# 1．Visual Studio 2015 安装

建议Windows配置推荐：2vCPUs,8G内存

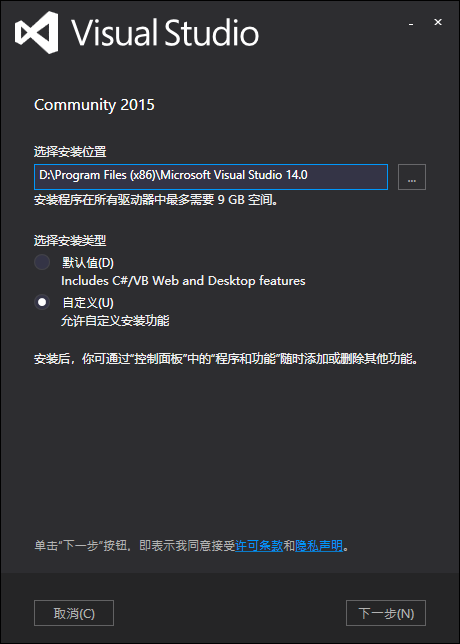


1）下载地址

<https://go.microsoft.com/fwlink/?LinkId=615448> （common）（推荐）

2）安装过程

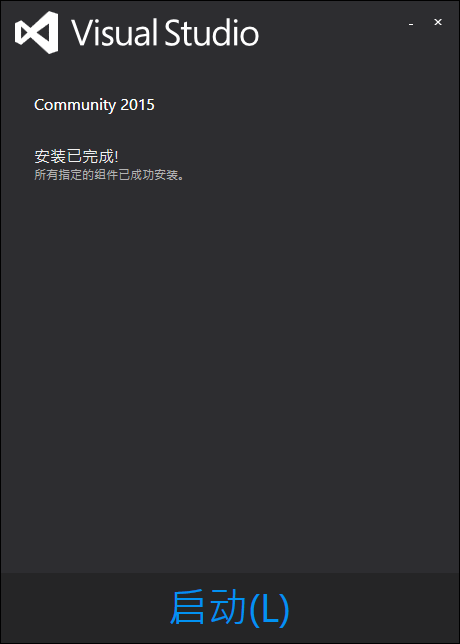
打开程序选择安装路径，这里选择自定义安装。



根据自己的需求选择安装内容



安装完成。



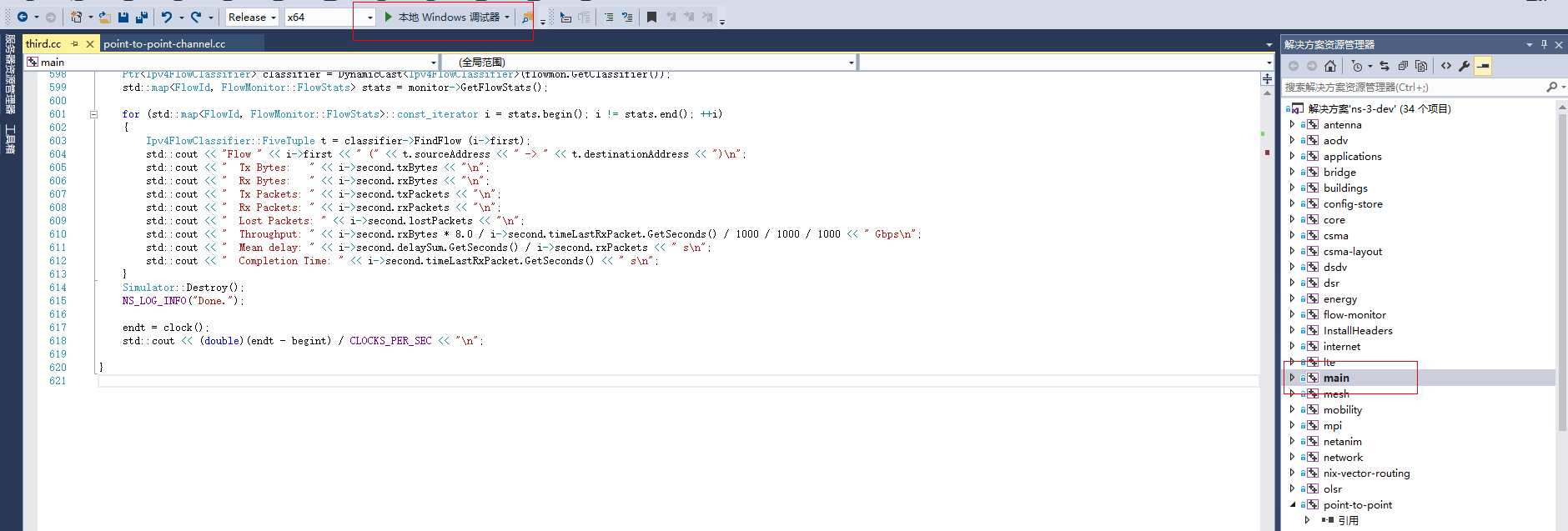
**注意事项：**如果安装过程在update或正在配置系统界面中无响应，可直接重启电脑，继续安装。

# 2．代码下载

1） 安装git代码管理工具。参考<https://blog.csdn.net/lyorz/article/details/127424055>配置git 和github

2） git clone [git@github.com:bobzhuyb/ns3-rdma.git](mailto:git@github.com:bobzhuyb/ns3-rdma.git)

3） 用Visual Studio 2015 打开windows/ns-3-dev/ns-3-dev.sln。



4）点击本地windows调试器即可编译完成，生成main.exe文件即可，**不用关注是否运行，有时候会打开一个运行窗口，这时关闭即可**。

5） 运行该代码仓提供的基本示例。

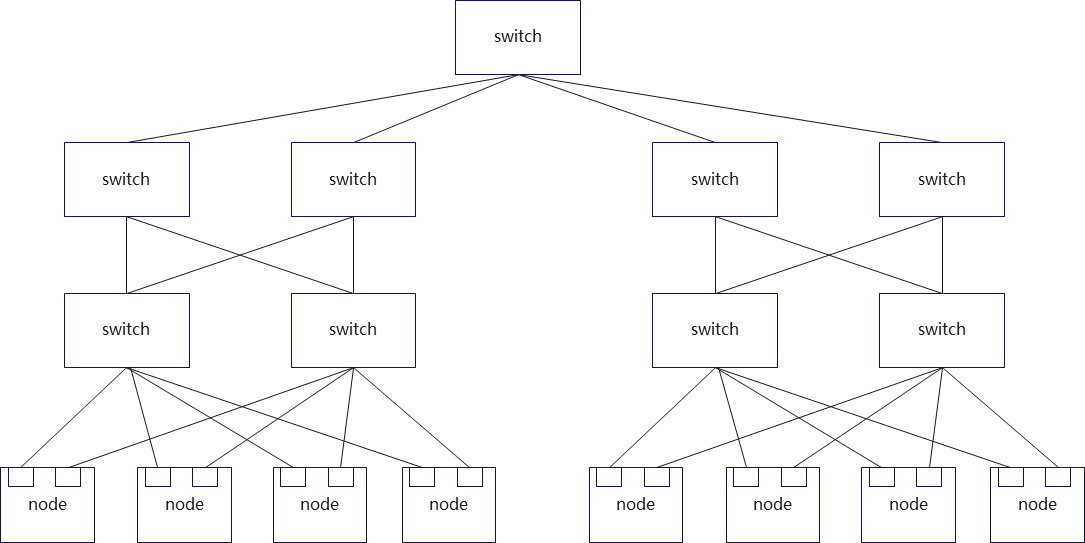
cd ns3-rdma\windows\ns-3-dev\x64\Release

执行./main.exe mix/config.txt

# 3. 补充说明

1）根据提供的拓扑结构图（附件1）搭建仿真拓扑，生成topology.txt ，替换原有目录下ns3-rdma\windows\ns-3-dev\x64\Release\mix\topology.txt。

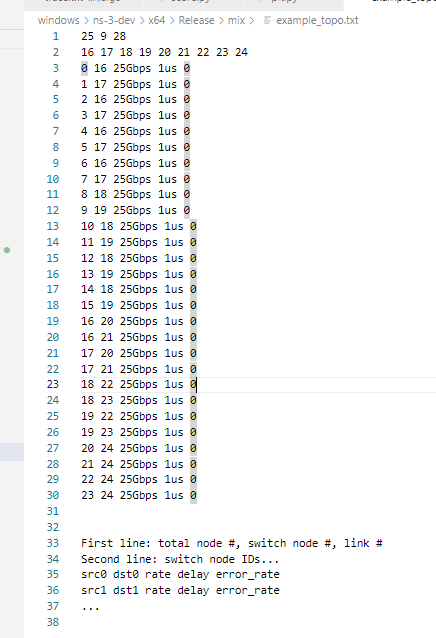
生成topology.txt方式可参考如下示例：



图一：三层胖树网络拓扑

以图一为例进行拓扑搭建，默认每一条链接物理带宽为25Gbps,延迟为1us。

对图一的节点进行编号（编号可以任意）后，生成新的topology.txt。举例实现：从左到右，从下至上开始编号。对于非交换机的节点，仿真中最小粒度按照**节点端口**来编号。按照哪两节点互连则可以生成topology.txt，如下所示



图二 topology.txt样例

可以看到共有 8（node）\* 2 + 4(L1) + 4 (L2) + 1(L3) = 25 个节点，其中交换机节点数为9， 总link数为 8 \* 2 + 2\* 2 + 2\* 2 + 4\*1 =28 第一行生成为25 9 28。交换的序列号为16~24。

第一层和第二层连接关系为：dst = 16(node占用数) + src / 8 \* 2 + src % 2 （src范围0-15）

第二层和第三层连接关系为：dst = 16 + 4（L1交换机数量） + src / 2 \* 2 + src % 2 （scr范围0-3）

第三层和第四层连接关系为：dst = 24

根据关系即可以生成topology.txt。

2）根据提供的附录2文件，生成flow.txt，替换原有目录下ns3-rdma\windows\ns-3-dev\x64\Release\mix\flow.txt。

flow.txt每一行含义为：原始节点端口，目标节点端口，包优先级（不用关心），包数量，开始时间，最大结束时间

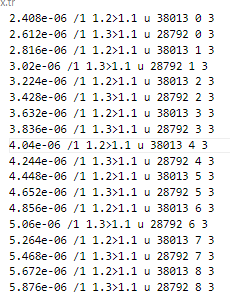
3）生成trace.txt文件，替换原有目录下的ns3-rdma\windows\ns-3-dev\x64\Release\mix\trace.txt，在trace.txt中填写所有存储节点，用以获取所有存储节点的仿真结果。

4）样例ns3-rdma\examples\tutorial\third.cc 文件中main()为主函数入口，在这里可以进行相关的自定义操作，在这里可以使用monitor对每一条的相关信息，比如吞吐，完成时间进行分析。

third.cc为作者提供的一个实现RDMA网络协议的一个用例。用例包括从配置文件中读取网络参数、创建节点、创建网络设备和链路、安装协议栈、分配ip地址、设置全局路由、创建应用程序**。参赛选手可参考官方用例实现**。

5）结果获取，默认输出文件mix.tr。

每一列分别为 时间，当前节点，源ip> 目的ip ，协议Udp，端口号，包序列号，包优先级。



选手不提交mix.tr（上传代码时手动删除该文件）,使用组办方提供的gen\_result.py工具，基于mix.tr文件生成result.txt，放在\ns3-rdma\windows\ns-3-dev\x64\Release\mix目录下，随源码一起提交。

6）选手可选择设计新的拥塞算法提升网络吞吐能力。拥塞控制算法实现，可以参考代码仓的两个分支master和timely（该方法需掌握数据中心中拥塞算法相关原理）。

7）选手可选择设计新的负载均衡算法提升网络吞吐能力。例如，考虑多条流在交换机中出现哈希碰撞、极化等问题，导致不同的流的完成时间长短不一，最终整体文件的完成时间大大增加，如果采取动态更改流的路径，则可以获取更快的文件传输时间。修改流的路径可以采取修改五元组中的端口号实现。

8） hash算法修改

代码位置：Ipv4GlobalRouting::RouteOutput (Ptr<Packet> p, const Ipv4Header &header, Ptr<NetDevice> oif, Socket::SocketErrno &sockerr)。

如果每一层交换机的哈希算法一致则会出现哈希极化，可以考虑在哈希算法中加入不同的偏移。

或者可以在主函数中实现自己的类比如class CustomHashRouting : public Ipv4RoutingProtocol继承该Ipv4RoutingProtocol函数，复写里面的RouteOutput函数，实现不同的hash算法。

9）config配置文件

注意**APP\_START\_TIME**为数据开始传输时间，应设置为0，**SIMULATOR\_STOP\_TIME** 为仿真停止时间。

10) 选手可在小规模场景下调试相关算法以及相关参数，最后再运行整个模型。

11) MTU修改位置

udp-client.cc：line76

**注意事项：**代码提交需要 git commit。可用git log --oneline 查看每一次提交记录。请**阅读代码中的README.md文件和代码仓中issue。**

# 4. 结果提交

提交内容包括：

1. 可编译运行的源代码；
2. result.txt；
3. 方案设计文档.docx（参考附件模板），放在ns3-rdma\下即可；

文件打包成xxxx.zip。删除Visual Studio运行生成的相关过程文件：

