

Ericsson's enriched messaging architecture

Åke Gustafsson and Anders Lenman

To achieve simple, efficient interworking between messaging services, Ericsson has factored out the message routing and storage functions from messaging services. Instead, it has introduced a new architecture with a common interworking function (IWF) that features a dispatcher and a common message store. Then, taking this approach one step further, Ericsson communalized other aspects of messaging servers. The resultant design is a horizontalization of traditional, vertical messaging solutions which gives rise to lower total cost of ownership (TCO) and fewer, better-quality components, and yields a simple, scalable, and robust messaging architecture that maintains all states in the shared message store file system.

To derive the full benefit of this enriched messaging solution with minimum impact on existing systems, Ericsson also provides a prepackaged messaging integration solution that converges vertical legacy solutions and IMS.

Introduction

Ericsson's messaging vision for end users: "A message is a message, regardless of the technology used."

Studies and market trends show that messaging solutions must be easier to use if they are to facilitate communication and increase traffic. Today, for example, to use SMS, MMS, e-mail, instant messaging (IM), voice mail, and so on, end users must be aware of the underlying technology and their intended recipients' capabilities.

Ericsson's enriched messaging architecture addresses this matter by allowing end users to exchange messages with anybody, at any time, and from any place without having to give the associated technologies a thought.

The new architecture provides mechanisms that accommodate interworking between messaging services. Some key functions in this context are addressing, routing, and message header mapping.

Traditional interworking is typically achieved by means of an interworking function component between two separate messaging services. An IWF component contains protocol handlers (PH) for each service, and logic for handling message header mappings. Each messaging service typically routes messages independently.

This approach works fine when there are only two messaging servers involved, but as the number of messaging servers grows the complexity of the associated interworking

becomes unmanageable, both for the IWFs and in terms of distributed routing logic (Figure 1).

Ericsson's enriched messaging solution, by contrast, introduces a new architecture with a common IWF function that features a dispatcher and a common message store. These two key components provide a generic IWF function and common message routing logic.

The dispatcher, which is both the brain and the heart of the enriched messaging core, ensures that messages are routed properly (brain). It also safeguards messages by means of scheduled retry events that make certain they do not get stuck in the system (heartbeat).

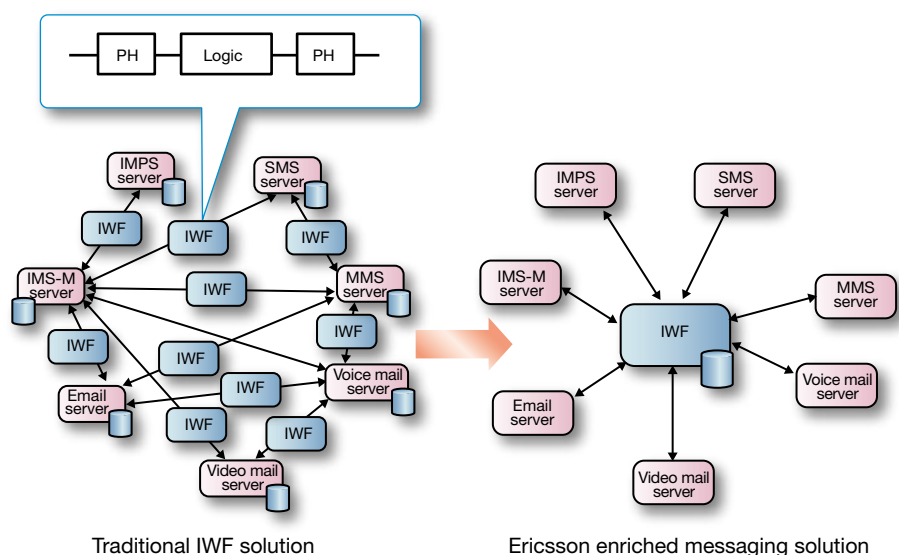
The message store, which is a common storage function for all messages, regardless of messaging service used, is structured into mailboxes – one mailbox per user.

Enriched messaging core architecture

To achieve simple yet efficient interworking between messaging services, Ericsson has factored out the message routing and storage functions from messaging services. In essence, the enriched messaging core is a store-and-forward engine that consists of a dispatcher, the common message store, and messaging servers.

Messaging server is a generic term for the applications that contain the messaging service business logic. Examples include MMS,

Figure 1
Interworking between messaging services.



SMS, IM, voice mail, and e-mail servers. A very important aspect in this context is the protocol handling of external interfaces.

The three entities, dispatcher, common message store, and messaging servers, interact with every message that passes through the enriched messaging solution. Figure 2 illustrates this interaction. Box A describes the flow of messages in the enriched messaging core architecture.

To preserve flexibility and optimize interworking, messages are stored both in native and MIME formats. Any messaging server may thus easily pick up a message for delivery.

Below follows an example of how the solution, based on user settings, interworks and delivers voice mail as MMS (see also Figure 2). An end user – “Alex” – wants all incoming voice-mail messages to be delivered as MMS, so she can play them locally in her terminal at any time. (1) An incoming voice mail message arrives and the voice-mail messaging server asks the dispatcher (2) how it should route the message. The dispatcher checks Alex’s preferences and replies with the preferred routing, in this case, “send as MMS.” The voice-mail system stores the message (3a) and hands the task over to the dispatcher, which requests an MMS messaging server to deliver the message (4). The MMS messaging server sends an MMS notification (5) to the client, indicating that a new message is waiting to be retrieved. When the client requests delivery, the MMS messaging server fetches the voice-mail message from message store and builds an MMS message; that is, the voice-mail message becomes audio content in an MMS message. The MMS messaging server handles all necessary audio transcoding to adapt the content to client capabilities. After Alex has received the MMS, she may play the voice-mail message locally at any time.

Components of the enriched messaging core: dispatcher, message store and messaging servers

The dispatcher’s two main functions are route resolver and event handler. As route resolver, it decides which messaging server should be used to deliver a message. It bases this decision on external input from different sources of user data and message metadata, such as size and type of content. External sources of user data include presence, user profile settings, and status of the home location register (HLR). The routing rules, defined via a script interface, are easy to customize.

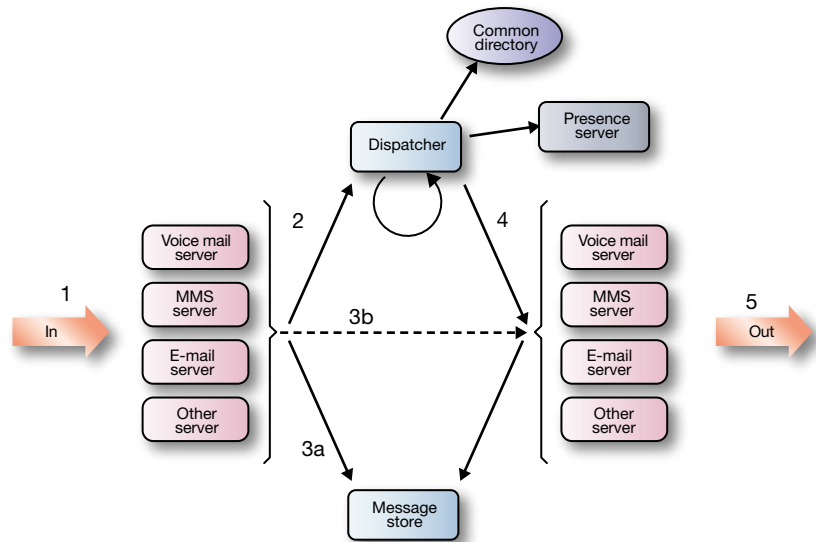


Figure 2
Interworking between messaging servers. Note: For descriptive purposes, the messaging servers have been depicted twice (once for incoming and once for outgoing messages).

BOX A, MESSAGE FLOW IN ENRICHED MESSAGING CORE ARCHITECTURE

Step 1

A messaging server receives a message, interacts with the sender, and runs its business logic; for example, it validates the user and screens the message for spam and virus content.

Step 2

The messaging server queries the dispatcher, asking how the message should be routed to the recipient. The dispatcher checks recipient preferences – for example, presence (presence server) and user settings (common directory) – and replies with the preferred routing.

Step 3

- a. The messaging server stores the message in the recipient’s inbox in message store and informs the dispatcher that a new message has arrived. The dispatcher safeguards the message by scheduling an event. When the messaging server receives acknowledgement from the dispatcher that it has taken over responsibility for delivering the message, the messaging server signals the sender that the message has been accepted and will be delivered. The messaging server (incoming side) is then released from the message flow.
- b. If this same messaging server (outgoing side) is tasked with delivering the message, it may, as an optimization, deliver the message directly without storing it.

Step 4

When the scheduled event is triggered, the dispatcher selects a messaging server type (based on preferred routing information) and asks it to deliver the message.

Step 5

The selected messaging server (outgoing side) initiates message delivery. This includes message notification if necessary. The message is fetched from the recipient’s inbox in message store when forwarded or retrieved, and if necessary, adapted to the recipient’s capabilities. Outstanding dispatcher events related to the message are deleted when the message has been successfully delivered.

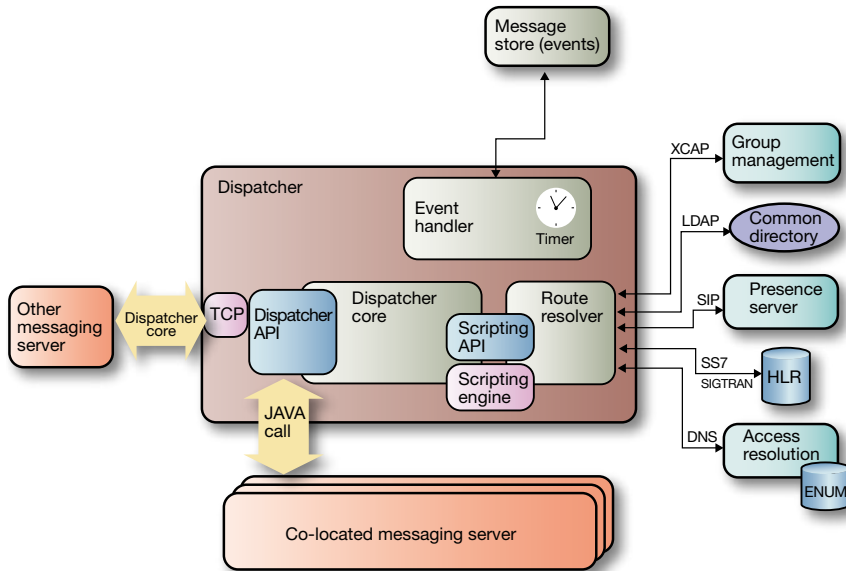


Figure 3
The internal structure and main interfaces of the dispatcher.

The event handler is a stateless, near-real-time timer function that stores all events in a shared area in the common message store. This approach allows the dispatcher to achieve high availability by applying simple $n+1$ redundancy. When an event is triggered, the event handler analyzes it and orders actions based on content – for instance, “retry sending message.”

Figure 3 depicts the dispatcher’s internal structure and main interfaces.

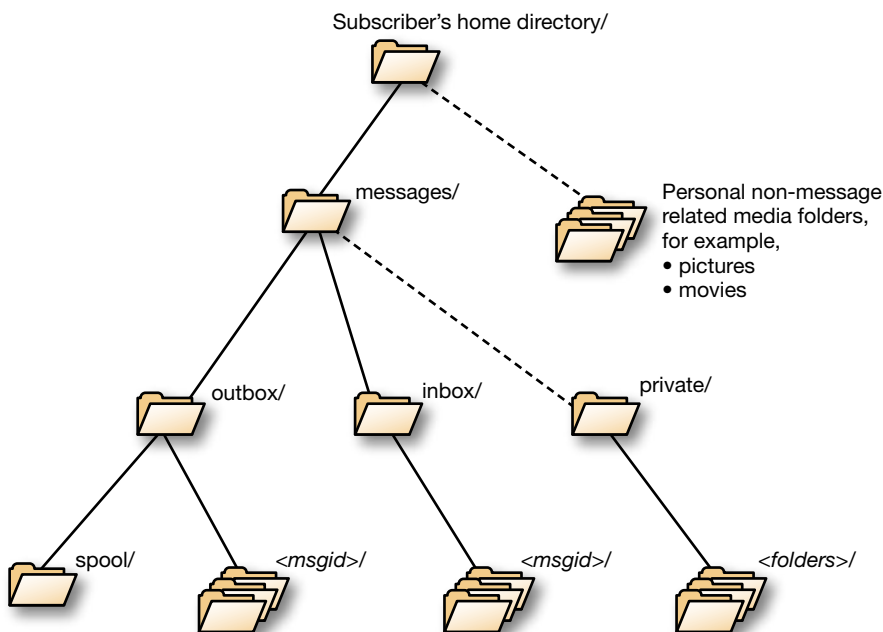
Message store

Message store is a standard network-file-system-mounted (NFS) file server that hosts one mailbox folder per user. The folder contains all of a given user’s messages regardless of messaging service used. Every mailbox folder has a set of subordinate folders, for instance, an Inbox and an Outbox. The mailboxes may also have other “private” folders for sent items and so on (Figure 4).

Message store has a separate folder structure for common storage of events scheduled by the dispatcher. This way, dispatchers can backup one another, safeguarding against dispatcher failure.

Message store is fully scalable by means of partitioning user accounts/folders over multiple mount points, where each mount point is backed by a high-availability or redundant NFS network-attached storage (NAS) server.

Figure 4
Internal structure of message store.



Messaging servers

Messaging servers implement messaging services – for instance, MMS. They interact with end users and execute the messaging feature set for a given service. They are responsible for triggering charging and for screening message content. In short, apart from routing and storage, which has been factored out to the dispatcher and common message store, they implement all messaging service business logic.

Horizontal view of enriched messaging components

To simplify interworking, designers at Ericsson have, as described above, factored the storage and routing functionalities out of messaging servers into common messaging components. Then, taking this approach one step further, they communalized other aspects of messaging servers.

A look at today’s messaging products

(SMS, MMS, e-mail, IM, voice mail, and so on) shows that they are clearly vertical silos that do not share or reuse components. These vertical solutions have resulted in a duplication of functions in operator networks. Most of them include message store, a directory, and an adaptation function. By breaking the solutions down into components, one sees that several of the components are repeated. Therefore, in a horizontal solution they could be shared or reused (Figure 5). The resultant design and architecture gives rise to lower total cost of ownership (TCO) and fewer, better-quality components.

Ericsson's enriched messaging architecture is an open, horizontal architecture based on internet protocols that implicitly support a heterogeneous environment with different control and connectivity layers – for example, a system that both supports SIP and SS7 while running over different access technologies (such as GSM, WCDMA, PSTN, broadband, and ISP).

The architecture is also open in the sense that it allows for easy integration of third-party vertical messaging solutions via standard protocol interfaces, such as SMTP and IMAP4 for e-mail.

Figure 6 gives an overview of Ericsson's messaging architecture in the service layer and Figure 7 outlines the interfaces of the different components.

We have already described messaging servers, the dispatcher and message store. Below follows a brief description of border gateways, messaging enablers, common functions, and the VASP gateway. Note: The set of messaging enablers and border gateways in Figure 6 is not exhaustive.

Border gateways

Border gateways are a set of proxies that translate access-dependent protocols into instructions that can be read by messaging servers. The voice gateway, for instance, translates circuit-switched (CS) voice into voice over IP (VoIP).

Enablers

Enablers are resources that perform common functions with service-independent business logic used by, say, the messaging servers, to fulfill their business logic. The Media Adaptation enabler, for example, is a resource that can transcode any given message content from one format to another. It can also change image size and resolution, the audio coding of a sound clip, and so on.

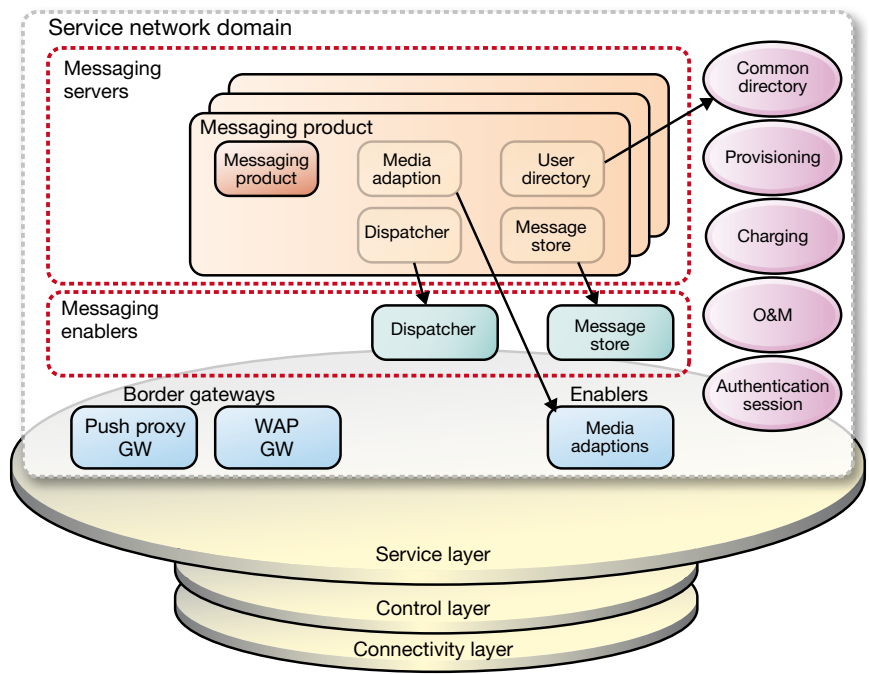
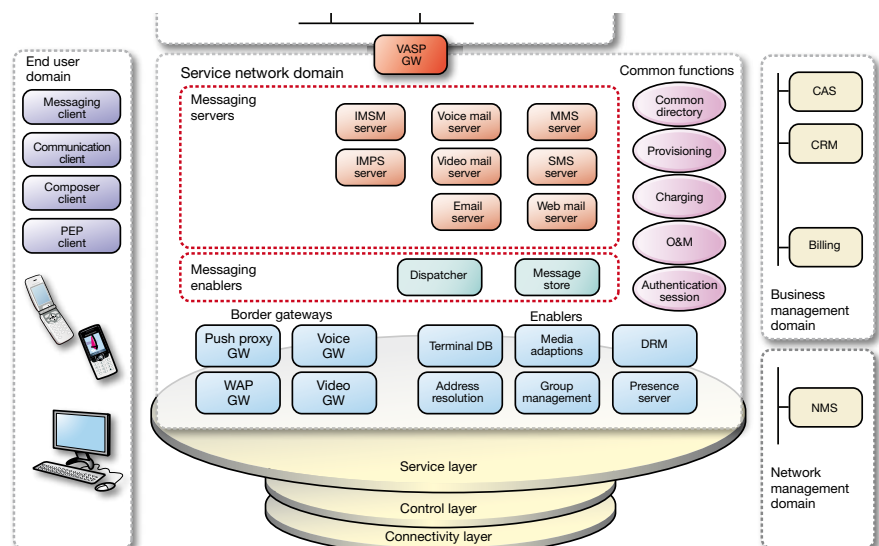


Figure 5
Horizontal approach to messaging solutions.

Figure 6
Ericsson's messaging architecture.



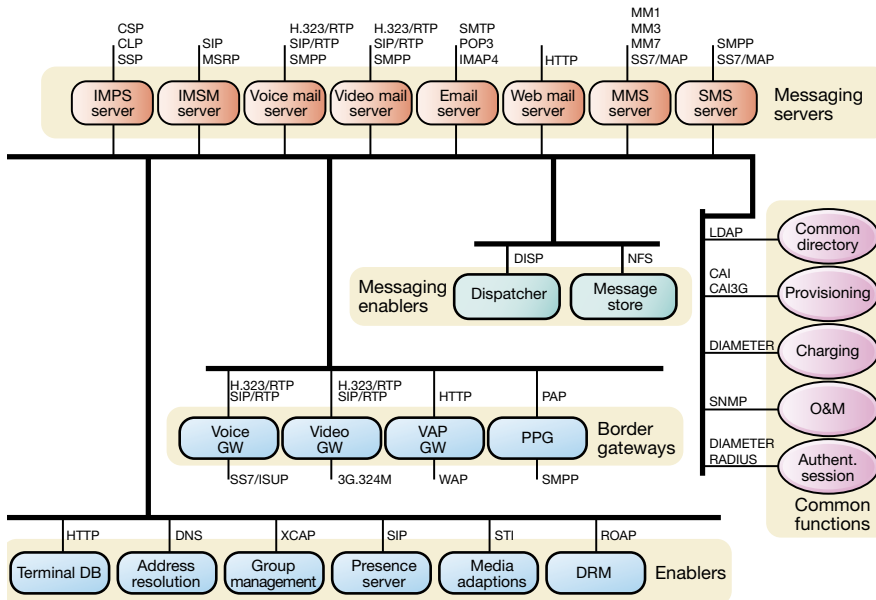


Figure 7
Interfaces of the messaging architecture.

Common functions

Common functions are a set of functions that support a horizontal service network of user provisioning, charging, single sign-on (SSO), and network node operation and maintenance (O&M).

VASP gateway

The VASP gateway gives value-added service providers (VASP) a single point of access for interacting with operators' service layer components. It exposes services to applications in order to simplify the integration of these applications. VASPs may then provide message content – short video clips, birthday greetings, and so on.

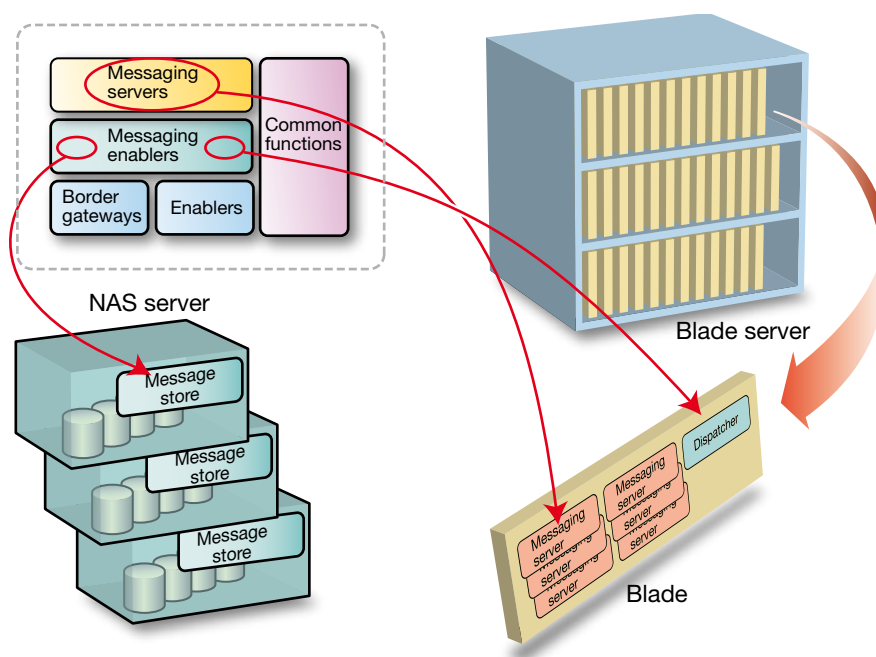
Deploying enriched messaging components

The architecture is designed for stateless business logic, yielding a simple, scalable, and robust messaging architecture ($n+1$ redundancy) that maintains all states in the shared message store file system.

To achieve near-linear scalability and robustness, the messaging servers and dispatcher are preferably co-located as shown in Figure 8. This way, the dispatcher is local and can invoke outgoing messaging servers without the need for cross-host/blade communication.

This configuration also reduces the need for copying data (the message) over the internal network. Instead, the design relies on local caches on the blade or host. Each blade or host thus becomes its own entity and can be managed independently (start/stop/restart).

Figure 8
Messaging architecture deployment.



Enriched messaging architecture as a tool for integration

The open, enriched messaging architecture is a powerful toolbox for integrating legacy solutions. The challenge, of course, is to derive the full benefit of Ericsson's vision of "a message is a message" with minimum impact on existing systems.

One can easily integrate legacy systems via their network-to-network interface (NNI). These systems may either send all messages to the enriched messaging solutions for routing analysis and interworking, or solely those messages that need to be interworked, as determined by the legacy system.

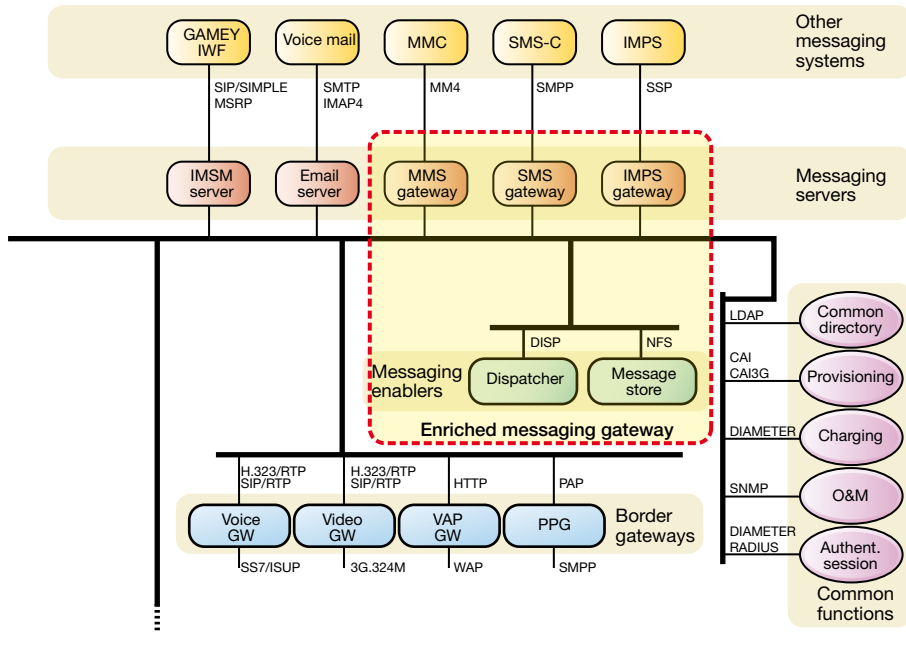


Figure 9
The enriched messaging gateway
(a lightweight messaging server).

Ericsson provides a prepackaged messaging integration solution that converges vertical legacy solutions and IMS. The enriched messaging gateway, for instance, contains lightweight messaging servers that provide a suite of interfaces for integrating multi-vendor legacy messaging solutions. These lightweight messaging servers basically only support an NNI with associated business logic, hence the name messaging gateway. Figure 9 shows an example of interfaces between Ericsson's enriched messaging solution and legacy systems.

Conclusion

Ericsson's enriched messaging architecture gives operators and end users full interworking between legacy messaging solutions and, more important still, between IMS messaging and legacy messaging solutions. The architecture may be likened to a toolbox for easy integration of legacy solutions. In addition, it makes the network smarter, so end users can concentrate on the message instead of on underlying technology. Remember, a message is a message, regardless of the technology used.

TERMS AND ABBREVIATIONS

CS	Circuit-switched	NAS	Network-attached storage
CSP	Client-to-server protocol	NFS	Network file system
DNS	Domain name system	NNI	Network-to-network interface
DRM	Digital rights management	O&M	Operation and maintenance
GAMEY	Internet IM providers (Google, AOL, MSN, E-bay, Yahoo)	PAP	Push access protocol
GSM	Global system for mobile communications	PH	Protocol handler
GW	Gateway	POP	Post office protocol
HA	High availability	PSTN	Public switched telephone network
HTTP	Hypertext transport protocol	RTP	Real-time transport protocol
IM	Instant messaging	SIMPLE	SIP for instant messaging and presence-leveraging extensions
IMAP	Internet mail access protocol	SIP	Signaling over IP
IMPS	Instant messaging and presence services	SMPP	Short message peer-to-peer protocol
IMS	IP Multimedia Subsystem	SMS	Short message service
IP	Internet protocol	SMTP	Simple mail transfer protocol
ISP	Internet service provider	SNMP	Simple network management protocol
ISUP	ISDN user part	SS7	Signaling system no. 7
IWF	Interworking function	SSP	Server-to-server protocol
LDAP	Lightweight directory access protocol	VASP	Value-added service provider
MAP	Mobile application part	VoIP	Voice over IP
MIME	Multipurpose internet mail extensions	WAP	Wireless application part
MMS	Multimedia messaging service	WCDMA	Wideband code-division multiple access
MSRP	Messaging session relay protocol	XCAP	XML configuration access protocol