

**METHOD AND DEVICE FOR EXTRACTING DATA FROM A DATA STREAM
FLOWING OVER AN IP NETWORK**

[0001] The present invention relates to analysis techniques relating to a data stream flowing over telecommunication networks, in IP ("Internet Protocol") packets.
5 More particularly, in this case it is sought to extract in real time data from one or more specified categories without having to deal with the enormous amount of data flowing over the network.

[0002] IP packet analysers such as that distributed under the name Wireshark carry out a global extraction from a stream of the content of the packets carrying data
10 in order to then subject it to a complete analysis allowing each of the different elements constituting this content to be identified. This methodology is not well adapted to the real time observation of multiple streams because it requires the extraction in real time of all of a stream for its analysis. Moreover, in the case of a change in the protocol according to which the stream is constructed, it is necessary
15 to modify the analyser, even if the change is minor or relates to aspects of the protocol which are not relevant to the information sought.

[0003] A need exists for a technique allowing efficient extraction and synchronic mapping of targeted information present in the data streams over IP type networks.

[0004] A method is proposed for extracting data from a data stream flowing over
20 an IP network, the data of the stream being organised according to an application layer protocol (layer 7 of the OSI model). In this context and from the point of view of the application layer, the stream is segmented in a virtually random manner by the processes of the network (IP) layer and/or transport (TCP, UDP, etc.) layer. Nevertheless it is desirable, given that the traffic flows at a very high bit rate over the
25 IP network, to have the capability to extract specific data contained in the stream for processing. The method comprises:

- a configuration phase in which a state machine is constructed with states and transitions configured according to at least one type of data to be extracted from the stream, the transitions between states being activated by the
30 respective conditions defined as a function of rules for organization of the data

of the stream according to the application layer protocol, and in which at least one state is selected for the extraction of data from the stream; and

- a real time analysis phase of the data stream.

[0005] The real time analysis phase comprises:

- 5 • observing the data of the stream originating from IP packets flowing successively over the network;
- when the state machine is in a current state, determining if an activation condition for a transition from the current state to a target state is fulfilled by the observed data of the stream and when this activation condition is
- 10 fulfilled, changing the state machine to the target state;
- extracting the data from the stream when the state machine is in a state selected in the configuration phase for the extraction of data from the stream; and
- disregarding the data of the stream when the state machine is in a state
- 15 not selected in the configuration phase.

[0006] The state machine, the nodes of which describe the relevant structural elements of the grammar of the protocol, makes it possible to extract the useful semantic and subsumption information when they appear in the stream. The state machine is composed of states and transitions. The transitions make it possible to

20 move from one state to another and are typically activated by lexemes during observation of the data stream.

[0007] The state machine used operates on data streams which can have a very variable content (textual or binary), the syntax of which is not necessarily perfectly known, which are likely to contain errors and which are not available in their entirety

25 at each moment in time.

[0008] In order to do this, without having to extract all of the stream, the method searches for the conditions allowing the activation of the different transitions by analysing the data of the stream in real time. The transition activations and the data present between two states can be managed separately.

30 [0009] Moreover, the method allows management of the storage in buffer memory of only those portions of the stream necessary to the search for activation

conditions of the transitions in the presence of fragmentation of the data. In such an embodiment, the activation conditions of transitions of the state machine comprise the presence of respective lexemes in the data of the stream, and the real time analysis phase of the data stream comprises, when the state machine is in a current state:

- storing in buffer memory at least N-1 characters situated at the end of the data of the stream observed in an IP packet, N being the largest number of characters of the lexemes corresponding to the transitions out of the current state; and
- on receiving next data of the stream originating from an IP packet flowing subsequently over the network, placing the content of the buffer memory in front of the data received in order to search for the possible presence of a lexeme split between the two packets.

[0010] The states of the state machine selected in the configuration phase can comprise one or more states in which the data extracted from the stream are directly transferred to an external processor.

[0011] The selected states of the state machine can also comprise one or more states in which the data extracted from the stream are stored in buffer memory then transferred to an external processor once the state machine leaves this selected state. The buffer memory receiving the data extracted from the stream in such a selected state of the state machine preferably has a size limited to a configurable number of characters.

[0012] Another aspect of the present invention relates to a device suitable for implementing the above method. This device comprises:

- a state machine having states and transitions configured according to at least one type of data to be extracted from the stream, the transitions between states being activated by the respective conditions defined as a function of rules for organization of the data of the stream according to the application layer protocol, at least one state of the state machine being selected for the extraction of data from the stream;
- an input for receiving in real time the data of the stream originating from IP packets flowing successively over the network;

- a detector of transitions for determining, when the state machine is in a current state, if an activation condition of a transition from the current state to a target state is fulfilled by the data of the stream received at the input and for changing the state machine to the target state when said activation condition is fulfilled.

5 The data of the stream are extracted when the state machine is in a selected state for the extraction of data from the stream, and disregarded when the state machine is in an unselected state for the extraction of data from the stream.

[0013] Other features and advantages of the present invention will become apparent from the following description of a non-limitative embodiment, with
10 reference to the attached drawings, in which:

- Figure 1 is a simplified diagram of an IP network to which the invention can be applied;
- Figure 2 is a block diagram of equipment which can be connected to the network of Figure 1 for implementation of the invention;
- 15 - Figure 3 is a block diagram of an example of a data extraction device according to the invention; and
- Figure 4 is a diagram of a state machine which can be used in a specific case presented by way of illustration.

[0014] With reference to Figure 1, an IP network 1 such as the Internet comprises
20 in a standard fashion different routing equipments, some (2) being internal to the network and others (3) being situated on the periphery for connecting various installations, such as user terminals 3, user computer installations 4, servers by which the network operators manage their subscribers, gateways to other networks, etc.

25 [0015] The links between the routers 2, 3 are carried out by very high bit-rate connections provided, for example, by optical fibre lines. Representative values of the bit rate are several tens of gigabits per second.

[0016] Within the different data streams that are carried by these high bit-rate links, certain applications require the extraction of specific data, for example for the
30 purposes of billing, security, management of the quality of services, etc.

[0017] It is desirable that the devices receiving the data extracted in this way are

not overwhelmed by the potentially immense quantities of data which flow over the IP network 1. To this end, equipment 10 can be used such as that represented diagrammatically in Figure 2.

[0018] This equipment 10 is typically installed at the level of a router 2, 3 so as to interface with one of the high bit-rate links between these routers. However, it will be understood that the equipment 10 can also be installed on a link situated between an end router 3 and gateways or user installations.

[0019] The equipment 10 represented in Figure 2 comprises a network interface 11 adapted to the physical layer and to the lower protocol layers of the link on which the equipment is mounted. The traffic seen by the network interface 11 is subjected to a classifier 12 capable of identifying the data stream to which the successive IP packets constituting this traffic belong. The classifier 12 implements a recognition and protocol analysis technique such as for example that described in WO 2004/017595 A2. Its architecture can optionally be distributed as described in WO 2006/120039 A1.

[0020] The traffic classifier 12 is configured to selectively present the IP packets which are taken from one or more data streams specified by the system administrator to a data extraction device 13. For each of these streams, the extractor 13 selects the relevant data as a function of a configuration performed by a user, extracts them from the stream and communicates them to an external processor 14 which carries out the required processing on the data extracted (for example processing for billing, for security or quality of service applications, etc.). The user to which reference is made in this case is that which manages the applications executed by the external processor 14. It is possible to combine this user with the system administrator specifying the data stream to be extracted. In another service-oriented architecture, the user can equally be separate from the system administrator if the latter offers the data extraction service to several people having different types of processing to carry out.

[0021] The data extractor 13 can have an architecture such as that illustrated by Figure 3. This architecture is centred on a state machine 20 and a detector of transitions 21 that the user configures via an appropriate human/machine interface 22. In a configuration phase, the user defines via the interface 22 the states and the

transitions of the state machine 20 as well as the activation conditions of the transitions between the states. The interface 22 can incorporate knowledge of the grammar of the protocol in order to assist the user in the configuration operation.

5 [0022] The data extractor 13 has an input 24 which receives in real time the data of the stream originating from IP packets, which are presented to it by the traffic classifier 12. The data extracted from this stream in accordance with the configuration carried out via the interface 22 are delivered to the external processor 14 by the data output 25.

10 [0023] The use of a state machine in the case of networks of IP type imposes operational constraints. In fact, the streams are then fragmented, each fragment originating from an IP packet being of variable size. This fragmentation can take place at any moment in the grammar of the protocol. The two following cases should be taken into account:

- 15 - the data necessary for the activation of the condition of a transition can be split between the end of a packet and the start of the following packet;
- the useful data to be processed in a state can be present over several packets whereas the user can request that they are not processed in a fragmented manner.

20 [0024] In order to take these two cases into account, the data extractor 13 comprises two buffer memories 26, 27 (in practice, these buffers 26, 27 can be produced inside a single memory plane).

25 [0025] The buffer memory 26 serves to manage situations where a lexeme sought by the detector of transitions 21 is found to be split between two IP packets successively received at the input 24. In a given state of the state machine 20, several lexemes can be sought for the possible activation of the transitions available out of this state. If N denotes the maximum number of characters of these lexemes, it is convenient to record in the buffer memory 26 the last N-1 characters of the data stream received in each IP packet received when the state machine is in this given state. On reception of the next IP packet containing data of the stream, the content
30 of the buffer memory 26 is placed in front of the first character of the data received at the input 24 so that the detector of transitions 21 can observe the possible presence

of one of the sought lexemes. This cooperation between the detector of transitions 21 and the buffer memory 26 ensures that the sought lexemes are not lost due to segmentation of the stream at the IP level. It will be noted that the buffer memory 26 can optionally contain slightly more than $N-1$ characters, its size must nevertheless remain considerably less than that of the data to be extracted.

[0026] The state machine 20 comprises a list of states established according to the structure of the grammar of the stream. Each state contains a list of transitions the activation conditions of which, sought by the detector of transitions 21, are typically the presence of lexemes received in the data stream. Each state is moreover associated with a procedure indicating the way in which to process the data which are received while the machine 20 is in this state.

[0027] Each transition of a state specifies an activation condition and the associated target state, as well as the minimum length of data that it is necessary to analyze in order to ensure the operation of the condition. The activation condition, specific to a state and to the transition on which it depends, can take one of the following forms:

- discovery of a given string of characters by using for example the Boyer-Moore search algorithm;
- discovery of a given string of characters by using a search tree (prefix tree or ordered tree data structure);
- progression in the stream of a given number of bytes;
- progression in the stream of a variable number of bytes which is a generalization of the previous case. In this case, the size of the hop is not indicated in the description of the state machine, but is given at the input of the state out of which the transition exists;
- a generic condition: in practice, the state machine is informed of this condition via a function capable of finding the transition, which makes a generalization thereof that encompasses all the preceding cases.

[0028] The conditions are generally activated following a progression in reading the data stream received at the input 24. When such a transition is activated, it is possible to exploit the data present before activation of the transition allowing

transition to the next state. According to the configuration requested by the user, the data present between two changes of state can be:

- disregarded; or
- transmitted to a function linked with the first state for processing. This function is typically the immediate sending of the data via the output 25; or
- stored in the buffer memory 27 then transmitted to a function linked with the first state for processing. This function is typically the sending of the data via the output 25 at the moment when the state machine 20 leaves the current state. The buffer memory 27 can be configured so as to have a maximum size in terms of number of characters. The state machine 20 can be forced to leave the current state when this maximum number of characters has been saved in the buffer memory 27. This allows an overflow of this buffer memory to be avoided.

[0029] By way of illustration, the particular examples of the implementation of the invention are presented hereafter in the case of the instant messaging protocol known as Jabber or XMPP ("Extensible Messaging and Presence Protocol"), based on the XML language ("Extensible Markup Language"). These examples can be generalized without difficulty to other extensions of the Jabber/XMPP protocol or to other protocols.

[0030] For the sake of performance (memory and processing time of the analysis of the text) and of robustness, the grammar of the Jabber protocol is not described in full. Only those markers are considered which constitute the invariants around the information which it is desired to extract.

[0031] In Example 1, the user seeks to extract the content of the messages transmitted in the Jabber protocol.

[0032] In Example 2, the extraction relates to contacts (e-mail addresses) transmitted in the Jabber protocol.

[0033] The state machine 20 can be constructed according to the diagram illustrated by Figure 4 for implementing these embodiments. Two types of states are selected by the user in the state machine 20 for carrying out the extraction of the data. In the first type (state 30 called "NODE_BODY"), the content of the messages

will never be memorized in the extraction device 13 as it can be very voluminous. In the second type (state 31 called "TR_CONTACT_ENTRY"), the e-mail addresses may be stored in buffer memory 27 in order to be transmitted in one go to the external processor 14.

5 [0034] In the two cases, during the callback procedure for the transfer of data in a selected state, the extractor 13 provides the processor 14 with the information on the current state (NODE_BODY or TR_CONTACT_ENTRY) as well as the data extracted, which gives the semantic and subsumption information necessary for the correct interpretation of the data.

10 [0035] From a node 32 called "NODE_BASE", the initial position of the state machine 20, a branch of the graph describing the state machine detects the start of the content of a message from the detection of the lexeme *<message*, while another branch searches for an item of information present only in the signalling data designated by the lexeme: *iq:roster*.

15 [0036] The states and transitions of the state machine 20 represented in Figure 4 are defined in the following manner in the configuration phase:

```
{ node: NODE_BASE, nextnode: NODE_MESSAGE, start: " <message " },
{ node: NODE_BASE, nextnode: NODE_CONTACT_LIST, start: ":iq:roster " },
{ node: NODE_MESSAGE, nextnode: NODE_BODY, start: " <body> " },
20 { node: NODE_BODY, nextnode: NODE_BASE, start: " </body> " },
{ node: NODE_CONTACT_LIST, nextnode: NODE_BASE, start: " </query> " },
{ node: NODE_CONTACT_LIST, nextnode: NODE_CONTACT_ENTRY, start:
" <item " },
{ node: NODE_CONTACT_ENTRY, transnode: TR_CONTACT_ENTRY, start:
25 " jid=' ", end: " /> ", flag: SM_TRUNCATE },
{ node: NODE_CONTACT_ENTRY, nextnode: NODE_BASE, start: " </query> " },
```

where:

- "node" is the name of the current state;

- "nextnode" is the name of the next state when the transition condition is activated;
- "start" denotes the marker for searching in the stream in order to activate this transition.

5 [0037] In the case where a temporary state linked to the storage in buffer memory 27 is used by the state machine, there are two additional items of information:

- "transnode" (transitory node): name of the temporary state of the rapid transition;
- "end": marker sought in order to leave the rapid transition;
- 10 • "flag": optional marker. In the present case, it indicates truncation of the data if the end marker cannot be found when the buffer memory 27 is full.

Example 1:

[0038] After detection of the lexeme `<message`, the first branch brings the state machine 20 to the `NODE_MESSAGE` (33) state where no extraction has yet been carried out. The detection of the lexeme `<body>` in the `NODE_MESSAGE` state will then bring the machine 20 to the `NODE_BODY` state. In this state 30, the callback sends all the received data from the successive IP packets to the output 25, until the end marker `</body>` is detected which returns directly to the initial state 32.

[0039] That way, the data extractor 13 isolates any content of the messages of the stream included between the XML tags `<body>` and `</body>` defined in the protocol. As the content of the message can be voluminous, it does not pass through the buffer memory 27. It will be transmitted in one or more goes, on successive reception of the IP packets.

[0040] The application stream can for example be presented thus:

```
25 <message xmlns="jabber:client" type="chat" to="cyberic99@gmail.com" id="aac3a">
    <body>Bonjour Eric</body>
    <active xmlns="http://jabber.org/protocol/chatstates"/>
    <nick xmlns="http://jabber.org/protocol/nick">sir swiss</nick>
    </message>
```

30 [0041] The extractor 13 will then provide the body of the message *"Bonjour Eric"* to

the external processor 14.

[0042] It can occur that the start marker is segmented over two IP packets, for example:

A. `<message xmlns="jabber:client" type="chat" to="cyberic99@gmail.com" id="aac3a"><b`

5 B. `ody>Bonjour Eric</body><active xmlns="http://jabber.org/protocol/chatstates"/><nick
xmlns="http://jabber.org/protocol/nick">sir swiss</nick>`

[0043] In the NODE_MESSAGE state where the state machine 20 is at the end packet A, the length of the longest lexeme (`<body>` = 6 characters) activating a transition, less one character (1 byte), i.e. five characters, is stored in buffer memory 10 26. At the end packet A, the memory 26 then contains "a">"b". On reception of the packet B, the strings of characters "<b" and "ody>" are reassembled and the message is transmitted in one go as it is complete in packet 2. The remainder of the stream is disregarded by the extractor 13.

[0044] It can also occur that the content of the message is segmented over 15 several IP packets, for example:

C. `<message xmlns="jabber:client" type="chat" to="cyberic99@gmail.com"
id="aac3a"><body>Bonjour`

D. `Eric</body><active xmlns="http://jabber.org/protocol/chatstates"/><nick
xmlns="http://jabber.org/protocol/nick">sir swiss</nick></message>`

20 [0045] In the NODE_BODY state where the state machine 20 is at the end of packet C, the length of the longest lexeme (`</body>` = 7 characters) activating a transition, less one character, i.e. six characters, is stored in buffer memory 26. Only "B" is transmitted to the processor 14 on reception of this packet C, "onjour" being held in memory 26. On reception of the packet D, the end marker `</body>` is detected, 25 which sends the data of the memory 26 and those of the new packet situated before the end marker to the output 25, i.e. in all: "onjour Eric". The processor 14 can then proceed with the reassembly of the strings "B" and "onjour Eric" successively received from the extractor 13.

[0046] It can also occur that the end marker `</body>` is segmented over two IP 30 packets, for example:

E. `<message xmlns="jabber:client" type="chat" to="cyberic99@gmail.com" id="aac3a"><body>Bonjour Eric</bo`

F. `dy><active xmlns="http://jabber.org/protocol/chatstates"/><nick xmlns="http://jabber.org/protocol/nick">sir swiss</nick></message>`

5 [0047] In the state NODE_BODY where the state machine 20 is at the end of packet E, six characters are stored in buffer memory 26 at the end of each IP packet. At the end of packet E, the memory 26 then contains "`ic</bo`". Only "`Bonjour Er`" is transmitted to the processor 14 during the reception of this packet E. On reception of packet F, the end marker `</body>` is reassembled and detected, which sends to the
10 output 25 the data of the memory 26 except those which belong with the marker detected, i.e.: "`ic`".

[0048] The method described here for two packets C, D or E, F is general for any splitting of the data stream. If several packets smaller than the size of the body of the message are received successively, they are transmitted as and when they are
15 received except the six last characters retained in buffer memory 26 until reception of the following packet of the stream, in order to make it possible to search for the largest transition out of the current state 30.

Example 2:

[0049] After detection of the lexeme: `iq:roster`, the second branch of the graph of
20 Figure 4 brings the state machine 20 to the NODE_CONTACT_LIST state (34) where no extraction is yet carried out. If the lexeme `</item` is detected in the stream at the NODE_CONTACT_LIST state, the state machine 20 moves to the NODE_CONTACT_ENTRY state (35) where no extraction is carried out either. The detector of transitions 21 must then observe the lexeme `jid='` in the stream in order to
25 move to the TR_CONTACT_ENTRY state. In this state 31, the content of the stream received on the input 24 is written to the buffer memory 27 until the end marker `>` is detected which resends to the NODE_CONTACT_ENTRY state 35. If the lexeme `</query` is detected in the stream at the state 34 or 35 (NODE_CONTACT_LIST or NODE_CONTACT_ENTRY), the state machine 20 returns to the base state 32. The
30 transition to the TR_CONTACT_ENTRY state is called a rapid transition, as the state machine 20 can only remain in this state 31 for the requirements of storing in buffer

memory the content between the two start and end markers *jid='* and */>*. When the end marker */>* has been detected, the state machine 20 returns to the state `NODE_CONTACT_ENTRY`. The detector of transitions 21 can thus continue to read the e-mail addresses of the contacts until the marker *</query* of the end of signalling is detected.

[0050] The data extractor 13 therefore searches in the XML token *<item>* for the content of the attribute *jid*, i.e. the text contained between *jid='* and */>*. As the e-mail addresses are by nature relatively small, it can be requested that it is transmitted in one go, in the case where it is segmented over several packets. The state machine must optionally put the content into buffer memory 27 until reception of the end marker */>*.

[0051] The application stream can for example be presented thus:

```
<iq from='qosmojab@swissjabber.org/dev1' type='result' id='aab6a'>
  <query xmlns='jabber:iq:roster'>
    <item subscription='both' jid='qosmojab@im.apinc.org'/>
    <item subscription='both' name='babydaisy' jid='babydaisy@binaryfreedom.info'/>
    <item subscription='to' jid='roiboo.crusher@gmail.com'/>
    <item subscription='to' jid='cyberic99@gmail.com'/>
  </query></iq>
```

[0052] When an e-mail address is not fragmented, the two start and end markers *jid='* and */>* being present in the same packet of the stream, the extraction of the address and its transmission by the processor 14 are carried out in one go, without use of the buffer memory 27.

[0053] If the start marker *jid='* is segmented between two IP packets, the procedure is the same as in Example 1, with recording in the buffer memory 26 of the length of the first transition less 1 byte. On reception of the second packet, the marker *jid='* is reconstituted and the transition activated, in order to enter into the temporary `TR_CONTACT_ENTRY` state.

[0054] An e-mail address can be segmented between two IP packets, for example:

G. *<iq from='qosmojab@swissjabber.org/dev1' type='result' id='aab6a'><query*

xmlns='jabber:iq:roster'><item subscription='both' jid='qosmojab@im

H. *.apinc.org'/><item subscription='both' name='babydaisy'*
jid='babydaisy@binaryfreedom.info'/><item subscription='to'
jid='roiboo.crusher@gmail.com'/><item subscription='to'
 5 *jid='cyberic99@gmail.com'/></query></iq>*

[0055] Once the transition of marker *jid='* is activated, the state machine 20 enters the temporary TR_CONTACT_ENTRY state where the data are written to the buffer memory 27 until the end transition *'/>*. This buffer memory 27 is of a limited size, the maximum value of which is for example 50 bytes (configurable) plus the size of the
 10 end marker sought *'/>* (i.e. 53 bytes in total). In the packet G, the string of characters "*qosmojab@im*" is stored in memory. Then, in the packet H, the data are added to this memory 27 until the maximum size calculated previously is reached. When the end marker *'/>* is found, all of the data recorded in the memory 27 are transmitted to the output of 25, and the state machine 20 leaves the TR_CONTACT_ENTRY state in
 15 order to return to the state 35 NODE_CONTACT_ENTRY.

[0056] If the end marker is not found, there are two possibilities between which a choice is made during the configuration using the options of the transitions. The general case is to continue searching for the end marker *'/>*, by replacing the oldest stored data by those which are read in the stream. The data transmitted when the
 20 end marker is found are then the 50 bytes which precede it. Another possibility consists of truncating the data at the maximum size that can be stored, and activating the end transition in order to force the change of state (SM_TRUNCATE option).

[0057] The embodiments described above are illustrations of the present invention. Various modifications can be made to them without exceeding the scope
 25 of the invention which is apparent from the attached claims.

C L A I M S

1. Method for extracting data from a data stream flowing over an IP network (1), the data of the stream being organized according to an application layer protocol, the method comprising:

- 5 - a configuration phase in which a state machine (20) is constructed with states and transitions configured according to at least one type of data to be extracted from the stream, the transitions between states being activated by the respective conditions defined as a function of rules for organization of the data of the stream according to said application layer protocol, and in which at least
 - 10 one state (30, 31) is selected for the extraction of data from the stream; and
 - a real time analysis phase of the data stream, comprising:
 - observing the data of the stream originating from IP packets flowing successively over the network;
 - when the state machine is in a current state, determining if an activation condition of a transition from the current state to a target state is fulfilled by the observed data of the stream and when this activation condition is fulfilled, changing the state machine to the target state;
 - extracting the data from the stream when the state machine is in a state selected in the configuration phase for the extraction of data from the stream; and
 - disregarding the data of the stream when the state machine is in a state not selected state in the configuration phase.

2. Method according to claim 1, wherein activation conditions of the transitions of the state machine (20) comprise the presence of respective lexemes in the data of the stream, and wherein the real time analysis phase of the data stream comprises moreover, when the state machine is in a current state:

- storing in buffer memory (26) at least N-1 characters situated at the end of the data of the stream observed in an IP packet, N being the largest number of characters of the lexemes corresponding to the transitions out of the current state; and

- on receiving next data of the stream originating from an IP packet flowing subsequently over the network, placing the content of the buffer memory in front of the data received in order to search for the possible presence of a lexeme split between the two packets.

5 3. Method according to any one of the preceding claims, wherein the selected states of the state machine (20) comprise a state (30) in which the data extracted from the stream are directly transferred to an external processor (14).

4. Method according to any one of the preceding claims, wherein the selected states of the state machine (20) comprise a state (31) in which the data
10 extracted from the stream are stored in buffer memory (27) then transferred to an external processor (14) once the state machine leaves said selected state.

5. Method according to claim 4, wherein the buffer memory (27) receives the data extracted from the stream in said selected state (31) of the state machine (20) has a size limited to a configurable number of characters.

15 6. Device for extracting data from a data stream flowing over an IP network (1), the data of the stream being organized according to an application layer protocol, the device comprising:

- a state machine (20) having states and transitions configured according to at least one type of data to be extracted from the stream, the transitions between
20 states being activated by the respective conditions defined as a function of rules for organization of the data of the stream according to said application layer protocol, at least one state (30, 31) of the state machine being selected for the extraction of data from the stream;
- an input (24) for receiving in real time the data of the stream originating from IP
25 packets flowing successively over the network;
- detector of transitions (21) for determining, when the state machine is in a current state, if an activation condition of a transition from the current state to a target state is fulfilled by the data of the stream received at the input and for changing the state machine to the target state when said activation condition is
30 fulfilled,

the data of the stream being extracted when the state machine is in a selected state for the extraction of data from the stream, and disregarded when the state machine is in an unselected state for the extraction of data from the stream.

7. Device according to claim 1, wherein activation conditions of the transitions of the state machine (20) comprise the presence of respective lexemes in the data of the stream, and wherein the detector of transitions (21) is associated with a buffer memory (26) controlled in order to receive, when the state machine is in a current state, at least N-1 characters situated at the end of the data of the stream observed in an IP packet, N being the largest number of characters of the lexemes corresponding to the transitions out of the current state, the detector of transitions being arranged in order to, on receiving the next data of the stream originating from an IP packet flowing subsequently over the network (1), search for the possible presence of a lexeme split between the two packets by putting the content of the buffer memory in front of the data received.

8. Device according to claim 6 or 7, wherein the selected states of the state machine (20) comprise a state (30) in which the data extracted from the stream are directly transferred to an external processor (14).

9. Device according to any one of claims 6 to 8, wherein the selected states of the state machine (20) comprise a state (31) in which the data extracted from the stream are stored in buffer memory (27) then transferred to an external processor (14) once the state machine leaves said selected state.

10. Device according to claim 9, wherein the buffer memory (27) receiving the data extracted from the stream in said selected state (31) of the state machine (20) has a size limited to a configurable number of characters.