Location-Aware Media Engine

System architecture document

Vincit Oy

Hermiankatu 6-8 A

33720 Tampere

+358 40 589 8316

Table of Contents

1. INTRODUCTION 4
1.1. SCOPE OF THE DOCUMENT 4
1.2. DEFINITIONS, TERMS AND ACRONYMS 4
1.3. OVERVIEW 5
2. SYSTEM OVERVIEW 6
3. COMPONENT INTERACTIONS 9
3.1. CLIENT AUTHENTICATION 9
3.2. CLIENT INTERACTION WITH STORAGE SESSION SERVER 9
3.3. REGISTRATION OF SESSION SERVERS 11
3.4. RESUMING A SESSION 11
4. PROTOCOLS 12
4.1. HTTP 12
4.2. RTP 14
4.3. LRS 15
4.4. TAGS 16
4.5. SECURITY 17
5. TIMING 17
6. APPENDIX 20

© 2008 Vincit Oy

Hermiankatu 6 A
33720 Tampere

+358 40 589 8316
info@vincit.fi

VAT reg no
FI 2113034-9

Change history

Version	Date	Author	Description
1.0	15.08.2008	Olli-Pekka Räsänen	Initial version.
1.1	18.08.2008	Olli-Pekka Räsänen	Added some information about security.
1.2	22.09.2008	Olli-Pekka Räsänen	Changes to 4.1.
1.3	01.07.2009	Olli-Pekka Räsänen	Modified HTTP protocol description. Added chapter about tags.

© 2008 Vincit Oy

Hermiankatu 6 A
33720 Tampere

+358 40 589 8316
info@vincit.fi

VAT reg no
FI 2113034-9

1. INTRODUCTION

1.1. Scope of the document

This document describes the system architecture of Location-Aware Media Engine ("LAME"). LAME is a software platform that enables a rapid development of location-aware content-sharing applications.

1.2. Definitions, terms and acronyms

LAME	Location-Aware Media Engine, a software platform enabling a rapid development of location-aware content-sharing applications.
Client	A producer application that generates content and sends it to (storage session) server.
Frontend	An application that consumes the content that it has requested from (retrieval session) server. Although frontend acts as a client to the retrieval session server, it must not be confused with the content-producing client application.
Storage session server	An application that serves requests sent by clients in order to store data.
Retrieval session server	An application that serves requests sent by frontends in order to retrieve data from database and send that data back to frontends.
Server registry	An application that keeps track of (both storage and retrieval session) servers and their load statuses.

© 2008 Vincit Oy

Hermiankatu 6 A
33720 Tampere
+358 40 589 8316
info@vincit.fi
VAT reg no
FI 2113034-9

Authentication server	An application that authenticates clients and authorizes them to connect to either storage or retrieval session servers.
Database	An application that stores and manages all data needed by session servers, authentication servers and server registry.
RTP	Real-time Transfer Protocol, a protocol that is used to send (media) streams (e.g. vid-eo/audio) over unreliable transport layer protocol (UDP).
LRS	LAME Reliable Stream, a protocol that is used to send streams (e.g. video/audio) over reliable transport layer protocol (TCP).
Session ID	Unique identifier for a session.
Stream ID	Unique identifier for a stream inside one session.
Octet	A group of 8-bits, usually equal to a byte.

1.3. Overview

Chapter 2 describes briefly the main components of the system and their relations.

Chapter 3 describes component interactions in more details.

Chapter 4 describes what protocols are used in communication between components.

Chapter 5 describes how time and timing is handled in LAME.

© 2008 Vincit Oy

Hermiankatu 6 A
33720 Tampere
+358 40 589 8316
info@vincit.fi
VAT reg no
FI 2113034-9

2. SYSTEM OVERVIEW

The system consists of seven different components. These components are client, frontend, authentication server, storage session server, retrieval session server, server registry and database (Figure 1).

A client is an application that produces content (e.g. video and audio), attaches (GPS) location and other information to it, and sends it to the server for the storing. The client is not bound to a specific operating system so it may as well be Symbian S60 application as it can be a Windows or Linux application. The only requirements for client are that it must be able to produce some kind of content that interest users and it must be able to communicate with LAME servers by using different protocols, mainly Web Services (SOAP), RTP and TCP. The client interacts directly with an authentication server and a storage session server.

A frontend is an application that uses contents stored on servers to interact with user. If the client is thought as a producer then the frontend is a consumer. For example, a web page that enables watching of videos stored on the server is a frontend. The frontend communicates directly with authentication server and retrieval session server.

An authentication server is a stateless application that is responsible for authenticating users (i.e. clients and frontends) and granting them access to both storage and retrieval session servers. Because of statelessness, it is possible to have an arbitrary amount of servers working at the same time. Authentication servers host a web service via which authentication is done as a remote procedure call. Authentication servers are also responsible for assigning each client and frontend to a specific session server in order to balance the load between these servers. Authentication servers communicate directly with clients, frontends and database.

A storage session server is responsible for handling the requests made by clients and storing the data sent by clients to a database. Storage session server is also stateless so it is possible to have an arbitrary amount of these servers working at the same time. Storage session servers communicate directly with clients, server registry and database.

© 2008 Vincit Oy

Hermiankatu 6 A
33720 Tampere
+358 40 589 8316
info@vincit.fi
VAT reg no
FI 2113034-9

A retrieval session server is responsible for handling the requests made by frontends, retrieving data from a database and sending it back to frontends. Retrieval session server is also stateless so it is possible to have an arbitrary amount of these servers working at the same time. Retrieval session servers communicate directly with frontends, server registry and database.

A server registry is an application that is responsible for keeping track of both storage and retrieval session servers and their load statuses. With the help of server registry, it is possible to add storage and retrieval session servers dynamically to the system. The server registry itself is not a database but instead it uses the database component for storing the information about operating session servers. The server registry communicates directly with session servers and database.

A database is simply an application that stores and manages all data needed by session servers, authentication servers and server registry. Although the database communicates with session servers, server registry and authentication server, it is always the passive side meaning it only responses to requests, never makes a request by itself.

© 2008 Vincit Oy

Hermiankatu 6 A
33720 Tampere
+358 40 589 8316
info@vincit.fi
VAT reg no
FI 2113034-9

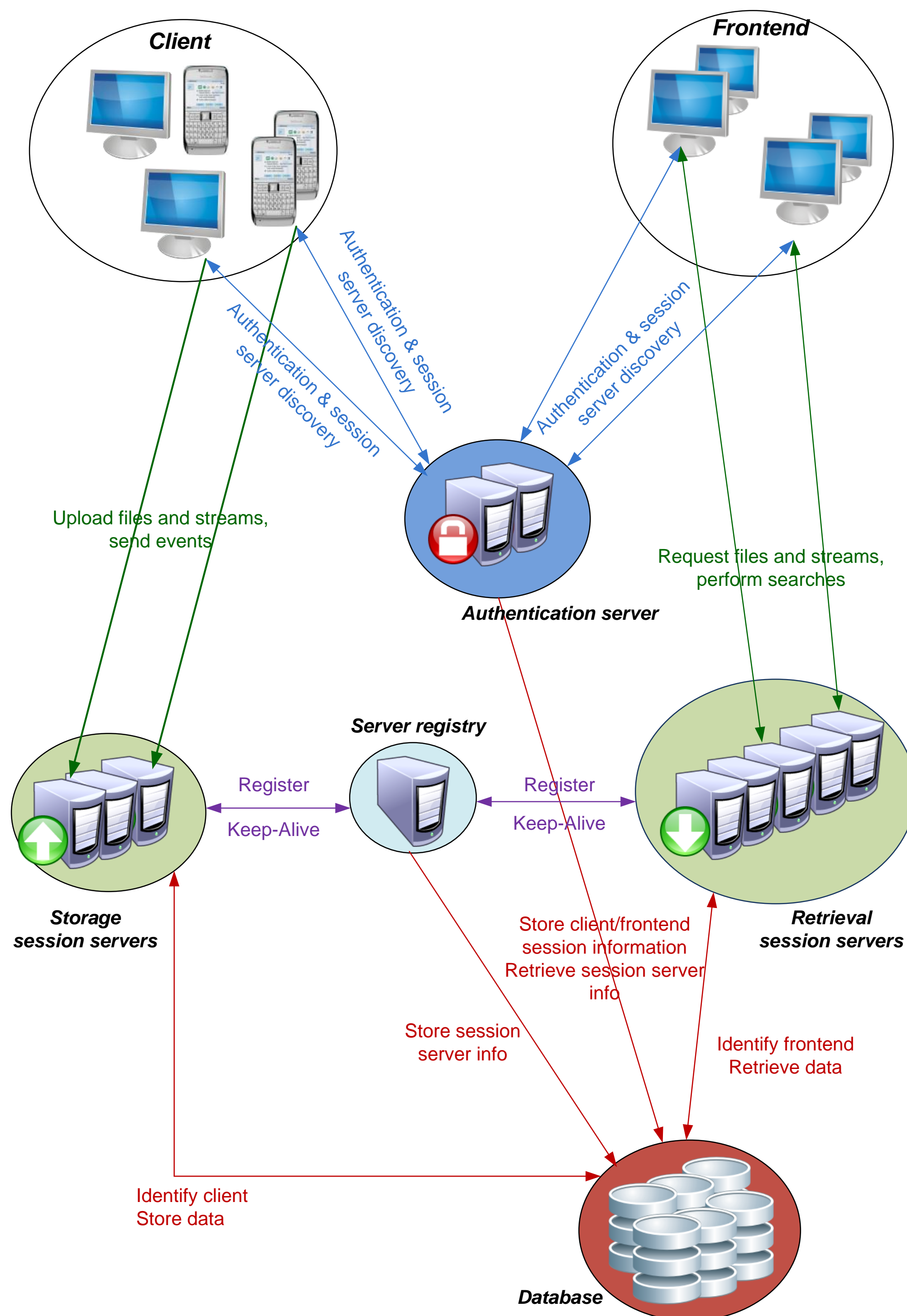

Figure 1: System overview.

© 2008 Vincit Oy

Hermiankatu 6 A
33720 Tampere
+358 40 589 8316
info@vincit.fi
VAT reg no
FI 2113034-9

3. COMPONENT INTERACTIONS

3.1. Client authentication

In order to store content to the server, client must first authenticate itself to an authentication server after which it can start session with a storage session server. Because there may be an arbitrary amount of authentication servers, clients cannot connect directly to a specific server. Instead a DNS load balancing technique is used - that is, only one URL exists which the clients use and DNS is responsible for translating that URL to different IP addresses at different times.

The authentication itself is done as a remote procedure call via a web service. The authentication server hosts a web service to which client makes a *session request* containing credentials (username and password) after which the server requests database to check if the given credentials are authentic. If so, the server generates a session ID, stores it to the database and chooses to which session server client should connect. The choice is made on the basis of which session server has the least load. After the choice has been made, the URL of that server and the session ID are returned to the client as a session request response.

It is also possible for client to request resuming of a session by providing a previously received session ID. In this case authentication server checks from database if session ID is found and then sends session server's URL to the client. Resuming a session is described in more details in section 3.4.

3.2. Client interaction with storage session server

After receiving session ID and session server URL, client can start interaction with session server. In most of the cases this means sending content, either files or streams, to the server.

If the client wants to send a stream it must first request a permission from the server to do so. This must be done because some protocols (for example RTP) require unique port for each transmission and there is no way how client could know the port to be used without

© 2008 Vincit Oy

Hermiankatu 6 A
33720 Tampere
+358 40 589 8316
info@vincit.fi
VAT reg no
FI 2113034-9

requesting it first. The request is done via a web service which is hosted by the server. The request contains session ID so server can check from database if the client has been authenticated. If everything is ok, server generates stream ID, stores it to database together with session ID, opens a port for transmission, and sends stream ID and port number back to the client. After that client can start the sending of stream.

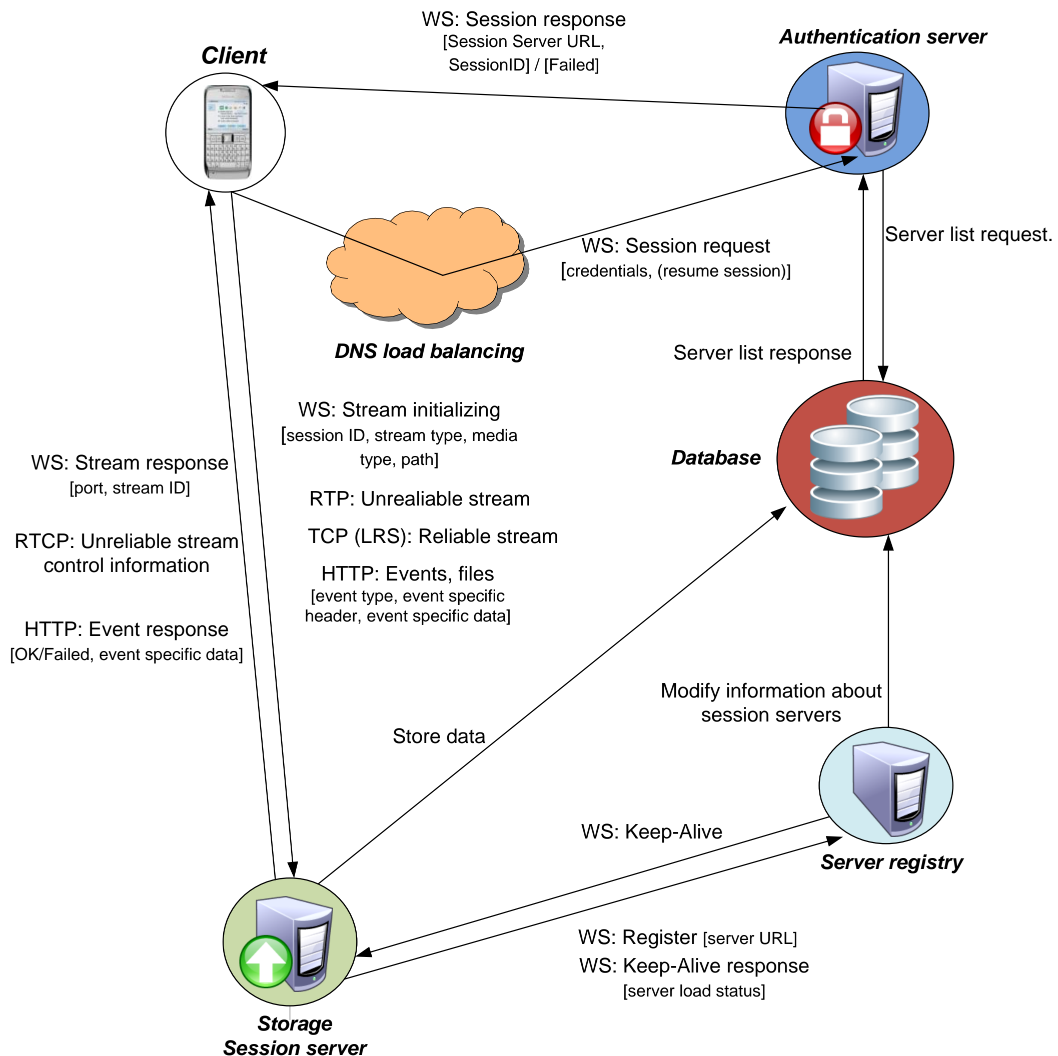

Figure 2: Component interactions.

It is possible to add location and other arbitrary information (for example tags) to a specific section of a stream. This can be done in two ways, either by adding that information directly to the stream or by sending an event to the server. When added directly to the stream, additional information is transported together with actual

© 2008 Vincit Oy

Hermiankatu 6 A
33720 Tampere
+358 40 589 8316
info@vincit.fi
VAT reg no
FI 2113034-9

stream data: actual data will be transported as a payload and information as a header. This convention does not work well with unreliable streams if the information is crucial because it is possible to lose some of the packets. Therefore it is possible to send this additional information as an event using HTTP POST. Events and protocols used to send streams are described in more details in section 4.

Sending a file is simpler than sending a stream because files are sent as HTTP POST messages and no permission request is needed. More details can be found from section 4.1.

3.3. Registration of session servers

The purpose of the server registry is to enable the addition of new (storage and retrieval) session servers dynamically to the system. When a new session server is added, it registers itself to the server registry via a web service which is hosted by the registry. The registry stores the URL of that server to the database. This enables authentication servers to get information about operating session servers.

With certain intervals, the registry goes thru all registered servers and sends them a *keep-alive* (ping) message to determine its load status and whether the server is still operating or not. Sending the message is done via a web service which is hosted by the session server. After receiving server's status the registry updates it to the database so that authentication servers can forward clients and frontends to session servers with little of load.

3.4. Resuming a session

A communications error may arise during a stream sending, for example because of client crashing. In these cases it is possible for a client to continue stream sending after connection is re-established. This is possible because all data concerning a session, such as stream IDs, are stored to database.

This is done by requesting a session from authentication server and providing the session ID of last session. As a response, client receives storage session servers URL with which it can start communicating as

© 2008 Vincit Oy

Hermiankatu 6 A
33720 Tampere
+358 40 589 8316
info@vincit.fi
VAT reg no
FI 2113034-9

usually. To resume a stream, client makes a *stream resume* request and provides the stream ID that was used previously with the stream that is to be resumed. It is client's responsibility to store both session ID and stream IDs to persistent memory - if the IDs are lost, resuming cannot be done.

It is not needed to resume a session when client wants to resume a file sending because files are identified by names that client provide, not with "file IDs". How resuming a file sending can be done is described in section 4.1.

4. PROTOCOLS

4.1. HTTP

HTTP is used to send files and events to storage session server. Both of these are sent using HTTP POST mechanism with a packet type shown in Figure 3. Events need not to be associated with streams or files, but they can simply contain arbitrary data, usually combined with GPS location.

HTTP header consists of several fields needed to identify the event. A field is a key-value pair where field name acts as a key and field content acts a value. Fields can be divided into three different groups: HTTP specific fields, LAME specific fields and user-defined fields. All field names **case-insensitive**. Also, all field values in HTTP specific fields are **case-insensitive**. Values of LAME specific fields and user-defined fields are **case-sensitive**. All fields take form of *<field name>: <field value><end-of-line>*, for example "Content-Length: 2312".

HTTP specific fields are used to transmit HTTP related data. The most common HTTP fields used in LAME are following: *content-type* and *content-length*.

Content type describes the media type of payload data. Media type takes form of *<type>/<subtype>*. In LAME, type is always "*x-lame*".

© 2008 Vincit Oy

Hermiankatu 6 A
33720 Tampere
+358 40 589 8316
info@vincit.fi
VAT reg no
FI 2113034-9

Note that prefix "*x-*" is needed because LAME events are not registered with IANA. Subtype can be one of the following: file or event.

Value of field *Content-Length* is the size of HTTP body in bytes. If client is not sending a file, body is empty and value of this field must be zero.

LAME specific fields are used to transmit metadata that is needed by servers in order to handle incoming requests. The most common LAME fields are following: *sessionID*, t*imestamp*, *streamID*, f*ilename*, *filemode*, *length* and *hash*.

Value of field *sessionID* is the session ID that the authentication server assigned to the client. Value of field *Timestamp* is the time when event occurs. Timestamps are described in more details in section 5.

```
POST / HTTP/1.1
Content-Type: lame/event
Content-Length: (length of body in octets)
SessionID: ID
[Filename: Path\Name] (only for files)
[Mode: New | Resume | Overwrite] (only for files)
[StreamID: stream ID] (only if event is associated with a stream)
Timestamp: Time
PredefinedKey: Value
CustomType: CustomTag: Value
CustomType2: CustomTag2: Value, CustomTag3: Value
- - - - - - - - - -(empty line) - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
[file data]
```

Figure 3: HTTP packet used in LAME

Fields *filename* and *file*mode exist only if event is actually a file. In this case, value of *filename* is the path and the name with which the file will be stored. The path can consist of multiple directories separated by backslash (\). Value of *Mode* is either *new*, *resume, overwrite* or *request.* Value *new* means that client is sending a new file, and the sending will fail if a file with the same name already exists. Value *resume* means that if a file with the same name already exists, the data that client is sends will be appended to that file. Value *overwrite* means that if a file with the same name exists, it will be deleted and new file will be created. Value *request* means that server

© 2008 Vincit Oy

Hermiankatu 6 A
33720 Tampere
+358 40 589 8316
info@vincit.fi
VAT reg no
FI 2113034-9

checks if file with given name already exists and returns its length and 128-bit MD5 hash value. Values are returned in HTTP response which contains fields *Length* and *Hash*. Hash value is a 16-character long string representing a hexadecimal number. With mode *request*, any data in HTTP body will be ignored. If client is sending file with mode *new* and a file with same name already exists in server, response is same as if client had used mode *Request*. Field *StreamID* exists only if event is associated with a stream. In this case, the value of this field is the stream ID.

In addition to these fields, user-defined fields can be added to files and events. Using user-defined fields makes it possible to add arbitrary user-defined meta- or control data to streams and files. A central concept in user-defined metadata is **tag**. Tags are described in more details in section 4.4. In HTTP message, name of user-defined field is the type of tag and value of the field is one or more key-value pairs separated by commas: *<tag type>: <tag1>: <tag1value>, <tag2>: <tag2value>, … , <tagX>: <tagXvalue>*. Because of HTTP standard, it is also possible to define same tag type more than once [RFC 2616, 4.2]. The result of this is the same as defining all tags in one line.

Server responds always to an event regardless of its type. This response contains always at least lines HTTP/1.1 (status code) (status code as a text) and Status-code: (status code). General status codes (and their corresponding textual versions) are 200 (OK) and 400 (Bad request). Response may contain additional key-value pairs or HTTP body if needed.

4.2. RTP

Real-time Transfer Protocol is a protocol (defined in RFC 3550) which is used in unreliable transferring of audio and video streams over the Internet. To support all features of LAME, RTP's header format must be adjusted a bit. The protocol will still follow the standard after this adjustment.

RTP supports adding an extension to the header that can be used to store user-defined information. In LAME, it must be possible to add

© 2008 Vincit Oy

Hermiankatu 6 A
33720 Tampere
+358 40 589 8316
info@vincit.fi
VAT reg no
FI 2113034-9

several extensions so RTP header extension is modified to be like shown in Figure 4.

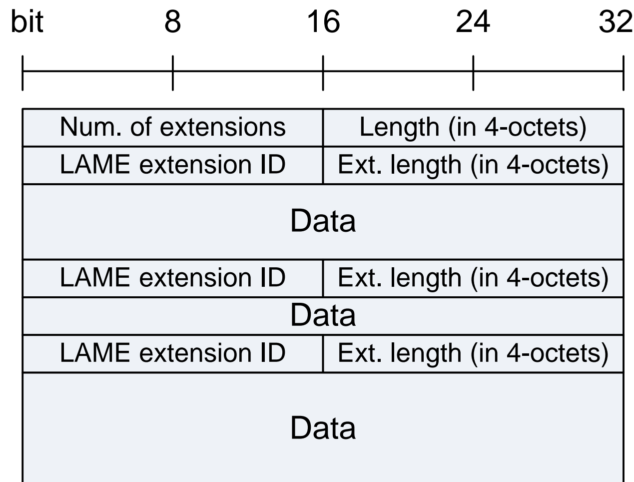

Figure 4: RTP header extension in LAME

Number of extensions tells how many LAME extensions there are. *Length* is the total length of extensions in 4-octects after this field, so zero is a valid value. These fields are followed by the extensions. An extension starts with 16-bit field *extension ID* that is either pre- or user-defined identifier. Second field is *extension length* which tells the length of actual extension data in 4-octects. Third field contains the actual data.

4.3. LRS

Reliable streams are transferred over TCP. As TCP does not directly support all features needed by LAME, a wrapper protocol is defined around it. This protocol is called LRS, LAME Reliable Stream. The packet format of LRS is shown in Figure 5.

© 2008 Vincit Oy

Hermiankatu 6 A
33720 Tampere

+358 40 589 8316
info@vincit.fi

VAT reg no
FI 2113034-9

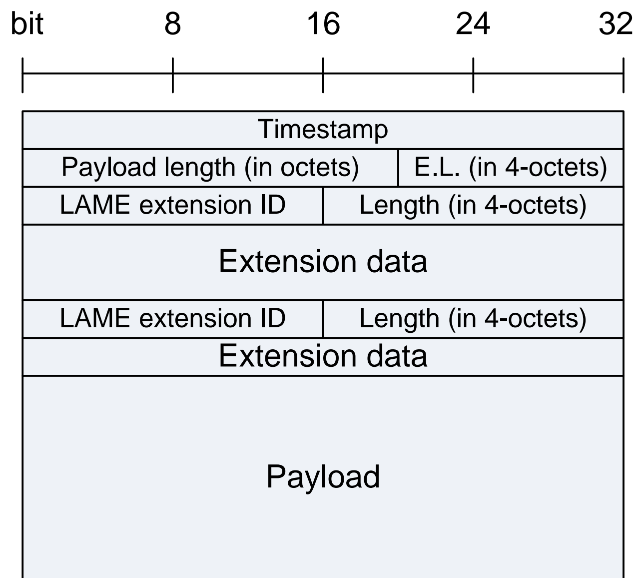

Figure 5: LRS packet format

Timestamp reflects the sampling instant of the first octet in the payload field. Time is informed in milliseconds and in relation to the starting time of the stream. Therefore timestamp of the first packet should usually be zero. Field *payload length* tells the length of field *payload* in octets. Field *extension length (E.L)* tells the length of extension fields in 4-octets. If this number is zero then no extensions exist. Last field, after the extensions, is the actual payload.

Extensions are defined in the same way as in the RTP header. First field is extension ID, second field is the length of extension data in 4-octets and the third field contains the actual data.

4.4. Tags

Tags are concept of adding user-defined (i.e. non-LAME) meta- or control data to streams, events and files. For example, an image could have tags *winter* and *night* implying that the image has been captured at night in winter. Also, image could have a tag *temperature* with value *-20* implying that the temperature was -20C at the time when image was captured.

A tag has three properties: type, name and value. A type defines a group of tags and name identifies a tag inside one group. That is,

© 2008 Vincit Oy

Hermiankatu 6 A
33720 Tampere
+358 40 589 8316
info@vincit.fi
VAT reg no
FI 2113034-9

only one tag name can exist inside one group but other tags with same name can exist in different groups. Name and value can be thought as a simple key-value pair where value can also be nil.

In example above, tag names were *winter*, *night* and *temperature* with corresponding values of *nil*, *nil* and -20. The type for two first tags could be *description* and *miscinfo* for the last one.

All tags that has been predefined in LAME (such as *location*) have type of *lame*.

4.5. Security

In LAME, some of the information that is sent over Internet are confidential. In addition to username and password that are used in authentication, also files and streams sent to storage session server may contain data that needs to be secured.

All information related to authentication is always secured with the help of SSL. This means that web services hosted by authentication servers use HTTPS instead of HTTP as their transport protocol. Web services hosted by session servers are not secure by default but it is also possible to use them above HTTPS instead of plain HTTP. As files and events are sent by using HTTP, securing them is also done by changing protocol from HTTP to HTTPS.

Reliable streams can be secured with the help of SSL. As SSL is designed to work above reliable transport protocol, it cannot be used with unreliable streams. Therefore for now, unreliable streams cannot be secured.

5. TIMING

Streams are timed with the help of reference clock and timestamp that is located in every packet. When requesting a permission to send a stream, client can either set the value reference clock itself or rely on server's system clock. Timestamps in packets tell the number of milliseconds that has passed since beginning of the stream.

© 2008 Vincit Oy

Hermiankatu 6 A
33720 Tampere
+358 40 589 8316
info@vincit.fi
VAT reg no
FI 2113034-9

Events are attached to streams with the help of timestamps and stream IDs as shown in Figure 6. Each event has a timestamp which behaves exactly in the same way as they do in stream packets. If event's timestamp does not exactly match timestamps in stream packets, event is attached to the first packet which timestamp is smaller than event's timestamp. For example, if there are two stream packets with timestamps 360 and 480, and one event with timestamp 440, the event is attached to the packet with timestamp of 360.

Files are considered to be unitary, that is files are not composed of small pieces such as packets. Therefore files have at the most only one timestamp, and this timestamp is the absolute value of time, not relative to reference clock.

Absolute timestamps are used in the same way as HTTP uses them. This is described in RFC 1123. The format is following "Sun, 06 Nov 1994 08:49:37 GMT". Therefore resolution of absolute timestamps is 1 second whereas with relative timestamps resolution is 1 millisecond.

© 2008 Vincit Oy

Hermiankatu 6 A
33720 Tampere
+358 40 589 8316
info@vincit.fi
VAT reg no
FI 2113034-9

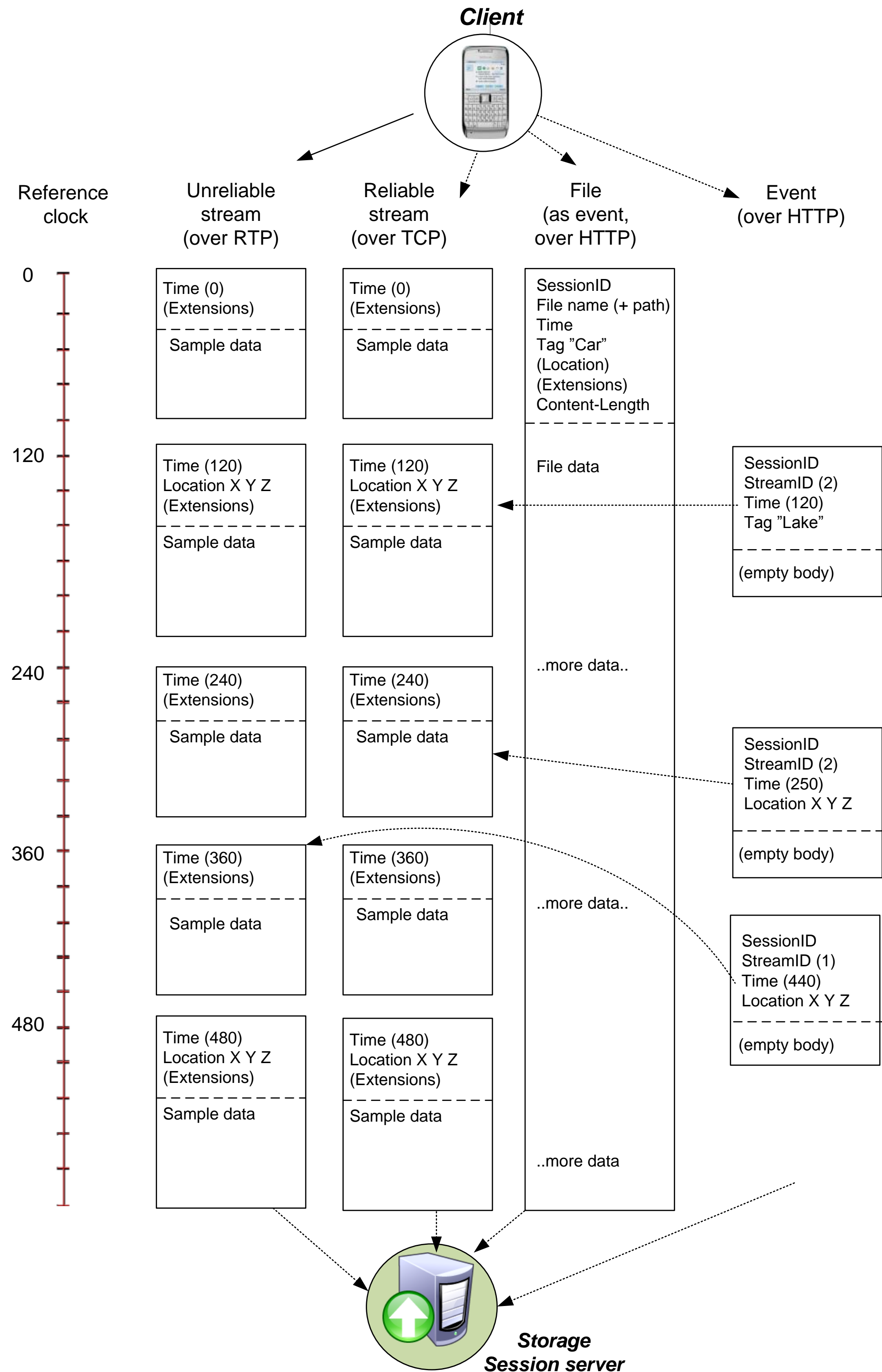

Figure 6: Timing

© 2008 Vincit Oy

Hermiankatu 6 A
33720 Tampere

+358 40 589 8316
info@vincit.fi

VAT reg no
FI 2113034-9

6. APPENDIX

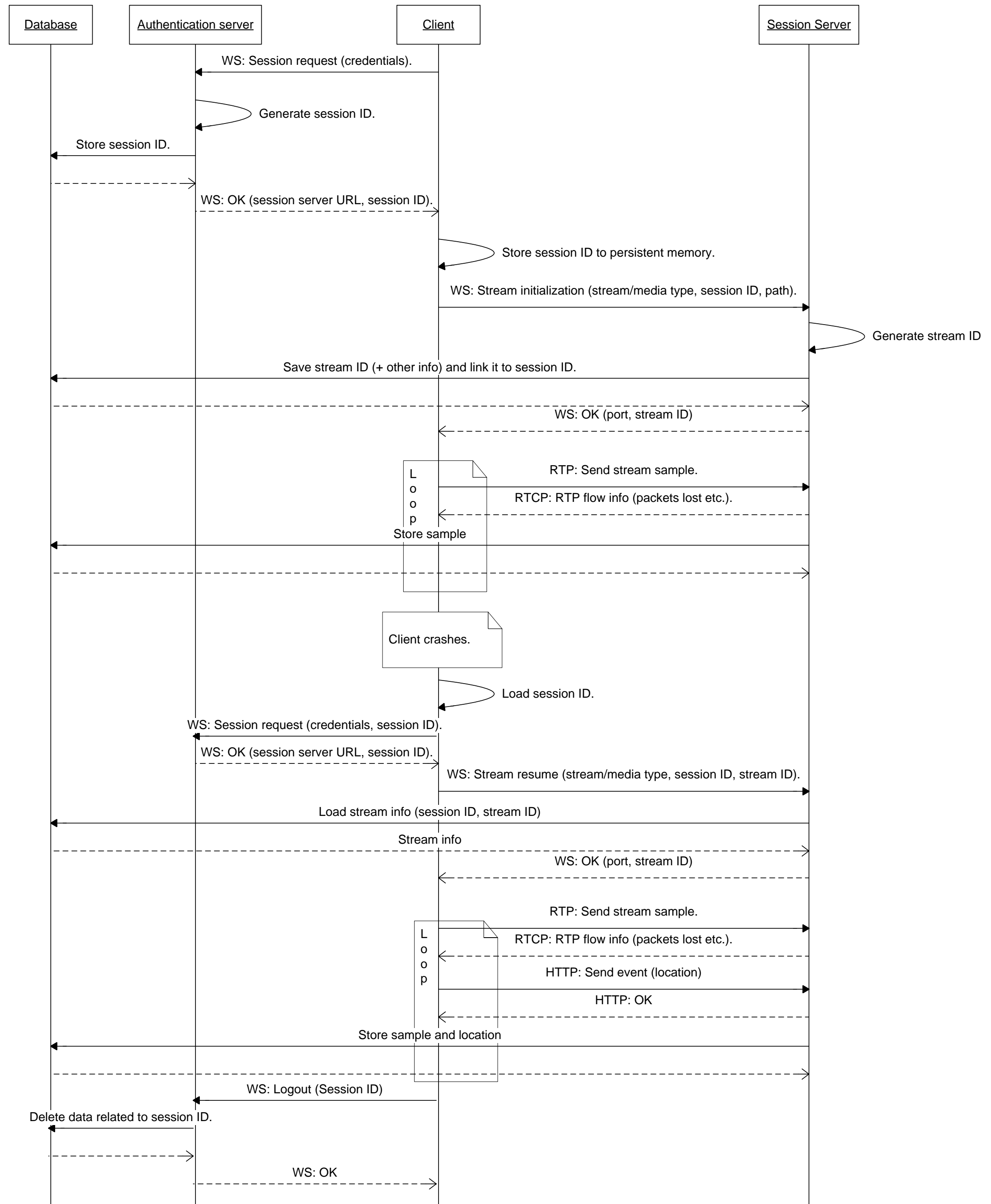

Figure 7: Storing a stream to the server.

© 2008 Vincit Oy

Hermiankatu 6 A
33720 Tampere

+358 40 589 8316
info@vincit.fi

VAT reg no
FI 2113034-9

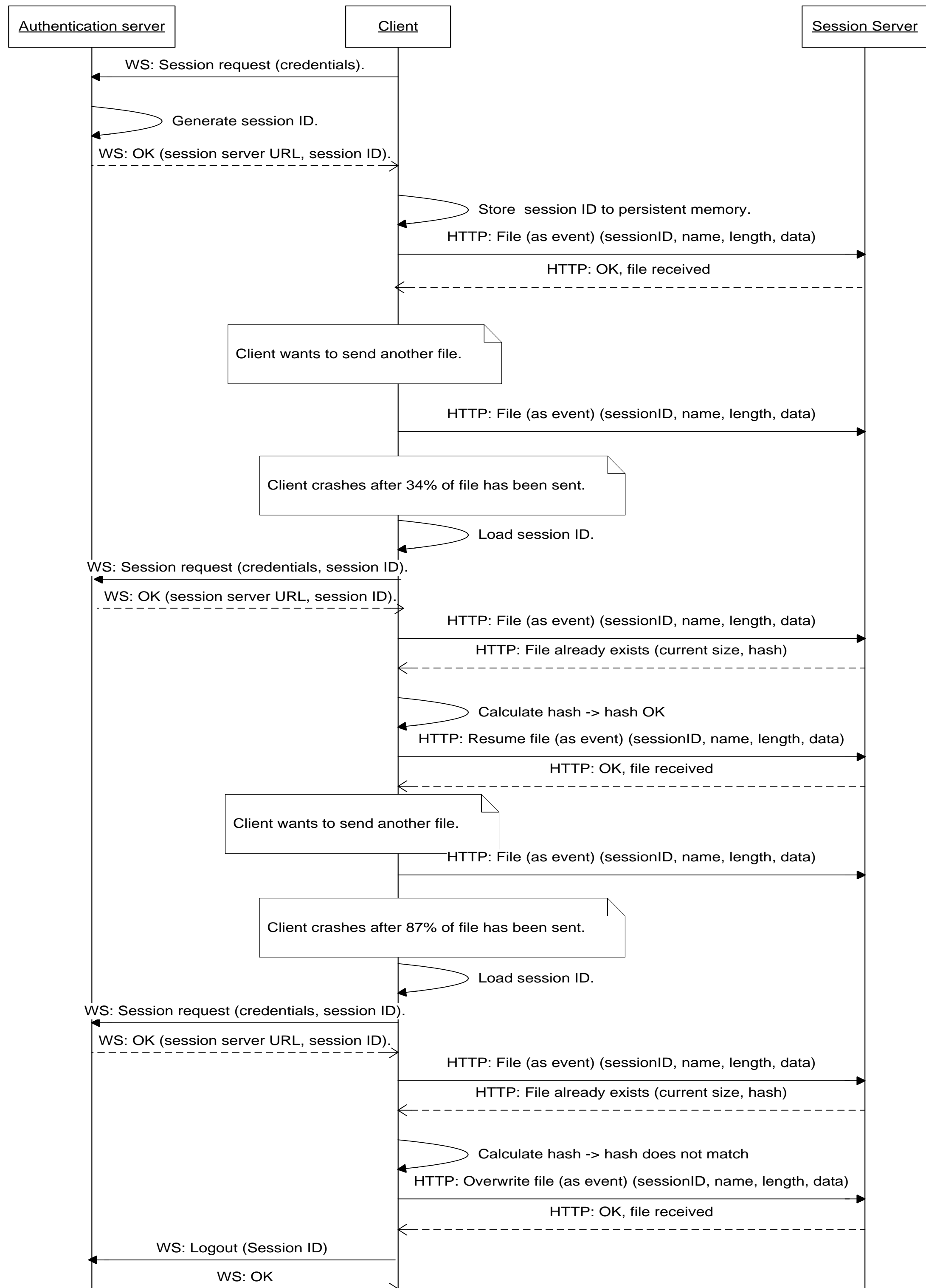

Figure 8: Storing a file to the server.

© 2008 Vincit Oy

Hermiankatu 6 A
33720 Tampere

+358 40 589 8316
info@vincit.fi

VAT reg no
FI 2113034-9