

Today, RADVision's H.323 Protocol Stack, GateKeeper and family of H.323-based OnLAN multimedia communications products are licensed and packaged as core technology "building blocks" for IP telephony and multimedia conferencing. As an active participant in the ITU experts group with real world implementation experience, the company plays a key role in the continuing evolution of the H.323 standard.

Executive Summary

This white paper:

- Defines "H.323 INTEROPERABILITY" from the point of view of different communities of interest.
- Provides an overview of WHERE AND WHY INTEROPERABILITY ISSUES ARISE within the H.323 universe.
- Discusses STANDARD-RELATED SOURCES of interoperability problems.
- Describes issues arising during implementation of the H.323 PROTOCOL STACK "CORE".
- Explains BROAD ARCHITECTURE ISSUES including supplementary services, security, multiplexing, etc.
- Illustrates APPLICATION-SPECIFIC H.323 ENTITY ISSUES relating to terminals, gatekeepers, gateways and multipoint calls.
- Discusses what developers can do at each level of implementation to REDUCE INTEROPERABILITY PROBLEMS in current and future product releases.
- Reviews the appropriate steps that developers, implementers and all those vested in the success of H.323 products can take to ASSURE SMOOTH INTEROPERABILITY of multi-vendor H.323 deployments.

H.323 Interoperability

"H.323 interoperability" means different things to different people. To the end user who just wants to get on with business and assume that everything

works, H.323 interoperability means being able to communicate with another person anytime, using any device. It's really that simple. And if it's not...it's up to network designers and managers to make it work!

To the vast majority of network managers, systems integrators and deployers, H.323 interoperability is the ability to take any new H.323-compliant product that a user or workgroup wants to introduce and "drop" it into an existing H.323-ready network with minimum planning and no risk of network disruption. There are very few "H.323-ready" networks in existence today that would make the life of network managers that simple. However, it is safe to predict that tools to simplify the configuration of networks for multimedia communication will be available and, perhaps, someday this IT nirvana can be achieved.

For developers, H.323 Interoperability has a specific technical definition. It means that all H.323-compliant products conform to H.323 specifications and that all messages and parameters can be exchanged and correctly interpreted by H.323 endpoints to produce a predictable result. Thus, messages will either be interpreted by the receiving entity as intended by the sending entity, or an appropriate "do not understand" message is returned to the sender who will then send a new message until the entities communicate their intents to one another.

This white paper focuses on what developers need to know in order to deliver products that can provide end-to-end interoperability. It attempts to clarify specific implementation issues that developers face when building fully interoperable H.323 products and services.

The H.323 Universe at a Glance

H.323 is an umbrella specification developed by a consortium of computing, telephony and computer networking experts in the internationally recognized ITU (International Telecommunications Union) forum. The standard provides a framework for developing H.323-compliant products and services, and its widespread adoption is driving the growth of IP telephony and multimedia conferencing. The ultimate goal behind the standard is to make end-to-end interoperability a reality.

To better understand the relationship between the standard itself and commercial products/services, it

helps to view the H.323 universe as a layered globe. The nucleus of the globe is the actual ITU specification. The intellectual property embodied in the specification is translated into "enabling software" which represents the first layer outside of the core.



Figure 1. H.323 Universe at a Glance

An H.323 stack is the most basic, low-level software product that is created by converting the specification into program code. All H.323-compliant products contain an embedded H.323 protocol stack. The specification defines four different H.323 entities as the functional units of a complete H.323 network: gatekeepers, MCUs, gateways and terminals. An H.323 gatekeeper can be considered another enabling-software building block that is required for controlling and managing H.323 networks. (For more information about H.323 gatekeepers, see RADVision's "H.323 Gatekeepers" White Paper.)

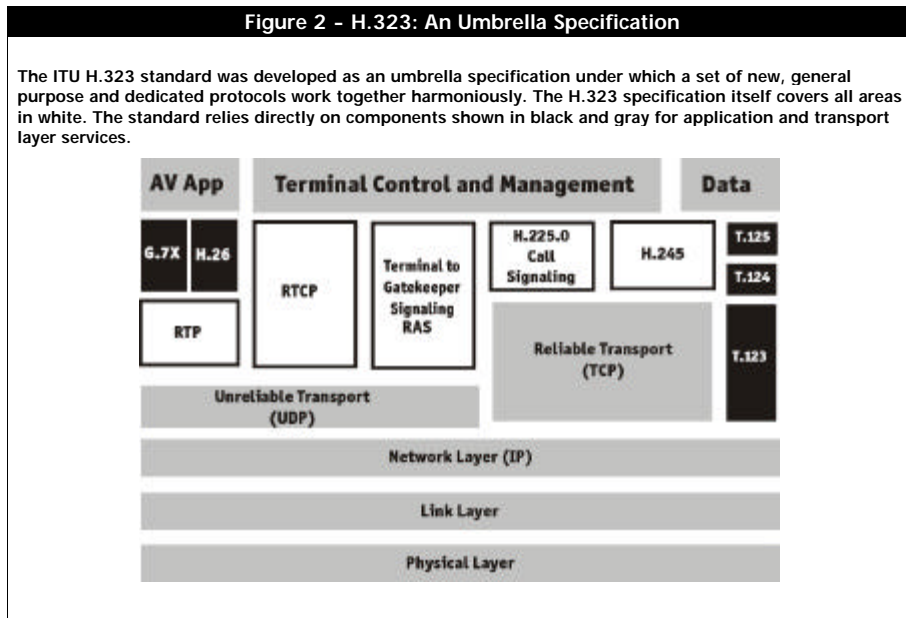
Developers use enabling software to build MCUs, gateways and terminals. These commercial products fall into the next layer of the H.323 globe. Since the standard only provides a development framework and does not define specific implementations, developers can use core H.323 software to develop H.323-compliant products with unique value-added features.

Network deployers, system integrators, and service providers select from an array of H.323 products to build a complete H.323 network capable of providing IP-based multimedia communication. This is the third layer of the H.323 universe.

The final layer of this model encompasses IP telephony and multimedia conferencing end-user applications.

As discussed earlier, developers, deployers and end-users all have different perspectives on interoperability. Companies like RADVision that implement core H.323 technologies must address the needs of all players in the H.323 universe. Product developers typically focus on deployers concerns. In turn, deployers are most concerned with what they should provide in order to satisfy users with business needs.

We will use the layers of the H.323 globe described above to organize our discussion of the different issues developers face when building interoperable solutions. This metaphor will allow us to take into account the different perspectives of developers, deployers and end users.



First, we will discuss interoperability issues arising directly from the specification itself. Next, we will discuss issues that arise from different protocol stack implementations. Then we will turn to a few higher level architectural issues that the development community is wrestling with. Finally, we will present a number of application-specific issues that need to be taken into account when building H.323 entities from lower level components.

- associating messages among different H.323 protocols
- backward compatible Fast Connect procedures
- multiplexing call signaling and call control messages
- emergence of single use devices (SUDs)

We will now look at each of these in more detail.

Encoding/decoding

Part of the current interoperability problems can trace their origins to how the H.323 standard was defined. H.323 incorporates H.225, RTP and RTCP. Each of these protocols needs to be encoded according to a consistent formula.

All H.225 is based on ASN.1 for encoding and decoding. Yet, RTP and RTCP, which come from the IETF, are based on TLV (Type Length Value) formula. There can be interoperability problems connected directly with the use of these encoding and decoding formulas.

Standard-Related Sources of Interoperability Issues

Interoperability issues can stem from many different sources in the H.323 specification itself. This section will examine potential interoperability problems arising in the following areas:

- different encoding and decoding methods
- inheriting/adapting portions of previous protocols
- assuring graceful backward compatibility

Example: Different implementations of H.323 use different ASN.1 compilers. Compilers differ in their limitations (such as OBJECT IDENTIFIER length) and the accessibility of configuration features in a particular compiler. Therefore, some protocol definitions and procedures can be performed using one compiler and not another. Although designers of the compilers are aware of this situation and have cooperated with developers to reduce problems, these differences can be a source of interoperability issues.

Inheriting/interpreting portions of previous protocols. As we indicate above, H.323 borrows protocols from diverse fields, including computing, data networking and telecommunications.

The committee that developed the H.323 specification not only adopted previously existing (and proven) protocols, but also interpreted them for the unique requirements of packet-based (which we use synonymously with IP) conferencing.

RTP/RTCP is a case in point. Real Time Protocol (RTP) and its control protocol (RTCP) were approved by the IETF prior to the development of H.323. RTP/RTCP is a relatively mature protocol with many details and fields for mandatory and optional procedures (especially RTCP) which are partially overlapping with other protocols defined within the H.323 umbrella. Some system designers will conform rigidly to the RTCP guidelines, while others may think "it's H.323" and the functionality is redundant, so these developers may choose to omit certain fields when interpreting the standard. Therefore, one implementation may differ and fail to correspond to implementations of other developers, leading to interoperability problems.

Example: In RTCP, the "cname" field is mandatory. It assigns a unique name to a media stream. Although it is required according to the original RTP/RTCP specification, it is redundant in the H.323 context. In H.323, stream identification and association is done by means of H.225 (using H.245). Some H.323 vendors don't provide the "cname" field. Others (using the original RTP/RTCP implementations) don't receive the messages without it! This prevents systems from interoperating.

Q.931 (in the context of H.225) is a call signaling protocol borrowed from ISDN and subsequently modified for use in H.323 products. Specifically, H.323 interprets this protocol for use in packet networks. When used in H.323, the layout of the original Q.931 protocol was kept but the meanings of many fields were redefined together with the

procedure's definitions for H.323. In addition, many fields in the original Q.931 remain undefined for H.323. Nevertheless, there are attempts to use these fields in the exact way they are used in ISDN.

Engineers working on H.323 deliberately interpreted and adjusted Q.931 because packet-based networks are inherently connectionless and rely on different architectures using completely different components and procedures. When developers use an original Q.931 implementation and run it as a part of H.323, they fail to comply with H.323 in the strict sense of the standard.

NOTE: The implementers' guide is a special resource that is intimately associated with the H.323 standard. It makes developers aware of special circumstances and clarifies language used in the specification where it is unclear, or where specific interpretations have been discussed and a solution adopted by the development community. Procedurally, the Implementers Guide becomes part of the standard (the guide is "automatically" integrated into the next version of H.323).

Example 1: Bandwidth allocation using Q.931 In H.225.0 v.2.0, call setup parameters for an outgoing SCN call are signaled using the bearer capability information element inherited from the Q.931 recommendation. The meaning of Bearer Capability Information Element (with its information transfer rate and rate multiplier octet fields) is defined differently for H.323 and H.320 due to the differences between packet and circuit switch networks.

Example 2: Overlapping digits (in conjunction with gatekeeper-assisted dialing) In the SCN world, where Q.931 assists in performing address resolution, the network devices begin address resolution while a terminal (an H.320 videoconferencing system, for example) is still dialing. This is called "overlapping".

In H.323 v.2 overlapping digits procedure is defined to work in conjunction with a gatekeeper because when it is present, the gatekeeper is responsible for the Address Resolution part of Call Setup. This procedure differs substantially from the "original" Q.931 procedure, although the same Setup and Setup Acknowledge messages are used. Where the gatekeeper is involved it performs address resolution using both RAS and Q.931 messages. And, to make matters even more complicated, in

H.323 v.1 the Setup Acknowledge message is not defined for use at all! This, in fact, leads us to the general topic of backward compatibility.

Backward compatibility

H.323 is a rapidly evolving specification. The process began in the spring of 1996 and version 1 was determined by the ITU in May 1997. Version 2 was already on its way when version 1 was determined. Version 2 passed final determination by ITU in January 1998. Similar to its precursor, version 3 is currently under development though version 2 products have yet to be developed. We will examine some of the opportunities and risks associated with this protocol's high rate of expansion/refinement.

Backward compatibility is required by the ITU. While each new version represents a significant improvement, in terms of features and functionality, it also seeks to "fix" broken parts of the preceding version...and is not itself, entirely perfect.

In version 2, support for some advanced features, such as support for supplementary services and security, was introduced. They were deemed necessary for standardization in order for H.323 products to be fully interoperable and successful in the marketplace.

However, it seems that some vendors who pushed for advanced features to be part of the standard are not implementing them yet because their version 1 products approached the features using proprietary mechanisms. This definitely makes the transition between version 1 and version 2 more difficult and is a very fertile area from which many interoperability issues emerge today.

Backwards compatibility with mixed version 1 and version 2 topologies is a very interesting problem. Although ASN.1 is fully backwards compatible (this is one of the strengths of H.323), it depends on the ability of the existing (already deployed) version 1 products to gracefully understand the "version 1 portion" of the version 2 messages (without crashing) and continue normal behavior.

Complete pure and perfect backward compatibility is a high goal. And version 2 achieves it, if and when the implementations of both version 1 and version 2 have been written in such a way that the resulting products behave gracefully in situations where both versions exist.

A rapidly evolving standard has some negative aspects which implementers and users should keep in mind.

Association among the H.323 protocols

One of the challenges that developers who are implementing H.323 today face is intelligently "tying" (or associating) all the H.323 messages and protocols together in the context of the same call.

This was defined one way in the first version of the standard, but has been changed during the evolution of the protocol.

In version 1, implementers could choose one of two options when making the association between different messages in various protocols. They could use either:

1. conferenceID and CRV – original Q.931 fields, or
2. Source and Destination Call Signaling Address, and AnswerCall – fields that are defined in RAS and Q.931

Now the association of different protocol messages is defined much more precisely using CallIdentifier. The CallIdentifier field was added to all of the H.323 protocols, and as a result, every message that belongs to the same call is identifiable and evident. Sets of messages are easily "associated" with their partners.

Another benefit of changing to CallIdentifier is that the new technique covers all the topologies, thus simplifying life for developers and implementers going forward. However, in order to be backward compatible with any version 1 protocol, all subsequent product implementations will need to support all 3 of the call association methods defined to date.

Backward compatible Fast Connect procedure

H.323 is a highly flexible protocol, defining the means for the users to establish, run and control an arbitrary number of different streams (such as audio and media) in the context of the same H.323 call. The original full procedure requires 4 round trips of messages between endpoints before the first media stream is exchanged between the peers. The full set of messages includes:

- Setup/Connect (Q.931 procedure)
- Master-Slave Determination (H.245 procedure)
- Capability Exchange (H.245 procedure)
- Open Logical Channel (also H.245)

Fast Connect (sometimes referred to as "Fast Start") is a new procedure added to H.323 in version 2 to reduce the number of round trips to 1 by allowing developers to combine the above procedures into a single H.225 transaction.

The design of this procedure is fully compatible with the original multi-step procedure (for version 1 or version 2 endpoints that do not desire to support Fast Connect) while at the same time doesn't require any one step being a precursor for the other steps.

To build a procedure that met these criteria was a significant technical challenge, especially without requiring additional configuration by the user or the network manager (no "out-of-band" set up).

In the final specification, Fast Connect preserves a critical version 1 feature called "Early H.245". This mode of operation permits media stream establishment (audio and video start running) before the "connect" message is sent from the called party. This feature was available and implemented in RADVision's version 1 protocol stack, but not in the products of (m)any other vendors.

A completely new benefit of the H.323 v.2 Fast Connect procedure is H.245 encapsulation. This specifies the procedure for H.245 messages to be included in a H.225 TCP channel, without opening a separate H.245 TCP connection. Moreover, Fast Connect is defined in such a way that, if a user or one of the sites wants to open a separate H.245 connection later during the call, it can be done without disrupting the integrity of an ongoing H.323 call.

NOTE: Fast Connect needed to be part of the H.323 standard in order for the different possible modes of operation to be done transparently to the user or higher level applications. In order for new applications to take advantage of the theory for smooth interoperability, a stack must strictly follow the Fast Connect procedure. In this case the code is very precise and the rules provided must be followed very strictly.

Multiplexing of call signaling and call control messages

In the original H.245 specification (created for H.320 and adapted for H.323), there is no CallIdentifier field on the application level. This is

why when a stand alone H.245 connection was established in version 1 of H.323, it used the actual TCP ports for this connection as identification of the H.245 signals.

When there are many H.323 calls between the same H.323 entities (e.g., between 2 gateways or between the gatekeeper and the gateway) substantial resources (memory, CPU and bandwidth) may be wasted in maintaining multiple parallel TCP ports. Developers determined that it would be very beneficial to run all these calls using the same TCP connection. This means that the same port is used on each side for all messages between parties.

Unfortunately, this was not possible in version 1 because there was no application level identification for H.225 and/or H.245 streams (only TCP connections). In version 2, CallIdentifier serves as this stream identifier.

With CallIdentifier, the entities may run any number of H.225 connections using the same TCP connection. In addition, when using CallIdentifier with H.245 encapsulation, the application can use the same TCP connection for all H.225 and H.245 messages. Only one connection needs to exist for all these call signals.

This feature (multiplexing of call signaling) will be formally standardized in version 3, but in version 2 it can be implemented because all operational fields are available. When implementing call signal multiplexing in version 2, a developer must be aware of the fact that H.323 v.2 doesn't currently have a protocol-based means to detect how a called-party's system is handling call signaling multiplexing. Without this knowledge call establishment will not be interoperable. The implementer must provide (via out-of-band means) a way to communicate its support for call signal multiplexing to another entity. One means can be during the configuration of the entity.

Single Use Devices

In the future, new interoperability issues may emerge when the specification defines single use devices (SUD). The potential problem with these is that they will use a subset of H.323 and this subset needs to be defined in a very rigid way, otherwise there will be no interoperability between devices. Developers of these future SUDs should take special precautions to make sure that out-of-band configuration is not needed in order for SUDs to be interoperable with fully implemented H.323 devices.

To this point we have identified the most critical interoperability issues that arise directly from the way the protocol is written or interpreted.

Issues Arising in the Implementation of the "core" in the Protocol Stack

Implementers of an H.323 protocol stack or other software building blocks, must be acutely aware of all of the standard related issues that were introduced above. It is the implementer's responsibility to find these problem areas, and use different approaches to reduce potential problems as they implement the stack.

The developers who use H.323 stacks to create H.323-compliant entities should never need to worry about the standards issues we have raised above. The application development process is sufficiently difficult that the developers should be oblivious of the underlying complexities and focus their resources on the unique value-add they bring to their products. This said, the stack must provide certain support for developers.

A robust and highly consistent API definition is an essential feature for a core technology component. Developers of H.323 network components and applications must also use the APIs as defined in order for them to support smooth application transition while upgrading the H.323 version.

With a well designed stack, the product developer should be able to quickly and easily port their software from version 1 to version 2 implementations of the protocol stack. The smooth transition between version 1 and version 2 is an example of the complex underlying changes that a core building block developer must think about in advance to save their in house or commercial (licensing) customers time.

APIs should always be backward compatible. In other words, for those applications that don't wish to take full advantage of the version 2 features, they may still use the version 2 core without changing the application itself.

The core technology developer must build every piece of software so that it explicitly "understands" that all future products will be backward compatible. This interface between the core product and the application is a very delicate area where

implementations will differ in their support for graceful version migration.

H.323 Broad Architectural Issues

A number of broad architectural issues remain to be addressed by the H.323 development community. For example, compared to version 1, version 2 has a much more robust set of features such as security and supplementary services. These are complex features. Therefore, as customers deploy H.323-compliant products in the future, more issues may emerge. We don't yet know how complex these problems will be to resolve, but we will discuss three of those identified to date here.

Implementation of supplementary services

All supplementary services, in version 1 and 2, are conveyed using facility messages.

The general layout of facility messages was defined in version 1 but the specific fields and procedures were missing. As a result, additional (optional) services with version 1 stacks could be supported using either these facility messages in a proprietary way, or implemented with the whole logic of supplementary services in the gatekeeper.

In version 2, new supplementary services were defined (see RADVision Gatekeeper White Paper for more details) using the facility messages. This time the use of these messages is explicitly defined by the standard.

While this is beneficial for long term interoperability, the architectural approach has shortcomings. Specifically, the logic for supplementary services has been defined to be implemented primarily in the endpoint, but in some cases, the H.323 gatekeeper can take responsibility for fulfilling these services. Both approaches to supplementary service provisioning are defined in version 2, making interoperability difficult to accomplish.

Security in H.323

In the first version, H.323 did not address security. Early implementers explored use of security features defined at the IP layer. Subsequently, a new standard (H.235) was developed and determined as part of H.323 v.2 of the standard.

In any situation where such choices exist, there will likely be interoperability problems.

Discussing H.235 specifically, one should recognize that this is a very flexible standard. At the same time, it can be interpreted in many different ways. In H.323 v. 3, those working on the standard are profiling certain aspects of security for implementers. Profiling will help implementers and developers have interoperable products. Some work in this area is also being done in the Telephony and IP Harmonization committee (TIPHON, a group within ETSI, the European Telecommunications Standards body) and the VoIP working group overseen by the International Multimedia Teleconferencing Consortium (IMTC).

Multiplexing media streams

From a resource utilization point of view, multiplexing is very desirable. Currently there is no procedure in the H.323 standard nor in the RTP/RTCP definition (under IETF) to describe interoperable media stream multiplexing.

Although some proposals have been discussed both in VoIP forum and in IETF, there is no clear direction to follow at this time. Until a standard technique is agreed upon, this can be a source of new interoperability issues in the future.

H.323 Entity/Application Level Issues

In this section we examine the possible pitfalls associated with endpoints, gatekeepers, gateways and MCUs.

Endpoints

Developers are seeking to add unique value to their products, while maintaining interoperability. In some situations, two H.323-compliant endpoints may not have a common denominator in terms of their media capabilities, and still be compliant with the H.323 standard. This is known as "unmatched capabilities" and is another area where interoperability issues arise.

H.323 (and H.245, as a part of H.323) allows users to take advantage of full, rich feature negotiation and communications of different media stream parameters. H.245, from the specification point of view, is a very highly developed/mature protocol. It comes with a very comprehensive description, especially when compared to other IP telephony protocols.

H.323 specifies certain mandatory codecs that any endpoint, or entity must support. To achieve interoperability, these are negotiated using H.245,

but there are portions of the standard that invite interoperability problems.

For instance, H.323 dictates that if the available bit rate is more than 64kbps, the master (head) codec is G.711. On the other hand, if the available bit rate is less than 64kbps, the rules are different. Specifically, if it is a voice-only call, the audio codec must be G.729, and if the call is voice and video, the codec is G.723. G.723 compresses voice more and uses less bandwidth than G.729 or G.711, leaving more room for the video portion of the call.

As a result, there is no minimum defined capability for all H.323 terminals. What is clear is that in order to select the right audio codec, the two endpoints need to be aware of the bandwidth available, yet the ability to detect the data rate is not explicitly a requirement of an H.323-compliant endpoint. And gateways are often another source of unknown parameters.

On the video codec selection there are other obstacles to interoperability. H.263 is the codec of choice, although H.261 support is required. Within H.263, there are several different "flavors". There is H.263-draft, H.263 (final) and H.263+ implementations. For example, when implementing NetMeeting, Microsoft used a different version of H.263 than that specified in H.323 version 1 (they used the initial draft of H.263, not the final draft). Using the field of type VendorIdentifier, a gatekeeper may determine what type of devices are in the network and then (as is the case in all RADVision products) support both the initial draft and the final draft of the H.263 standard.

Again, in version 2 there is a field to allow product developers to specify which version of H.263 the terminal or gateway is using, so the protocol negotiations don't always have to go to vendor ID field (a work-around developed for use in version 1 deployments).

Gatekeepers

Currently, gatekeeper to gatekeeper communications is undefined. The H.323 developer community is working very hard to fill this void. When ready, gatekeeper to gatekeeper communications will be defined in Annex G of H.225.

Example: Address resolution is defined in H.323 for specific topologies. In version 2 the concept of aliases has been expanded to support different addressing schemes. The version 1 list was expanded with new version 2 formats. Currently,

there is no clearly defined way for gatekeepers to deal with both of these schemes, nor does the standard indicate if these are to be combined when communicating with another gatekeeper.

Today, deployers and developers are looking for generic solutions. And potentially having different address resolution schemes will lead to interoperability problems.

Gateway devices/entities

There are certain inherent interoperability problems that gateway developers must resolve to support a network of H.323 users with a high degree of reliability and interoperability.

Bandwidth allocation and requests

As previously noted, there are differences between the original Q.931 and the H.323 interpretation of this protocol. H.323 internetworking gateways connecting H.323 and H.320 terminal users will need to use Q.931 the way it was originally approved for ISDN and the adapted version for the IP network.

Elsewhere in this white paper we allude to the fact that this might cause interoperability problems associated with bandwidth allocation. Bandwidth allocation is defined differently for H.323 and H.320. There is no rigid definition for transitioning between the two worlds. For example, there is no defined way to distinguish a request for a 384Kbps Hybrid call (H0), from a request for a 384Kbps call produced by aggregating together six B channels, when originating a call from an H.323 terminal.

Today these issues are open for interpretation while implementing gateways.

GW to GK communication

Gatekeeper to gateway communication is another huge source of confusion. Today, the gatekeeper to gateway communications resides at the application level. This is the source of many problems.

To understand where these problems arise, consider the endpoint to gateway path as two segments of a system. One segment is defined by the endpoint and gatekeeper for that endpoint's zone (see [b] below) and the other segment is from the gatekeeper to a number of possible gateways (see [a] below). The gatekeeper has to be able to pass information (e.g., address resolution and routing information) back and forth between the gateway and the endpoint.

(a). Today the gateway is defined and treated in the H.323 specification as an endpoint. The messages between this special endpoint and the gatekeeper are defined using RAS. It is almost the same (set of messages) interface as is defined for gatekeeper to "simple terminal" communications. In fact, there is only one message that is different. A message provides the possibility for the gateway to publish (or advertise) its connectivity modes on the non-H.323 side. For example, it can say what bandwidth it has, what kinds of interfaces to the circuit switched network it can provide for calls/users, and what prefixes should be dialed to get access to a specific service.

The problem is that the messages are there, but until more implementations have been tested, it is unclear if interfaces will interpret the procedures in the same way. Further definition may be necessary.

(b). Another part of the solution has to do with how an end-point requests service. When the calling endpoint wants to go through a gateway, it explicitly asks the gatekeeper for the gateway access. This necessitates that the end-point user know what kind of service it wants to receive in terms of address resolution and bandwidth/quality of service.

This is communicated from the user interface to the network entities using RAS and Q.931 call signaling, but the end to end solution is not sufficiently defined and this leads to interoperability problems. Everyone who builds a gatekeeper, terminal or gateway has developed a slightly different approach. For example, some solutions require that:

- the user either dials a prefix, to tell the gatekeeper which service to receive,
- or the other way is to put this request message in the user interface but hidden from the user's explicit control.

In the later case the user receives the service without remembering a dial plan number, but the endpoint application has to consistently use the appropriate protocol fields to convey the request for the service.

The protocol fields are there, but most of the endpoints don't currently make use of them or interpret them in different ways.

Multipoint devices

The problems in using multipoint devices in H.323 start with the fact that there are many different ways to bring up a multipoint session in H.323.

First, we need to define a few concepts unique to H.323 multipoint. H.323 has separate logical entities for different tasks. A multipoint controller (MC) is software that directs multipoint sessions. The multipoint processor (MP) is the hardware that may or may not be used during a multipoint session to mix audio or video streams for users. An MCU is the combination of an MC and an MP. An MCU is not always required, if an MC is present in a zone.

Multipoint commands in an H.323 MC are defined using two different protocols: Q.931 and H.245. A user can JOIN (a call) or INVITE (to a call) on the Q.931 level and (then), using an active MC of the call, "build" a conference using H.245 commands. Media stream transmission is also very flexible. Through an MC's control messages, a multipoint conference may be centralized, decentralized, or a mixture of both.

A centralized multipoint conference is one in which all participating terminals communicate in a point-to-point fashion with an MCU. The terminals transmit their RTP streams to the MCU. The MP within the MCU receives media streams, and, after processing, returns them to each terminal.

A decentralized multipoint conference is one in which the participating terminals multicast their audio and video to all other participating terminals without using an MCU. In this case the terminals are responsible for selecting one or more of the received video streams for display, and for mixing the audio streams.

There are a number of undefined situations that present themselves during a multipoint session. Interoperability issues will arise when different H.323 entities define the situations and attempt to control them in unique, proprietary ways, for example:

- How does an endpoint automatically, or through a discovery process, know which of the multipoint modes is supported?
- Does the client directly support (issue) the commands in Q.931, in H.245, both, or not at all?
- Today there are no real (fully compliant) H.323 MCUs on the market. Moreover, the main terminal players don't support multipoint related commands.

As a result, the prevalent solution is to put both the MC and MP logic inside the MCU. In this situation the whole multipoint issue becomes transparent to participants' terminals. The multipoint conference is a group of point-to-point calls between a terminal

and the MCU where the whole (proprietary) logic is implemented.

Since these are issues between MC and MP separation, it is safe to assume that interoperability issues regarding MCU cascading are still far from being fully understood, much less resolved.

Conclusion

With all these possible sources of problems, how do different vendors work towards H.323 interoperability?

RADVision's core H.323 software comes complete with application level interfaces. This is one way to test the stack software itself for compliance.

RADVision also provides sample applications that simulate a terminal, sending requests and performing many different H.323 defined functions.

Finally, RADVision and its partners can use Remote Procedure Calls (RPC) to run a terminal application with H.323 call software and test the responses of network infrastructure components to these calls.

A second level of interoperability testing must be conducted in a multi-vendor environment. Vendors who belong to the International Multimedia Teleconferencing Consortium (IMTC) use quarterly (standard-specific) and annual (SuperOp!) events to test interoperability of their multimedia communications products.

There are two factors currently holding back the pace of the interoperability testing in the IMTC and the rate at which vendors can resolve the issues they encounter in these multi-vendor environments, as follows:

1. First, the number of H.323 application developers who come to the testing events is growing rapidly. Therefore, the number of potential interoperability problems (and test procedures) also grows rapidly.
2. Second, version compatibility is a factor slowing down progress. Most of the vendors who have participated in H.323 interoperability tests to date have been testing version 1 implementations. Vendors that have implemented version 2 can test for backward compatibility, but can not really test the interoperability of new version 2 features.

It is important to note that a final level of interoperability testing must be conducted in real-world customer deployments. Since so many third-party products are built around the RADVision H.323 protocol stack, relatively few stack related problems have arisen in real-world deployments.

In the near future, there will be islands of H.323 interoperability. In the early deployments, network designers and those testing products for ultimate use will be satisfied with an island of interoperable devices. Gradually, as more developers understand the sources of interoperability problems and how to address them, we can expect full H.323 end-to-end

interoperability to become the standard state of affairs.

This white paper was published by RADVision to provide practical information about H.323 to product developers, total solution providers and deployers. To learn more about H.323 technology components, products, their integration in total solutions, or the issues associated with developing or implementing H.323 systems in today's networks, please refer to companion volumes in this series.