

# Open Source VoIP Traffic Monitoring

Luca Deri  
*ntop.org*  
Email: [deri@ntop.org](mailto:deri@ntop.org)  
<http://luca.ntop.org/>

## Abstract

These days voice over IP (VoIP) is quite popular as it is a cost effective way to reduce telephony costs using the Internet. Although many projects are focusing on developing tools and solutions for building the voice infrastructure, there is very little available in terms of tools and metrics for measuring the impact of VoIP on a network.

This paper describes the design and implementation of open source tools for detecting and measuring VoIP traffic based on both standard and proprietary protocols.

## Keywords

Voice Over IP, passive packet capture, network traffic measurement, NetFlow.

## 1. Introduction

VoIP is a solid technology available since some years that allows people to communicate via voice using the IP protocol instead of telephone lines. Unfortunately this technology has been relegated in a niche market due to several factors such as proprietary standards, high price tag, limited integration with existing telephony environments. However in the last couple of years the situation changed dramatically since some open source tools such as asterisk [asterisk] as well as low-cost VoIP telephone adapters and services become available. In fact, today it is quite common for internet providers to provide their customers VoIP calls at very low cost, if any, in addition to standard xDSL connectivity.

Standard VoIP protocol such as SIP [sip] and H.323 [h323] are very popular in the carrier environment and in many other fields not limited to VoIP, such as messenger and chat. In addition to these standards-based applications, there are other applications such as Skype [skype] or voipstunt [voipstunt] that instead are based on proprietary communication protocols and codecs, and other hybrid applications partially based on open standards such as google talk [googletalk] and gizmo [gizmo]. The result is that VoIP is becoming in some ways similar to P2P (peer to peer), as:

- new applications appear, grow and disappear very often.
- some VoIP applications (e.g. Skype) are using P2P as communication transport for building the communication infrastructure and crossing firewalls, a typical scenario where many standard-based VoIP application fail to operate.

In a nutshell, VoIP solutions are often used at corporate level as a cost effective solution to telephone communications, whereas proprietary VoIP applications are used for letting people talk either computer-to-computer or computer-to-telephone using a PC equipped with a special application and a headset.

## 2. Motivation

In this complex and evolving scenario, VoIP traffic monitoring tools are very few, often integrated into packet sniffers such as ethereal [ethereal] [hollis] and used for finding issues (e.g. severe packet loss or incompatible codecs) in specific situations, rather than for permanently monitoring VoIP and non-VoIP traffic. Other tools such as Vomit [vomit] or RTP-tools [rtp-tools] are suitable for capturing voice communications but not for providing a comprehensive permanent monitoring tool. This has been the author motivation for this work, namely to develop an open source VoIP-aware traffic monitoring tool able to:

- Provide long-term monitoring, contrary to what available VoIP monitoring tools do.
- Handling standard VoIP protocols as well, as much as possible, proprietary protocols.
- Decode calls, hence identify peers (who's calling who) and client applications. This is useful for VoIP accounting, billing or fraud detection.
- Provide VoIP metrics such as packet loss and latency, as well as voice quality.
- Generate traffic trends in order to identify how VoIP traffic is changing over the time.

In order to achieve the above goal, the author decided to use a dual approach:

- Enrich ntop [ntop], a home-grown open-source passive traffic monitoring application, for making it VoIP traffic aware.
- Develop some metrics suitable for monitoring key VoIP traffic characteristics and export them via Netflow [netflow] v9/IPFIX [ipfix], by means of nProbe [nprobe] an open-source netflow probe also developed by the author.

This decision has been made because:

- It allows users to exploit the available traffic analysis facilities provided by ntop, without having to run any specialized VoIP traffic analysis application. In this way VoIP traffic is not treated as first-class citizen but it is at the same level as other traffic (e.g. http or email).
- It enables VoIP measurements computed by nProbe to be exported using the standard Netflow/IPFIX protocol, so that they can also be used by ntop and other commercial netflow applications such as Cisco NetFlow Collector. This is particularly important when open source solutions are deployed in an enterprise that is using an existing/commercial management console

The following sections describes the design and the implementation of the extensions to ntop and nProbe for monitoring VoIP traffic.

## 3. VoIP Basics

As stated before there are three main VoIP protocol families, namely those based on:

- standards protocols such as SIP/H.323/RTP [rtp];
- proprietary but well documented protocols such as Cisco skinny [skinny];
- proprietary protocols such as Skype.

Note that the use of standard or known protocols does not always means that it is possible to monitor everything as protocols such as RTP, used to carry voice and video, may transport data encoded with proprietary codecs. This is true for instance for Google Talk whose voice is encoded with a proprietary codec. In general all the protocols are

based on a connectionless protocol such as UDP.

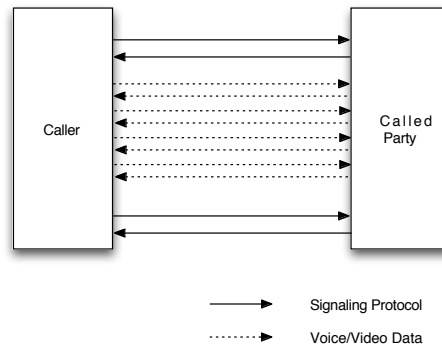


Figure 1. - VoIP Protocol Architecture

The figure above shows the basic of VoIP. Every communication is made of three basic steps:

1. When a caller wants to communicate with another party it initiates a communication either with the remote party or with a gateway/PBX (this depends on the protocol being used and on the local network setup) using a signaling protocol. This step is responsible for:
  - Verifying party credentials, credit (if applicable), ability to call the specified number;
  - Negotiating the call (e.g. is it voice only or voice and video).
  - Agreeing on a common codec (e.g. H.264).
  - Negotiating the ports used for exchanging voice/video data.
2. The call takes place on the ports previously selected, and the payload is encoded using the specified codec. If a standard protocol is used, usually RTP is the one selected. In case of a video-call there are two independent RTP streams, one for voice and one per video.
3. When one of the parties decides to complete the call, using the signaling protocol the call is terminated.

The following table lists some popular signaling and transport protocols used for VoIP.

Signaling	Transport
<ul style="list-style-type: none"> <li>• SIP (Session Initiation Protocol)</li> <li>• Cisco Skinny</li> <li>• H.323</li> </ul>	<ul style="list-style-type: none"> <li>• RTP (Realtime Transport Protocol)</li> <li>• RTCP XS (RTP Control Extended Reports)</li> </ul>

Figure 2. - Popular Signaling and Transport Protocols

From the traffic monitoring point of view:

- The signaling protocol contains important information such as parties identity, type of call (voice or video-call), codecs, duration, and information about the RTP session(s) that usually do not take place on fixed ports. In general without properly decoding the signaling protocol, it is not possible to guess the ports used for RTP.

- The voice/video transport protocol is used to extract information such as jitter, packet loss, and packet latency that are the building blocks for evaluating the call quality. Note that as the RTP packets contains information about the codec being used, with appropriate software it is also possible to decode the RTP and extract further call information.
- Protocols such as RTCP XS [rfc3611] are used to report information about RTP streams including informations such as packet loss, burst/delay, call and transmission quality metrics. These reports are not always available for VoIP calls as they are often generated by VoIP equipments.

For other closed and proprietary protocols such as Skype, it is not really possible to decode the call. Although some researchers tried to reverse-engineer the protocol [baset], to date it is not possible to grab any call information from the traffic stream. Actually the first problem is the detection of the phone call, as Skype is based on the eDonkey P2P [edonkey] protocol. Therefore at best a monitoring application can only detect the phone call and the parties IP, but not the party identities. For this reason, this work will focus only on (partially) open VoIP protocols as they allow monitoring applications to decode relevant call information.

#### **4. Monitoring VoIP Traffic**

VoIP traffic monitoring is divided in two big families: proprietary and standard VoIP protocol monitoring. This section describes how VoIP traffic has been monitored using two open source applications developed by the author.

##### **4.1. Proprietary VoIP Traffic Monitoring**

The main VoIP protocol that falls into this category is Skype. As the protocol implementation is currently unknown and the packet payload encrypted, the best way to monitor this protocol is to treat it as special eDonkey protocol communication. Typically Skype is detected as follows:

- The underlying protocol must be eDonkey. This can be detected by dissecting the initial session payload as described in [karagiannis], and partially relying on the default port being used. Patterns searching for Skype detection has been implemented using the popular PCRE [pcre] library. This library that allows patterns to be efficiently searched into into a data buffer, has been used to search for Skype pattern into the packet payload. The protocol pattern definition has been borrowed by the popular I7-filter [I7-filter] tool that includes several patters not limited only to P2P/VoIP protocols. Thanks to this solution, it is possible to detect not only Skype in general, but also the conversation type (skype2skype or skype-in/out call).
- As Skype traffic looks similar to the original eDonkey traffic, it is necessary to further characterize the traffic in order to distinguish eDonkey from Skype. As protocol payload is encrypted, the only choice left is the analysis of traffic conversations. In particular the main differences between a P2P and Skype conversation are:
  - During a Skype conversation, traffic is bidirectional, packet frequency is high (in general around 64 packets/sec regardless of peers speaking or not) with limited jitter, packet size is limited (usually below 250 bytes).
  - A eDonkey P2P session instead is mostly unidirectional (from the source of data to the host where data is directed), packet rate is not constant and packet size is much

larger.

In a nutshell the only thing that a monitoring application can do with respect to Skype traffic, is to provide evidence of calls without furnishing any other information such the nickname of the people who held the conversation. For this reason Skype detection has been implemented only inside ntop and not on nProbe as there are almost no metrics to export while analyzing Skype traffic.

#### 4.2. Standard VoIP Traffic Monitoring

As explained before, VoIP traffic analysis is divided in two parts:

- Signaling protocol analysis.
- Voice traffic analysis.

The author decided not to analyze legacy signaling protocols such as H.323 but instead focus on modern protocols or industry standards as SIP and Cisco Skinny. For voice traffic analysis the choice is simple as RTP is basically the only protocol being used; this is because RTP has been designed flexible enough to carry various type of data (e.g. not only voice) coded in various formats.

The implementation of VoIP monitoring is slightly different in ntop and nProbe. From a user survey, ntop users are more interested in having a “simple to use and understand” traffic analysis overview. Instead, nProbe users are usually professional network administrators, who prefer precise traffic metrics that can be meaningless for non-professionals. For this reason, ntop has been designed to provide VoIP traffic evidence with some simple metrics, whereas nProbe sports precise VoIP traffic metrics that can be used by netflow collectors for building accurate analysis applications. However it is worth to note that as ntop can act as a flow collector, ntop can also receive and take advantage of nProbe traffic metrics.

The following table shows the metrics that nProbe is currently able to measure.

SIP Metrics	RTP Metrics
<ul style="list-style-type: none"><li>• SIP_CALL_ID</li><li>• SIP_CALLING_PARTY</li><li>• SIP_CALLED_PARTY</li><li>• SIP_RTP_CODECS</li><li>• SIP_INVITE_TIME</li><li>• SIP_TRYING_TIME</li><li>• SIP_RINGING_TIME</li><li>• SIP_OK_TIME</li><li>• SIP_ACK_TIME</li><li>• SIP_RTP_SRC_PORT</li><li>• SIP_RTP_DST_PORT</li></ul>	<ul style="list-style-type: none"><li>• RTP_FIRST_SSRC</li><li>• RTP_FIRST_TS</li><li>• RTP_LAST_SSRC</li><li>• RTP_LAST_TS</li><li>• RTP_IN_JITTER</li><li>• RTP_OUT_JITTER</li><li>• RTP_IN_PKT_LOST</li><li>• RTP_OUT_PKT_LOST</li><li>• RTP_OUT_PAYLOAD_TYPE</li><li>• RTP_IN_MAX_DELTA</li><li>• RTP_OUT_MAX_DELTA</li></ul>

Table 1. - VoIP Traffic Metrics

Note that these metrics can be exported only using NetFlow v9 or IPFIX - supported by both nProbe and IPFIX - as previous NetFlow version such as v5 have no room for carrying extra information. Instead v9/IPFIX have the ability to define flow templates that dynamically define the flow format and attributes. Those metrics have been implemented in order to satisfy basic traffic measurements such as:

- SIP
  - Unique call identifier used for accounting/billing and tracking problems.
  - Call parties: caller and called party.
  - Codecs being used, useful for identifying voice quality issues due to the use of codecs with poor quality.
  - Time of important call events such as beginning of the call. These times can be used to identify performance issues on the SIP gateway.
  - RTP ports where the call will take place. This information is necessary for associating a signaling flow with the phone call just negotiated.
- RTP
  - Source identifiers and time-stamp for the first and last RTP flow packet.
  - Jitter calculated in both (in to out, and out to in) directions.
  - Number of packets lost as well as maximum packet time delta in both directions.
  - Identifier of RTP payload type as specified in [rfc2862].

With the above metrics it is possible to create a wide range of measurement applications such as simple “who’s talking to who” CDR (Call Data Record) used for accounting and billing, and complex traffic analysis applications able to identify communication problems due to the use of poor codecs or high network jitter.

nProbe implements VoIP support in two plugins, one for SIP and the other for RTP. VoIP measurements are exported inside v9/IPFIX flows using custom flow templates. The following example defines a simple SIP flow template whose identifier is 257.

```
nprobe -n 192.168.0.1:2055 -U 257 -T "%LAST_SWITCHED %FIRST_SWITCHED %IN_BYTES
%IN_PKTS %OUT_BYTES %OUT_PKTS %SIP_CALL_ID%SIP_CALLING_PARTY %SIP_CALLED_PARTY
%SIP_RTP_CODECS %SIP_RTP_SRC_PORT %SIP_RTP_DST_PORT"
```

Figure 3. - Simple NetFlow v9 flow template definition

As stated before ntop is able to collect those flows and understand the VoIP metrics that can be defined into the flows. However as v9/IPFIX is an open architecture where flow format is defined using standard templates, commercial applications such as Cisco NetFlow Collector are also able to collect flows and use the VoIP traffic measurements.

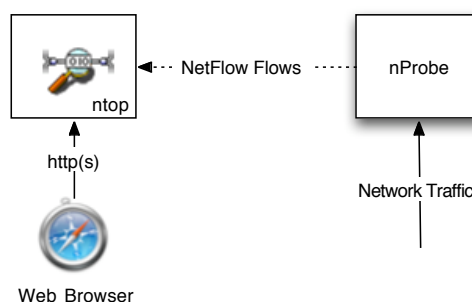


Figure 4. - nProbe flow export towards ntop

The figure above depicts a common setup where ntop collects flows (including VoIP calls) emitted by nProbe in NetFlow v9 format. Nevertheless ntop can also analyze VoIP traffic without having to use flows as feeds. In fact it is also possible for ntop to analyze traffic natively by means of the libpcap [pcap] library. From the user point of view, there is virtually no difference from analyzing VoIP traffic with ntop using netflow or libpcap. This is because one of the main design goals of ntop is to hide differences to the user in terms of traffic capture techniques or network interface types.

83.175.52.136   :49650 <VoIP>	83.175.54.75   :25000	27.0 KB	055487214 called 055470167
83.175.52.136   :49652 <VoIP>	83.175.54.75   :quake	39.5 KB	055487214 called 055470167

Figure 5. - ntop: VoIP Session Detail

For each host that generates VoIP traffic, ntop puts an icon next to it. Clicking on the host, ntop displays further information such as user alias or telephone number as seen in the VoIP traffic.



<b>Host Type</b>	VoIP Host 
<b>Known Users</b> 	055470167 [ VoIP ]

Figure 6. - ntop: Host Detail

In the sessions list, ntop lists the ongoing phone calls complete with call information such as peer telephone number. ntop can also keep track of video calls as shown in figure 5, where two simultaneous sessions (one for voice and one for video) are active from the same peers. Basically ntop handles VoIP calls as sessions even if they are based on UDP and not TCP, and reports call details into each session. As of today, ntop can handle both Skype/Skinny/SIP/RTP, whereas nProbe handles only SIP/RTP.

In case of Skype traffic, ntop puts an icon next to the host but as explained before it is not able to display any additional call information. From users experience, the use of patterns for detecting skype traffic is quite reliable but not the ultimate solution. This is because sometimes the pattern is reporting false positives (e.g. non-Skype traffic is sometimes marked as such) even on HTTP connections, definitively used for tunneling Skype traffic. On the other hand the use of patters do not seem to have false negatives (e.g. inability to detect Skype traffic).

Both ntop and nProbe have been deployed on networks with both proprietary and standard VoIP traffic. The performance of the tools seems to be acceptable and their use helped significantly to unhide details of VoIP communications. The main issue is instead the way these tools are deployed, in case they need to be used for analyzing only VoIP traffic. In fact VoIP traffic is very limited compared to the overall traffic, in terms of both packets and bytes volume. This means that if a Gbit link needs to be analyzed, most of the work of these tools is packet discard of non-VoIP packets; this activity that can take quite some time and waste all the CPU cycles if some packet acceleration facilities [deri] are not used. Furthermore as RTP traffic is flowing on dynamic ports, packet filtering facilities provided by standard equipment such as Juniper routers are not suitable as they are static and not able to be reconfigured on the fly based on the

signaling protocol. The conclusion is that on fast links, it is advisable to either analyze only the signaling protocol without taking into account RTP, or use packet filtering and acceleration if RTP needs also to be used.

## **5. Open Issues and Future Work**

The main open issue is the inability to properly handle Skype, to date probably the most widespread VoIP protocol. As explained before, this is due to the lack of documentation about the protocol, and the use of payload encryption. This is not only a limitation of ntop and nProbe but of any other VoIP analysis tool.

VoIP support is relatively new into ntop/nProbe hence several extensions can be added to their implementation. The current measurements focus mainly on high-level metrics such as jitter or packet loss, and are independent of the codecs being used. However as new codecs such as H.264 [h264] are becoming increasingly popular, a planned enhancement is the ability to decode some of these codec formats in order to also provide precise information about the RTP payload (e.g. voice quality), as well provide support for RTP XS reports. The implementation of these voice analysis metrics has been delayed with respect to the original plan, as they are described in ITU documents (e.g. ITU E.411 recommendation) that are not freely available on the Internet, that is usually a problem for the open source community. Nevertheless in the next release two new common metrics such as MOS score and r-factor will be implemented thanks to bits and pieces found googling on the Internet.

## **6. Final Remarks**

This paper described the challenges of VoIP traffic monitoring and presented two open-source traffic monitoring applications able to also monitor VoIP traffic.

This work is novel in many aspects:

- Beside traffic sniffers, this is the first open-source traffic monitoring application able to continuously monitor VoIP traffic.
- nProbe is probably the most flexible and advanced NetFlow probe available, and definitively the first probe able to monitor VoIP traffic using v9/IPFIX.

Furthermore thanks to packet capture acceleration [ncap], it is also possible to monitor VoIP traffic on gigabit links using nProbe/ntop with almost no packet loss.

## **7. Availability**

This work is distributed under the GPL2 license and is available at the ntop home page (<http://www.ntop.org/>) and other mirrors on the Internet (e.g. <http://sourceforge.net/projects/ntop/>).

## **8. Acknowledgment**

The author would like to thank Urmet TLC (<http://www.urmet.it/>) for partially funding this research work.



## 9. References

- [asterisk] B. Schwarz, *Asterisk open-source PBX system*, Linux Journal, February 2004.
- [baset] S. Baset and H. Schulzrinne, *An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol*, <http://arxiv.org/pdf/cs.NI/0412017>, Columbia University, September 2004.
- [deri] L. Deri, *Improving Passive Packet Capture: Beyond Device Polling*, Proceedings of SANE 2004, 2004.
- [edonkey] K. Tutschku, *A Measurement-based Traffic Profile of the eDonkey Filesharing Service*, PAM 2004
- [ethereal] G. Combs, *Ethereal*, <http://www.ethereal.com/>.
- [gizmo] *The Gizmo Project*, <http://www.gizmoproject.com/>.
- [googletalk] Google Inc., *Google Talk*, <http://talk.google.com/>, 2005.
- [h264] The MPEG Expert Group, *MPEG-4 Part 10*, ITU-T H.264, <http://ftp3.itu.ch/av-arch/jvt-site/>, 2005.
- [h323] H. Liu and others, *Voice over IP Signaling: H. 323 and Beyond*, IEEE Communications Magazine, 2000.
- [hollis] E. Hollis, *Monitoring & Analyzing VoIP Traffic*, Certification Magazine, February 2005.
- [ipfix] B. Claise and others, *IPFIX Protocol Specification*, Internet Draft, 2005.
- [karagiannis] T. Karagiannis and others, *Transport Layer Identification of P2P Traffic*, Internet Measurement Conference, 2004.
- [l7-filter] *Application Layer Packet Classifier for Linux*, <http://l7-filter.sourceforge.net/>.
- [ncap] L. Deri, *nCap: Wire-speed Packet Capture and Transmission*, E2EMON, May 2005.
- [netflow] B. Claise, *NetFlow Services Export Version 9*, RFC 3954, October 2004.
- [nprobe] L. Deri, *nProbe: an Open Source NetFlow Probe for Gigabit Networks*, TNC 2003, May 2003.
- [ntop] L. Deri and others, *Monitoring networks using ntop*, IM 2001, May 2001.
- [pcap] Lawrence Berkeley National Labs, *libpcap*, Network Research Group, <http://www.tcpdump.org/>.
- [pcre] P. Hazel, *PCRE - Perl Compatible Regular Expressions*, <http://www.pcre.org/>.
- [rfc2862] M. Civanlar and G.Cash, *RTP Payload Format for Real-Time Pointers*, RFC 2862, June 2000.
- [rfc3611] T. Friedman, and others, *RTP Control Protocol Extended Reports (RTCP XR)*, RFC 3611, November
- [rtp] H. Schulzrinne and others, *RTP: A Transport Protocol for Real-Time Applications*, RFC 1889, January 1996.
- [rtp-tools] H. Schulzrinne, RTP Tools, <http://www.cs.columbia.edu/IRT/software/rtptools/>.
- [sip] M. Handley, *SIP: Session Initiation Protocol*, RFC 2543, March 1999.
- [skinny] Selsius Inc., *Skinny Client Control Protocol*.
- [skype] D. Bergström, *An analysis of Skype VoIP application for use in a corporate environment*, <http://www.geocities.com/bergstromdennis/>, October 2004.
- [voipstunt] *VoIPstunt*, <http://www.voipstunt.com/>.
- [vomit] N. Provos, *Vomit - voice over misconfigured internet telephones*, <http://vomit.xtdnet.nl/>.