Nabto Tunnels

NABTO/001/TEN/030

目录

1. 概念 2
2. 参考文献 3
3. 简介 4
4. uNabto Tunnel应用程序 5
4.1. 前提条件 5
4.2. 编译uNabto Tunnel 应用程序 5
4.3. 使用OpenSSL编译 5
4.3.1. 为一个特定目标编译OpenSSL 5
4.3.2. 使用OpenSSL确定代码大小 6
4.4. 在设备上使用uNabto Tunnel 7
4.5. 为资源有限的linux的IP摄像机实现高性能的数据渠道 7
4.5.1. CPU 优化 7
4.5.2. 缓冲优化 8
4.5.3. OpenSSL 编译 8
4.6. 常见问题解答 9
4.6.1. 问题一: 当系统死机时 9
4.6.2. 问题二: 找不到可执行文件 9
4.6.3. 问题三: gethostbyname 警告 9
4.6.4. 问题四: 视频流性能差和/或 uNabto tunnel占用太大 CPU 9
5. Nabto Tunnel客户端 10
5.1. iOS 和Android系统演示应用程序 10
5.2. 桌面演示win32/.NET 10
5.3. 命令行演示 11
5.4. 建构特定的客户端渠道 12

1. 概念

本文档描述了如何使用Nabto通道技术构建如P2P RTSP流媒体解决方案.

2. 参考文献

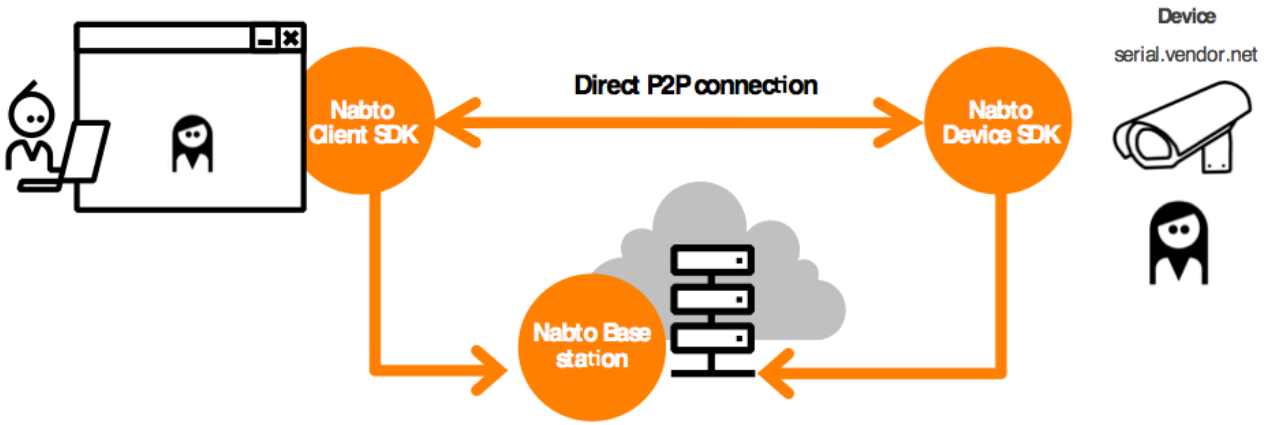
所有文件都可从 https://nabto.com 下载

TEN029	NABTO/001/TEN/029: **Nabto平台规范(Nabto概述)**
TEN023	NABTO/001/TEN/023: uNabto SDK –uNabto 设备应用文档
TEN025	NABTO/001/TEN/025: uNabto SDK – Nabto API用户端应用文档
TEN036	NABTO/001/TEN/036: Nabto安全解决方案

3. 简介

本文详细的描述了如何构建和使用Nabto Tunnel客户端和服务器软件。该软件常用在视频流应用程序中作为RTSP代理客户端和服务器，同时允许与现有的应用程序进行简单集成.

Nabto Tunnel解决方案是建立在uNabto SDK (设备端)和the Nabto 客户端SDK (如移动、桌面和m2m客户机应用程序)之上的:



从设备上来讲,Nabto Tunnel应用程序位于现有的TCP服务器(例如：RTSP服务器)的前端 ,本质上作为一个反向代理(或Nabto-TCP网关)。从客户端来讲,Nabto Tunnel客户端端点服务于TCP服务端点使现有应用程序和透明通道数据流能够实现设备端需求的目标服务,本质上作为一个设备代理(或TCP-Nabto网关)。

在[TEN023]和 [TEN025]中分别介绍了两个单独的SDK.

Nabto 提供了一个用uNabto SDK构建的开源渠道应用程序 -这通常是安装在目标系统上通过视频解决方案提供商 (如。、摄像机、DVR或NVR)或轻微的修改。这个应用程序在如下第四节描述中。.

供应商可以用Nabto Client SDK整合现有的视频播放器组件. Nabto Nabto提供了SDK整合的演示应用程序,可从苹果应用程序商店和谷歌玩下载 (搜索Nabto视频). 桌面演示也可以从https://nabto.com下载.。客户端应用程序描述如下第五节。

隧道设备可用于任何TCP数据传输，并必须安全地进行和毫无疑问的为用户配置防火墙。Nabto平台的其他用途,请参阅[TEN029]的总体介绍。

至关重要的是要了解如何确保整个远程访问解决方案的安全:请仔细阅读和了解文档的主要部分“Nabto安全解决方案”(TEN036)如何使用Nabto平台中的设施构建一个安全的解决方案.

4. uNabto Tunnel应用程序

提供的应用程序是作为uNabto SDK的一部分作为一个使用Nabto流抽象的例子. 更多uNabto流信息在设备方面,请看[TEN023]中的第七节。

4.1. 前提条件

uNabto SDK可从下载，https://nabto.com>下载 > uNabto服务器> uNabto SDK.

建立通道的应用程序,需要一个C工具链以及cmake构建工具(从http://www.cmake.org下载)。.

4.2. 编译uNabto Tunnel 应用程序

下载完后, 解压源代码.通道应用程序在 unabto_sdk/unabto/演示 /通道:

```
mkdir build_unabto_tunnel
cd build_unabto_tunnel
export CC=<path to c compiler, only necessary if this is a crosscompilation.>
cmake <path to unabto tunnel folder, e.g. ~/Downloads/unabto_sdk/unabto/demo/tunnel>
make
```

4.3. 使用OpenSSL编译

作为uNabto SDK一部分的通用加密模组方案不能优化所有平台。如果需要高性能流媒体, 加密模组方案必须被更快的模组方案所取代。系统可以运行OpenSSL库,推荐使用这个库。

如果OpenSSL为目标平台所用,，那么下一步就可以跳过 - 它描述了如何为一个特定平台编译OpenSSL libcrypto.a库 (也见下面4.5.3部分和4.6.4微调).

4.3.1. 为一个特定目标编译OpenSSL

从 https://www.openssl.org 下载最新稳定的OpenSSL资源包

配置uNabto SDK来使用它，并如下建构库:

```
mkdir unabtotunnel
cd unabtotunnel
export TUNNEL_DIR=`pwd`
# The openssl target should be a valid target for the platform in use.
export OPENSSL_TARGET= # chose a valid openssl target like linux-armv4 linux-generic32
wget <openssl 1.x>
tar xf <openssl 1.x>
cd <openssl 1.x>
export CC=<path to c compiler, only necessary if this is a cross compilation>
./Configure $OPENSSL_TARGET -prefix=$TUNNEL_DIR/external
make
make install_sw
```

现在安装目录应该包含一个 libssl.a和一个 libcrypto.a文件.

为了能够使用新编译的openssl 库来编译unabto_tunnel, 如下选项

UNABTO_EXTERNAL_BUILD_ROOT 应该被添加到CMake命令:

```
cmake -DUNABTO_EXTERNAL_BUILD_ROOT=$TUNNEL_DIR/external <path to unabto tunnel folder>
```

4.3.2. 使用OpenSSL确定代码大小

为Raspberry PI arm linux 平台编译，此通道有以下可执行文件大小:

没有 libcrypto.a， unabto_tunnel可执行文件为 137kB.

有 libcrypto.a ，unabto_tunnel 可执行文件为 1.3MB.

这个大小可能根据OpenSSL是否调整了平台来优化.

4.3.3. 在Windows CE编写uNabto Tunnel

使用 CMake 工程的话，可以把unabto_tunnel 程序能编写到 Windows CE 中。

```
mkdir tunnel_wince
cd tunnel_wince
cmake -G "Visual Studio 8 2005 <SDK> (<Architecture>)" <path to unabto/demo/tunnel>
cmake --build .
```

4.3.3.1. 在 Beckhoff Windows CE 6.0 PLCs 中编写 uNabto Tunnel

请参看Beckhoff PLC 从下面链接使用SDK的一个具体例子 ftp://ftp.beckhoff.com/Software/embPC-Control/CE/Solutions/SDK/Beckhoff_HMI_600_V2.2_SDK.msi

```
cmake -G "Visual Studio 8 2005 Beckhoff_HMI_600 (x86)" <path to unabto/demo/tunnel>
cmake --build .
```

4.4. 在设备上使用uNabto Tunnel

通道应用程序开始，设备id 和相应的秘钥是必要的，参见[TEN036]，特别是章节9 "安装 uNabto 设备加密钥匙".

对于开发和测试目的,设备的id 和秘钥可从开发者门户网站https://portal.nabto.com获取.

unabto_tunnel 二进制可像如下运行:

```
./unabto_tunnel -d <Device ID> -s -k <Key>
```

究竟密钥和设备ID时如何被提供给可执行文件是取决于积分器的 - 既可以使用一个适当的包装器脚本，例如读取设备上某处永久存储器的密钥. 或者自定义可执行的uNabto tunnel.

4.5. 为资源有限的linux的IP摄像机实现高性能的数据渠道

4.5.1. CPU 优化

对于大多数处理器的一些性能可以通过指定不同的编译器选项来获得。例如下列ARM处理器将受益于CFLAGS选项-mcpu=arm1136j-s.

```
~ # cat /proc/cpuinfo
Processor       : ARMv6-compatible processor rev 5 (v6l)
BogoMIPS        : 384.20
Features        : swp half thumb fastmult edsp java
CPU implementer : 0x41
CPU architecture: 6TEJ
CPU variant     : 0x1
CPU part        : 0xb36
CPU revision    : 5
```

因此，从特定处理器的优化中获益，当建设uNabto tunnel应用程序时使用如出口CFLAGS=“ - mcpu=arm1136J-S”.

4.5.2. 缓冲优化

Linux设备在本例中是一个带有单个h264 源的高清摄像机. 视频源将是1Mbit / s，所以流缓冲区应配置来处理网络路径上相当数量的未确认数据。.

这样说来，最大延迟一个环节是1.5秒 - 然后1Mbit / s的*1.5秒给了1.5Mbit（187KB）缓冲需要。默认段大小为每个分段的1311个字节。该NABTO_STREAM_ * _ WINDOW_SIZE然后应该是187/1.311=142。请注意，继电器连接时，必须考虑到所有往返时延，包括网关（设备 - >网关+网关 - >客户端+客户端 - >网关+网关 - >设备)。

因为它是一个存储有限的高清摄像机，并行流的数目要为保存存储器中的目标进行优化。合适的值为16时，相机将具有良好的视频流性能，它可以处理一些简单的HTTP请求。

unabto_config.h 如下:

```
// unabto_config.h
#ifndef UNABTO_CONFIG_H
#define UNABTO_CONFIG_H

#include <modules/log/dynamic/unabto_dynamic_log.h>

#define NABTO_ENABLE_STREAM 1
#define NABTO_STREAM_MAX_STREAMS 16

#define NABTO_STREAM_RECEIVE_WINDOW_SIZE 142

#define NABTO_SET_TIME_FROM_ALIVE 0

#define NABTO_APPLICATION_EVENT_MODEL_ASYNC 1
#define NABTO_ENABLE_EXTENDED_RENDEZVOUS_MULTIPLE_SOCKETS 1

#define NABTO_APPREQ_QUEUE_SIZE NABTO_CONNECTIONS_SIZE
#define NABTO_ENABLE_DEBUG_PACKETS 1
#define NABTO_ENABLE_DEBUG_SYSLOG_CONFIG 1

#ifdef LOG_ALL
#define NABTO_LOG_ALL 1
#endif

#endif
```

4.5.3. OpenSSL 编译

要建立一个优化的OpenSSL库，而不是通常旧版本附带的工具链 (即未针对ARM 构架进行了优化)，一个新的 OpenSSL分配必须重新编译，为同一个目标优化如上uNabto所述:

```
mkdir build/camera
tar xf openssl-1.0.1j.tar.gz
cd openssl-1.0.1j
export CC=/opt/toolchains/arm-2009q3-67/arm-2009q3/bin/arm-none-linux-gnueabi-gcc
export CFLAGS="-mcpu=arm1136j-s"
./Configure --prefix=`pwd`/../external linux-armv4 $CFLAGS
make install
```

请谨慎链接该库而不是由工具链提供的默认的那个库! 如果运行时没有看到性能改进，如从默认的0.9.8库优化到1.0.x库, 它很可能是因为连接器使用了错误的库. 如果一切失败并且没有性能的提升可见，请尝试简单地删除或重命名该默认库.

4.6. 常见问题解答

4.6.1. 问题一: 当系统死机时

当运行unabto_tunnel 可执行系统死机时。

解决方案: 这可能是由于过多的内存使用尝试调整缓冲区大小.

4.6.2. 问题二: 找不到可执行文件

当运行unabto_tunnel可执行的系统提示：

```
sh ./unabto_tunnel: not found
```

解决方案:可执行文件不符合系统，可尝试编译一个小测试程序,并使用与通道编译相同的选项.

4.6.3. 问题三: gethostbyname 警告

当编译通道编译器提示:

```
Warning: Using 'gethostbyname' in statically linked applications requires at runtime the
shared libraries from the glibc version used for linking
```

解决方案：选用测试方案，你可以在通道使用-A选项，为生产来接触Nabto.

4.6.4. 问题四: 视频流性能差和/或 uNabto tunnel占用太大 CPU

请参见上面0章节关于如何构建一个性能调整库. 尤其要注意建设一个OpenSSL的优化版本，如第4.5.3概述 - 以及如何确保使用正确版本的库.

如果视频性能很差甚至隧道CPU消耗低, 请检查网络质量-你有足够的可用宽带的流问题吗?隧道已经配置了一个足够大的窗口，例如章节4.5.2所述延迟问题? 你考虑过更大的延迟转发连接吗? 网络质量是什么 -你有一个很大的数据包丢失在对等的网络吗(例如一个差的wifi连接)?

通常情况下，标准的Ping工具证明诊断这类问题非常有用的 - 试图衡量同行之间的等待时间（相对的同行WAN地址）和衡量各同行和基站之间。后者可以测量通过ping每个对等设备的DNS名称,它总是解决basesetation相关联的IP地址。此外，Ping工具报告网络丢包问题。

如果接触Nabto支持有关的性能问题，请给本节中的问题提供答案- 什么是在流的CPU使用率和上述ping测试的结果. 如果你观察到的对等或中继连接在表现不佳的情况下.

5. Nabto Tunnel客户端

Nabto通道客户端如[TEN025]中的第5.4章节所述那样发展为Nabto客户端SDK应用程序 .在开发和测试期间准备使用几个现成的例子.

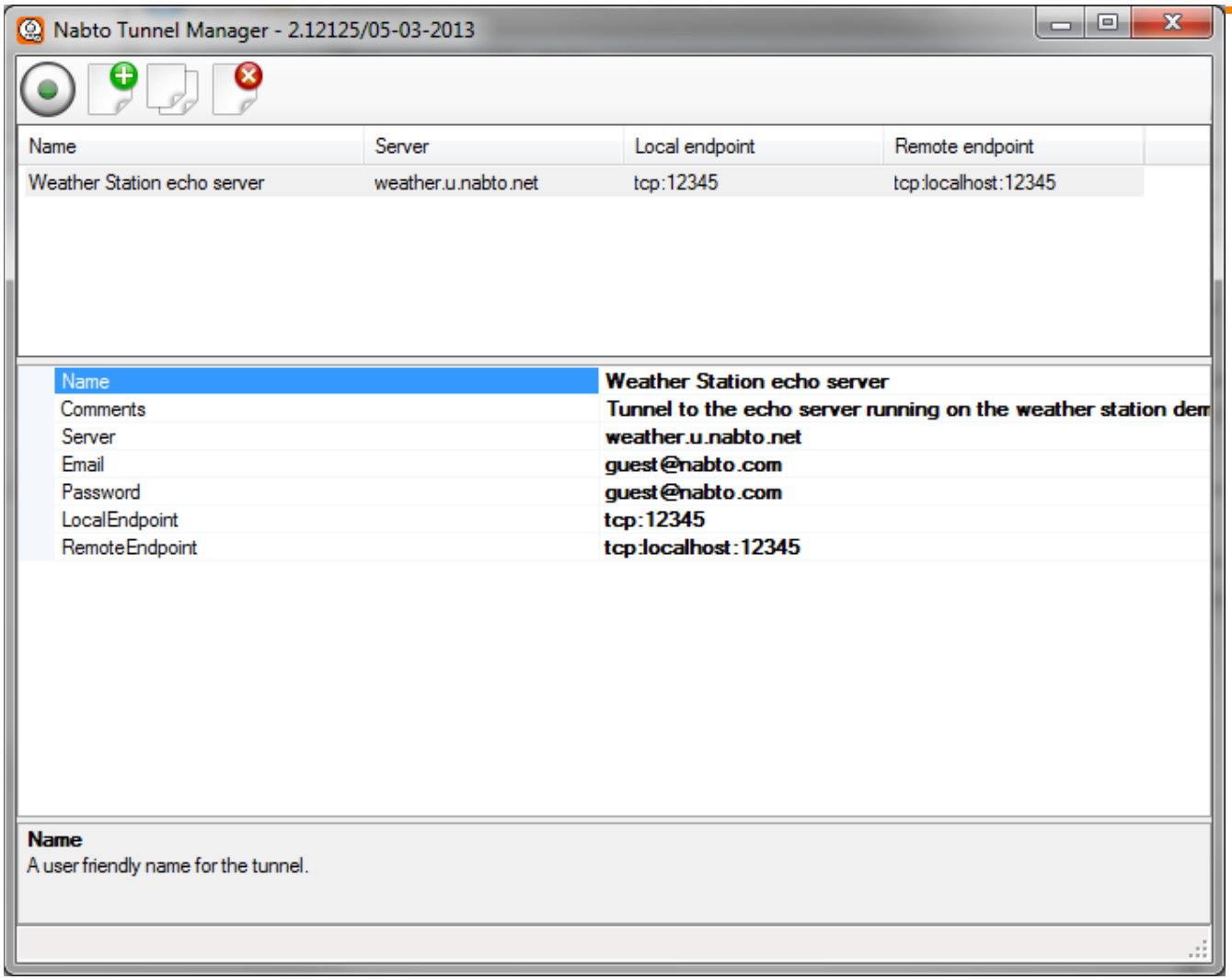
5.1. iOS 和Android系统演示应用程序

在苹果应用程序商店和/或谷歌市场搜索"Nabto 视频". 该应用程序简单的呈现了平台的流媒体功能. 你主要就是输入设备ID – 或者是手动, 通过扫描二维码或在本地网络中找到它. 然后应用程序建立一个通道并展示一个视频输入.

请注意，Nabto在应用程序商店中有好几个应用程序，只有标注“视频”的那个是能够建立TCP通道和流媒体视频的.

5.2. 桌面演示win32/.NET

可从 https://nabto.com 下载用于管理通道应用程序的Nabto Tunnel 管理器.它可设置TCP-Nabto通道——实际的客户端应用程序运行外部应用程序. 例如, 这个程序可能是VLC视频播放器.



5.3. 命令行演示

由Nabto客户端SDK建构的各种简单的命令行工具也可从https://nabto.com下载: .NET 演示包和简单的测试实用工具 "简单的客户端", 可在"Nabto 客户端 SDK"的子页面下载.

后者被使用如下:

```
$ simpleclient_app -q streamdemo.nabto.net --tunnel 12345:127.0.0.1:80
No username specified, defaulting to guest.
Connecting to streamdemo.nabto.net . connected
connected tunnelVer: 1  tunnelStatus REMOTE_P2P (4)
state has changed for tunnel 0xedbb1773 status REMOTE_P2P (4)
```

5.4. 建构特定的客户端渠道

以下出自 [TEN025]第5.4章节的代码段演示需要使用 Nabto客户端SDK来打开客户端应用程序中的一个Nabto通道端点

```
nabto_status_t st = nabtoStartupNULL);
nabto_session_t session;
st = nabtoOpenSession(&session, "user@example.org", "12345678");

nabto_tunnel_t tunnel;
st = nabtoTunnelOpenTcp(&tunnel, session, 8080, “streamdemo.nabto.net”, “localhost”, 80);
while (st == NABTO_OK) {
    nabto_tunnel_state_t status;
    st = nabtoTunnelInfo(tunnel, NTI_STATUS, sizeof(status), &status);
    // ignore error handling / status updates for simplicity
    sleep(1);
}

// if status is ok, use the TCP tunnel

nabtoTunnelClose(tunnel);
nabtoCloseSession(session);
nabtoShutdown();
```

详情请参阅[TEN025] 关于如何与客户端相互作用.