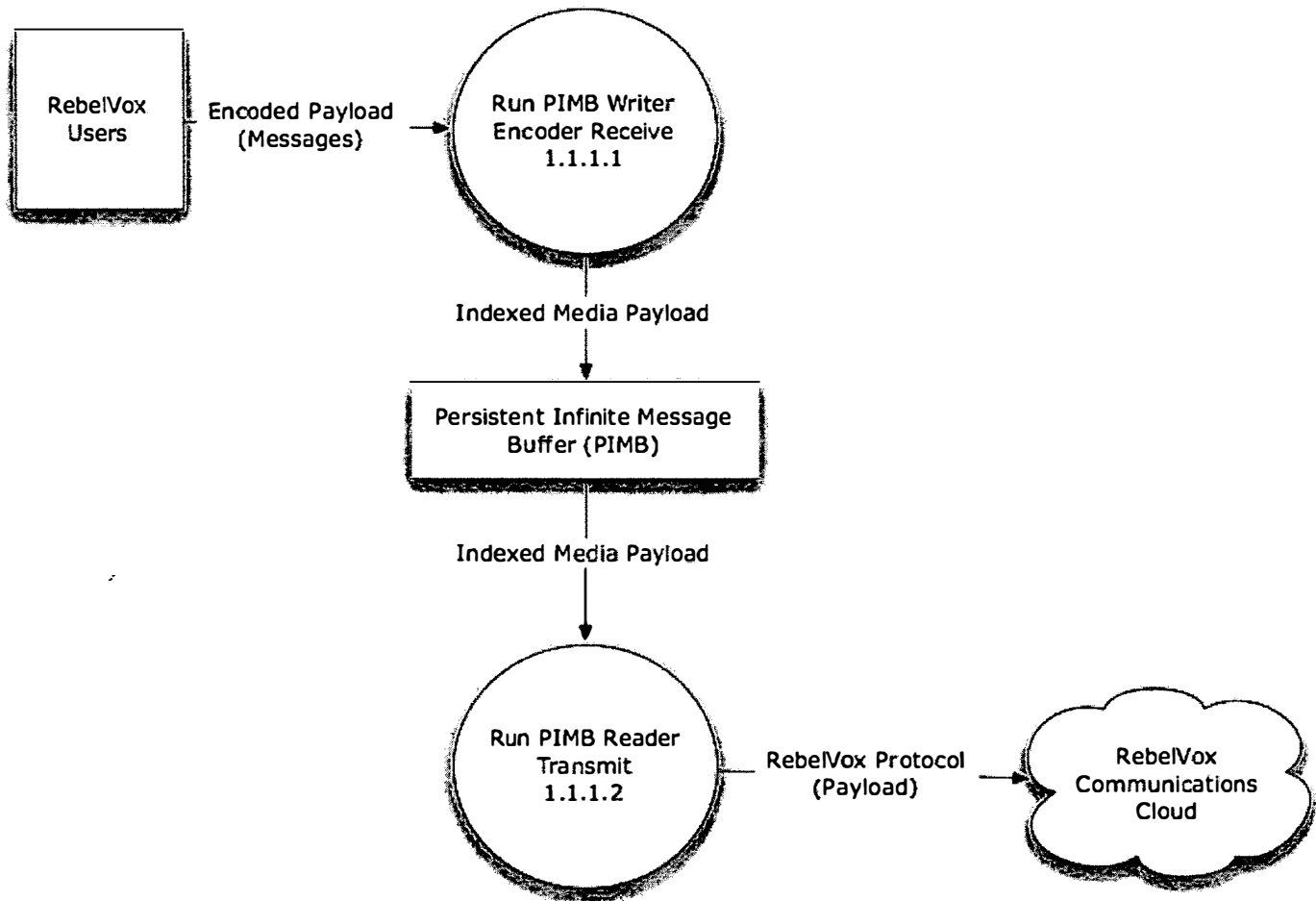


Diagram 1.1.1 : SAS: Process and Send Originating Payload

For more explanation, see section PR 4.2.1.2.1.3.2.

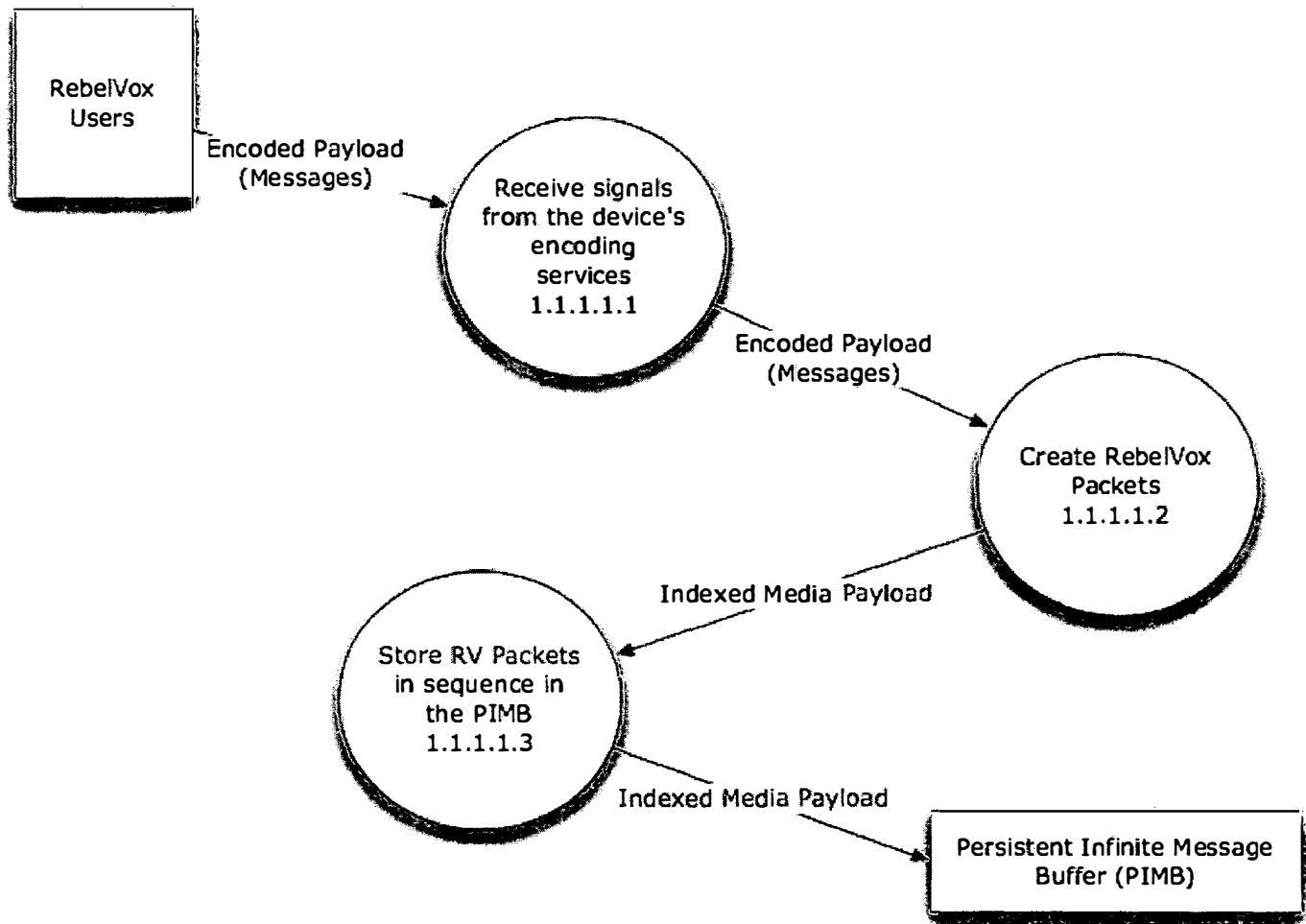


Diagram 1.1.1.1 : SAS: Run PIMB Writer Encoder Receive

For more explanation, see section PR 4.2.1.2.1.2.1.

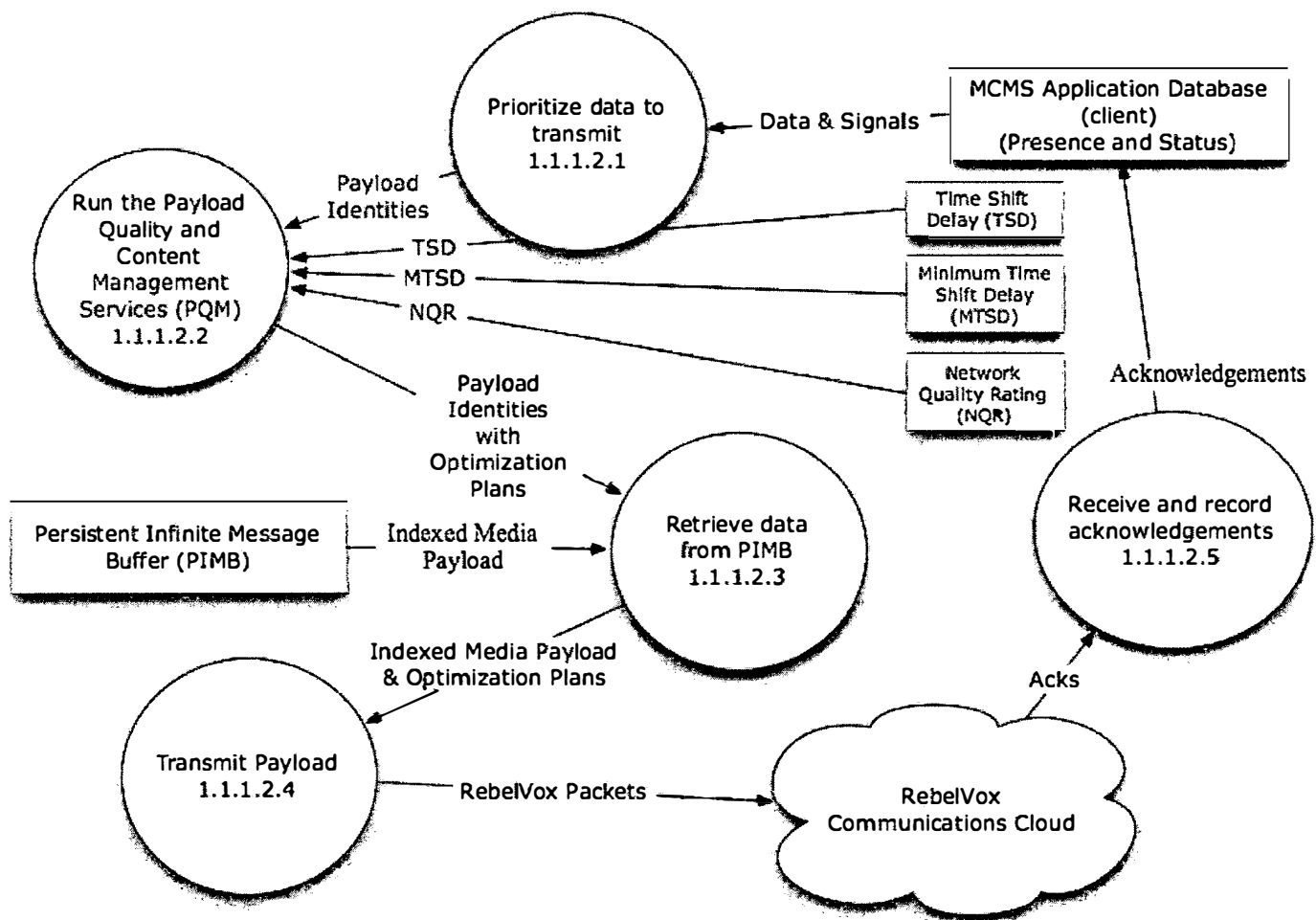
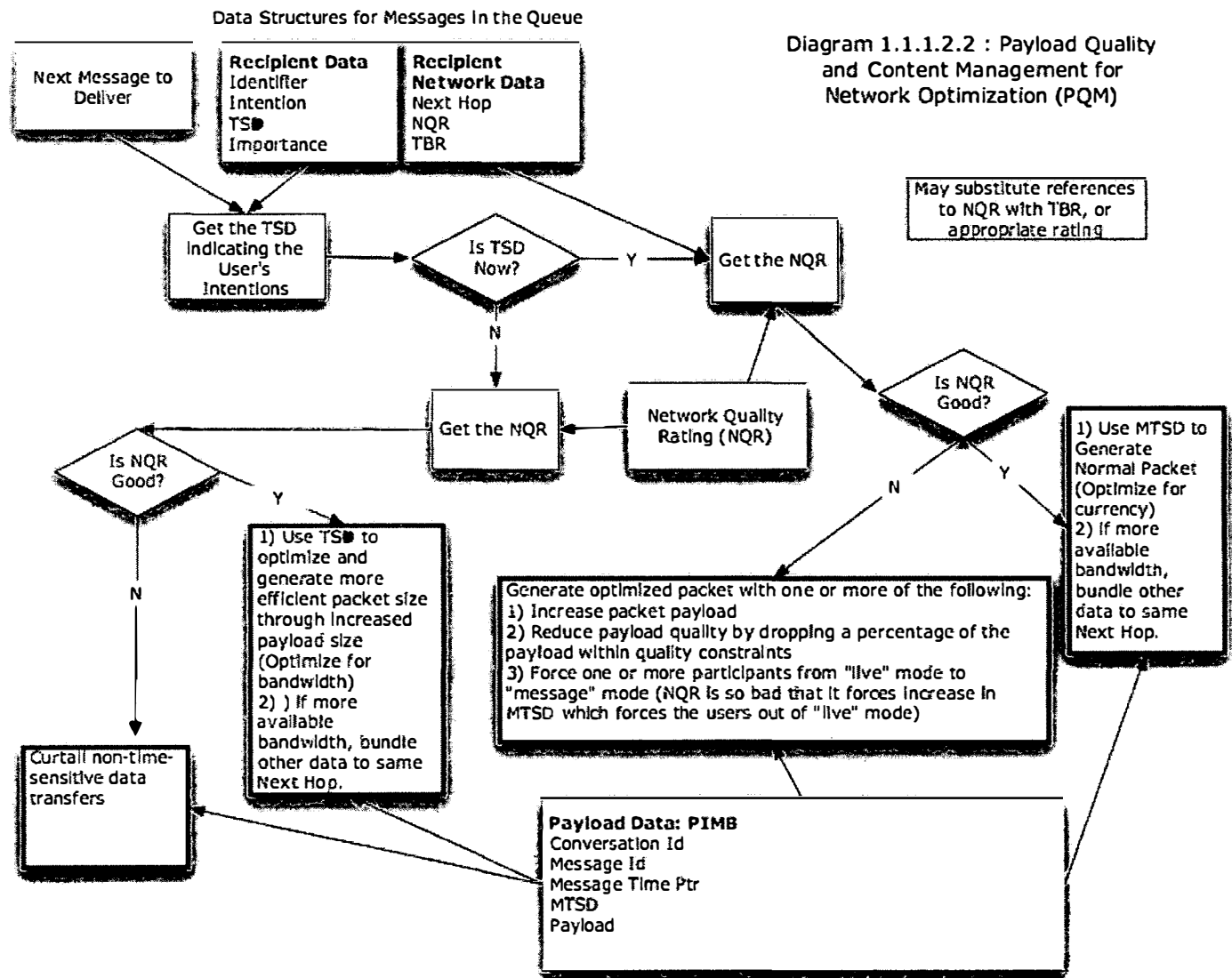


Diagram 1.1.1.2 : SAS: Run PIMB Reader Transmit

For more explanation, see section PR 4.2.1.2.2.3.



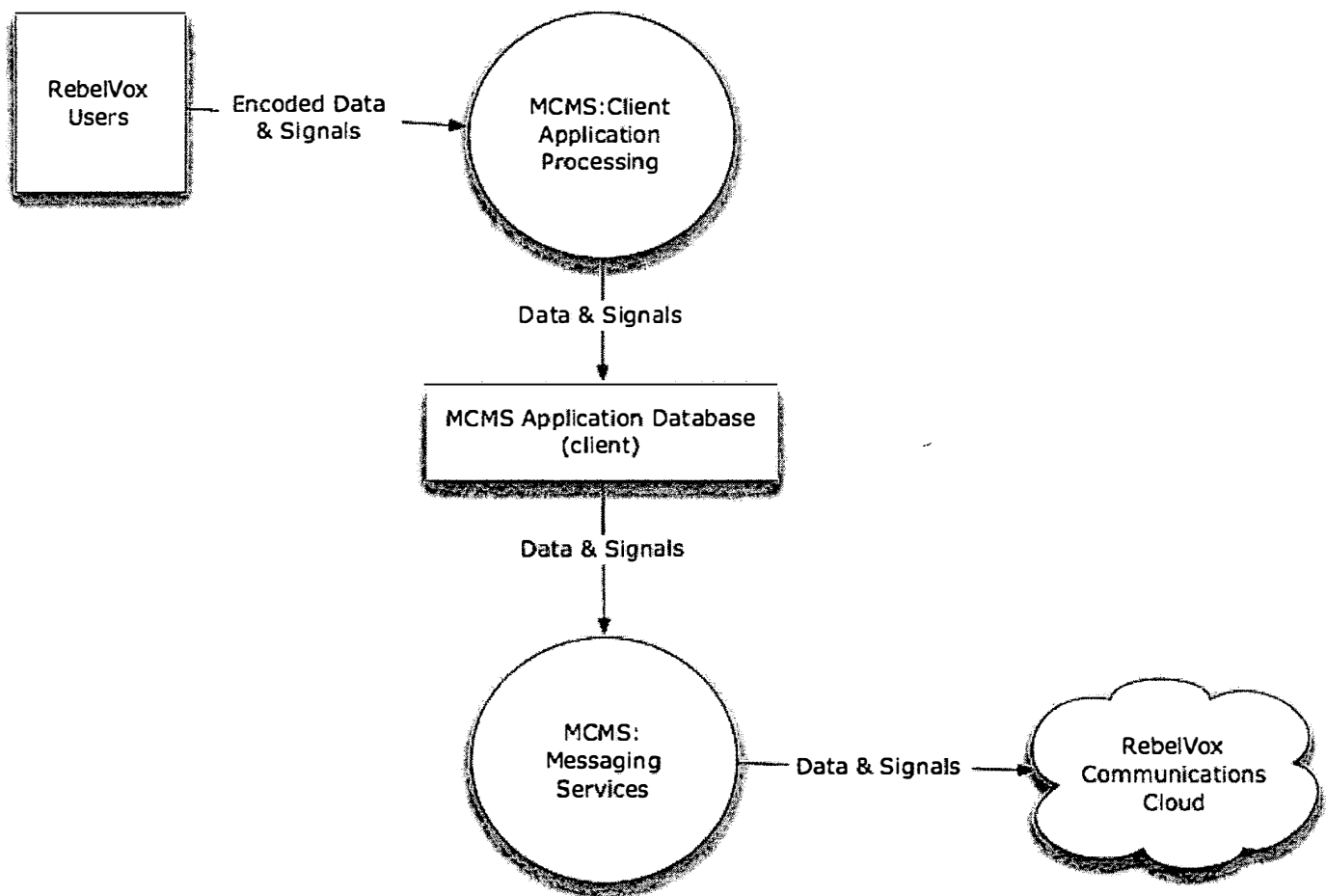


Diagram 1.1.2 MCMS: Process and Send Originating Data & Signals

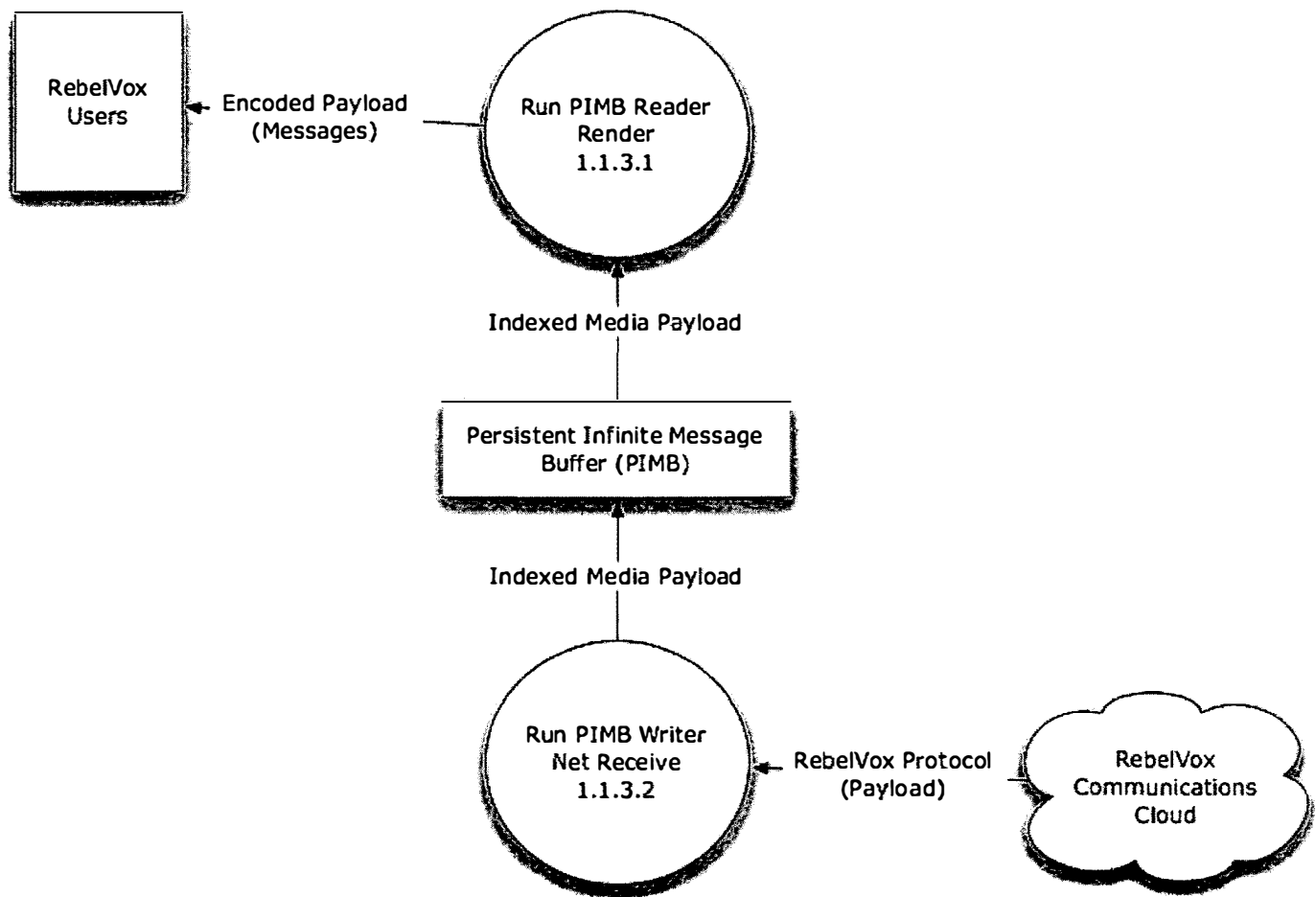


Diagram 1.1.3 SAS: Process and Render Incoming Payload

For more explanation, see section PR 4.2.1.2.1.2.2.

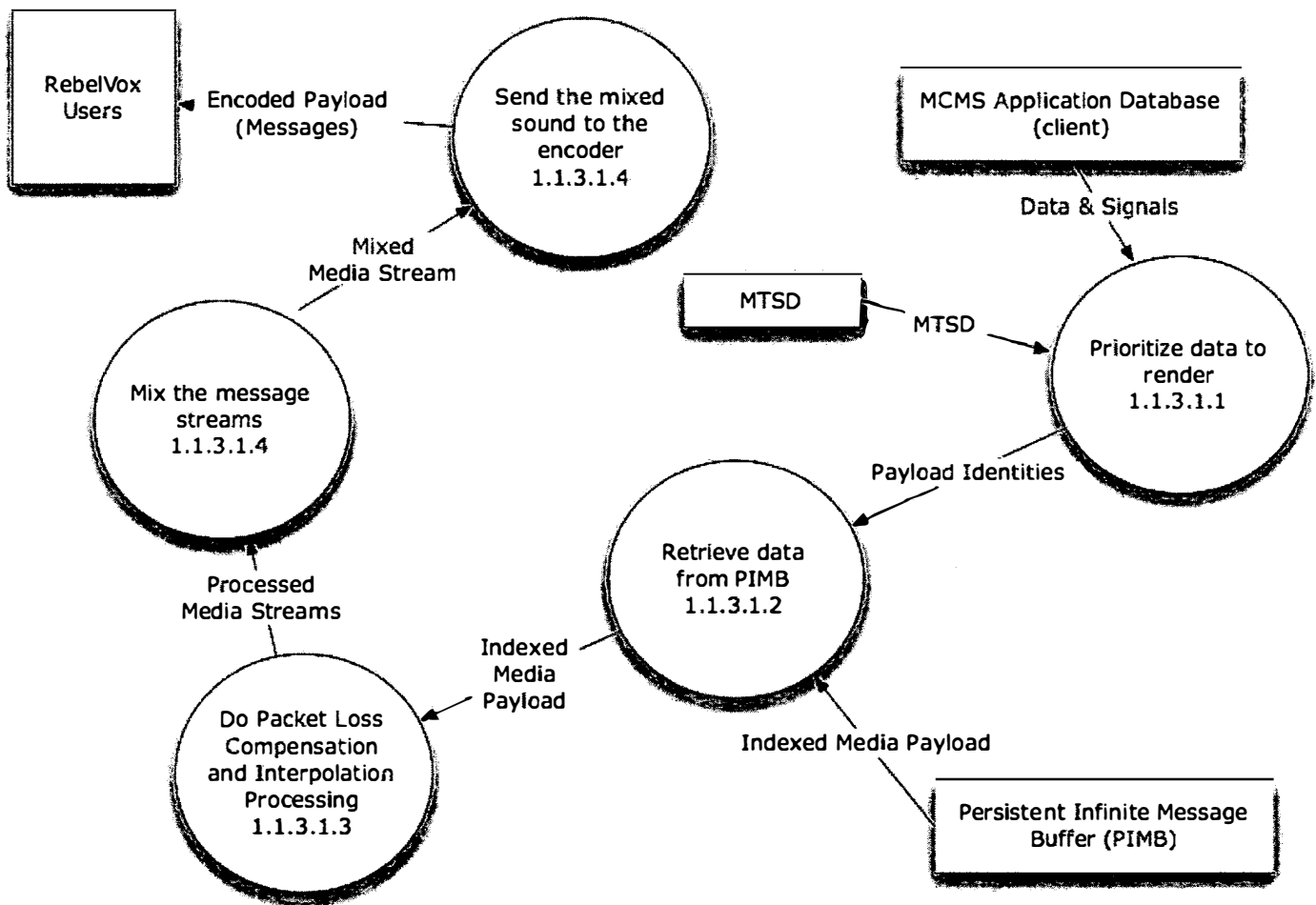


Diagram 1.1.3.1 SAS: Run the PIMB Reader Render

For more explanation, see section PR 4.2.1.2.1.3.1.

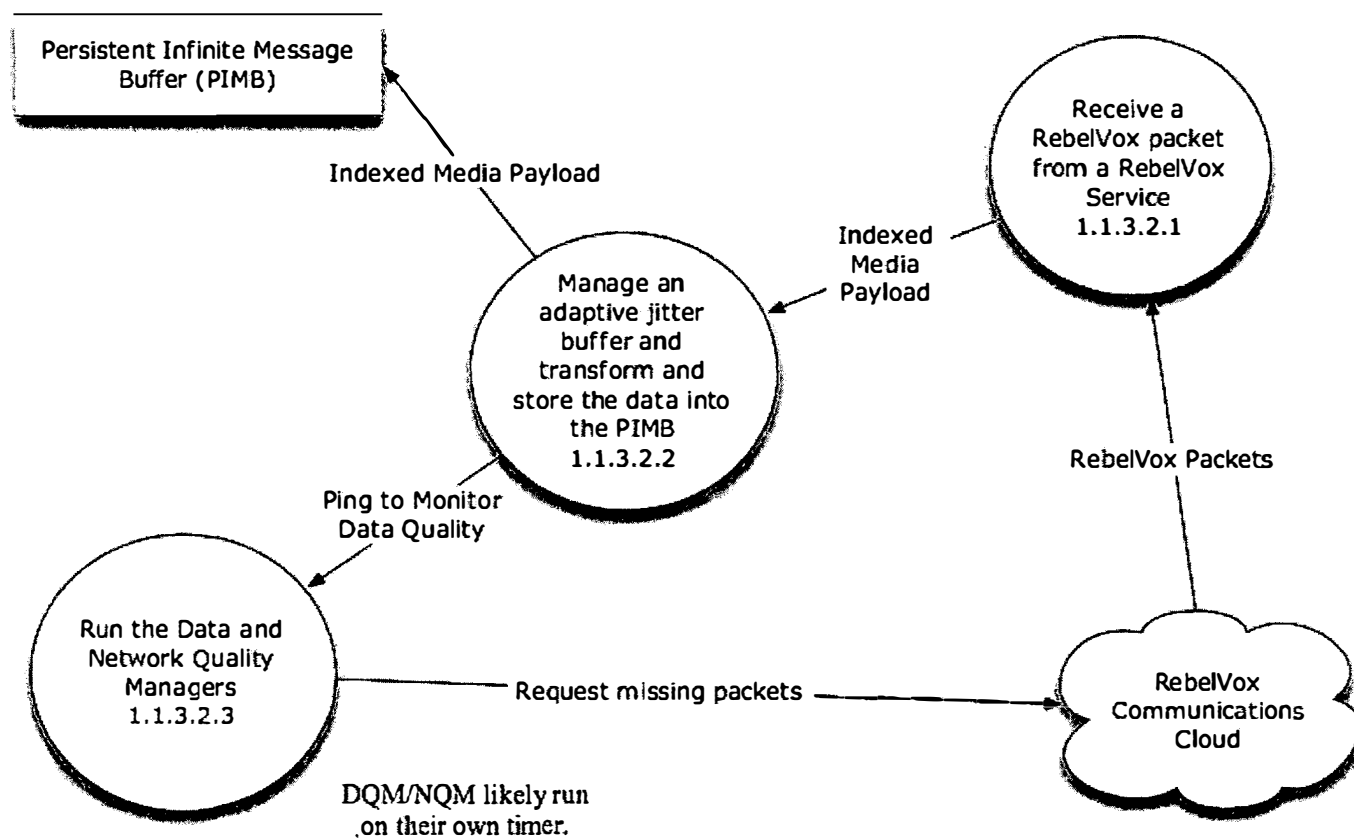


Diagram 1.1.3.2 SAS: Run the PIMB Writer Net Receive

For more explanation, see section PR 4.2.1.2.3.

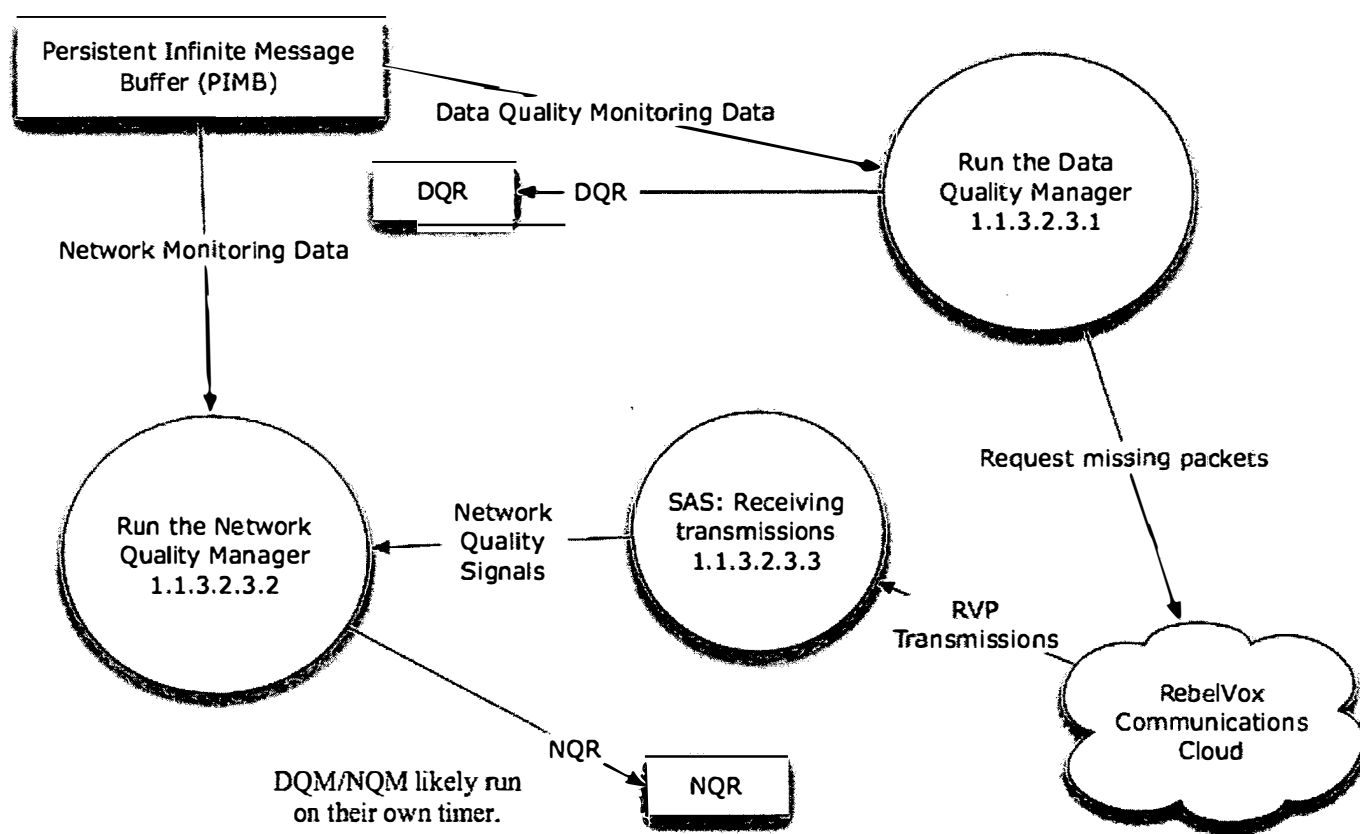


Diagram 1.1.3.2.3 SAS: Run the Data and Network Quality Managers

For more explanation, see section PR 4.2.1.2.3.1.

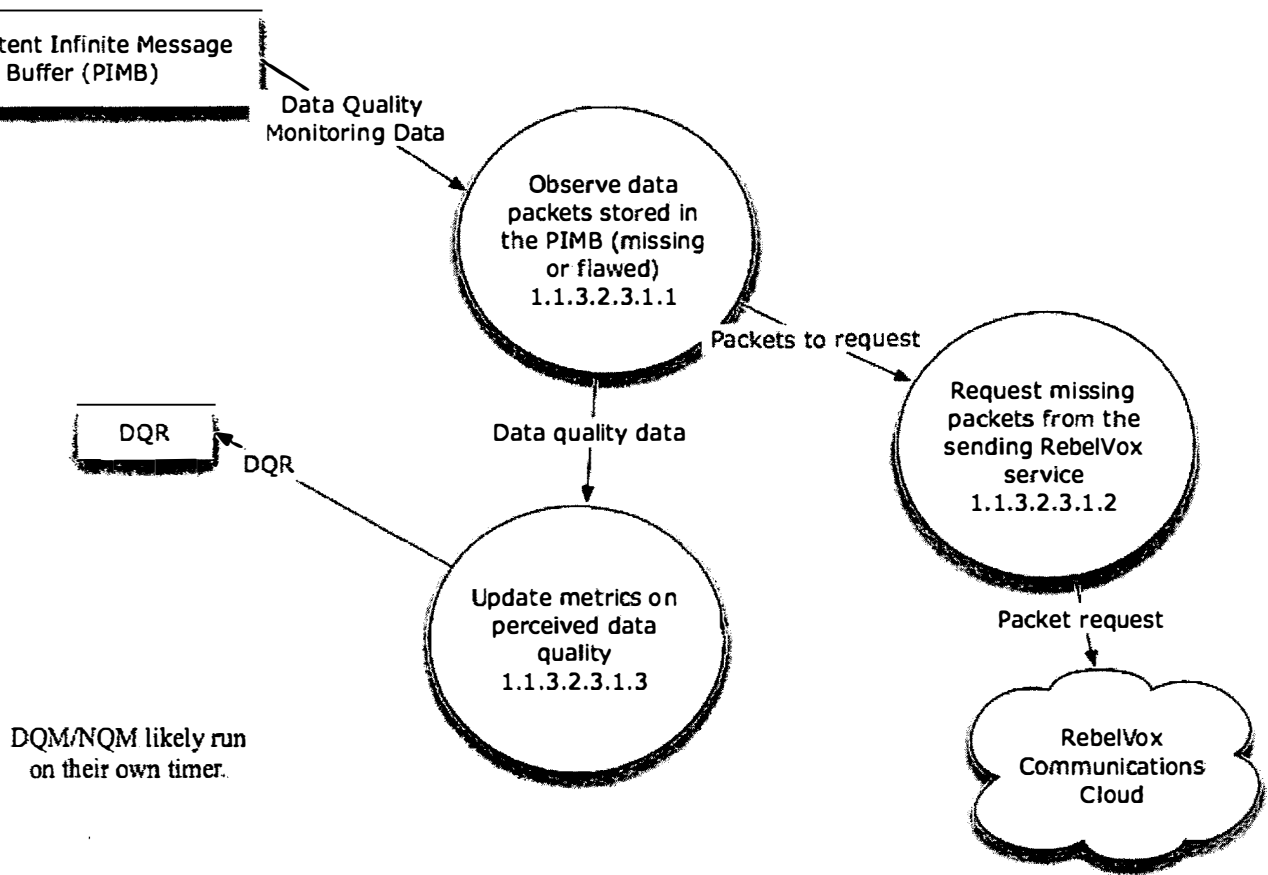


Diagram 1.1.3.2.3.1 SAS: Run the Data Quality Manager

For more explanation, see section PR 4.2.1.2.3.2.

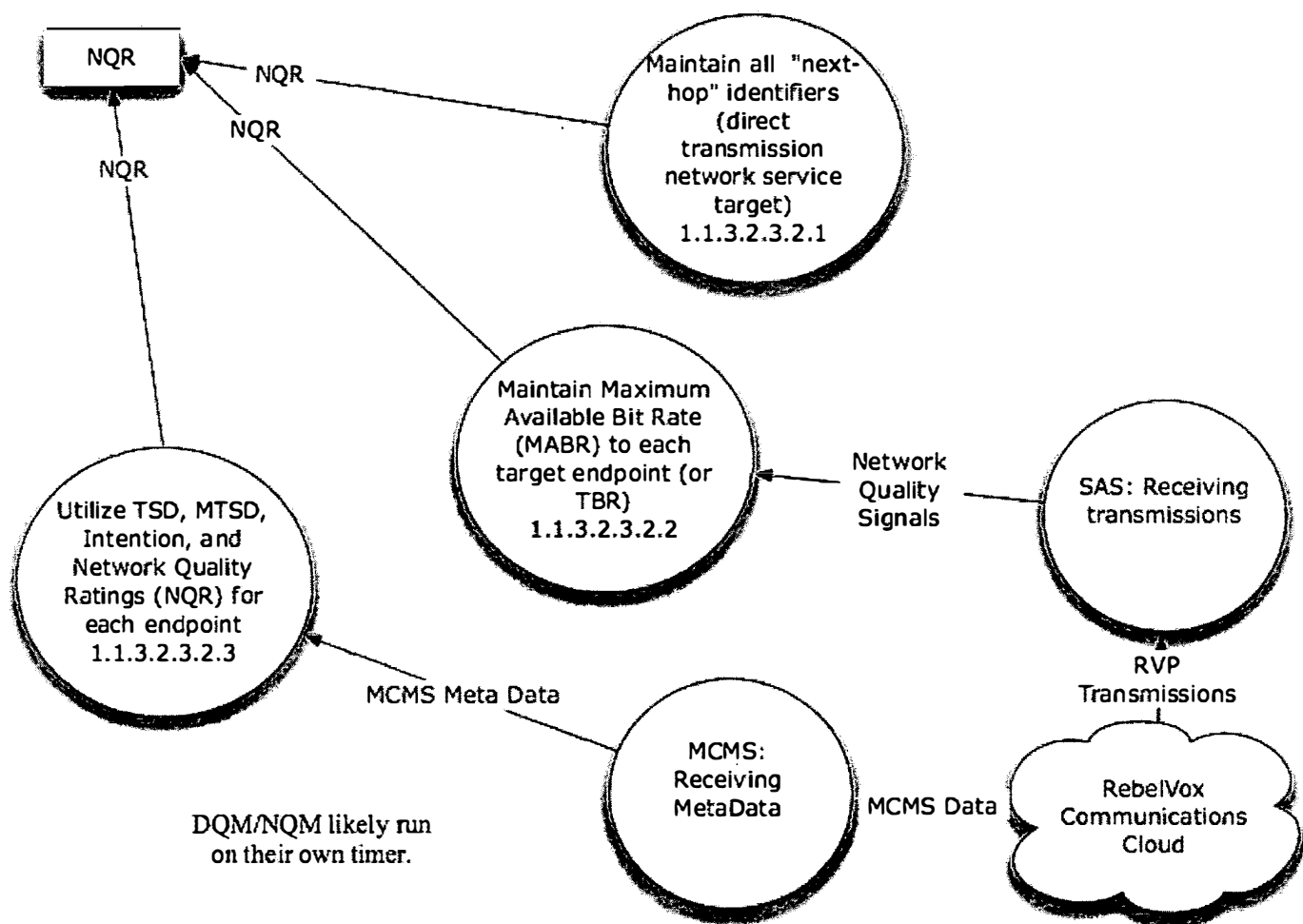


Diagram 1.1.3.2.3.2 SAS: Run the Network Quality Manager

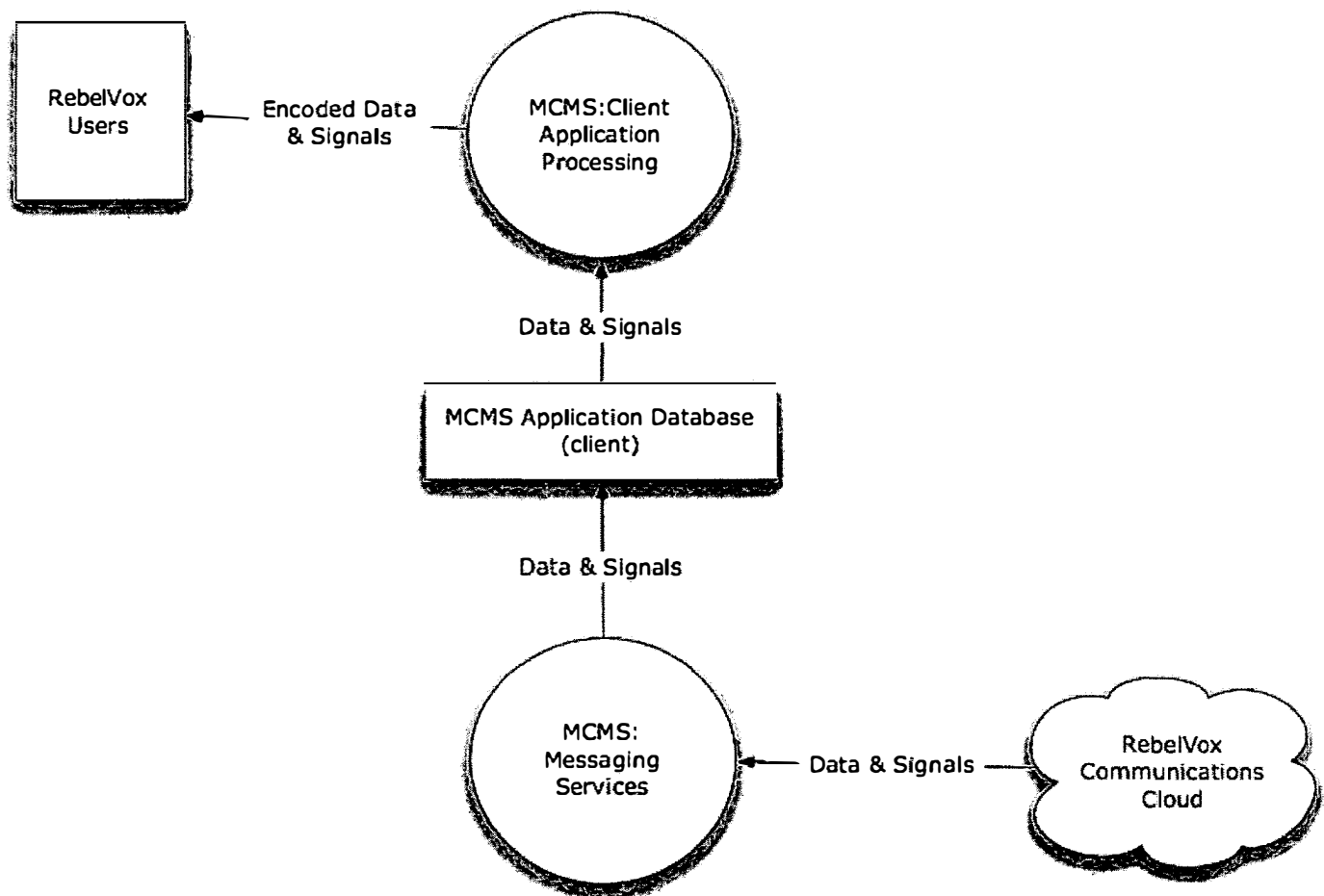


Diagram 1.1.4 : Manage Client-side Communications Processing

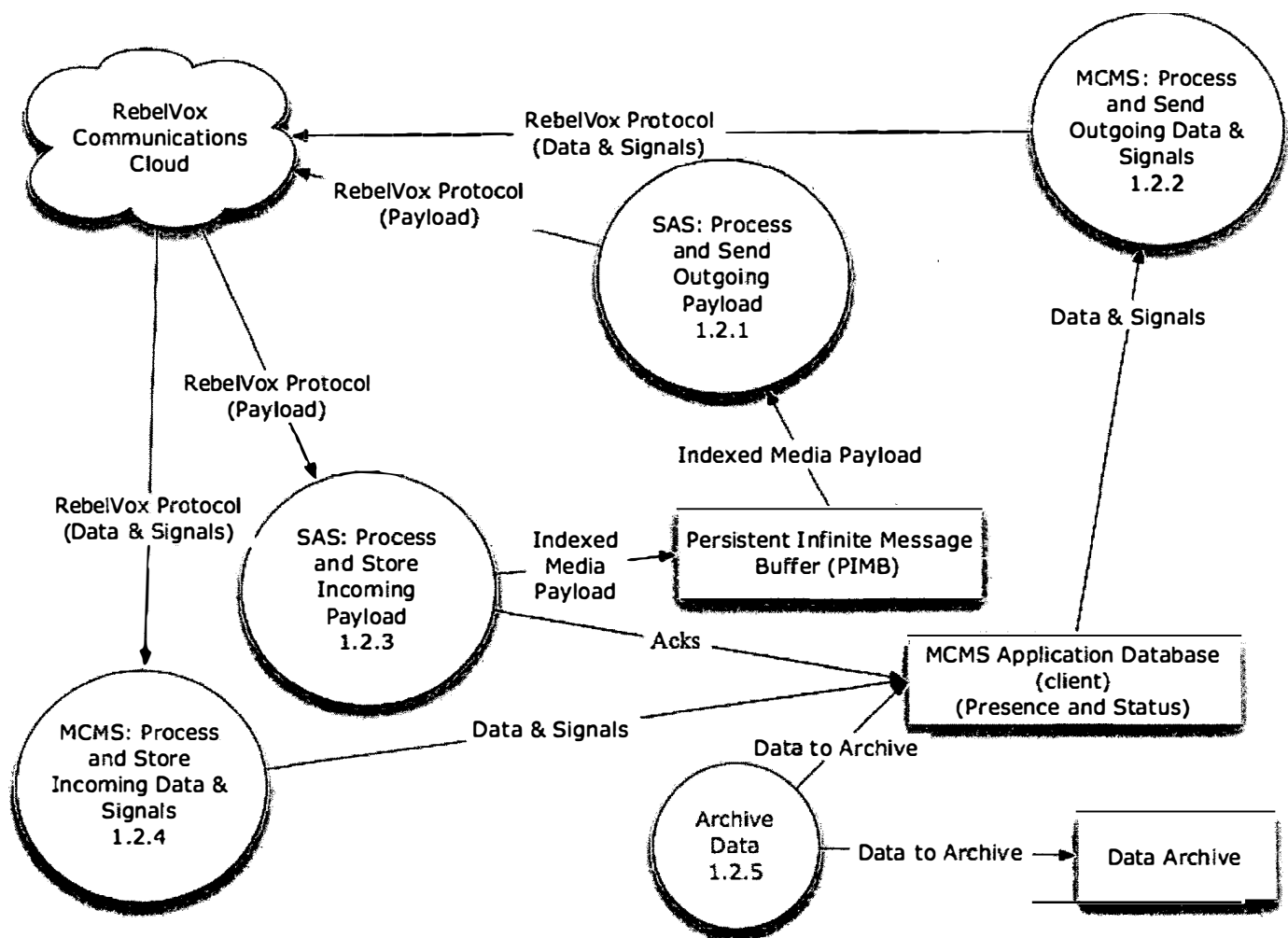


Diagram 1.2 : Manage Server-side Communications Processing

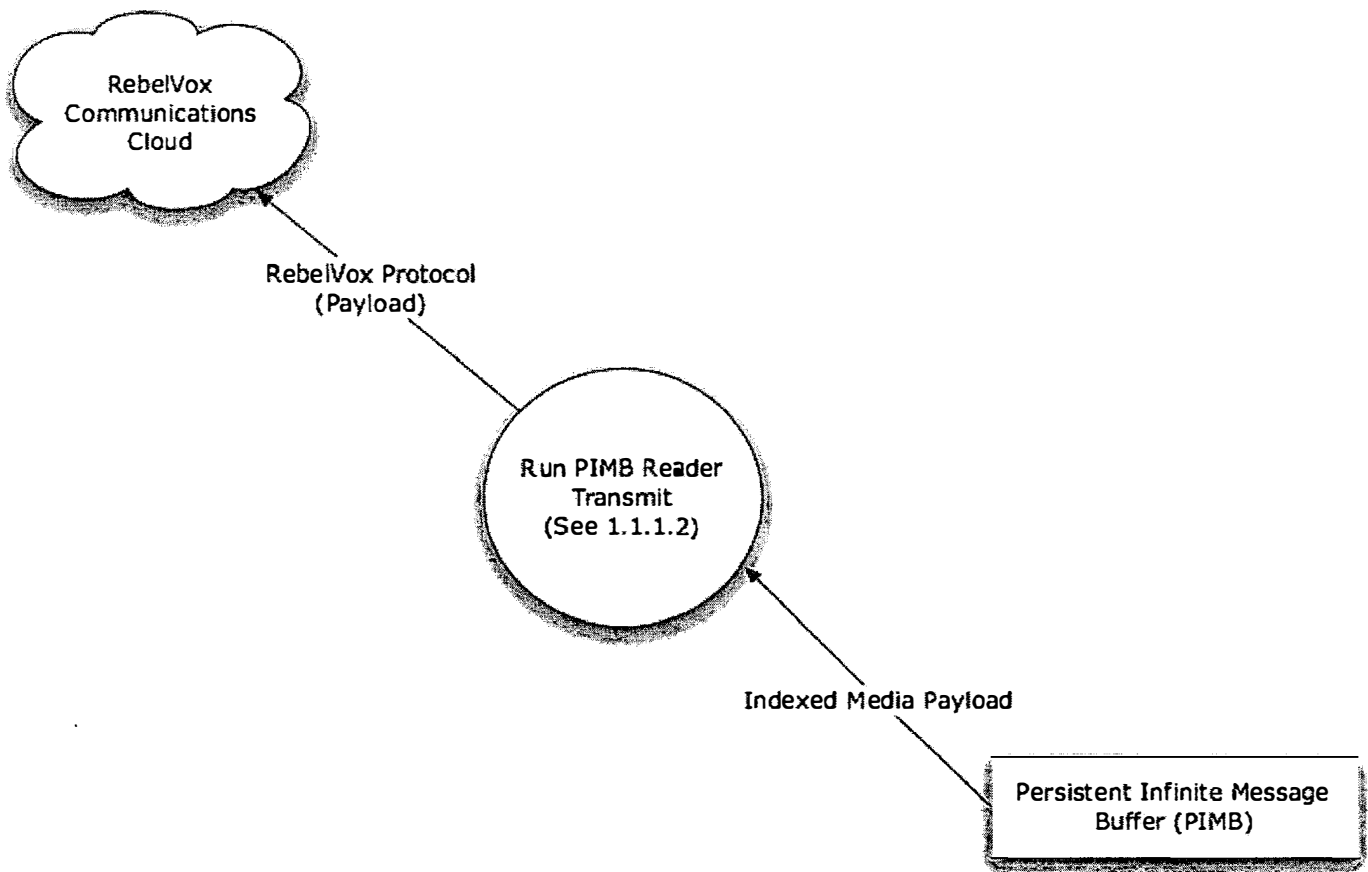


Diagram 1.2.1 : SAS: Process and Send Outgoing Payload

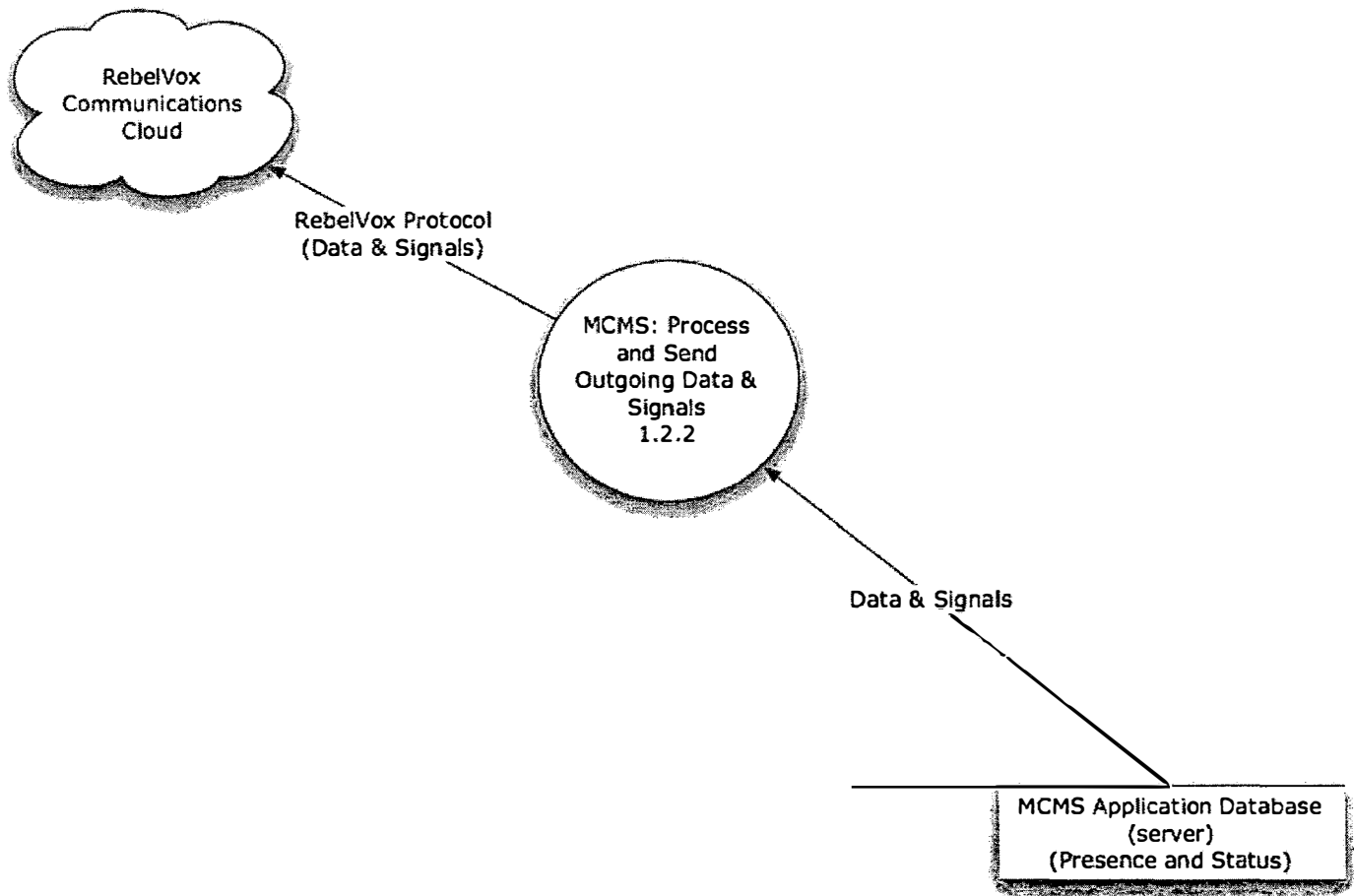


Diagram 1.2.2 MCMS Process and Send Outgoing Data & Signals

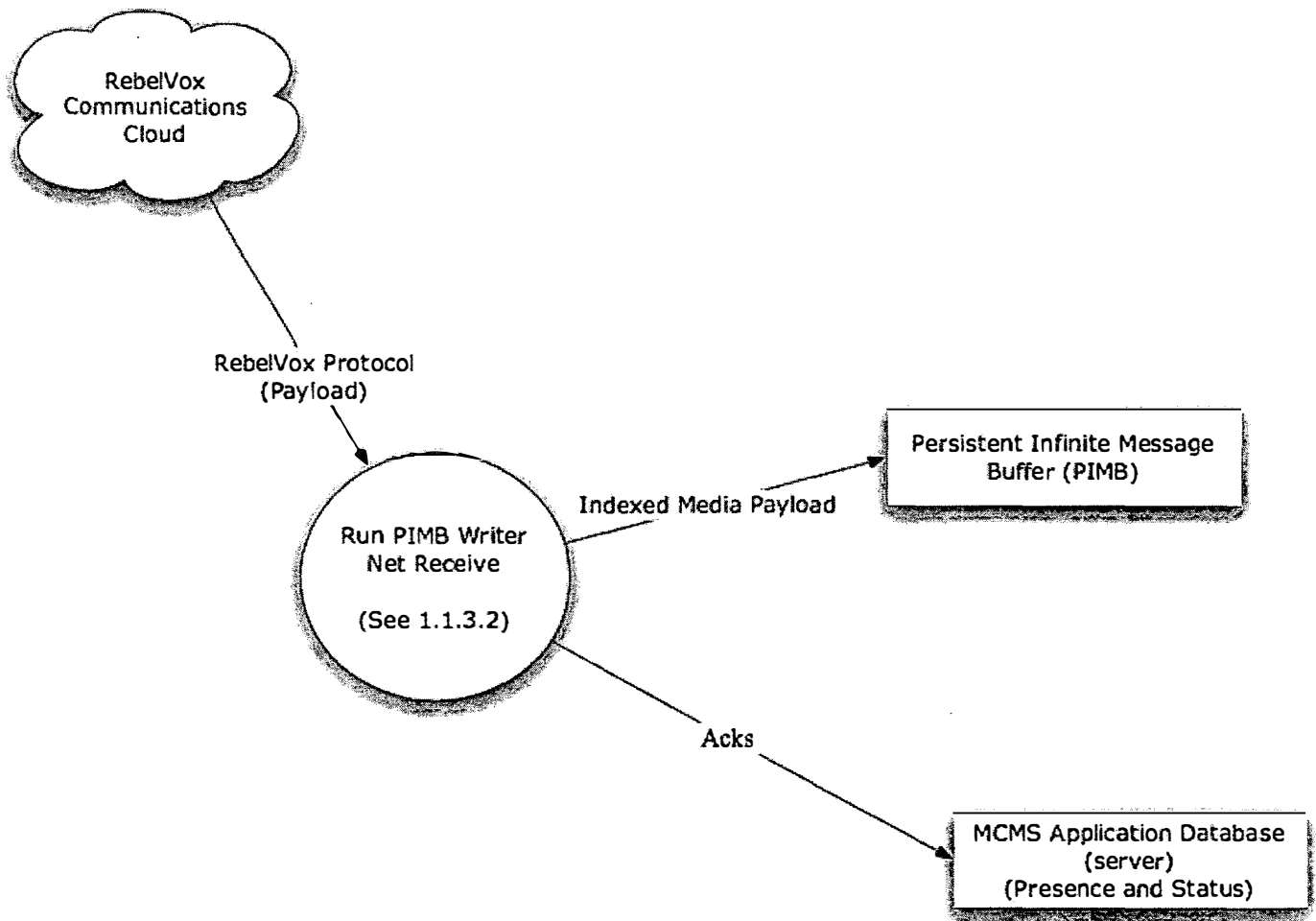


Diagram 1.2.3 : SAS: Process and Store Incoming Payload

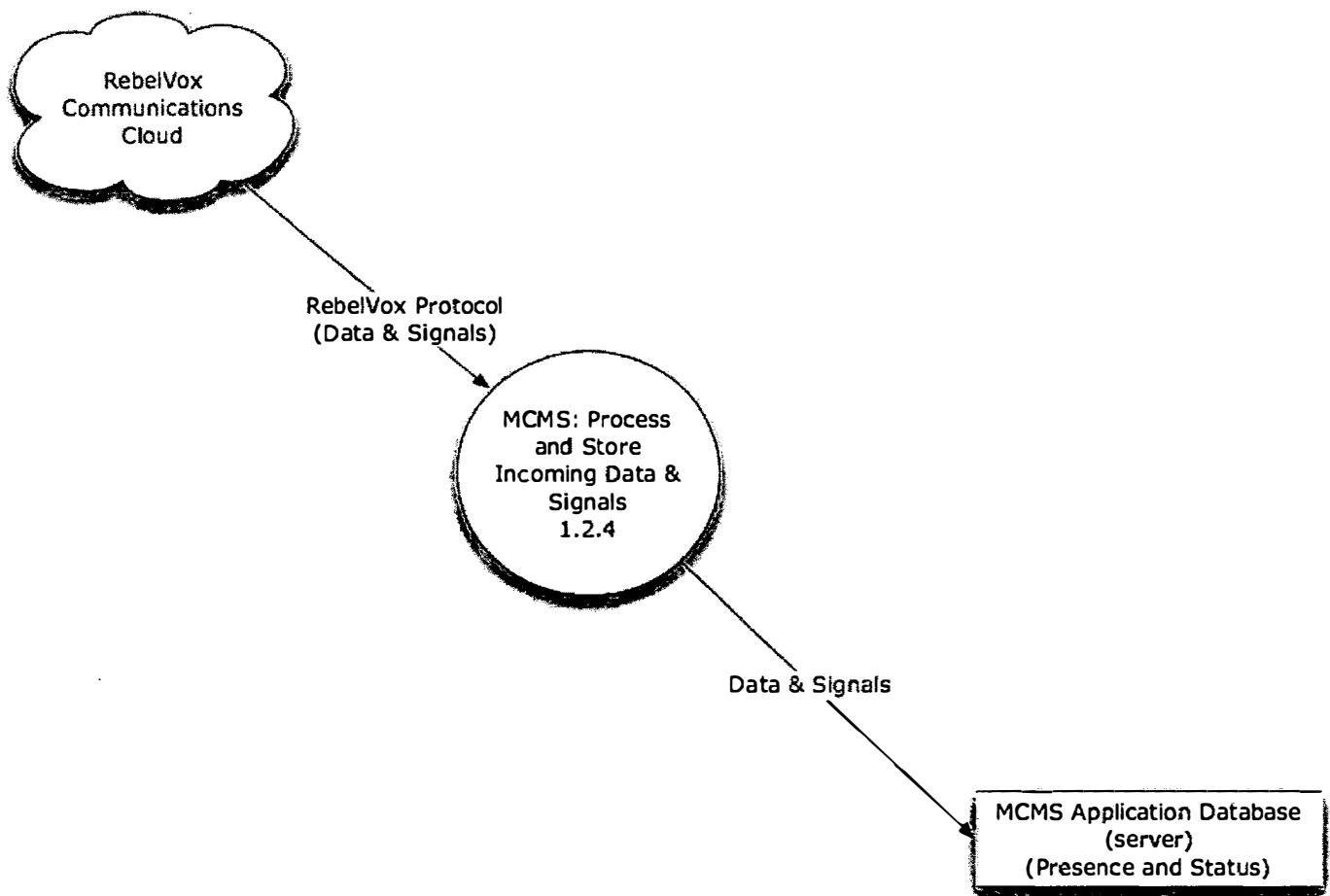


Diagram 1.2.4 : MCMS: Process and Store Incoming Data & Signals

For more explanation, see section PR 4.2.1.4.

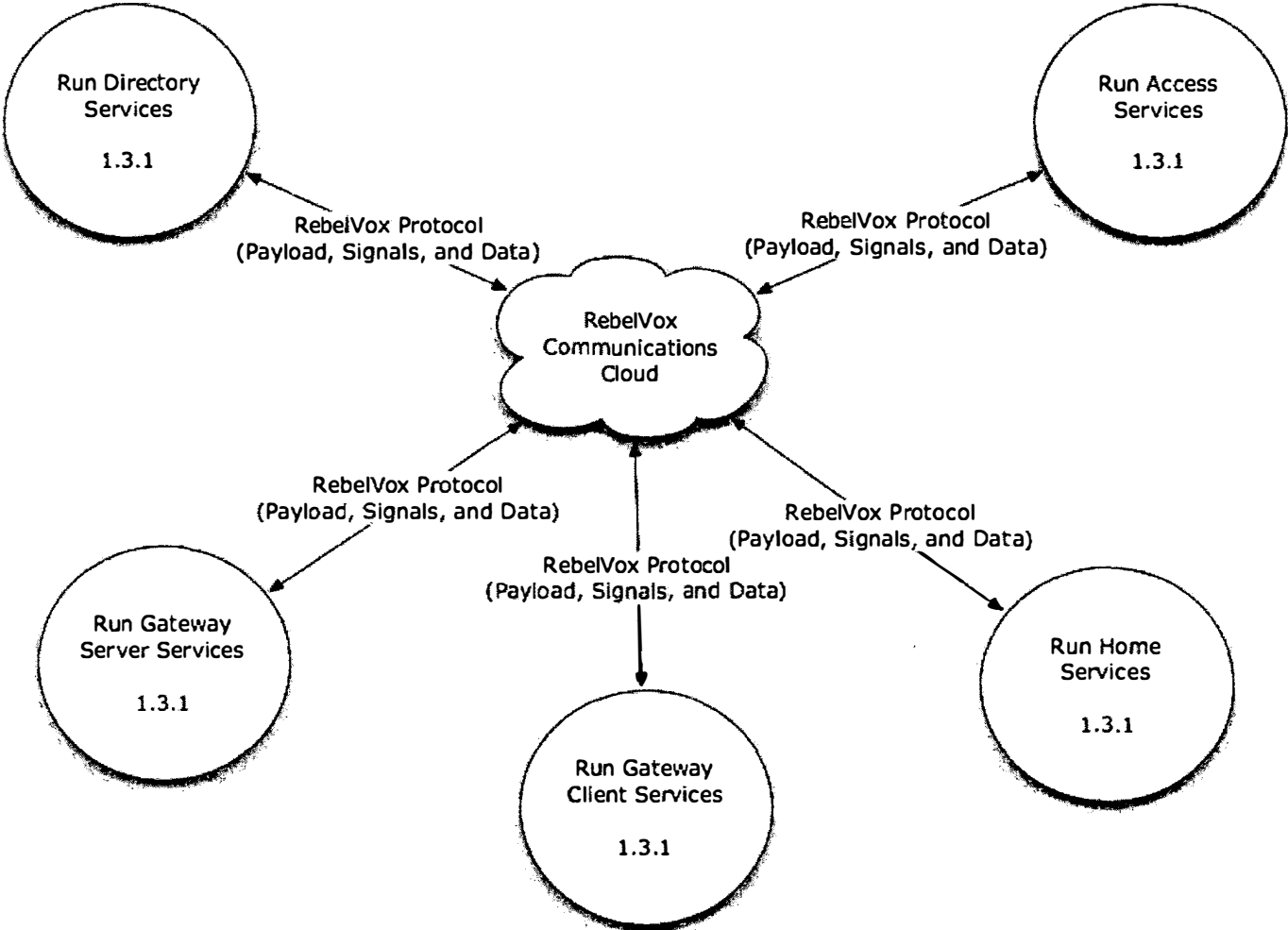


Diagram 1.3 : Manage Wide-area Network Routing