EXata 扩展（三）：VoIP ous 和 ILBC 编码

目标：了解 VoIP 实现的细节，添加 opus、iLBC 语音新编码。

参考：《EXata5.1-MultimediaENterprise Model Library》

几个文件：

app_voip.h/cpp

1. EXata VoIP 实现方式

在《Multimedia库》8.4 节，EXata 中，VoIP 涉及三个方面的模块：

- 流量发生器（VoIP Initiator/Receiver）：sec 8.4
- 呼叫信令（H323 或 SIP）：sec 8.3
- RTP/RTCP 媒体传输：sec 8.2

1.1 呼叫建立过程：SIP

启动 EXata GUI，新建一个最简单的 SIP 场景 ex_sip.config，包含三个节点：节点 1，别名“Alice”，节点 2，别名 Bob；节点 3:Proxy，作为 SIP Proxy，无论 SIP 呼叫模式为 Direct 或者 Proxy-routed，必须有一个 Proxy。在1-2 之间添加 VoIP 一个应用，SIP Call Model： Direct；节点 1 和 2: Multimedia Signaling 设置为 SIP。两个个节点位于一个有线子网。

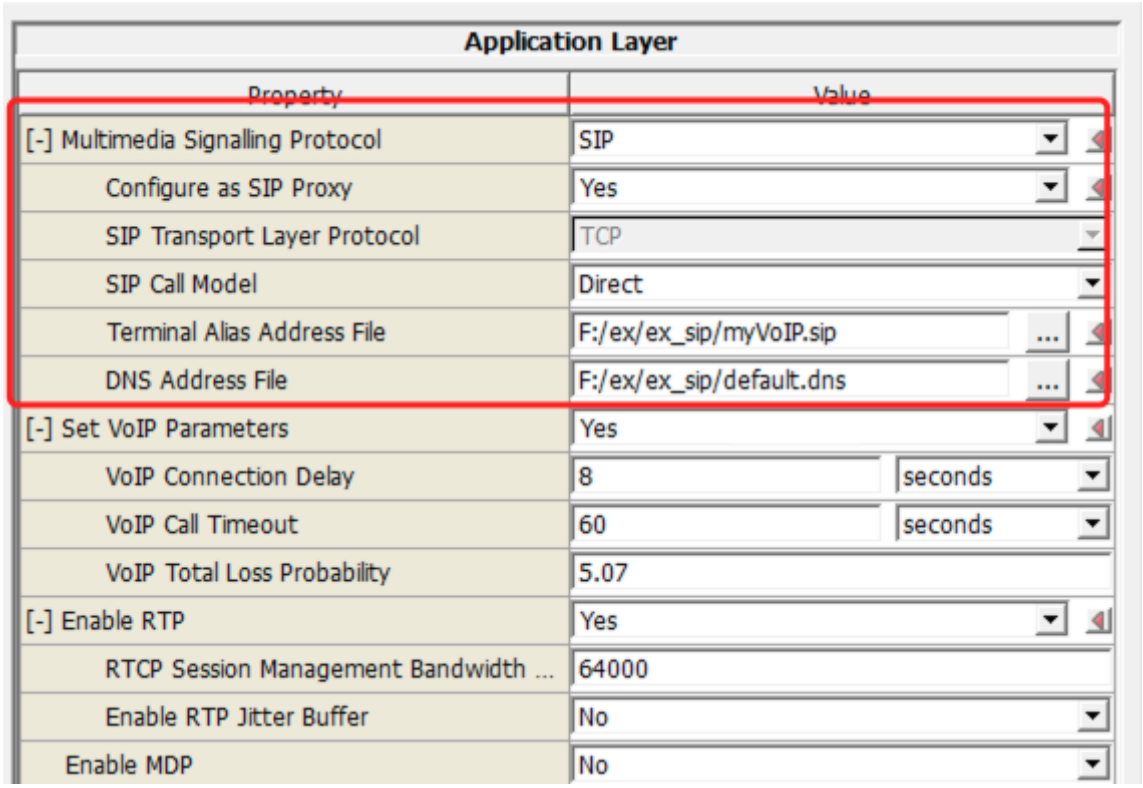
关于 VoIP 场景实例，参加笔记EXata学习（12）：VoIP Proxy 场景 Step by Step。

SIP 节点地址薄文件采用 myVoIP 文件，内容如下：

```
1    190.0.1.1    Alice        cqupt.com    5    190.0.1.5
2    190.0.1.2    Bob          cqupt.com    5    190.0.1.5

3    190.0.1.3    Proxy        cqupt.com    5    190.0.1.3
```

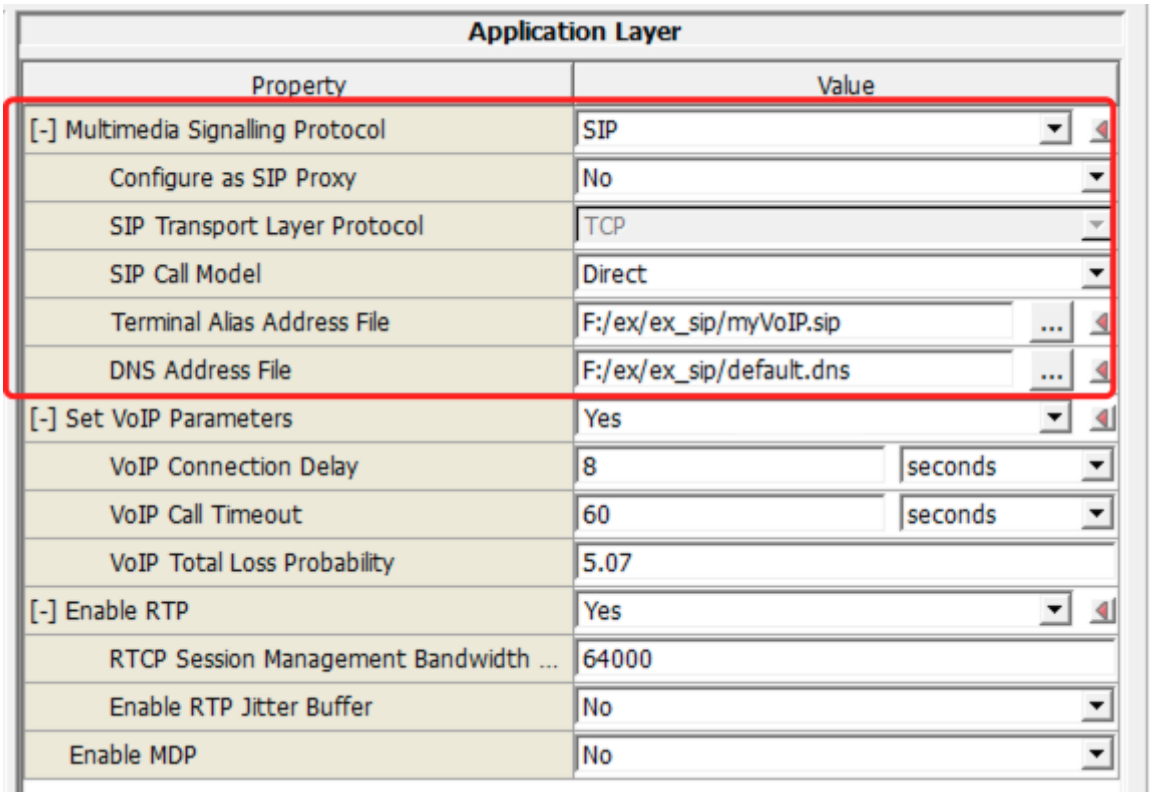
只有一个域，DNS 暂时不发挥作用，只给一个default.dns，内容为空。

Proxy 节点按以下对“Multimdedia Signaling Protocol”部分进行节点配置：



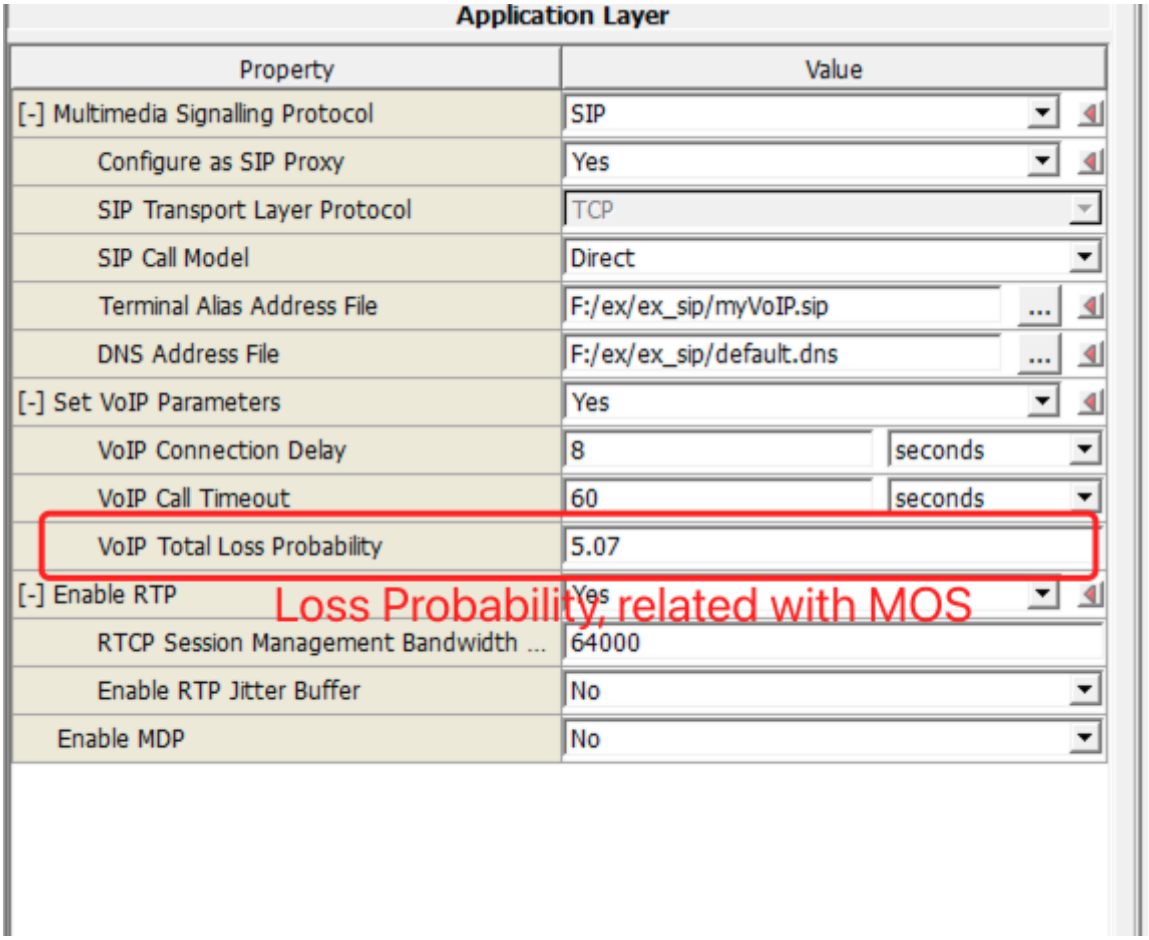
Application Layer	
Property	Value
[-] Multimedia Signalling Protocol	SIP
Configure as SIP Proxy	Yes
SIP Transport Layer Protocol	TCP
SIP Call Model	Direct
Terminal Alias Address File	F:/ex/ex_sip/myVoIP.sip
DNS Address File	F:/ex/ex_sip/default.dns
[-] Set VoIP Parameters	Yes
VoIP Connection Delay	8 seconds
VoIP Call Timeout	60 seconds
VoIP Total Loss Probability	5.07
[-] Enable RTP	Yes
RTCP Session Management Bandwidth ...	64000
Enable RTP Jitter Buffer	No
Enable MDP	No

对 Alice 和 Bob 进行如下设置，与 Proxy 不同的 是“Configure as SIP Proxy”为“No”



Application Layer	
Property	Value
[-] Multimedia Signalling Protocol	SIP
Configure as SIP Proxy	No
SIP Transport Layer Protocol	TCP
SIP Call Model	Direct
Terminal Alias Address File	F:/ex/ex_sip/myVoIP.sip
DNS Address File	F:/ex/ex_sip/default.dns
[-] Set VoIP Parameters	Yes
VoIP Connection Delay	8 seconds
VoIP Call Timeout	60 seconds
VoIP Total Loss Probability	5.07
[-] Enable RTP	Yes
RTCP Session Management Bandwidth ...	64000
Enable RTP Jitter Buffer	No
Enable MDP	No

注意：其中“Set VoIP Parameters”部分与 SIP 呼叫的应答以及话音流的丢包率有关，节点丢包率默认为 5.07%，这与后期的 MOS 评分有关！！！修改节点丢包率默认值，该参数应出现在场景配置文件中。



保存场景后，上述节点配置，体现在ex_sip.config 的<Node Configuration> 部分：

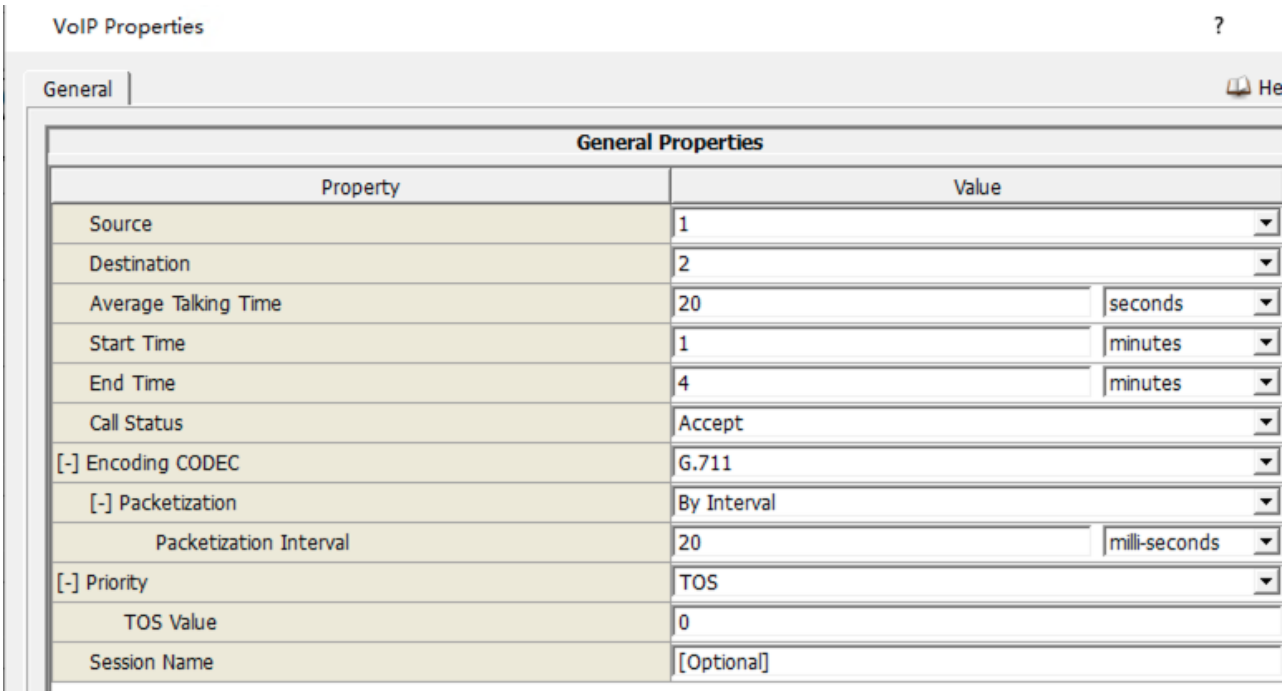
```
#******************Node Configuration************************************

[1 thru 3]       MULTIMEDIA-SIGNALLING-PROTOCOL SIP
[1 thru 3]       DUMMY-VOIP-APPLICATION-EXISTS YES
[1 thru 3]       RTP-ENABLED YES
[1 thru 3]       RTCP-SESSION-MANAGEMENT-BANDWIDTH 64000
[1 thru 3]       DNS-ADDRESS-FILE F:/ex/ex_sip/default.dns
[1 thru 3]       VOIP-CONNECTION-DELAY 8S
[1 thru 3]       VOIP-CALL-TIMEOUT 60S
[1 thru 3]       SIP-CALL-MODEL DIRECT
[1 thru 3]       TERMINAL-ALIAS-ADDRESS-FILE F:/ex/ex_sip/myVoIP.sip
[1 thru 3]       RTP-JITTER-BUFFER-ENABLED NO
[1]              HOSTNAME Alice
[2]              HOSTNAME Bob
[3]              HOSTNAME Proxy
[3]              SIP-PROXY YES
[3]              GUI-NODE-2D-ICON SoftX.png
[1 thru 3]       SIP-TRANSPORT-LAYER-PROTOCOL TCP
[3]              GUI-NODE-3D-ICON SoftX.png
[1 thru 3]       NODE-PLACEMENT FILE
NODE-POSITION-FILE ex_sip.nodes
```

- 修改节点 3 的节点丢包率为 4.07%，则config 文件中将多出一行：VOIP-TOTAL-LOSS-PROBABILITY，【这里TOTAL 怎么理解？】

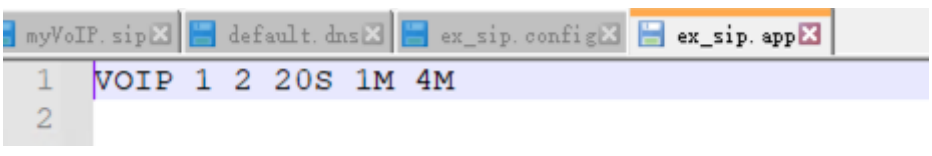
```
#******************Node Configuration*************************************

[3]                VOIP-TOTAL-LOSS-PROBABILITY 4.07
[1 thru 3]         MULTIMEDIA-SIGNALLING-PROTOCOL SIP
[1 thru 3]         DUMMY-VOIP-APPLICATION-EXISTS YES
[1 thru 3]         RTP-ENABLED YES
[1 thru 3]         RTCP-SESSION-MANAGEMENT-BANDWIDTH 64000
[1 thru 3]         DNS-ADDRESS-FILE F:/ex/ex_sip/default.dns
[1 thru 3]         VOIP-CONNECTION-DELAY 8S
[1 thru 3]         VOIP-CALL-TIMEOUT 60S
[1 thru 3]         SIP-CALL-MODEL DIRECT
[1 thru 3]         TERMINAL-ALIAS-ADDRESS-FILE F:/ex/ex_sip/myVoIP.sip
[1 thru 3]         RTP-JITTER-BUFFER-ENABLED NO
[1]                HOSTNAME Alice
[2]                HOSTNAME Bob
[3]                HOSTNAME Proxy
[3]                SIP-PROXY YES
[3]                GUI-NODE-2D-ICON SoftX.png
[1 thru 3]         SIP-TRANSPORT-LAYER-PROTOCOL TCP
[3]                GUI-NODE-3D-ICON SoftX.png
[1 thru 3]         NODE-PLACEMENT FILE
NODE-POSITION-FILE ex_sip.nodes
```

1.2 添加 VoIP 应用

在 Alice 和 Bob 之间添加一个 A--> B 的 VoIP 应用，参数全部采用默认。默认语音编码格式为 G.711.

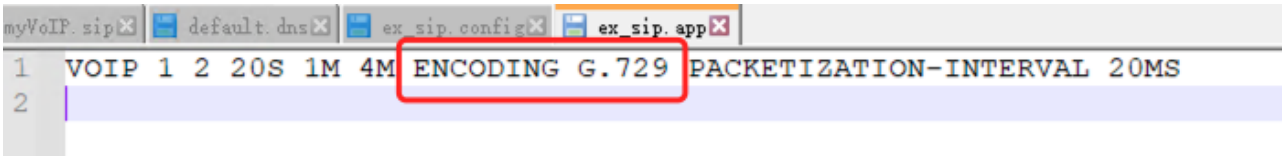


场景保存后，场景文件增加一行，“APP-CONFIG-FILE ex_sip.app”，而场景目录下新生成 ex_sip.app 文件，描述 应用的输入参数：



问题：G711 跑哪里了呢？【答：由于G711 为默认编码方式，因此 app 文件中可以不加参数，只要其后都为默认参数】

将语音编码格式改为：G729，重新保存，app 文件中将显示编码格式参数

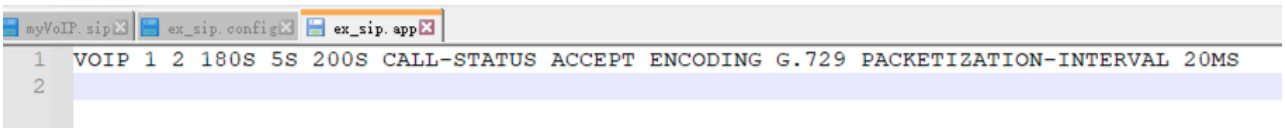


1.3 VOIP 应用实现

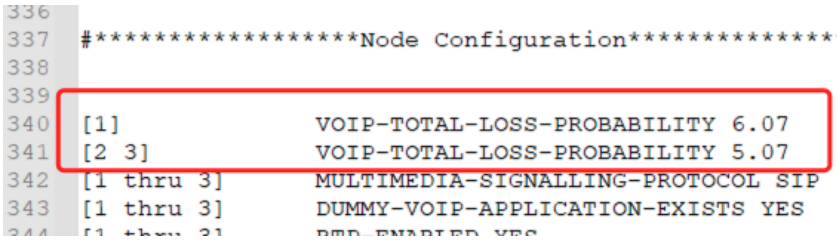
注意：在调试前先确保场景在 GUI 下正常运行。

比如：修改 VoIP 应用开始时间，确保在仿真时间内业务完成。Analyzer 部分能生成 MoS 评分。

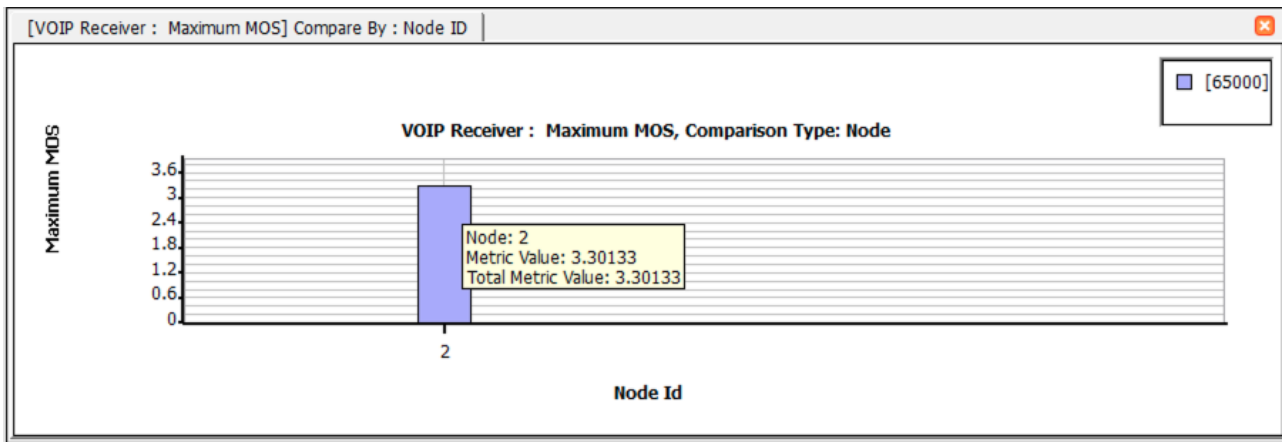
- VoIP 业务设置：

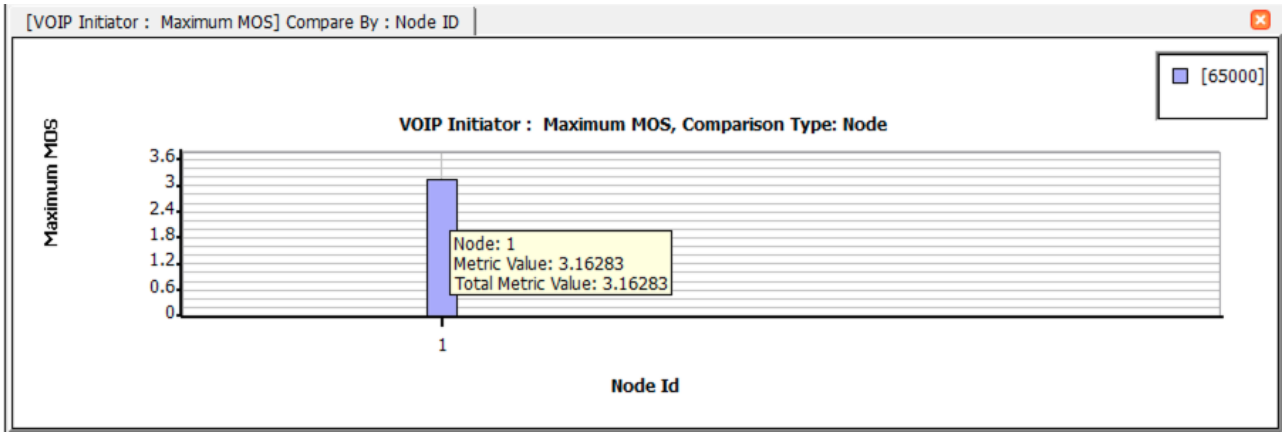


- 节点丢包率设置：



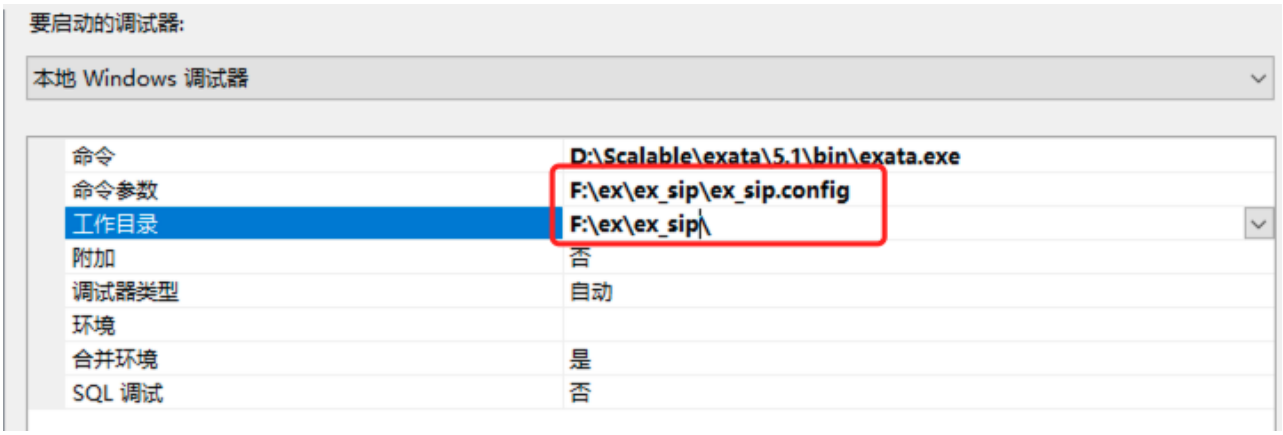
- 发起端和接收端的 MOS 评分不一样：发起端为3.16，接收端位3.30





这部分通过代码调试，分析 EXata 的呼叫建立、语音编码和 MOS 评分的实现。

1. 在 vs2010 中打开 exta 工程，设置调试器，将新创建的 ex_sip.config作为命令参数，相应设置工作目录。



2. VoIP 流量发生和 MOS 评分由 app_voip.h 和 app_voip.cpp 两个文件定义。
3. VoIP 发端发一个包的调用过程

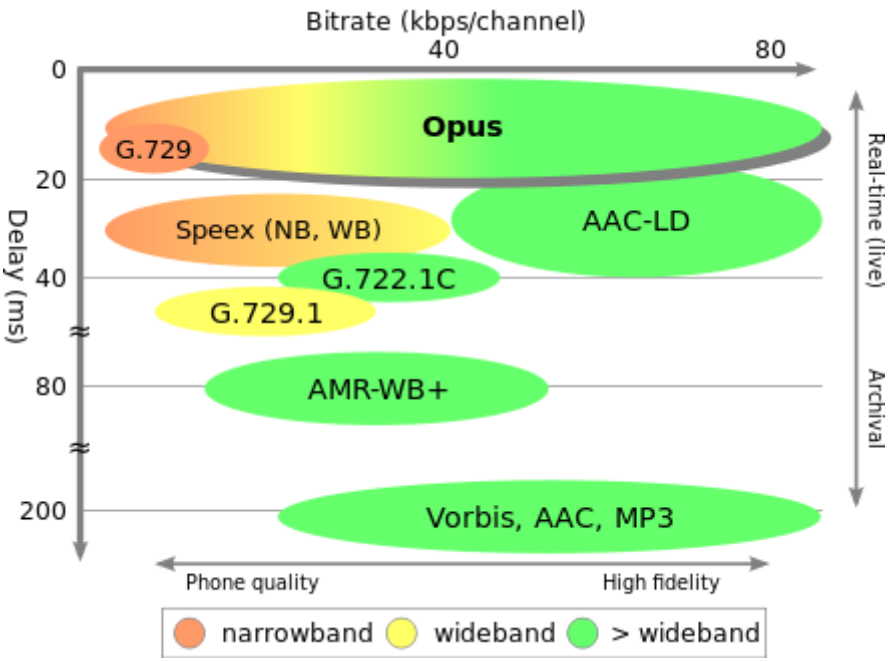
```
@startuml
start
:呼叫端初始化: VoipCallInitiatorInit;
:其中1, 计算数据包大小: bitsPerInterval = packetizationIntervalInSeconds *
bitRatePerSecond。这里与语音编码方式有关;
:其中2, 创建新会话: VoipInitiatorCreateNewSession;
:其中3, 通过 MESSAGE_Send 设定新事件: VOIP_INITIATE_CALL;
:呼叫端事件处理: VoipCallInitiatorHandleEvent;
if (VOIP_SEND_DATA 事件? ) then (true)
:VoipSendDataToRtp,发送数据和发端指针给 RTP;
:其中1, 添加 RTP 包头;
:其中2, 生成 UDP消息: APP_UdpCreateMessage;
:其中3, 添加净荷: APP_AddPayload;
:其中4, 交 UDP 发送: APP_UdpSend;
:其中5, 复制消息, 即timer: MESSAGE_Duplicate, 并发送, 即设定发送下一个包: MESSAGE_Send;
:其中, 时间间隔为 packetizationInterval, 这个与编码方式有关;
else if (VOIP_INITIATE_CALL 事件?) then (true)
:呼叫建立处理;
```

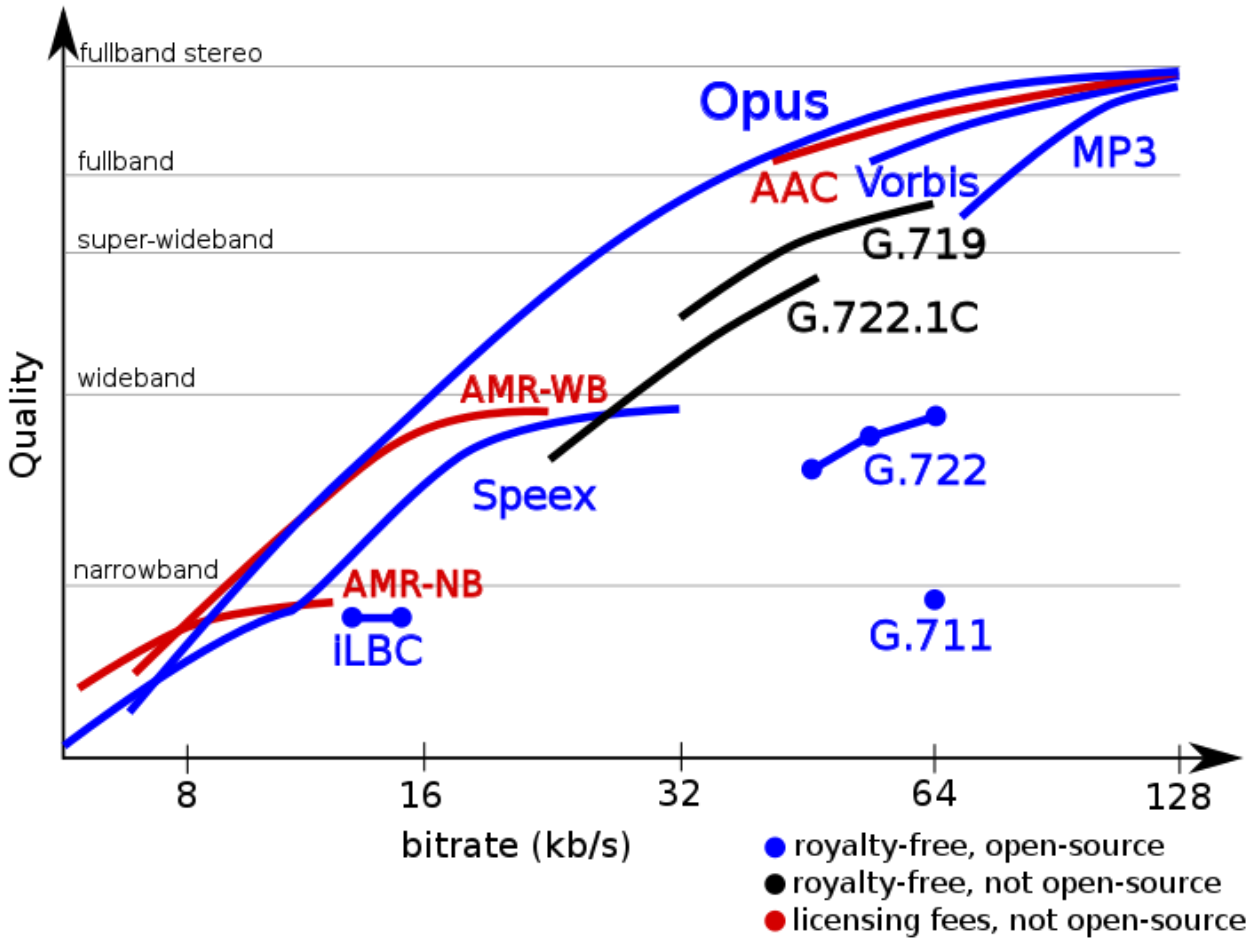
```
else (其他事件)
:其他事件处理;
end
endif
stop
@enduml
```

1.4 opus 语音编码

来自维基百科：Opus是一个有损音频压缩的数字音频编码格式，由Xiph.Org基金会开发，之后由互联网工程任务组（IETF）进行标准化，目标是希望用单一格式包含声音和语音，取代Speex和Vorbis，且适用于网络上低延迟的即时声音传输，标准格式定义于RFC 6716文件。Opus格式是一个开放格式，使用上没有任何专利或限制。

Opus集成了两种声音编码的技术：以语音编码为导向的SILK和低延迟的CELT。Opus可以无缝调节高低比特率。在编码器内部它在较低比特率时使用线性预测编码在高比特率时候使用变换编码（在高低比特率交界处也使用两者结合的编码方式）。





Opus可以处理各种音频应用，包括IP语音、视频会议、游戏内聊天、流音乐、甚至远程现场音乐表演。它可以从低比特率窄带语音扩展到非常高清音质的立体声音乐。支持的功能包括：

1. **6 kb/秒到510 kb/秒的比特率；单一频道最高256 kb/秒**
2. 采样率从8 kHz（窄带）到48 kHz（全频）
3. 帧大小从2.5毫秒到60毫秒
4. 支持恒定比特率（CBR）、受约束比特率（CVBR）和可变比特率（VBR）
5. 支持语音（SILK层）和音乐（CELT层）的单独或混合模式
6. 支持单声道和立体声；支持多达255个音轨（多数据流的帧）
7. 可动态调节比特率，音频带宽和帧大小
8. 良好的鲁棒性丢失率和数据包丢失隐藏（PLC）
9. 浮点和定点实现

- opus 编码支持两种不同的模式：
  - **话音（voice）模式**：较低的编码速率，主要用于话音通信。
  - **音频（audio）模式**：中等或最高的编码速率，用于音乐、立体声等编码。
- 推荐比特率
  - 8–12 kb/s for NB speech,
  - 16–20 kb/s for WB speech,
  - 28–40 kb/s for FB speech,

    - 48–64 kb/s for FB mono music, and
    - 64–128 kb/s for FB stereo music.
- 对应采样率（https://chenliang.org/2020/03/15/opus-format/）

缩写（全称）	音频带宽	应用采样率
NB (narrowband)	4 kHz	8 kHz
MB (medium-band)	6 kHz	12 kHz
WB (wideband)	8 kHz	16 kHz
SWB (super-wideband)	12 kHz	24 kHz
FB* (fullband)	20 kHz	48 kHz

- Opus 的帧长
    - 2.5ms、5ms、10ms、20ms、40ms、60ms。
- 采用技术

Opus 编码格式应用了两种技术：一个是**线性预测**（Linear Prediction，LP），另一个是**改进的离散余弦变换**（Modified Discrete Cosine Transform，MDCT）。线性预测技术在低频信号的编码上更加高效，适合处理语音数据。对于包含高频信号的音乐，改进的离散余弦变换这种域变换技术处理效率更高。

Opus 编码格式采用的技术不是全新的，它使用的线性预测技术来自于 Skype 发明的 SILK 编解码器，而改进的离散余弦变换技术来自于 CELT （Constrained-Energy Lapped Transform）。CELT 也是由 Xiph.Org 基金会早期发明一种音频编码格式，现在合并入 Opus 项目后，就不再有独立的 CELT 格式了。

- 工作模式
    - Opus 工作在 SILK 模式时，支持 NB、MB、WB 频率带宽的音频，并且帧长在 10ms ~ 60ms 之间。工作在 CELT 模式时，支持 NB、WB、SWB、FB 音频带宽，并且帧长在 2.5ms ~ 20ms 之间。Opus 还可以工作在混合模式（Hybrid），也就是 SILK 和 CELT 同时起作用，这种情况下只支持 SWB 、FB 音频带宽，并且帧长为 10ms 或 20ms。
- Opus 包结构
    - Opus 编码器处理原始数据输出一串包（Packet），一个包里面可能包含多个编码后的音频帧数据，只是这些音频帧的参数必须是一致的，例如：编码模式、音频带宽、帧大小以及声道数。
- 其他技术：
    - Discontinuous Tranmission（DTX）不连续发射：对应沉默的部分发送端将不发送，而接收端会产生一些舒服噪音来填补。
    - in-band FEC （带内前向纠错）：对之前的 n-1个包，多发送一个冗余包。
    - 自适应变速率（？）
- EXata 的实现评估
    - EXata 实现会极其困难，主要基于以下观察：
        - EXata 目前主要针对固定比特率和采样率的编码，对于 opus 这种变速率和采样率的编码方案很难

添加到下面的结构体中，或者参考 VBR 的流量发生来实现，但将重新实现与多媒体信令（如 SIP）的接口。

```
VoipEncodingScheme encodingScheme[] =
        {
            {"G.711", 64000, "20MS"},
            {"G.729", 8000, "20MS"},
            {"G.723.1ar6.3", 6300, "30MS"},
            {"G.723.1ar5.3", 5300, "30MS"},
            {"G.726ar32", 32000, "20MS"},
            {"G.726ar24", 24000, "20MS"},
            {"G.728ar16", 16000, "30MS"}
         };
```

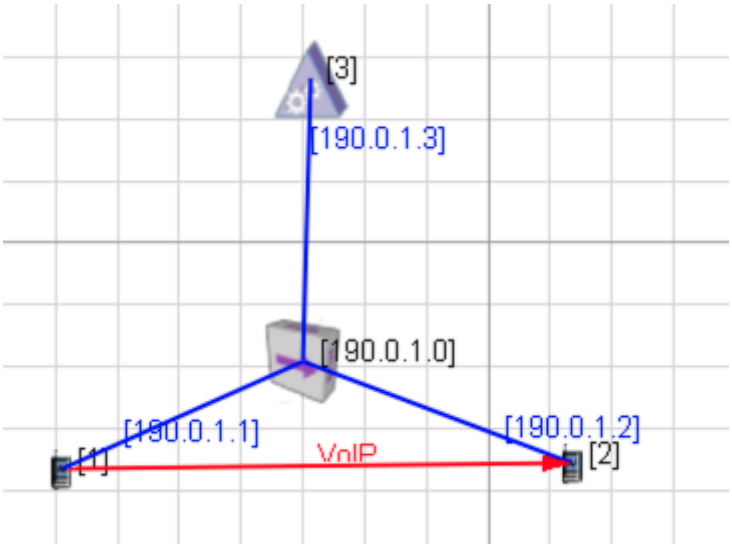
- 实现的价值不大。添加 opus、iLBC 的目标是利用 EXata 的复杂网络环境的模拟能力、以及 MOS 评分能力。复杂网络模拟能力，同样可以对半实物接入流量发挥作用；而EXata 自带的 MOS 评分功能主要基于 PESQ 或 R-factor 实现（VOIPGetMeanOpinionSquareScore 方法），主要考虑了单边延迟（oneway delay）和总的丢包率（Node Configuration 中设置的值），与编码方式无关，并没有实现 opus 更复杂发包机制的考虑（自适应变速率、in-band FEC等），因此，即使添加成功，EXata 的 MOS 打分规则并不能体现 opus 编码方案的整体优势。
- VOIPGetMeanOpinionSquareScore 方法实现如下，位于 app_voip.cpp中：

```
void VOIPGetMeanOpinionSquareScore(Node *node,
                                   VoipHostData* voipHostData,
                                   clocktype oneWayDelay)
{
    //Calculate Transmission rating Factor R Factor
    //  Using the formula R = 93.2 - Id - Ief
    //  Where Id is Delay Impairment Factor, caused by delay in network.
    //        Ief is Equipment impairment factor, caused by low bit rate
    //               coders,the effect of frame loss on the coder.
    //      Id = 0.024d + 0.11(d - 177.3) H(d - 177.3)
    //   Where d is one way delay (coding + network + de-jitter delay)(ms)
    //  And H(x) = 0 for x<0
    //      H(x) = 1 for x>=0
    double functionOfHx = 0.0;
    double functionOfHx1 = 0.0;

    double delayImpairmentFactor;
    double equipmentImpairmentFactor;
    double transmissionRatingFactorR;

```

```
    AppMultimedia* multimedia = node->appData.multimedia;
    double totalLossProbablity = multimedia->totalLossProbablity;

    double scoreMOS = 0.0;
    double oneWayDelayinMS;

    oneWayDelayinMS = (double)(oneWayDelay / MILLI_SECOND);

    if ((oneWayDelayinMS - 177.3) < 0.0)
    {
        functionOfHx = 0.0;
    }
    else if ((oneWayDelayinMS - 177.3) >= 0.0)
    {
        functionOfHx = 1.0;
    }

    delayImpairmentFactor = (0.024 * oneWayDelayinMS)
                    + (0.11 * (oneWayDelayinMS - 177.3) * functionOfHx);

    // Ief = 30 ln(1+15e) H(0.04-e)+19ln(1+70e) H(e-0.04)
    if ((0.04 - totalLossProbablity) < 0)
    {
        functionOfHx = 0.0;
    }
    else if ((0.04 - totalLossProbablity) >= 0)
    {
        functionOfHx = 1.0;
    }

    if ((totalLossProbablity - 0.04) < 0)
    {
        functionOfHx1 = 0.0;
    }
    else if ((totalLossProbablity - 0.04) >= 0)
    {
        functionOfHx1 = 1.0;
    }
    equipmentImpairmentFactor
         = ((30.0 * log((double)(1.0 + 15.0 * totalLossProbablity))
```

```
                                    * functionOfHx)
          + (19.0 * log((double)(1.0 + 70.0 * totalLossProbablity))
                                    * functionOfHx1));

    transmissionRatingFactorR = 93.2 - delayImpairmentFactor
                                     - equipmentImpairmentFactor;

    if (transmissionRatingFactorR < 0.0)
    {
        scoreMOS = 1.0;
    }
    else if ((transmissionRatingFactorR >= 0.0)
                && (transmissionRatingFactorR <= 100.0))
    {
        scoreMOS = 1.0 + 0.035 * transmissionRatingFactorR
                       + 7.0 * transmissionRatingFactorR
                       * (transmissionRatingFactorR - 60.0)
                       * (100.0 - transmissionRatingFactorR) * 1.0e-6;
    }
    else if (transmissionRatingFactorR > 100.0)
    {
        scoreMOS = 4.5;
    }
    voipHostData->scoreMOS += scoreMOS;
    if (voipHostData->maxMOS < scoreMOS)
    {
        voipHostData->maxMOS = scoreMOS;
    }
    if (voipHostData->minMOS > scoreMOS)
    {
        voipHostData->minMOS = scoreMOS;
    }
}

```

- 在ex_sip.config场景中，VoIP 业务的编码由 G.729换成 G.711， 节点丢包率不变，最后MOS 评分没有改变。当然，由于网络比较简单，语音包大小没有带来延迟或丢包的变化。
- 语音编码：G711，节点1丢包率：10.07%，发起者 MOS：2.73793
- 语音编码：G729，节点1丢包率：10.07%，发起者 MOS：2.73793，一模一样！



1.5 结论：

- 不再探索在 EXata 上实现 Opus。
- 可以添加 iLBC 练练手，它支持两种固定比特率的窄带 IP 语音。

1.6 iLBC 编码

来自维基百科：

- **标准：RFC 3951, RFC 3952 （iLBC RTP payload 格式）**
- **RTP：RFC 3550**
- **Internet Low Bitrate Codec** (**iLBC**) is a royalty-free narrowband speech audio coding format and an open-source reference implementation (codec), developed by Global IP Solutions (GIPS) formerly Global IP Sound，后来被 Google 收购。
- 2011 年后，iLBC 作为开源软件出现，作为WebRTC 的一部分。
- 技术参数：
  - 采样频率：8kHz/16bit
  - 两种固定比特率：
    - 15.2 kbps （20ms 帧长）
    - 13.33 kbps （30ms 帧长）
  - 固定帧大小：
    - 304 bits（32字节） （20ms 帧长）
    - 400 bits （50字节） （30ms 帧长）
  - 一个 RTP packets 可以装载多个 iLBC 帧 （RFC3952）

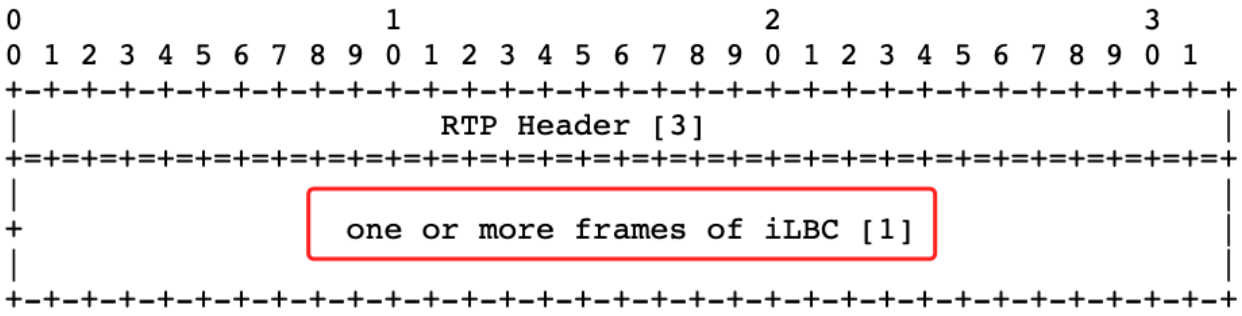


Figure 1, Packet format diagram

- 发端的封包策略
  - 时延敏感的实时应用：单个 RTP 封装尽量少的语音帧。
  - 带宽受限或时延不敏感的应用：可以单个RTP 多个语音帧。

1.7 添加 iLBC

1.7.1 主要数据结构

多媒体信息应用信令信息

```
// /**
// STRUCT      :: AppMultimedia
// DESCRIPTION :: Store multimedia signalling related information
// **/
struct AppMultimedia
{
    // MULTIMEDIA SIGNALLING related structures
    // 0-INVALID, 1-H323, 2-SIP
    unsigned char sigType;

    // currently SIP and H323 are available, H323Data and SipData
    // structure pointer is stored depending on the signalling type
    void* sigPtr;

    InitiateConnectionType  initiateFunction;
    TerminateConnectionType terminateFunction;

    IsHostCallingType       hostCallingFunction;
    IsHostCalledType        hostCalledFunction;

    double                  totalLossProbablity;
    clocktype               connectionDelay;
```

```
    clocktype               callTimeout;
};
```

- SIP 协议实现：
  - multimedia_sipdata.h: SIP协议数据结构
  - multimedia_sipmeg.h：SIP 消息结构

1.7.2 几个相关方法

- VoipReadConfiguration:
  - 这个方法用于读取场景配置文件中，VoIP 相关参数的设置，要传递给 SIP 实例。其输入参数中有编码方案 char* codeScheme，自*APP_InitializeApplication* 中从 XXX.app中读取VoIP 属性框而获取。
    - VOIP-CONNECTION-DELAY【src/dest SIP】
    - VOIP-CALL-TIMEOUT 【src/dest SIP】
    - VOIP-TOTAL-LOSS-PROBABILITY 【src/dst Node】

```
#******************Node Configuration********************************

[1]             VOIP-TOTAL-LOSS-PROBABILITY 10.07
[2 3]           VOIP-TOTAL-LOSS-PROBABILITY 5.07
[1 thru 3]      MULTIMEDIA-SIGNALLING-PROTOCOL SIP
[1 thru 3]      DUMMY-VOIP-APPLICATION-EXISTS YES
[1 thru 3]      RTP-ENABLED YES
[1 thru 3]      RTCP-SESSION-MANAGEMENT-BANDWIDTH 64000
[1 thru 3]      DNS-ADDRESS-FILE F:/ex/ex_sip/default.dns
[1 thru 3]      VOIP-CONNECTION-DELAY 8S
[1 thru 3]      VOIP-CALL-TIMEOUT 60S
[1 thru 3]      SIP-CALL-MODEL DIRECT
[1 thru 3]      TERMINAL-ALIAS-ADDRESS-FILE F:/ex/ex_sip/myVoIP.sip
[1 thru 3]      RTP-JITTER-BUFFER-ENABLED NO
```

- VoIPGetEncodingSchemeDetails：利用输入 codeScheme 字符串，根据VoIP 编码方案获取详细参数，
  - VoipEncodingScheme 结构体（in app_voip.h）

```
typedef struct
{
    char codecScheme[20];
    unsigned int bitRatePerSecond;
    char packetizationIntervalInMilliSec[6];
} VoipEncodingScheme;
```

  - encodingScheme 编码方案（in VoIPGetEncodingSchemeDetails）

```
//-------------------------------------------------------------------
void VoipGetEncodingSchemeDetails(char* codecScheme,
                                  clocktype& packetizationInterval,
                                  unsigned int& bitRatePerSecond)
{
    int index;
    char intervalMS[MAX_STRING_LENGTH];

    VoipEncodingScheme encodingScheme[] =
            {
                {"G.711", 64000, "20MS"},
                {"G.729", 8000, "20MS"},
                {"G.723.1ar6.3", 6300, "30MS"},
                {"G.723.1ar5.3", 5300, "30MS"},
                {"G.726ar32", 32000, "20MS"},
                {"G.726ar24", 24000, "20MS"},
                {"G.728ar16", 16000, "30MS"}
            };

    int upperIndex = sizeof(encodingScheme) / sizeof(VoipEncodingScheme);
```

- 添加新的固定码率的编码方式，只需要修改VoIP 属性框、扩展此encodingScheme 数组即可。

1.7.3 VoIP 初始化过程

- 应用程序中进行 VoIP 初始化进行的操作【此时：似乎与编码格式无关，已经转换成interval等参数】
  - 初始化目的端 SIP 信令（Bob）；
  - 初始化客户端 SIP 信令（Alice）；
  - VoIP 呼叫（媒体流）发起端初始化（VoipCallInitiatorInit）
    - 创建新会话：~CreateNewSession
    - 设定新事件（Timer Message）：VOIP_INITIATE_CALL
    - 添加 Client 地址信息： AppVoipClientAddAddressInformation
  - 呼叫状态为 Accept，则将呼叫添加到目的节点的列表中：VoipAddNewCallToList
  - VoIP 呼叫（媒体流）接收端端初始化（VoipCallReceiverInit）
    - 检查接受的呼叫列表：VoipCheckCallList
    - 如果接受，则创建新回话：VoipReceiverCreateNewSession
    - 添加客户端地址信息： AppVoipClientAddAddressInformation

1.7.4 扩展编码方案

在 encodingScheme 数组中添加 iLBC。

由于 iLBC 有两种速率和帧长，姑且分别命名为 iLBC.20 和 iLBC.30，分别对应 20ms 和 30ms 的帧长，添加后如下：

```
94      VoipEncodingScheme encodingScheme[] =
95          {
96              {"G.711", 64000, "20MS"},
97              {"G.729", 8000, "20MS"},
98              {"G.723.1ar6.3", 6300, "30MS"},
99              {"G.723.1ar5.3", 5300, "30MS"},
100             {"G.726ar32", 32000, "20MS"},
101             {"G.726ar24", 24000, "20MS"},
102             {"G.728ar16", 16000, "30MS"},
103             // LuoJT: add two iLBC schemes
104             {"iLBC.20", 15200, "20MS"},
105             {"iLBC.30", 13330, "30MS"}
106         };
```

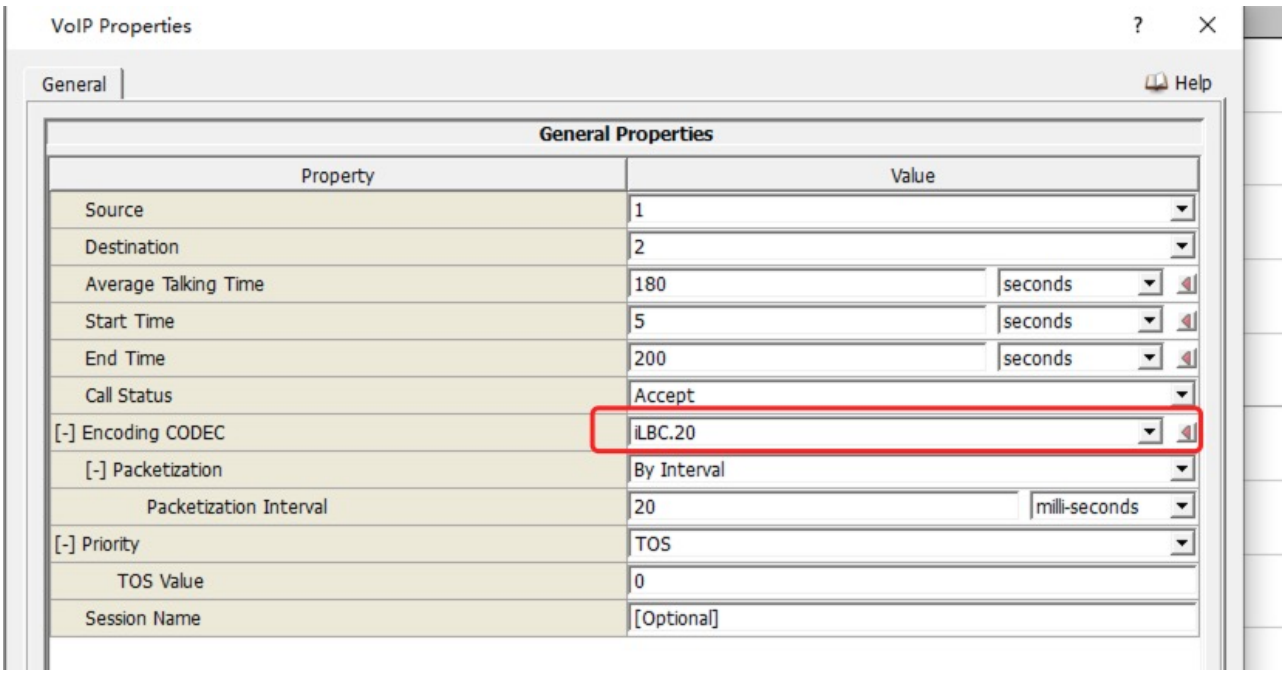
1.7.4 VoIP GUI 添加 iLBC 编码

VoIP 的属性定义在 CMP 文件“voip.vmp”中，添加新的编码方式，只需要在其中 variable – “Encoding CODEC”下添加一个新的 option即可

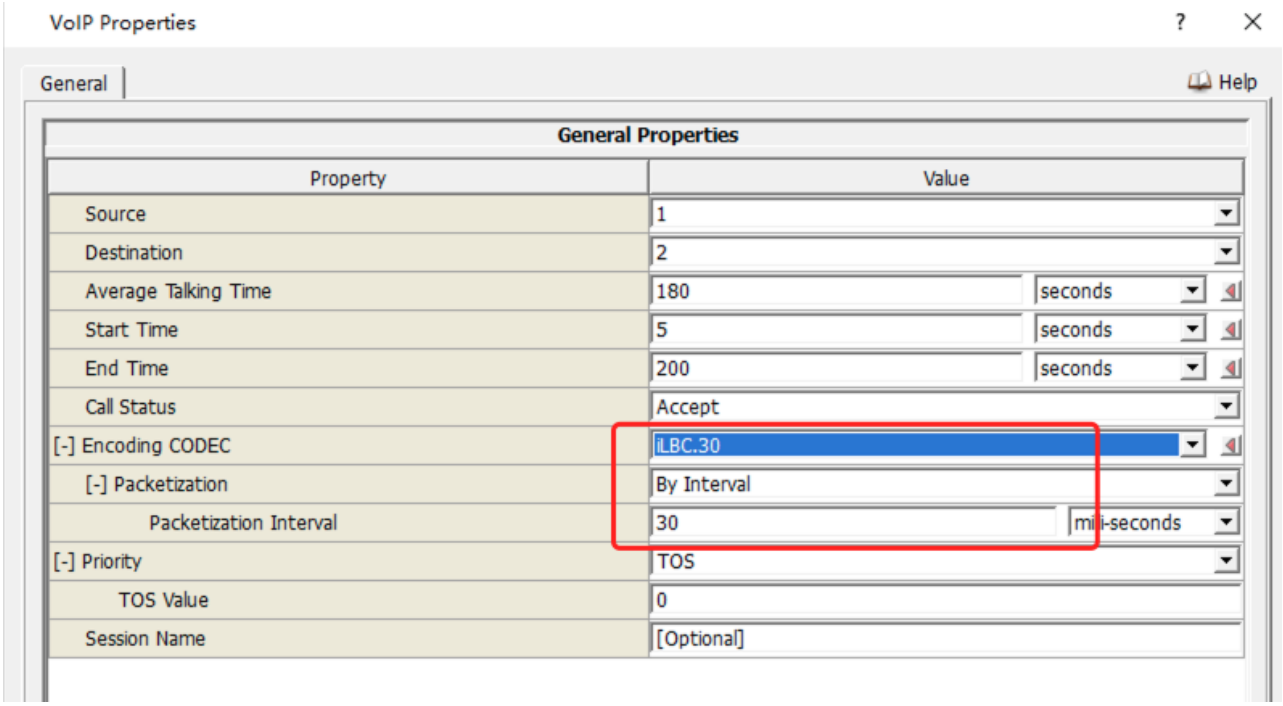
1. 首先添加一个 iLBC.20 ,对应20ms 帧长的编码，注意 key 值递增 1，default 取“20MS”

```
    <option value="iLBC.20" name="iLBC.20">
  <variable name="Packetization" key="PACKETIZATION-DUMMY-8" type="Selection" default="PACKETIZAT
    <option value="PACKETIZATION-INTERVAL" name="By Interval">
      <variable name="Packetization Interval" key="INTERVAL-8" type="Time" default="20MS" min="
    </option>
  </variable>
    </option>
```

2. 效果如下：

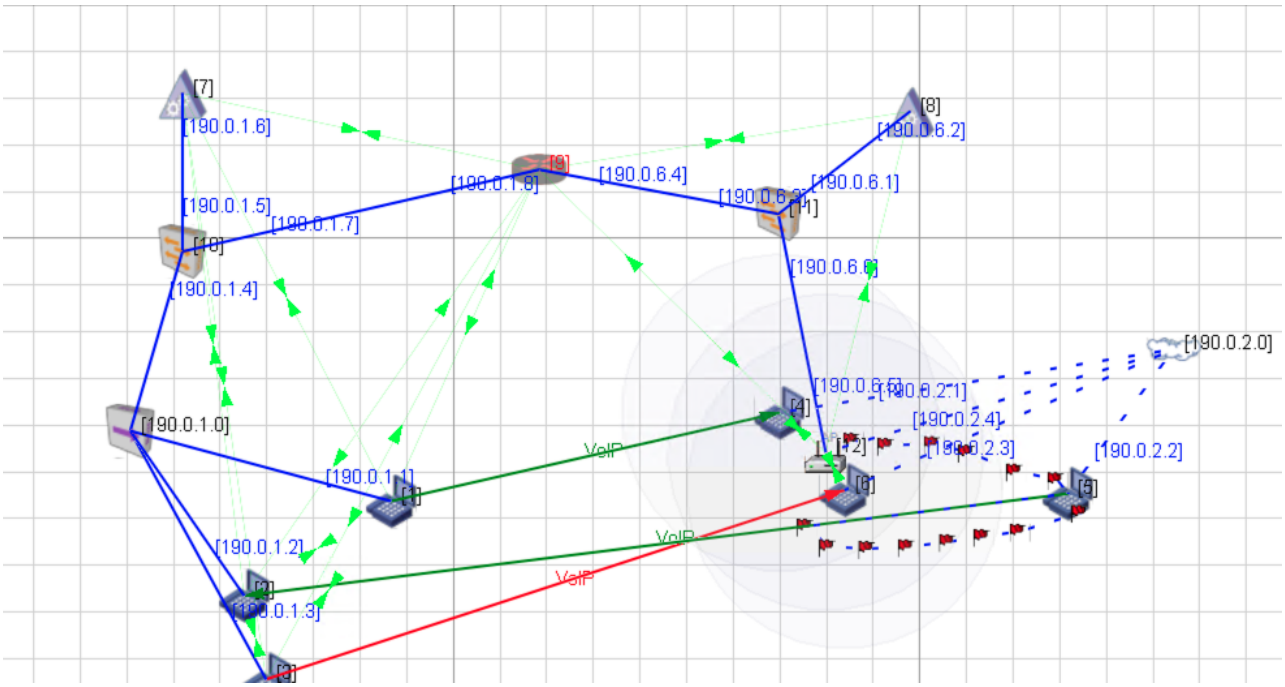


3. 添加 30ms 帧长
4. 效果如下：

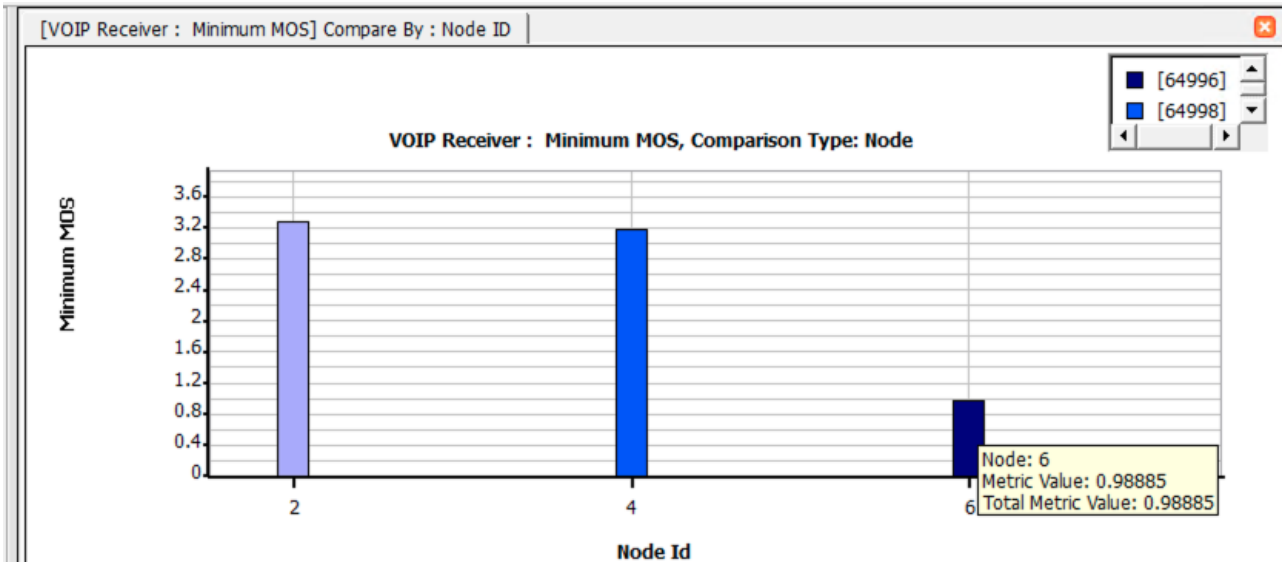


5. 对比 iLBC20 和 iLBC30 的 MOS 评分，节点丢包率取默认值（5.07%）

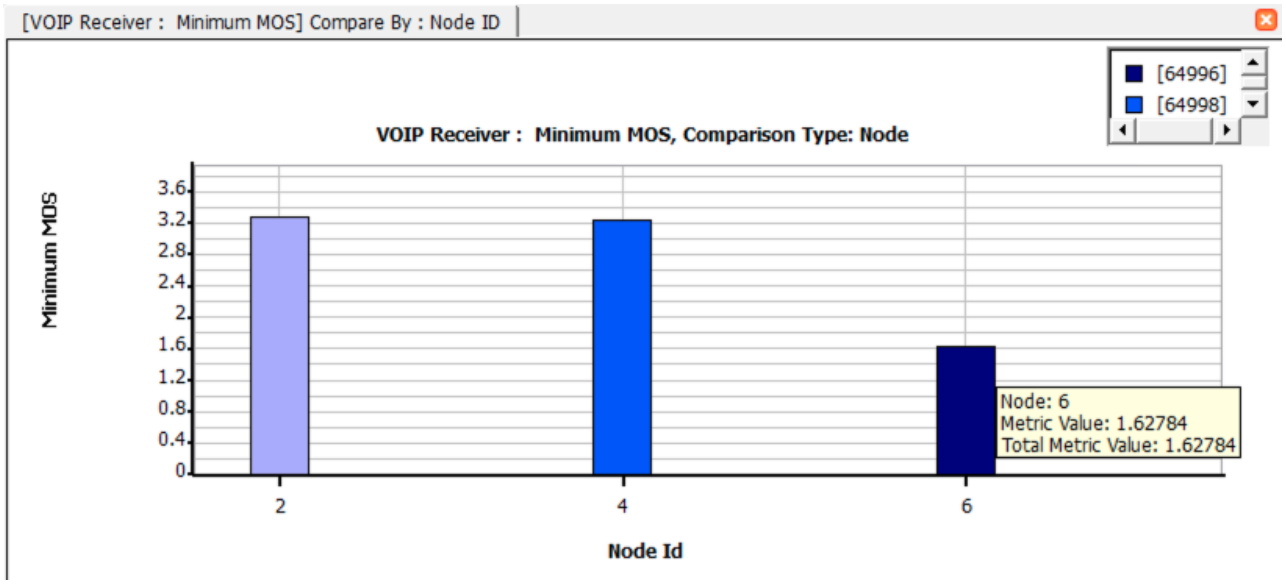
6. 选用一个复杂的场景 myVoIP_Proxy2_wifi，其中，节点 6 会运动到 WiFi 范围以外导致通信中断。



7. 全部采用 G.711，接收端最差 MOS 0.988



8. 3-6 采用 iLBC20，其余仍采用 G711. 节点6 minmum MOS 为 1.6 提升比较明显。其MOS 低于其他两个节点是因为由于移动过程中脱离WiFi 通信范围而存在中断。这个对比表明存在中断情况下 iLBC 编码的优越性。



9.