# FAST RTP DETECTION AND CODECS CLASSIFICATION IN INTERNET TRAFFIC

Petr Matoušek Ondřej Ryšavý Martin Kmeť
Faculty of Information Technology
Brno University of Technology, Czech Republic
{matousp,rysavy}@fit.vutbr.cz,ikmet@.fit.vutbr.cz

## ABSTRACT

This paper presents a fast multi-stage method for on-line detection of RTP streams and codec identification of transmitted voice or video traffic. The method includes an RTP detector that filters packets based on specific values from UDP and RTP headers. When an RTP stream is successfully detected, codec identification is applied using codec feature sets. The paper shows advantages and limitations of the method and its comparison with other approaches. The method was implemented as a part of network forensics framework NetFox developed in project SEC6NET. Results show that the method can be successfully used for Lawful Interception as well as for network monitoring.

**Keywords**: Network Forensics, RTP Detection, Codec Identification, VoIP

## 1. INTRODUCTION

Network monitoring and traffic analysis either for the purpose of network management or network forensics faces the challenges of handling big real-time data streams. Filtering input data is necessary to obtain only relevant information for further processing. Many applications for analysis and monitoring of VoIP (Voice over IP) and multimedia communication rely on the efficient identification of RTP streams or sessions[1] in order to isolate VoIP communication from the rest of Internet traffic.

Transmission of multimedia streams over packet networks is very popular today. It includes transmission of voice over IP that almost replaced traditional telephony, transmission of video streams like video on demands (VoD), online streaming (radio/TV streams), video conferencing, etc. These services work on the application level of OSI model using signalling protocols for establishing and maintaining communication, e.g., SIP, H.323, or RSTP, and data transmission protocols for passing voice and video data, e.g., RTP/RTCP (Schulzrinne, Casner, Frederick, & Jacobson, 2003). Since RTP streams are routed independently over the Internet, it is not always easy to detect these streams if signalization is missing at the point of observance.

Detection of RTP streams and classification of RTP payload is an important task for network administrators in order to find out how many VoIP/video sessions are established and what bandwidth is required for these sessions according to QoS requirements. RTP detection and classification is also needed in the area of Network Forensics and Lawful Interception (LI) where Law Enforcement Agencies (LEAs) should detect and analyze a communication that deals with criminal activities. The task is more complex than eavesdropping in classical telephony, where communicating parties use a dedicated line for all communication. In VoIP, each direction of communication is transmitted independently using a sequence of RTP packets that are routed over the dynamic topology of packet-based network.

[1]By a RTP stream/session we understand a one-way sequence of RTP packets transmitting a multimedia content (voice, video) between two communicating parties.

In this work we focus on the detection of RTP protocols without knowing signalization, i.e., without knowing L4 identification of RTP traffic (ports). Our RTP detector identifies and filters RTP packets on the fly using selected values from IP and UDP headers. Since RTP uses dynamic ports on L4, other UDP traffic can be mistaken for RTP transmission. Our results prove that with a sufficient number of RTP packets per stream we are able to detect an RTP stream with high probability in real time. If the minimum number of RTP packets per stream is set to 10, the probability of false positives comes near to zero depending on the type of traffic.

Another part of our work deals with the classification of codecs used to encode multimedia contents encapsulated in a RTP stream. Common audio and video codecs can be identified using a payload type (PT) value from the RTP header. This value corresponds to RTP Audio/Video Profiles as defined in RFC 3551 (Schulzrinne & Casner, 2003). However, there are two limitations. First, RTP Audio/Video Profiles list only well-known codecs with a static PT value. Such codecs are called *static codecs*. Many RTP streams transmit audio and video data encoded using *dynamic codecs* where a value of the codec is not standardized and signalling protocols like SIP/SDP, or H.323 are needed to inform communicating sites about the codec type. The latter limitation reflects the fact that open source codecs used in VoIP softphones and hard phones have different PT numbers depending on implementation even for the same codec, see Table 1.

This table shows PT types of common audio and video codecs in soft phones (Ekiga, X-lite, SJ Phone), hard phones (Well T20, Linksys WRP 400-G2), and video terminal software Polycom PVX. You can see that for static codecs like G.711 PCM $A$-law, $\mu$-law, GSM, G.722, H.261, H.263, or H.264 payload type can be determined using a PT value in RTP header. However, for dynamic codecs like iLBC, Speex, or G.726 PT value can differ. Even the same software (Ekiga) differs in versions under Windows (Ekiga W) or under Unix (Ekiga U).

It means that the PT value in the RTP header cannot be used without validation as a unique codec identifier. That is the reason why our codec classifier implements advanced codec classification using a *specific feature set*. This method employs specific features of RTP packets related to codecs, e.g., length of the packet, time delay between two adjacent RTP packets, and typical patterns of the payload in the codec classification.

## 1.1 Contribution

The main contribution of our work is a fast method for RTP detection and codec classification that combines several approaches in order to detect a RTP stream and its codec with high probability on the fly. This is especially valuable for high-speed networks where fast and less resource-demanding processing is strongly required. Another contribution related to our work is a creation of an annotated reference dataset that contains real RTP streams encapsulating multimedia payload encoded using different codecs. The dataset was generated in order to test our application for accuracy and for the comparison with other tools. The dataset is freely available to researchers who work with RTP detection and codec classification[2]. It contains 60 annotated PCAPs (Packet Capture File Format) with RTP streams encoded using different codecs and sampling frequencies.

## 1.2 Structure of the Paper

The remainder of the paper is structured as follows. Section 2 gives a short overview of related work in the area of RTP detection and codecs classification. It discusses advantages and limitations of current methods and compares them with our approach. Section 3 describes our method of RTP detection and codec classification. This section shows how payload type (PT) in RTP header should be understood in terms of classification and how PT values of dynamic codecs depend on the end-point application or device. Section 4 shows how our tool RTPinfo was tested on well-known classified data. The result is compared with other tools (Wireshark,

[2]Codecs Database is available for download at `https://nes.fit.vutbr.cz/ansa/pmwiki.php?n=Main.Codecs`.

| **Codec** | **Ekiga (W)** | **Ekiga (U)** | **X-lite** | **SJ Phone** | **Well T20** | **Linksys** | **PVX** |
|---|---|---|---|---|---|---|---|
| G.711 $\mu$-law | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| G.711 $A$-law | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| iLBC | 110 | 116 | 98 | 97,98 | - | - | - |
| Speex-16 | 125 | 113 | 100,106 | - | - | - | - |
| Speex-8 | 124 | 112 | 97,105 | 110 | - | - | - |
| GSM | 3 | 3 | 3 | - | - | - | - |
| MS-GSM | 106 | 123 | - | - | - | - | - |
| G.721 | - | - | - | - | - | - | 101,102,103 |
| G.722 | 9 | 9 | - | - | 9 | - | 9 |
| G.726-16 | 105 | 122 | - | - | - | - | - |
| G.726-24 | 104 | 121 | - | - | - | - | - |
| G.726-32 | 103 | 120 | - | - | - | 98 | - |
| G.726-40 | 102 | 119 | - | - | - | - | - |
| G.728 | - | - | - | - | - | - | 15 |
| G.729 | - | - | - | - | 18 | 99 | 18 |
| H.261 | 31 | 31 | - | - | - | - | 31 |
| H.263 | 34 | - | 34 | - | - | - | 34 |
| H.263-98 | 108 | - | 115 | - | - | - | 96 |
| H.264 | 109 | - | - | - | - | - | 109 |

Table 1: Values of RTP payload types (PT) in different applications.

PacketScan, Cisco nBAR). The last part of the paper summarizes our results and discusses directions for the future work.

## 2. RELATED WORK

The classification of network applications, including RTP detection, was explored by research teams (Costeux, Guyard, & Bustos, n.d.) or (Zhang et al., 2008), as well as by commercial companies (Cisco Systems, 2002). These approaches mostly use pattern-based methods, i.e., packet classification using a check of selected field in IP, UDP, and RTP headers, or well-defined patterns in RTP payload. Cisco Network Based Application Recognition (nBAR) (Cisco Systems, 2002) looks deeper into RTP header and successfully classifies RTP packets based on multiple attributes in the RTP header rather than UDP port numbers. However, nBAR is not able to identify codecs in RTP payload, as our tests showed. Costeux et al. (n.d.) use *per-packet checking* for RTP identification. Their algorithm is based on validation check described in (Schulzrinne et al., 2003). Unfortunately their paper does not reveal how their approach is accurate. According to our tests, *per-packet RTP checking* can be inaccurate for short RTP streams where possibility of false positives rises. Thus, in our approach we implement a two-stage detection combined with a codec classification. The first stage of the detection works on *per-packet* basis using a similar algorithm as described by Costeux et al. (n.d.), however we use different filters to check if a packet is RTP. The second stage of our detection validates previous results using *per-flow checking* that minimizes false positives and false negatives.

Another part of our work is codec identification in detected RTP streams. There are many algorithms for the audio and video codec identification proposed in research papers. Most of these algorithms is based on the machine learning and observation of statistical properties of codecs. These approaches usually require a set of features of the incoming stream that is compared with a set of training profiles using neural networks (Yargicocglu & Ilk, 2012), chaotic sets (Hicsonmez, H.T.Sencar, & Avcibas, 2011), etc. These methods give very precise results. Their main disadvantages are (i) the need of well-established training sets of different codecs, and (ii) complex implementation of a chosen method. Thus, these methods are not suitable for the fast on-line processing of a high-volume Internet traffic that is usually required by monitoring devices and LI probes in network forensics.

Another approach described by Bestagini, Allam, Milani, Tagliasacchi, and Tubaro (2012) analyses video codecs using coding-based footprints. The idea of codec footprints was applied in our approach where we formed a set of specific features for each codec to be identified. These features are derived directly from RTP headers what makes the classification really fast.

## 3. RTP DETECTION AND CODEC CLASSIFICATION

RTP sessions use dynamically negotiated ports, so it is not easy to identify that a given UDP packet transmits RTP. At the first glance, identification is nontrivial since RTP packets do not contain an explicit protocol identifier. However, by observing specific header fields over several packets, we can identify RTP streams with high probability. In our approach, we combine two types of check of RTP traffic (Perkins, 2003):

***Per-packet checking*** is based on fixed known values of the header field. RTP header fields recommended for validity checks are described in (Schulzrinne et al., 2003, Appendix A) and include (i) RTP version (must be 2), (ii) payload type (must be known and not equal to SR or RR, i.e., not reserved values 72–26 (Schulzrinne & Casner, 2003)), (iii) padding bit P (if is set, then the last octet must contain a valid octet count), (iv) extension bit X (must be set to zero if the profile does not specify it), and (v) the length of the packet (must be consistent with CC and PT).

***Per-flow checking*** processes a sequence of possible RTP packets with a unique SSRC[3] identifier and checks (i) if the sequence numbers are properly incremented, and (ii) if timestamp intervals correspond with the payload type and sampling frequency.

[3]SSRC stands for Synchronization Source. It is a 32-bit integer identifying participants in a RTP session.

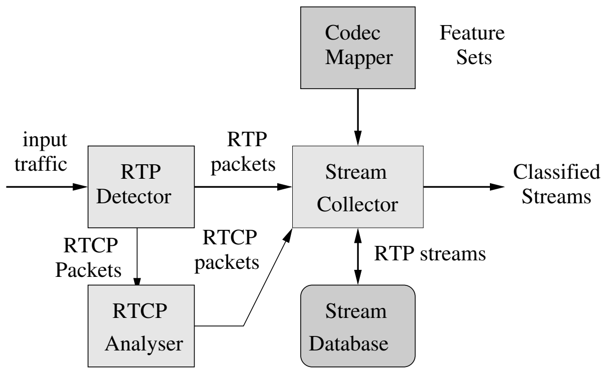


Figure 1: Scheme of RTP detection and codec identification

Our method implements combination of *per-packet* and *per-flow checking.* At first, *per-packet checking* processes incoming UDP packets and possible RTP packets are selected. Then, the selected packets are grouped into RTP streams by *per-flow checking.* The *per-flow checking* process also provides a codec classification using *a specific feature set* described later. In the following text we describe the algorithm of detection and classification. Moreover, the results of classification on a testing dataset are presented.

### 3.1 Multi-stage Filtering in RTP Detector

Here we describe how the packet processing in our tool *RTPinfo* works. At first, incoming packets are processed one by one by the *RTP detector* that implements enhanced *per-packet checking.* If a packet is classified as an RTP packet, it is grouped with other RTP packets into RTP streams by *a stream collector* that performs the *per-flow checking* and codec identification, see Figure 1. If a possible RTCP packet is detected, is analyzed using RTCP Analyser and then it is assigned to a corresponding stream.

The first-stage of the *per-packet checking* in the *RTP detector* filters incoming packets by rules based on RTP validity checks (Schulzrinne et al., 2003, App. A) and our observations:

1. Only IPv4 or IPv6 datagrams with UDP payload are permitted.

2. Source and destination ports of UDP must be higher than 1023.

3. The length of an application header must be at least minimal RTP header length according to CSRC Count (CC), i.e., higher than $12 + 4 \times CC$ bytes.

4. RTP version must be 2.

5. RTP payload type must be within the range defined by RFC 3550. Packets with PT type containing *unassigned* or *reserved* values are filtered out.

6. If padding bit P is set, the last byte of the padding is checked with the total length of the packet. However, some devices (like Tanberg Video Conferencing System) set P bit but did not properly set the last byte of the padding. Thus, the padding bit filtering can be switched off in our tool.

If a packet successfully passes all the above written filtering rules, it is marked as an RTP packet. Then, the second stage of detection is applied: RTP packets with the same source and destination IP addresses, source and destination ports and the same SSRC identifier are grouped together into a possible RTP stream using the *per-flow checking*. If the number of packets in an RTP stream is higher than a required minimum, the RTP stream is successfully detected. Otherwise, the packets are considered false negatives and labeled as non-RTP data.

There are also additional possibilities of the *per-flow checking* like checking sequence numbers of packets, checking incrementation of timestamps in adjacent packets, etc. However, these checks don't work properly with some video streams. If an inter-leaved video is transmitted using RTP, RTP packets with the same timestamp would have different sequence numbers. Also, if two audio or video sources are mixed together, it can happen that a recently received packet could have an older timestamp than a previously received packet. Even though this behavior violates the basic rules of audio transmission, it is valid for video traffic. Thus, we decided to check only the number of packets in RTP streams without losing the accuracy of detection.

Table 2 shows, how the correctly chosen minimum number of RTP packets per stream can decrease number of false positives and false negatives. In the table, you can see four tests with minimum packets per stream set to 100, 10, or 1. If minimum is set to 1, only *per-packet checking* (i.e., the first stage of detection) is applied. You can see in Test no. 4, that there is an enormous number of false positives in this case. For some inputs, *per-packet checking* can generate inaccurate results. This is the reason, why the multi-stage detection was implemented. A naive approach would expect that the higher value of minimum number of packets per stream (e.g., 100) is more efficient since shorter streams cannot transmit any useful video or audio records. Nevertheless, our results in Table 2 reveal that there can be real RTP streams with just few packets and value 100 can be too high for them. We can see that in Test no. 1 and 2, the considerable number of false negatives was detected because streams shorter than 100 RTP packets were not labeled as an RTP stream. For some kind of traffic, even value 10 is too high to be the minimum of packets per stream. Our tests showed that the optimal value for the minimum packets per stream varies between 2 to 5 depending on the input traffic.

### 3.2 Codecs Classification Using a Specific Feature Set

Similar to the previously mentioned approaches (Yargicocglu & Ilk, 2012; Hicsonmez et al., 2011; Jenner & Kwasinski, 2012), we are able to detect a set of known codecs, that are specified in *Codec Mapper Table (CMT)*. If an RTP stream contains a payload with a codec whose features are not specified in CMT, the codec is not recognized and it is classified as *unknown*. Currently, we are able to classify about 20 common audio codecs and their variants. Classification of video codecs will be added in a new version of the tool.

CMT table uses four distinguished features (payload type, $\Delta$ time, payload size, and $\Delta$ ratio). By these features we are able to classify a codec of the RTP payload using RTP header values only. The set of specific features for the most common audio codecs is shown in Table 3. In the following text we describe how these features are determined and how the classification proceeds.

| | **Test no. 1** | | | **Test no. 2** | | |
|---|---|---|---|---|---|---|
| **Min Pkts/stream** | **100** | **10** | **1** | **100** | **10** | **1** |
| No. of real RTP streams | 7 | 7 | 7 | 30 | 30 | 30 |
| Detected RTP streams | 5 | 7 | 7 | 20 | 26 | 30 |
| No. of real RTP pkts | 36 603 | 36 603 | 36 603 | 99 646 | 99 646 | 99 646 |
| Detected pkts | 36 490 | 36 603 | 36 603 | 99 478 | 99 638 | 99 646 |
| Total Pkts in pcap | 38 193 | 38 193 | 38 193 | 101 169 | 101 169 | 101 169 |
| False Positives | 0 | 0 | 0 | 0 | 0 | 0 |
| False Negatives | 113 | 0 | 0 | 168 | 8 | 0 |
| Exec Time (s) | 46 | 46 | 46 | 121 | 121 | 121 |
| Test pcap size (MB) | 21 | 21 | 21 | 59.9 | 59.9 | 59.9 |

| | **Test no. 3** | | | **Test no. 4** | | |
|---|---|---|---|---|---|---|
| **Min Pkts/stream** | **100** | **10** | **1** | **100** | **10** | **1** |
| No. of real RTP streams | 6 | 6 | 6 | 4 | 4 | 4 |
| Detected RTP streams | 6 | 6 | 8 | 2 | 2 | 11 060 |
| No. of real RTP Pkts | 184 107 | 184 107 | 184 107 | 19 311 | 19 311 | 19 311 |
| Detected pkts | 184 107 | 184 107 | 184 111 | 19 300 | 19 300 | 30 367 |
| Total Pkts in pcap | 197 163 | 197 163 | 197 163 | 126 257 | 126 257 | 126 257 |
| False Positives | 0 | 0 | 4 | 0 | 0 | 11 056 |
| False Negatives | 0 | 0 | 0 | 11 | 11 | 0 |
| Exec Time (s) | 227 | 227 | 227 | 217 | 218 | 218 |
| Test pcap Size (MB) | 180.8 | 180.8 | 180.8 | 107.3 | 107.3 | 107.3 |

Table 2: Impact of minimal packets per stream on false positives/negatives.

#### 3.2.1 Codec Features.

The first feature is a *payload type*. This feature is extracted directly from RTP header field PT. Our tests proved that static values of the payload type as defined in standard RFC 3551 (Schulzrinne & Casner, 2003) are constant for end-point devices and can be used as a unique feature to identify a corresponding audio or video codec. For some codecs (e.g, G.723) the value of PT points only to the codec category and does not say anything about sub-categories what is often important for the proper decoding. In VoIP, the missing information (e.g., sampling period) is included in signalling protocols like SIP/SDP. For the classification without signalization, we need to use additional features to classify the codec sub-category precisely, e.g., to determine if it is G.723.1-5k codec or G.723.1-6k codec. If a PT value falls into the dynamic codecs range, this first feature cannot be used as a unique codec identifier (see Table 1). In this case, the value is internally set to $-1$, it means *a feature not used*, and other features are tested.

The second feature of the codec classification is $\Delta$ *time*. $\Delta$ *time* is given as the difference between timestamps of adjacent packets $i$ and $j$, i.e., $\Delta\ time = t_j - t_i$, where $j = i + 1$, $t_x$ is a timestamp of packet $x$. $\Delta$ *time* is computed for each pair of adjacent RTP packets of the stream. It should be the same for all RTP packet of the given RTP stream. If the value of $\Delta$ *time* differs between any two adjacent packets of the RTP stream, it means that the packetization time was changed during RTP transmission. This may happen for video transmissions when adjacent samples are very similar (static scenes) and the sender decides to change the sampling period in order to safe bandwidth. Looking at audio codecs, this was also observed for Silk codecs only. In case of changing $\Delta$ *time* value, the feature is set to zero and the feature is invalidated for the further classification.

*Payload size*, or the voice payload size, contains the length of RTP payload. This value is mostly fixed for many codecs. However, the payload size can be manually configured at end-point devices for some codecs. This is typical for G.726 (all variants) and G.729b (if VAD is switched on, see * in CMT table). In these cases, also $\Delta$ *time*

| **Codec** | **Payload Type** | **Δ time** | **Payload Size** | **Δ ratio** |
|---|---|---|---|---|
| G.711 $\mu$-law | 0 | 160 | 160 | 1:1 |
| G.711 A-law | 8 | 160 | 160 | 1:1 |
| Speex8 | dyn | 160 | 20 | 8:1 |
| Speex16 | dyn | 320 | 52 | 80:13 |
| GSM | 3 | 160 | 33 | 160:33 |
| G.722 | 9 | 160 | 160 | 1:1 |
| G.722.1 | dyn | 320 | 60 | 16:3 |
| G.723.1-5k | 4 | 240 | 20 | 12:1 |
| G.723.1-6k | 4 | 240 | 24 | 10:1 |
| G.726-16 | dyn | 80/240 | 20/60 | 4:1 |
| G.726-24 | dyn | 80/240 | 30/90 | 8:3 |
| G.726-32 | dyn | 80/240 | 40/120 | 2:1 |
| G.726-40 | dyn | 80/240 | 50/150 | 8:5 |
| G.729 | 18 | 160 | 20 | 8:1 |
| G.729a | 18 | 160 | 20 | 8:1 |
| G.729b | 18 | 160* | 20* | var. |
| AMR-WB | dyn | 320 | 62 | 160:31 |
| AMR-12k | dyn | 160 | 33 | 160:33 |
| Silk8 | dyn | 320 | var. | var. |
| Silk16 | dyn | 640 | var. | var. |

Table 3: Codec Mapper Table (CMT) with specific codec features.

changes but the ratio between Δ *time* and *the payload size* remains unchanged. Thus, Δ *ratio* is further used as the fourth classification feature. Some codecs (for example Silk8, Silk16) use a variable payload size, so this feature cannot be applied in their case.

Δ *ratio* is an important feature especially for codecs with a variable packetization period (e.g., G.726 codecs). Δ *ratio* is fixed because timestamp values in RTP packets are related to the packetization time and the payload size. So, when the packetization period changes, timestamps are changed too but Δ *ratio* remains constant as seen in Table 3 for G.726 codecs. This ratio is also used to distinguish sub-categories of G.723 as seen in that table.

Using a specific feature set, we are able to uniquely detect at least 20 different audio codecs with high probability. Additional features were also considered, like payload patterns. Payload patterns can be successfully used for the identification of most video codecs and some audio codecs like GSM, G.723, or Speex. However, the current set of four features seems to be sufficient for the successful identification of most common audio codecs as our experiences show. The main advantages of this method are its simplicity, quality of accuracy, and high performance for large data volumes.

### 3.2.2 Codec Classification.

The process of classification is implemented in *Stream Collector* during *per-stream checking.* The classification works as multi-stage filter using the set of codecs from CMT. Input RTP streams are compared with each CMT entry. When a match on the specific feature is found, the next feature is examined. Using four features, four steps of comparison are provided for each CMT entry during the processing. Then, the best codec is selected according to the number of features matched.

This classification algorithm works with constant time complexity. Its input values are taken only from RTP headers. The accuracy of the algorithm depends on the feature set specified in CMT. Adding a new codec to the CMT table is a simple operation that includes (i) determination of a specific feature set of the new codec, and (ii) insertion of a new entry into CMT.

Currently, our CMT contains 20 entries of the most common audio codecs. Now, we are working on adding video codecs to the database. For some new codecs, it can be useful to add a new feature into CMT, e.g., payload pattern and offset of the pattern in the payload. Adding a new feature is straightforward operation in our classifier because each filtering stage is independent.

## 4. RESULTS

RTP detection and codec classification was tested on a RTP dataset that was created for the project. The comparison with three other tools for RTP detection and codec identification was done: we used open source packet analyzer Wireshark, professional analyzer PacketScan from GL Communications, Inc., and Cisco nBAR feature on Cisco 2911 routers.

### 4.1 RTP Dataset

For testing purposes, an annotated dataset containing RTP streams with known codecs was created. To our best knowledge, we are not aware of any available dataset with real VoIP communication encoded by different codecs.

The RTP dataset was generated using the following tools: soft phone Ekiga (Ekiga Set), Cisco ISR router with Call Manager Express (CME Set), and IXIA XM2 tester (IXIA Set). Each generated set contains several RTP streams with signalling protocols (mostly SIP signalization). For the codec classification, we modified these RTP datasets so that all signalling packets were removed from PCAP files. Thus, the classification is based on processing of RTP packets only.

Our RTP dataset contains several PCAP files with RTP streams transmitting a voice encoded by following codecs, see Table 4.

### 4.2 Testing and results

The testing was performed on all PCAP files from our RTP Datasets. In Wireshark configuration, decoding of RTP packets outside conversation was allowed in Edit/Preferences/Protocols. PacketScan had a default configuration without additional changes. The results of classification tests are in Tables 5, 6, and 7 for each tested application, i.e., Wireshark (W), PacketScan (P), and our application *RTPinfo* (R). For each application, two values are shown: a number of detected RTP packets (RTP) and a name of the identified codec (Codec).

In Table 5 we can see, that not all applications were able to detect RTP packets. Wireshark and *RTPinfo* detected the same number of RTP packets while PacketScan missed some RTP packets. The number of false negatives of PacketScan was 10. Concerning codec identification, our tool *RTPinfo* (R) was the most accurate. Each tool had problems with identification of Silk codec that uses a variable packetization time and payload size, see Table 3. RTP packets with Silk payload generated by Ekiga had value PT=92. According to standard RFC 3551 (Schulzrinne & Casner, 2003), this value is restricted and should not be used. Wireshark was able to recognize all static codecs, however it did not classify dynamic codecs correctly. A similar result was observed by PacketScan where static codecs classification was successful but the tool did not match any dynamic codec properly.

Similar results were detected for CME set, see Table 6. Wireshark and *RTPinfo* were able to detect all RTP packets properly, Packet Scan missed some packets. All applications were able to determine all codecs successfully.

The reason is that all codecs of this set are static with well-defined fixed PT value. We can see that *RTPinfo* was able to detect even subcategory of G.729 that is important for proper decoding.

Last tests were done using IXIA set, see Table 7. Similarly to the previous tests, the number of detected RTP packets by Wireshark and by *RTPinfo* is correct. Again, PacketScan missed few RTP packets. Wireshark was able to classify all static codecs but it was not able to distinguish sub-categories of G.729 and G.723. As in previous tests, it was not able to identify dynamic codecs. PacketScan was successful in the static codec classification but it did not match any dynamic codec properly.

We also made tests with Cisco nBAR on Cisco ISR G2 router. Data from our datasets were sent through a network where ISR router with nBAR was present. We discovered that nBAR was able to detect all RTP packets properly, so the result would be the same as Wireshark or RTPinfo in

| **Codec** | **Ekiga Set** | **CME Set** | **IXIA Set** |
|---|---|---|---|
| G.711 A-law | pcma.pcap | g711alaw.pcap | alaw.pcap |
| G.711 $\mu$-law | pcmu.pcap | g711ulaw.pcap | ulaw.pcap |
| Speex8 | speex8.pcap | | |
| Speex16 | speex16.pcap | | |
| GSM | gsm.pcap | | |
| G.722 | g722.pcap | | |
| G.722.1 | g722-1.pcap | | |
| G.723.1 5,3 kbps | | | g7231-5k.pcap |
| G.723.1 6,3 kbps | | | g7231-6k.pcap |
| G.726-16 | g726-16.pcap | | g726-16-a.pcap |
| G.726-24 | g726-24.pcap | | g726-24-a.pcap |
| G.726-32 | g726-32.pcap | | g726-32-a.pcap |
| G.726-40 | g726-40.pcap | | g726-40-a.pcap |
| G.729 | | g729r8.pcap | g729.pcap |
| G.729a | | | g729a.pcap |
| G.729b | | g729br8.pcap | g729b.pcap |
| AMR-WB | amr-wb.pcap | | |
| AMR 12,2 kbps | | | amr-12.pcap |
| Silk8 | silk8.pcap | | |
| Silk16 | silk16.pcap | | |

Table 4: Overview of our RTP Datasets

| **File** | **RTP (W)** | **Codec (W)** | **RTP (P)** | **Codec (P)** | **RTP (R)** | **Codec (R)** |
|---|---|---|---|---|---|---|
| pcma.pcap | 822 | G.711 *A*-law | 812 | G.711 *A*-law | 822 | G.711 *A*-law |
| pcmu.pcap | 791 | G.711 $\mu$-law | 781 | G.711 $\mu$-law | 791 | G.711 $\mu$-law |
| speex8.pcap | 804 | Unknown | 794 | Speex 16kHz | 804 | Speex 8kHz |
| speex16.pcap | 1015 | Unknown | 1005 | iSAC | 1015 | Speex 16kHz |
| gsm.pcap | 796 | GSM 06.10 | 786 | GSM 06.10 | 796 | GSM 06.10 |
| g722.pcap | 736 | G.722 | 726 | G.722 | 736 | G.722 |
| g7221.pcap | 846 | Unknown | 838 | AMR-WB | 846 | G.722.1 |
| g726-16.pcap | 609 | Unknown | 599 | G.726 40kbps | 609 | G.726 16kbps |
| g726-24.pcap | 583 | Unknown | 573 | EVRCC | 583 | G.726 24kbps |
| g726-32.pcap | 588 | Unknown | 578 | G.711 $\mu$-law | 588 | G.726 32kbps |
| g726-40.pcap | 623 | Unknown | 615 | G.711 *A*-law | 623 | G.726 40kbps |
| amr-wb.pcap | 717 | Unknown | 707 | G.726 32kbps | 717 | AMR-WB |
| silk8.pcap | 663 | Unknown | 653 | Unknown | 663 | Unknown |
| silk16.pcap | 631 | Unknown | 621 | Unknown | 631 | Unknown |

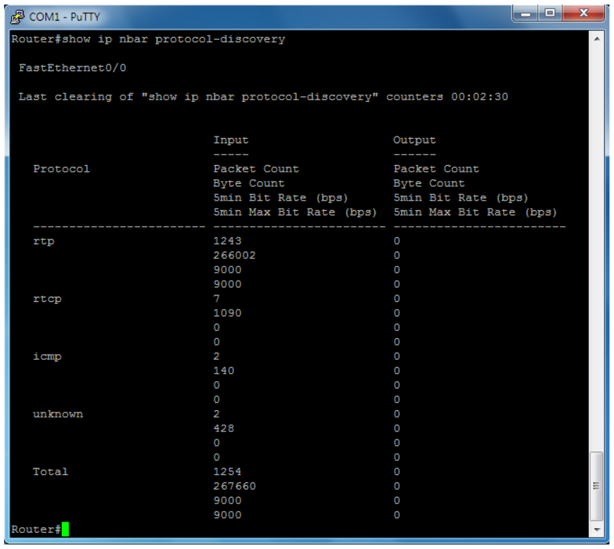
Table 5: Testing on Ekiga Set (W—Wireshark, P—PacketScan, R—RTPinfo)

| **File** | **RTP (W)** | **Codec (W)** | **RTP (P)** | **Codec (P)** | **RTP (R)** | **Codec (R)** |
|---|---|---|---|---|---|---|
| g711alaw.pcap | 1245 | G.711 *A*-law | 1235 | G.711 *A*-law | 1245 | G.711 *A*-law |
| g711ulaw.pcap | 1379 | G.711 $\mu$-law | 1369 | G.711 $\mu$-law | 1379 | G.711 $\mu$-law |
| g729r8.pcap | 2797 | G.729 | 2787 | G.729 | 2797 | G.729a |
| g729br8.pcap | 1291 | G.729 | 1281 | G.729 | 1291 | G.729b |

Table 6: Testing on CME Set (W—Wireshark, P—PacketScan, R—RTPinfo)

| File | RTP (W) | Codec (W) | RTP (P) | Codec (P) | RTP (R) | Codec (R) |
|---|---|---|---|---|---|---|
| alaw.pcap | 48016 | G.711 $A$-law | 48006 | G.711 $A$-law | 48016 | G.711 $A$-law |
| ulaw.pcap | 48016 | G.711 $\mu$-law | 48006 | G.711 $\mu$-law | 48016 | G.711 $\mu$-law |
| g7231-5k.pcap | 32012 | G.723.1 | 32002 | G.723.1 | 32012 | G.723.1 5,3kbps |
| g7231-6k.pcap | 32012 | G.723.1 | 32002 | G.723.1 | 32012 | G.723.1 6,3kbps |
| g726-16.pcap | 96032 | Unknown | 96022 | Unknown | 96032 | G.726 16kbps |
| g726-24.pcap | 96032 | Unknown | 96022 | Unknown | 96032 | G.726 24kbps |
| g726-32.pcap | 96032 | Unknown | 96022 | Unknown | 96032 | G.726 32kbps |
| g726-40.pcap | 96032 | Unknown | 96022 | GSM-EFR | 96032 | G.726 40kbps |
| g729.pcap | 48016 | G.729 | 48006 | G.729 | 48016 | G.729/G.729a |
| g729a.pcap | 48016 | G.729 | 48006 | G.729 | 48016 | G.729/G.729a |
| g729b.pcap | 40882 | G.729 | 40872 | G.729 | 40882 | G.729b |
| amr-12.pcap | 48016 | Unknown | 48006 | G.726 24kbps | 48016 | AMR 12,2kbps |

Table 7: Testing on IXIA Set (W—Wireshark, P—PacketScan, R—RTPinfo)



```
Router#show ip nbar protocol-discovery

 FastEthernet0/0

 Last clearing of "show ip nbar protocol-discovery" counters 00:02:30


                            Input                    Output
                            -----                    ------
   Protocol                 Packet Count             Packet Count
                            Byte Count               Byte Count
                            5min Bit Rate (bps)      5min Bit Rate (bps)
                            5min Max Bit Rate (bps)  5min Max Bit Rate (bps)
   ------------------------ ------------------------ ------------------------
   rtp                      1243                     0
                            266002                   0
                            9000                     0
                            9000                     0
   rtcp                     7                        0
                            1090                     0
                            0                        0
                            0                        0
   icmp                     2                        0
                            140                      0
                            0                        0
                            0                        0
   unknown                  2                        0
                            428                      0
                            0                        0
                            0                        0
   Total                    1254                     0
                            267660                   0
                            9000                     0
                            9000                     0
Router#
```

Figure 2: nBAR detection of RTP

tables 5, 6, 7. However, nBAR is not able to identify RTP payload, so we skipped other tests. An example of nBAR detection is in Figure 2.

This figure shows that 1243 packets were properly identified as RTP. Since we used g711alaw.pcap dataset with 1245 RTP packet and the total number of packets was 1267, some packets were missing. We can see that nBAR totally processed only 1254 packets. After repeating the test with lower injection rate we found out that nBAR identification works properly, however there is higher packet loss related to higher rate of incoming packets.

Our tests show that the proposed method of multi-stage detection of RTP packets and codec classification using the specific feature set is viable and gives very good results in comparison with Wireshark or PacketScan. Unlike other approaches, our method is fast and extremely simple in terms of computation. The tests show that its accuracy is also high.

## 5. CONCLUSIONS

In this paper we presented a simple but efficient method for finding RTP streams in network traffic captures and identification of encapsulated codecs. Finding RTP streams without possessing additional information from VoIP signalization is based on integrity checking of RTP packets. It uses a threshold value to refine the number of individual packets in a valid RTP stream. Our experiments show that this simple approach is sufficient enough to detect the most of RTP streams. We also presented a method for identification of codecs in successfully detected RTP streams. The codec classification employs the prior identification of the specific feature set characterizing different RTP codecs properties. Feature values for different codecs are organized in the Codec Mapper Table structure, which is optimized for quick identification of candidate codecs of an analyzed RTP stream.

The benefit of the method lies in its simplicity that enables cheap and efficient implementation suitable for fast on-line traffic monitoring. Our method results from an observation that RTP streams have certain characteristics distinguishing RTP traffic from other communication.

The presented method is usable in network monitoring applications and network forensics. With the increase of the VoIP communication a tool that precisely measures characteristics of VoIP traffic would greatly help administrators in the network management. The method can be also useful during network forensic analysis in situations when VoIP signalization is missing or corrupted.

## ACKNOWLEDGEMENTS

Research in this paper was supported by project "Modern Tools for Detection and Mitigation of Cyber Criminality on the New Generation Internet", no. VG20102015022 granted by Ministry of the Interior of the Czech Republic and an internal University project "Research and application of advanced methods in ICT", no. FIT-S-14-2299 granted by Brno University of Technology.

## REFERENCES

Bestagini, P., Allam, A., Milani, S., Tagliasacchi, M., & Tubaro, S. (2012). Video codec identification. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2012* (pp. 2257–2260).

Cisco Systems, I. (2002). Network Based Application Recognition RTP Payload Classification (White Paper). Cisco Systems, Inc.

Costeux, J.-L., Guyard, F., & Bustos, A.-M. (n.d.). Detection and comparison of RTP and skype traffic and performance. In *IEEE Global Telecommunications Conference, GLOBECOM'06* (pp. 1–5).

Hicsonmez, S., H.T.Sencar, & Avcibas, I. (2011, Nov). Audio codec identification through payload sampling. In *IEEE International Workshop on Information Forensics and Security (WIFS), 2011* (pp. 1–6).

Jenner, F., & Kwasinski, A. (2012, March). Highly accurate non-intrusive speech forensics for codec identifications from observed decoded signals. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2012* (pp. 1737–1740).

Perkins, C. (2003). *RTP: Audio and Video for the Internet.* Addison-Wesley.

Schulzrinne, H., & Casner, S. (2003, July). RTP Profile for Audio and Video Conferences with Minimal Control (RFC 3551).

Schulzrinne, H., Casner, S., Frederick, R., & Jacobson, V. (2003, July). RTP: A Transport Protocol for Real-Time Applications (RFC 3550).

Yargicocglu, A. U., & Ilk, H. G. (2012). Speech Coder Identification Using Chaotic Features Based On Steganalyzer Models. *Communications, Zilina, Slovakia*, *12*, 63–69.

Zhang, G., Xie, G., Yang, J., Min, Y., Zhou, Z., & Duan, X. (2008, Jan). Accurate Online Traffic Classification with Multi-Phases Identification Methodology. In *5th IEEE Consumer Communications and Networking Conference* (pp. 141–146).