RAVENNA

Draft 1.0

Operating Principles

This document describes the operating principles of RAVENNA - the Real-time Audio Video Enhanced Next-generation Networking Architecture.

ALC NetworX

ALC NetworX GmbH

Am Loferfeld 58
81249 Munich
GERMANY

Fon: +49 (89) 44236777-0
Fax: +49 (89) 44236777-1

@: info@alcnetworx.de

www.alcnetworx.de

Version information:

Draft 1.0 - 2011-06-01

Any information contained in this document may be changed or amended at any point in time without further notice. The document may also be replaced as a whole.

RAVENNA - Operating Principles

DRAFT 1.0

TOC

1 INTRODUCTION

RAVENNA is a solution for real-time distribution of audio and other media content in IP-based network environments. Utilizing standardized network protocols and technologies, RAVENNA can integrate and operate on existing network infrastructures. Performance and capacity are scaling with the capabilities of the underlying network architecture. It emphasizes data transparency, tight synchronization, low latency and reliability. It aims at applications in professional environments, where networks are planned and managed, and where performance has to surpass the requirements of consumer applications.

This document describes RAVENNA and specifies what implementers of compatible equipment need to comply with in order to successfully interoperate with equipment from other manufacturers.

This is a first draft document that may be amended or supplanted with further issues in the future.

1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 (1).

1.2 Abbreviations

ADC	Analog-to-Digital Converter
AES	Audio Engineering Society
AES11	a standard covering the synchronization of digital audio signals
AES3	a digital audio standard used for carrying digital audio signals between devices (also known as AES/EBU)
AM824	audio data format defined in IEC 61883-6
DAC	Digital-to-Analog Converter
DHCP	Dynamic Host Configuration Protocol (RFC 2131)
DiffServ	Differentiated Services, a mechanism for providing Quality of Service (QoS) guarantees on IP networks (RFC 2474)
DNS	Domain Name System (RFC 1034)
DNS-SD	DNS-based Service Discovery (IETF draft-cheshire-dnsext-dns-sd-10)
DSCP	Differentiated Services Code Point (RFC 2474)
GPS	Global Positioning System
HTTP	Hypertext Transfer Protocol (RFC 2616)

I/O	Input / Output
ICMP	Internet Control Message Protocol (RFC 792)
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol (RFC 2236)
IP	Internet Protocol (RFC 791)
IPv4	Internet Protocol version 4 (RFC 791)
IPv6	Internet Protocol version 6 (RFC 2460)
L16	Linear PCM 16-bit audio
L24	Linear PCM 24-bit audio
LAN	Local Area Network
mDNS	multicast-DNS (part of Zeroconf specification)
MTU	Maximum Transmission Unit, the size of the largest packet that a network protocol can transmit
multicast	simultaneous transmission of messages to a group of network destinations identified by a virtual multicast group address (one-to-many transmission)
NIC	Network Interface Controller
Node	a device acting as connector or link between two domains or layers
OSI	Open Systems Interconnection, a joint ISO and ITU-T standard for computer networks and communication protocols
OSI model	a layered description for communications and computer network protocol design
PTP	Precision Time Protocol (an acronym for IEEE 1588)
QoS	Quality of Service
RAVENNA	Real-time Audio Video Enhanced Next-generation Network Architecture
RFC	Request for Comments, an IETF memorandum on Internet standards and protocols
RTCP	Real-time Transport Control Protocol (RFC 3550)

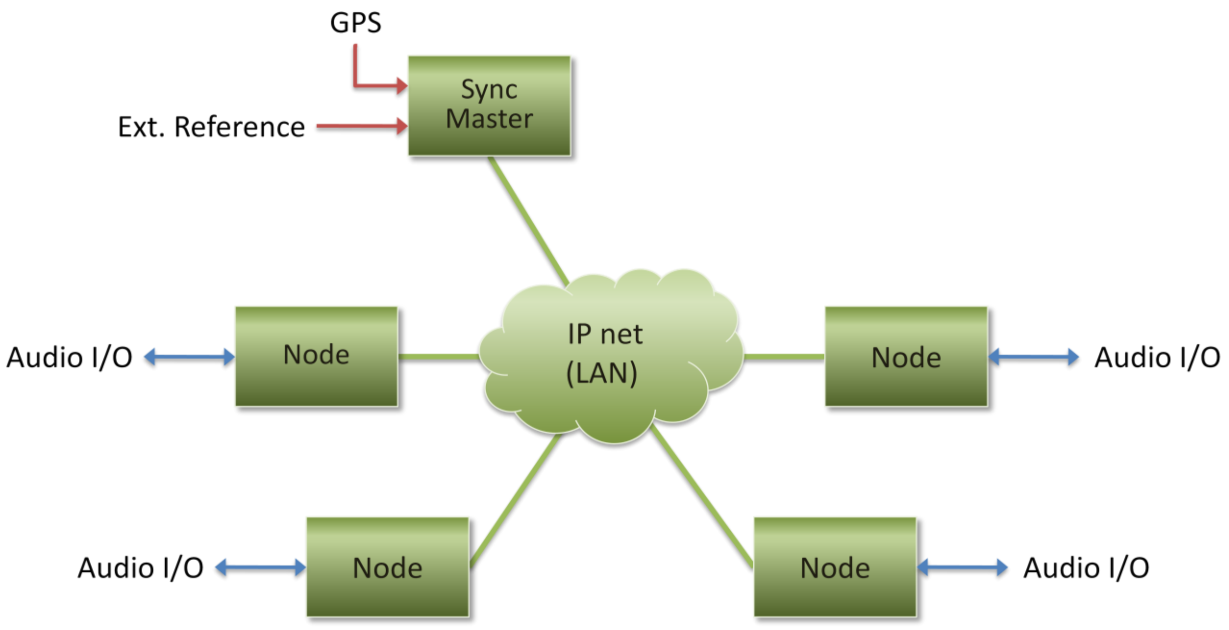
RTP	Real-time Transport Protocol (RFC 3550)
RTP/AVP	RTP Audio Video Profile (RFC 3551)
RTSP	Real-Time Streaming Protocol (RFC 2326)
SDP	Session Description Protocol (RFC 4566)
TCP	Transmission Control Protocol (RFC 793)
UDP	User Datagram Protocol (RFC 786)
unicast	transmission of messages to a single network destination identified by a unique address (one-to-one transmission)
URI	Uniform Resource Identifier, consists of a URL and a URN (RFC 3305)
URL	Uniform Resource Locator (RFC 1738)
URN	Uniform Resource Name (RFC 1737)
WAN	Wide Area Network
Zeroconf	Zero configuration networking (RFC 3927)

2 TECHNOLOGY OVERVIEW [INFORMATIVE]

2.1 Basic Components

Basic components of a RAVENNA system comprise of:

- An IP-capable network infrastructure
- A master clock device
- Any number of RAVENNA-enabled I/O nodes



BASIC COMPONENTS OF A RAVENNA SYSTEM

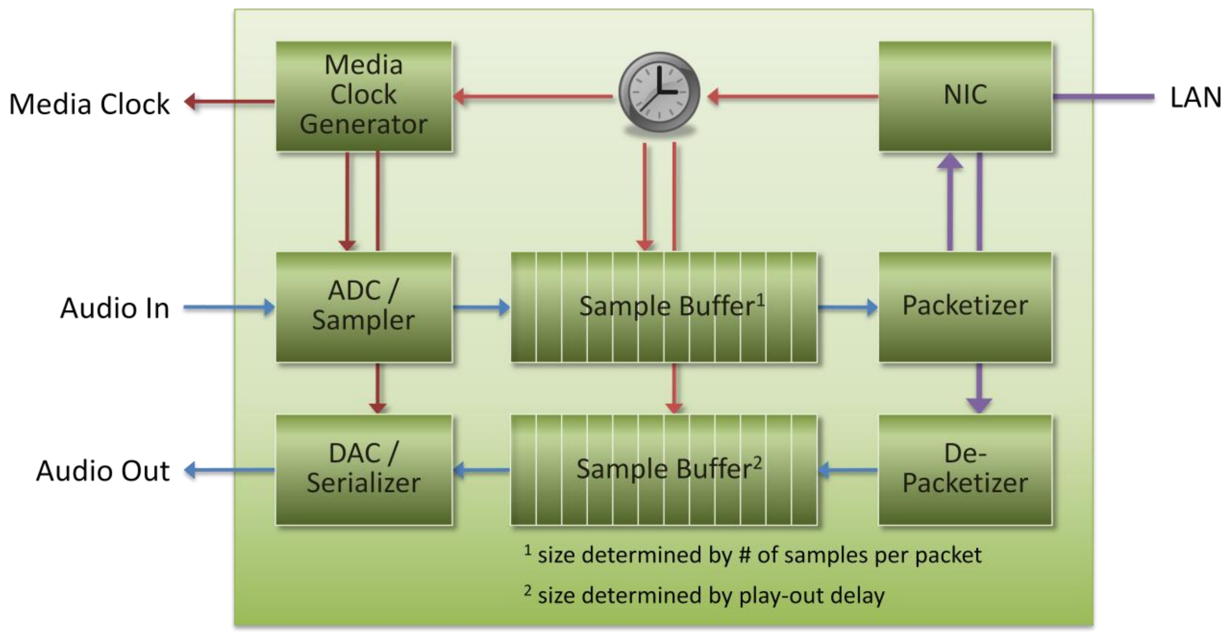
The network infrastructure needs to be able to transport IP packets and has to support several standardized operating protocols. As the network infrastructure or its behavior is not controlled or managed by any RAVENNA node or service, any configuration required to meet the expected performance has to be administered by the system operator or network administrator.

The master clock can either be a dedicated device or any RAVENNA node capable of serving as a grandmaster. Using GPS as the preferred time domain reference is an option that can help with reliable and reproducible operation.

RAVENNA nodes translate the real-time media signals into IP packets and vice versa. A functional block diagram of a typical RAVENNA node is discussed below.

2.2 Basic Operation of a RAVENNA Node

The following drawing highlights the basic functional blocks of a RAVENNA node:



BASIC COMPONENTS OF A RAVENNA NODE

2.2.1 Time Synchronization & Media Clock Generation

- PTP packets arrive through the NIC and synchronize the local clock to the wall clock (determined by a grandmaster)
- The desired media clock is synthesized from the local clock
- The synthesized media clock can be output for external device synchronization (house or word clock functionality)

2.2.2 Transmit Operation

- Incoming audio is sampled with the internal media clock
- Each sample receives a precise time stamp from the local clock and is stored in the transmit sample buffer
- Once the desired number of samples has been acquired, they are packetized into RTP packets and sent out via the NIC (or via both NICs in case of network redundancy operation)

2.2.3 Receive Operation

- Incoming RTP packets are depacketized and stored into the receive buffer according to their individual time stamps (samples depacketized from packets received on the second NIC are stored into the same buffer location potentially overwriting previously stored samples)
- Play-out time offset is determined by maximum expected jitter delay and potential correlation offset with other streams

- At due time samples are transported from the sample buffer to the play-out stage with the internal media clock

2.3 Standard protocols

All protocols and mechanisms used within RAVENNA are based on widely deployed and established methods from the IT and audio industry or comply with standards as defined and maintained by international standardization organizations like IEEE, IETF, AES and others. One way of looking at RAVENNA is to see it as a collection of recommendations on how to combine existing standards to build a media streaming system with the designated features. Adherence to RAVENNA therefore also means that these base standards must be followed, to this end the base standards are to be considered part of RAVENNA.

2.3.1 IP

RAVENNA is an IP-based solution. As such it is based on protocol levels on or above layer 3 of the OSI reference model. IP can be transported by virtually any LAN and is used as the base layer for communication across WAN connections. Although Ethernet will be deployed in most cases as underlying data link layer, IP is in general infrastructure-agnostic and can be used on virtually any network technology and topology.

2.3.2 Streaming

As IP has been chosen as a basis, it's only natural to use RTP for streaming of media content. RTP is widely used and supported by numerous applications and comes with a large number of standardized payload formats. This would even allow standard media player applications to potentially subscribe to RAVENNA streams.

Streaming is supported both in unicast and multicast mode providing the highest flexibility to match the distinct requirements of different applications.

A receiver can subscribe to any existing RAVENNA stream through RTSP / SDP protocol. Again, this is one of the standard methods used across the internet and is therefore supported by most common media players.

2.3.3 Synchronization

While simple streaming across a network can be achieved without any synchronization at all, in pro-audio applications a tight synchronization between all devices and streams is absolutely mandatory. While playback synchronization in most applications requires sample accuracy, it has been the goal for RAVENNA to optionally provide superior performance by providing phase-accurate synchronization; this would render the separate distribution of a reference word clock throughout a facility or venue obsolete.

In RAVENNA, synchronization across all nodes is achieved through IEEE1588-2008 (PTPv2 Precision Time Protocol), another standard protocol which can be operated on IP. PTPv2 provides means for synchronizing local clocks to a precision as defined in AES-11. Accurate synchronization can even be reached across WAN connections, when GPS is used as a common time domain.

2.3.4 QoS

As different services can co-exist with RAVENNA on the same network, it has to be ensured that RAVENNA streams will be expedited with priority.

For IP-based traffic, several QoS schemes have been defined as standards over time. For RAVENNA, DiffServ has been selected as QoS mechanism as it is widely supported by most modern managed switches. RAVENNA packets receive a high priority classification ensuring expedited transport across the network, while other packets with lower priority will be treated as best-effort traffic.

2.4 Full Network Redundancy

Although a modern network infrastructure can be configured to guarantee a high level of transport security and reliable 24/7 operation, a RAVENNA setup can optionally support full network redundancy. Each RAVENNA device may exhibit two independent network interfaces which can be connected to independent physical networks. By duplicating any outgoing stream to both network links, any destination device will receive the full stream data on both network interfaces independently. If data from one link is corrupted or one network link fails completely, the uncorrupted data is still present on the other link. An internal mechanism ensures flawless and uninterrupted continuation of play-out.

3 OPERATING PRINCIPLES [NORMATIVE]

RAVENNA operation is based on these fundamental principles:

- Time synchronization & media clock generation
- Streaming of media data, including payload formatting and packaging
- Connection management
- Service discovery and advertisement
- Monitoring

While each of the principles is described separately, they only form a complete solution when implemented in conjunction.

3.1 Prerequisites

RAVENNA uses IP network infrastructure as a basis to build upon. Nodes therefore need network interfaces and protocol support for the IP family of protocols. This support is a standard component of many operating systems and programming environments, but not necessarily present in an embedded system. Specifically, a minimum implementation MUST support IP (2), UDP (3), TCP (4), IGMP (5), ICMP (6) and protocols specific to the underlying transport, most commonly Ethernet.

In this first version of RAVENNA, we use IPv4 only. IPv6 (7) support is planned for a future version of RAVENNA.

RAVENNA makes use of multicast. The IP infrastructure MUST therefore support IP multicast in addition to unicast, including support for IGMP.

Support for DiffServ (8) is REQUIRED for both end-nodes and for the network infrastructure. Media and sync data is transported in packets that are tagged with a different class of service than those for ordinary traffic, allowing the network infrastructure to use prioritized forwarding of packets, thereby minimizing jitter effects and reducing the risk of dropping these packets in a congestion situation. Since usage of specific DSCP (8) values is highly depending on application- and network-specific scenarios, these values SHALL be configurable. Configuration guidelines can be found in (9). However, since DiffServ is not a bandwidth allocation scheme, no guarantee can be offered that streams will always have the bandwidth they need for uninterrupted streaming. If any sort of bandwidth reservation scheme is available in a particular network environment, it MAY be deployed or configured by administration to help prevent media packet loss due to congestion.

We do not specify how a network node obtains its basic network configuration, i.e. IP-address and netmask, amongst other things. While some simpler installations may be content with a fixed, manual configuration, most applications will benefit from more sophisticated services, such as DHCP (10) and DNS (11) (12). Whilst it is the decision of each manufacturer what level of support she or he is willing to integrate into a device, we RECOMMEND to include the possibility to let the user select between at least manual and DHCP-supported network configuration, preferably with the additional alternative of Zeroconf Address Autoconfiguration (13). RAVENNA assumes that this kind of configuration has already taken place, in other words it assumes that a node is already connected to the network and has its basic network configuration in force.

For media clock synchronization, RAVENNA relies on PTPv2 (14), which imposes its own requirements to the underlying network. Whilst native support for PTPv2 is desirable in the network infrastructure, RAVENNA can work with somewhat reduced synchronization accuracy when such support is missing.

RAVENNA assumes that packet delivery in the underlying network is quite reliable and incurs little delay variance. It does not automatically guard against packet loss, hence lost packets will usually be noticeable as dropouts in a stream. Redundant streams are offered as a way to guard against this, but they must be used specifically when required, and they obviously consume more network resources, potentially aggravating the problem. Before employing RAVENNA, one must verify whether the underlying network provides sufficient reliability and latency for the task at hand. The open internet will often be too unreliable or unpredictable for a satisfactory usage with RAVENNA. If RAVENNA is to be used for WAN connections, one must usually obtain connections with better quality than those that can be established on the general internet. This is by design.

3.2 Time synchronization & media clock generation

In order to provide data-accurate streaming over unlimited amounts of time, while maintaining low latency across the link, the media clocks of sender and receiver are synchronized tightly. Most commonly known IP-based networked media solutions derive the media clock information from incoming streams or work with sample rate conversion. With RAVENNA, all participating nodes derive their media clocks from a synchronized time base. Thus, all RAVENNA nodes have to run local clocks synchronized to a common wall clock. RAVENNA allows compliance with AES11 (15), as long as there is PTPv2 support in the network infrastructure. This chapter describes how this is achieved.

In order to determine AES11 compliance, the network - including end-nodes - is being regarded as a single device in the context of AES11. It follows that there are restrictions in the relative phase of reference signals being accepted or generated by those end-nodes. RAVENNA provides a framework for permitting the network to comply with AES11, but in order to bring this about, the nodes themselves also need to be constructed to comply with AES11. RAVENNA RECOMMENDS each node that has inputs and/or outputs which can serve as reference points for AES11 to be constructed such that the timing of the signal at those points is AES11 compliant.

Media clock distribution in RAVENNA is built upon PTPv2, which is used to distribute an accurate notion of absolute time on the network. Each node derives a local copy of absolute time, which is used as the basis for synthesizing a local media clock. The task hence divides itself into two parts:

3.2.1 Distribution of absolute time

PTPv2 is a protocol that can distribute absolute time on an IP network. A grandmaster acts as a source of time, and any number of slave nodes replicate the distributed time locally. The accuracy of the local clocks relative to the grandmaster's clock depends on the network infrastructure. With support throughout the network, the accuracy can be significantly better than +/- 100 ns, but even without specific support accuracies of a few µs can be achieved.

Any node can in principle assume the role of a grandmaster; PTPv2 automatically selects the highest quality clock that is available (using its Best Master Clock Algorithm). This can be used for clock backup purposes, to allow a grandmaster change while the system runs uninterrupted.

Larger systems will usually be best served by having a dedicated grandmaster similar to a "house-clock" that is common in installations that use traditional interconnections. Such dedicated grandmasters can be referenced to primary time sources such as GPS, allowing the media clocks to run in lockstep with "official" time.

RAVENNA nodes are REQUIRED to support PTPv2 slave operation with a configurable clock domain (default is domain "0"). Operation as a grandmaster is OPTIONAL.

A RAVENNA node is not required to derive all its time information from PTPv2. It is acceptable and normal for a node to depend on separate and asynchronous time bases for different functions. Hence, one should not assume that the media clock and/or the PTP time has any defined relationship with "wall clock time" or "system time". In some systems it may, in other systems it may not. This includes the possibility that different PTP domains are being used simultaneously in the same network for different purposes.

In a network that uses several PTP domains in parallel, we RECOMMEND to use PTP domain "0" as the domain that distributes wall-clock time, possibly synchronized to a primary standard. With the informal term "wall clock time", we associate a time scale that you would normally use in a system (such as a general purpose computer) to represent the actual calendar time. It does not imply that there is a technical link between wall clock time and a primary standard, but such a link is certainly often desirable.

Detailed PTP recommendations for RAVENNA are contained in a separate document (16)

3.2.2 Media clock synthesis

A node synthesizes its own media clock from the absolute time reference established by PTPv2. Nodes which require the clock as a hardware function, i.e. to drive interface chips or converter chips, need to have a hardware clock synthesizer that is referenced to the PTP timer that is built into the network interface hardware.

If no hardware clock is required, the media clock may be treated as an abstract concept realized in software only. For example, a device which records media streams onto persistent storage may not need a hardware media clock.

The choice of media clock frequencies which can be generated in a node is decided by the manufacturer of the device. For audio, we RECOMMEND to support at least a rate of 48000 Samples/s, to provide some common ground for stream interchange. Depending on the application, further rates may need to be supported.

RAVENNA assumes that the local media clock generator is configured outside of RAVENNA, and does not attempt to change the settings dynamically, i.e. depending on the streams. Rather, it is assumed that the user of a system will want to configure this based on operational requirements outside the realm of RAVENNA. RAVENNA takes the local media clock as given, and determines if a stream that is to be sent or received has properties that are compatible with this setting. If the stream's synchronization source and media clock frequency does not match that of the local node, the node rejects the connection attempt.

3.2.2.1 Precise clock frequency

The local media clock synthesizer MUST be precise. This means that the clock frequency or period must be related to the time reference in such a way that no cumulative error is introduced, which would cause a growing phase difference between the actual clock relative to the ideal clock. Relative phase between the time reference and the media clock is such that if the clock would extend indefinitely into the past and future, one sampling point would coincide with the epoch of the time reference. The epoch is "point zero" of a time scale; in the case of PTPv2, the epoch is 1st January 1970, 00:00 UTC.

Other properties of the synthesized clock, for example the phase jitter and its spectrum, are application dependent. Equipment manufacturers would specify those according to their needs in the device, or requirements of other standards they need to comply with.

A node may have several media subunits with independent clock generation. For example, a node may have several audio interfaces which can be clocked independently.

3.2.2.2 Pull-up or pull-down clocks

Some RAVENNA nodes may elect to support pull-up and pull-down rates in addition to the nominal sampling rates. These are clock frequencies that deviate from the nominal value by a defined rational number. Such deviations have applications in conjunction with video, but might also be useful to tune a sampling rate to an existing instrument. Support for this is OPTIONAL, including the choice of deviation ratios that a node supports. If support is present for this, the generated clock frequency MUST still be precise.

A pull-up or pull-down clock is described by a nominal frequency, modified by a ratio of two integers. A ratio that commonly occurs in conjunction with video is 1000/1001.

3.2.2.3 A stream as time reference

There are cases when one may not want to reference the media clocks to absolute time. For example, one may want to run a stream asynchronously (i.e. its timing is derived from a free-running oscillator which is unrelated to absolute time), and reference the media clock to this stream. The stream may exist solely to serve as the time reference, or it may also carry media data. This situation is more common in consumer-type installations, but it does occur in professional settings, too. For such situations, a future revision of RAVENNA will specify means of running a stream asynchronously, and using it as the time reference for other streams. PTPv2 will still be needed in this case, in order to establish a common time base for relating the streams to each other. Support for this will remain OPTIONAL.

3.3 Media streaming

This chapter describes how media data is formatted and packaged for transmission on the network.

RAVENNA nodes MUST support multicast and SHOULD also support unicast. The media data is packaged according to RTP/AVP (17) (18), which uses UDP over IP. The set of supported payload formats is an application-dependent decision made by each manufacturer. For audio nodes, support for the following payload formats is REQUIRED: L16 (18) and L24 (19). Nodes SHOULD accept any reasonable number of channels per stream on reception, from 1 up to a certain performance-related limit, even if they don't support this on transmission.

An additional RECOMMENDED payload format for audio nodes is not yet formally standardized: It is a 32-bit payload format to allow transparent transmission of all data included in AES3 (20) (21) (22) (23) sub frames, including sideband-data. We have a proposal on the basis of AM824 (24) which needs to be ratified in due course.

Further payload formats MAY be supported at the manufacturer's discretion. A further revision of RAVENNA may recommend additional payload formats.

RTCP (17) messages MUST be sent by transmitter and receiver as mandated by the RTP protocol.

3.3.1 Packetization

The number of samples per channel in a packet is a decision made by the transmitter. The maximum packet size (MTU) SHALL not be exceeded to avoid performance problems associated with packet fragmentation. Jumbo frames SHOULD not be used, because they require support by the network infrastructure throughout, and because they increase packet jitter (even when background traffic is allowed to use jumbo frames).

A receiving node MUST NOT assume any particular number of samples per channel in each packet, except when a payload format definition mandates this. It MUST even be prepared for this number to change during the lifetime of a stream. Most transmitting nodes will pick a number and stick to it while sending a stream, and this is indeed the RECOMMENDED behavior of a transmitting node, but there may be valid reasons for changing the packet size mid-stream, or for alternating between two sizes to approach a non-integer size on average. The choice is made by the transmitting node, and it is not communicated to receivers in advance.

3.3.2 Stream timing

A transmitting node is REQUIRED to transmit packets as soon as possible after the samples that fill the packet have been assembled. This somewhat fuzzy rule is intended to give some headroom for implementers while emphasizing that buffering should be minimized on transmit, in the interest of minimizing overall latency. Transmit timing is thus bound to the local media clock in the transmitter, with some timing uncertainty introduced by concurrent network traffic and concurrent tasks on the transmitting node.

Receiving nodes provide a receive buffer of a certain minimum size to cover latency requirements of various application domains and network environments. The actual size of the buffer is application dependent and decided by the manufacturer; in the case of audio nodes we RECOMMEND a buffer size of at least 1024 samples per channel.

The timestamp in each RTP packet denotes the "creation time" of the first sample in the packet, with an arbitrary offset that is fixed before the stream starts. The offset is communicated along with other stream parameters to allow proper decoding and time-aligning at the receiver, as described later. In essence, the stream can be thought to have its own local media clock, in the same way as transmitter and receiver have their local media clocks. All of these clocks have their own arbitrary offset which can be expressed as the clock count that was reached at the epoch of the time reference. Apart from individual (but constant) offsets, the three clocks must run synchronized, or else the connection cannot be made. Transmitting and receiving nodes must relate their internal media clock to the media clock of the stream, correcting the offset in the process. The receiver also takes into account the selected latency at this point.

3.3.3 Stream description

All information about a stream is collected in a document as defined by SDP (25). An SDP document describes a session, which in turn contains one or more streams. Exchanging the document between transmitter and receiver nodes is the business of connection management, which is described later.

A receiver needs the SDP document in order to be able to connect to the stream and to receive and decode it correctly. The receiver determines from the document the IP address and port numbers for a stream, the synchronization source (a RAVENNA-specific extension), the payload format along with channel count and sampling rate (in case of an audio stream), and additional stream parameters if required. Those are checked against the settings of the local media interface in order to determine whether the stream can be correctly received.

3.3.3.1 SDP-extension: session-level attribute to specify synchronization source

All streams of a session are assumed to relate their timing to a common synchronization source, which is specified as a session-level attribute of the following form (in the case of PTPv2 absolute time as the synchronization source):

```
a=clock-domain:PTPv2 <domainNumber>
```

RAVENNA streams SHALL always include this attribute in their SDP description. In the case that a receiver encounters a session description where this attribute is missing, it SHOULD assume that the synchronization source for the streams in the session is unknown, and MAY reject reception because of this. It MAY alternatively allow a local user to override the setting and choose a synchronization source manually, but this incurs the risk that a stream suffers incorrect and intermittent reception when the choice is poorly made. Receivers also MAY use rate adaptation techniques as described in RTP to compensate for rate mismatches between transmitter and receiver; this is essentially what other RTP implementations would do, at the expense of signal quality and/or data transparency. The latter could be called a "non-RAVENNA mode".

At present, only PTPv2 is defined as a valid synchronization source, and the PTP domain number must be given with it – omission of the domain number is not permitted.

Further valid synchronization sources may be added in future versions of RAVENNA.

3.3.3.2 SDP-extension: stream-level timestamp association

In order to relate the RTP timestamps of a stream to absolute time of the specified synchronization source, a further attribute is given at stream-level:

```
a=sync-time:<RTP-format-timestamp>
```

The RTP-format-timestamp is a decimal value representing a 32-bit unsigned integer. It specifies the RTP timestamp that would have been used in the stream in a packet that corresponds to the epoch of the synchronization source (if the stream had been running at that time). This takes into account the wrap-arounds that are inherent when using a 32-bit number.

Since in RAVENNA, the media clocks of both transmitter and receiver are derived from the same synchronization source with zero cumulative error, they don't drift with respect to each other. The same applies to the stream between them. Hence the timing can be established in advance of stream reception, and need not be aligned or corrected while streaming. The attribute relates the stream time to the common synchronization source, and hence allows reproducible timing on all receivers of the same stream.

If a receiver sees this attribute, a clock-domain attribute (see above) must also be present in the session, otherwise the attribute is ignored. If a clock-domain attribute is present, but the stream lacks a sync-time attribute, then the receiver cannot determine with certainty how the stream relates to absolute time, hence phase-accurate playback is not possible. The receiver MAY reject reception of a stream because of this, or it MAY employ other means to guess at a reasonable timing.

3.3.3.3 SDP-extension: clock-deviation for a stream

A stream may be marked to have a clock deviation from its nominal rate. This can be used for pull-up / pull-down applications as described above. The stream-level attribute is given like this:

```
a=clock-deviation:<numerator>/<denominator>
```

Both numerator and denominator must be given, and they must be unsigned decimal integers. The attribute as a whole is OPTIONAL. When it is not given, the stream's nominal rate is assumed to be in force; in other words, if omitted, the clock-deviation is assumed to be 1/1.

The actual rate for the stream results from a multiplication of the nominal rate by the given rational number. For example, a clock-deviation of 1001/1000 results in a 0.1% pull-up, which converts a nominal 48000 Samples/s into an actual 48048 Samples/s.

Note that such calculations must be made with no error. This typically requires rational arithmetic, as floating point arithmetic will produce rounding errors in certain cases.

Support for clock deviations is OPTIONAL in both transmitters and receivers. Receivers, however, MUST recognize the attribute, so that they can reject streams that don't match the local media clock.

3.4 Connection management

RAVENNA uses RTSP (26) to establish and control streaming sessions. RTSP is a protocol similar to HTTP, which allows establishing connections and controlling them from remote nodes. Transmitting nodes act as RTSP servers, receiving nodes as RTSP clients. Thus, RTSP support MUST be implemented.

The RTSP DESCRIBE method is used by the client to retrieve the session description in SDP format from the server.

In addition, it is RECOMMENDED that each node offers a basic configuration and control service through a web page accessible via HTTP (27). Page design is up to the manufacturer, but it should attempt to offer a simple and comprehensive means of configuring the parameters of the device, and to control/monitor the streams transmitted and/or received by the node.

HTTP control of the device configuration SHOULD always be available, even when additional means of control and connection management are implemented by the node. Note that it is perfectly reasonable to provide several concurrent ways of control and management in a device.

3.4.1 RTSP client and server behavior

With RTSP a client can connect to an existing session (by using the URL (28) (29) path of the session in the RTSP request). Once a connection is established the RTSP server can inform a client (asynchronously) of session changes by pushing an updated SDP to the client using RTSP ANNOUNCE.

Within RAVENNA it is desirable that streaming sessions, that are not advertised using DNS-SD (see below), can still be discovered by clients if at least the RTSP service hosting the session is known and the session name or ID is known (usually the session name will be more useful than the session ID in this scenario). However, if the session is not fully configured (or does not exist at all) the RTSP server would normally reply with a NOT FOUND message forcing the client to poll the server until the session becomes available.

In order to allow this type of discovery, the RECOMMENDED behavior for a Ravenna RTSP server is to respond with a NOT FOUND message as usual, but mind the client and keep the connection alive. This allows the server to push SDP updates to the client once a new session is available and fully configured and avoids the need for the client to poll.

3.5 Service Discovery and Advertisement

Service discovery requires a net-wide database that can be queried for available services. For each service, an entry must be created in the database, and it must be removed when the service becomes unavailable. RAVENNA uses DNS-SD (30) for this purpose. DNS-SD is not formally ratified as an internet standard at this point in time. The defining document is a draft only, even though implementations have been available for a number of years.

DNS-SD harnesses the infrastructure of DNS (11) for service discovery. In environments where DNS is already being used and a DNS server infrastructure is available, service discovery can be covered by the existing servers. For dynamic addition and removal of services, the server MAY have to offer Dynamic DNS Update (31) (32).

For environments where no DNS server is available or where dynamic update is not possible, a multicast-based alternative mDNS (33) SHOULD be supported. mDNS is part of the Zeroconf protocol suite, so Zeroconf implementations will contain what is needed. No formal standard has been issued for mDNS so far, the specification is in draft stage.

Each RAVENNA node acting as a session source (i.e. a microphone) MUST publish its RTSP service. In turn each node acting as a session sink (i.e. a speaker) MUST publish its HTTP configuration service. Additionally a node MAY publish sessions that are available at the node and thus can be subscribed to using the RTSP service.

3.5.1 Basic service advertisement

The HTTP and the RTSP service are advertised with their standard service types. The service names used for the registration are:

- a "vendor node ID" combined from the vendor ID, the product ID and the product serial number, i.e. "`My-Corp devicex 1234-5678`"
- the user defined device name

This results in 4 service names registered:

- `<vendor node id>._http._tcp.`
- `<vendor node id>._rtsp._tcp.`
- `<user defined node name>._http._tcp.`
- `<user defined node name>._rtsp._tcp.`

To enable browsing specifically for RAVENNA services, we additionally register "ravenna" sub types with the "vendor node ID":

- `<vendor node id>._ravenna._sub._http._tcp.`
- `<vendor node id>._ravenna._sub._rtsp._tcp.`

The registration of these services allows a (DNS-SD) browser to discover all RAVENNA nodes in a network or a specific (known) RAVENNA node using the "user defined node name".

3.5.1.1 DNS-SD Service Text

With DNS-SD a service MAY provide an optional service text when publishing the service. This service text (according to the standard) SHOULD NOT contain any information that is not also available via the service protocol run by the service. Within RAVENNA this service text MAY be used to optimize service discovery by adding information about the node (i.e. the node type).

3.5.1.2 Name collisions

The service registration with the "vendor node ID" should never fail since this ID is always unique for each device (responsibility of the vendor).

The service registration of the "user defined node name" may fail if the user sets the same name on different devices. In this case a node SHOULD flag the problem to the user (i.e. via the User Interface). A device SHOULD NOT generate a unique service name and register that instead (which is a standard strategy for many other DNS-SD services).

3.5.2 Streaming session advertisement

Nodes MAY also advertise the streaming sessions configured in order to make the sessions directly discoverable. A streaming session is advertised with:

3.5.2.1 Session name

Session names are REQUIRED to be unique (within this DNS-SD realm) and are published as a separate DNS-SD service. Using unique session names allows for instance to pre-configure a backup node and bring it into the network if the primary node fails. In this case the session name is REQUIRED to have exactly the same session configuration (i.e. same multicast addresses, same stream configuration(s) ...).

If the session name is advertised, it MUST also carry the "ravenna_session" sub service.

3.5.2.2 Content identifier

Content identifiers are not unique. They are registered as DNS-SD sub services of the session name. Sessions that share the same content identifier MUST carry the same content, however, the streams MAY have different configurations (i.e. different sample rates etc.). Content identifiers may be used for backup scenarios or may also be used for load balancing.

Note: If a content identifier is given, but no session name, the RTSP session ID SHOULD be used to register the service.

A streaming session may be advertised once it is fully configured and started. Streaming sessions that are not fully configured or currently disabled by a user SHOULD not be visible via DNS-SD.

Note: Changing the session name should not affect any already connected streaming clients. If an existing session is changed in any incompatible way, then the session should be destroyed and re-created.

3.5.3 Addressing streaming sessions

Streaming sources are addressed within RAVENNA using URIs (29).

3.5.3.1 RTSP service addressing

The RTSP service of any RAVENNA device can be addressed by either:

- a standard RTSP URL: i.e. "`rtsp://<host>:<port>`"
  RTSP services addressed this way obviously will try to connect to whichever node (RAVENNA node or not) uses that address
- a Zeroconf service address using the RAVENNA URI notation: `ravenna:<vendor node id> | <user defined node name>` i.e. "`ravenna:someVendorNodeId`"
  A URI of this type can be transformed into an RTSP URL by resolving the RAVENNA RTSP service via DNS-SD.

3.5.3.2 Session paths using the session ID

Every streaming session configured on a device MUST have a unique session ID (usually represented as an integer value). An URL path addressing a streaming session via its ID takes the form: "`/by-id/<session id>`". Since session IDs are not advertised in any way, a receiver node must receive the session ID by some external means (i.e. a node with no configuration options may only ever create a single session with the ID "1".

Valid URIs using a session path with a session ID are for example:

```
"rtsp://some_host/by-id/0815"
"ravenna:some_vendor_node_id:/by-id/1"
```

3.5.3.3 Session URLs of named sessions

Named sessions may also be addressed using the format: "`/by-name/<session name>`". So instead of addressing a session by its ID it also is valid to address a session by its name, i.e.

```
"rtsp://some_host/by-name/mySpecialSession"
"ravenna:some_vendor_node_id:/by-name/Line"
```

Sessions that have been advertised, however, may also be addressed using the following URI (29) notation: "`ravenna_session:<session name>`". This URI can again be transformed into a full RTSP URL by resolving the RAVENNA session with DNS-SD: `rtsp://<resolved host>:<resolved port>/by-name/<session name>`. Note that this time resolving the URI with DNS-SD will yield the resolved host / port address as usual. But combining it with the "/by-name" addressing permits to form a complete RTSP URL and allows to directly open an RTSP session on the node. Here is a complete example:

The session "special-session" on node "xyz" port 8080 is advertised as

```
"special-session._rtsp._tcp."
```

with the host / port address of the service set to "xyz" and 8080 respectively.

The URI "ravenna_session:special-session" is used to address the session and will resolve into:

```
"rtsp://xyz:8080"
```

By adding the path of the named service we get the full RTSP URL of the stream:

```
"rtsp://xyz:8080/by-name/special-session"
```

3.6 Monitoring

Streams can be monitored remotely by receiving their RTCP traffic. Additionally, each node is RECOMMENDED to provide monitoring functions for the local streams on the web pages that it serves through its HTTP service.

4 REFERENCES

1. **Bradner, S.** *Key words for use in RFCs to Indicate Requirement Levels.* s.l. : IETF, March 1997. RFC 2119.

2. **Postel, J.** *Internet Protocol.* [ed.] IETF. September 1981. RFC 791.

3. —. *User Datagram Protocol.* [ed.] IETF. August 28, 1980. RFC 768.

4. —. *Transmission Control Protocol.* [ed.] IETF. September 1981. RFC 793.

5. **Cain, B., Deering, S., Kouvelas, I., Fenner, B. and Thyagarajan A.** *Internet Group Management Protocol, Version 3.* [ed.] IETF. October 2002. RFC 3376.

6. **Postel, J.** *Internet Control Message Protocol.* [ed.] IETF. September 1981. RFC 792.

7. **Deering, S. and Hinden, R.** *Internet Protocol, Version 6 (IPv6) Specification.* [ed.] IETF. December 1998. RFC 2460.

8. **Nichols, K., Blake, S., Baker, F. and Black, D.** *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers.* [ed.] IETF. December 1998. RFC 2474.

9. **Babiarz, J., Chan, K. and Bakers, F.** *Configuration Guidelines for DiffServ Service Classes.* [ed.] IETF. August 2006. RFC 4594.

10. **Droms, R.** *Dynamic Host Configuration Protocol.* [ed.] IETF. March 1997. RFC 2131.

11. **Mockapetris, P.** *Domain Names -- Concepts and Facilities.* [ed.] IETF. November 1987. RFC 1034.

12. —. *Domain Names - Implementation and Specification.* [ed.] IETF. November 1987. RFC 1035.

13. **Cheshire, S., Aboba, B. and Guttman, E.** *Dynamic Configuration of IPv4 Link-Local Addresses.* [ed.] IETF. May 2005. RFC 3927.

14. **TC-9.** *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems.* [ed.] IEEE. July 24, 2008. IEEE Std 1588-2008.

15. **Grant, J.** *AES recommended practice for digital audio engineering - Synchronization of digital audio equipment in studio operations.* [ed.] AES. 2009. AES11-2009.

16. **ALC NetworX.** *Ravenna PTP Generic Profile Version 1.0.* 2011.

17. **Schulzrinne, H., Casner, S., Frederick, R. and Jacobson, V.** *RTP: A Transport Protocol for Real-Time Applications.* [ed.] IETF. July 2003. RFC 3550.

18. **Schulzrinne, H. and Casner, S.** *RTP Profile for Audio and Video Conferences with Minimal Control.* [ed.] IETF. July 2003. RFC 3551.

19. **Kobayashi, K., Casner, S. and Bormann, C.** *RTP Payload Format for12-bit DAT Audio and 20- and 24-bit Linear Sampled Audio.* [ed.] IETF. January 2002. RFC 3190.

20. **Grant, J.** *AES standard for digital audio — Digital input-output interfacing — Serial transmission format for two-channel linearly-represented digital audio data — Part 1: Audio Content.* [ed.] AES. 2009. AES3-1-2009.

21. —. *AES standard for digital audio — Digital input-output interfacing — Serial transmission format for two-channel linearly-represented digital audio data — Part 2: Metadata and Subcode.* [ed.] AES. 2009. AES3-2-2009.

22. —. *AES standard for digital audio — Digital input-output interfacing — Serial transmission format for two-channel linearly-represented digital audio data — Part 3: Transport.* [ed.] AES. 2009. AES-3-2009.

23. —. *AES standard for digital audio — Digital input-output interfacing — Serial transmission format for two-channel linearly-represented digital audio data — Part 4: Physical and electrical.* [ed.] AES. 2009. AES-4-2009.

24. **IEC.** *Consumer audio/video equipment - Digital interface - Part 6: Audio and music data transmission protocol .* October 28, 2005. IEC 61883-6 ed2.0.

25. **Handley, M., Jacobson, V. and Perkins, C.** *SDP: Session Description Protocol.* [ed.] IETF. July 2006. RFC 4566.

26. **Schulzrinne, H., Rao, A. and Lanphier, R.** *Real Time Streaming Protocol (RTSP).* [ed.] IETF. April 1998. RFC 2326.

27. **Fielding, R., Gettys, J., Mogul, J. C., Frystyk, H., Masinter, L., Leach, P. and Berners-Lee, T.** *Hypertext Transfer Protocol -- HTTP/1.1.* [ed.] IETF. June 1999. RFC 2616.

28. **Berners-Lee, T., Masinter, L. and McCahill, M.** *Uniform Resource Locators (URL).* [ed.] IETF. December 1994. RFC 1738.

29. **Berners-Lee, T., Fielding, R. and Masinter, L.** *Uniform Resource Identifier (URI): Generic Syntax.* [ed.] IETF. January 2005. RFC 3986.

30. **Cheshire, S. and Krochmal, M.** *DNS-Based Service Discovery.* [ed.] IETF. February 27, 2011. draft-cheshire-dnsext-dns-sd-10.txt.

31. **Vixie, P., Thomson, S., Rekhter, Y. and Bound, J.** *Dynamic Updates in the Domain Name System (DNS UPDATE).* [ed.] IETF. April 1997. RFC 2136.

32. **Wellington, B.** *Secure Domain Name System (DNS) Dynamic Update.* [ed.] IETF. November 2000. RFC 3007.

33. **Cheshire, S. and Krochmal, M.** *Multicast DNS.* [ed.] IETF. February 14, 2011. draft-cheshire-dnsext-multicastdns-14.

Document information:

Authors:

Stefan Heinzmann

Ralf Michl

Andreas Hildebrand

Editor: Andreas Hildebrand

Version: Draft 1.0

Last revision: 2011-06-01 by hildeba1

File: RAVENNA Operating Principles - Draft 1.0_final.docx