GDB 调试说明

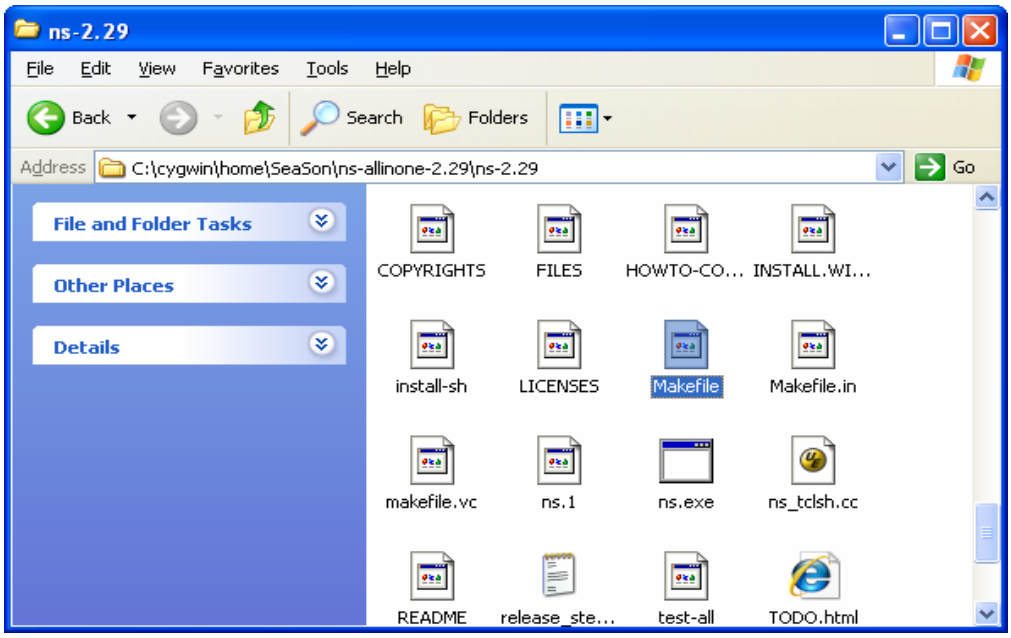
---- SeaSon from DB-LAB of HIT

安装篇

下面介绍的方法是以在 cygwin 中安装 gdb 为例说明。

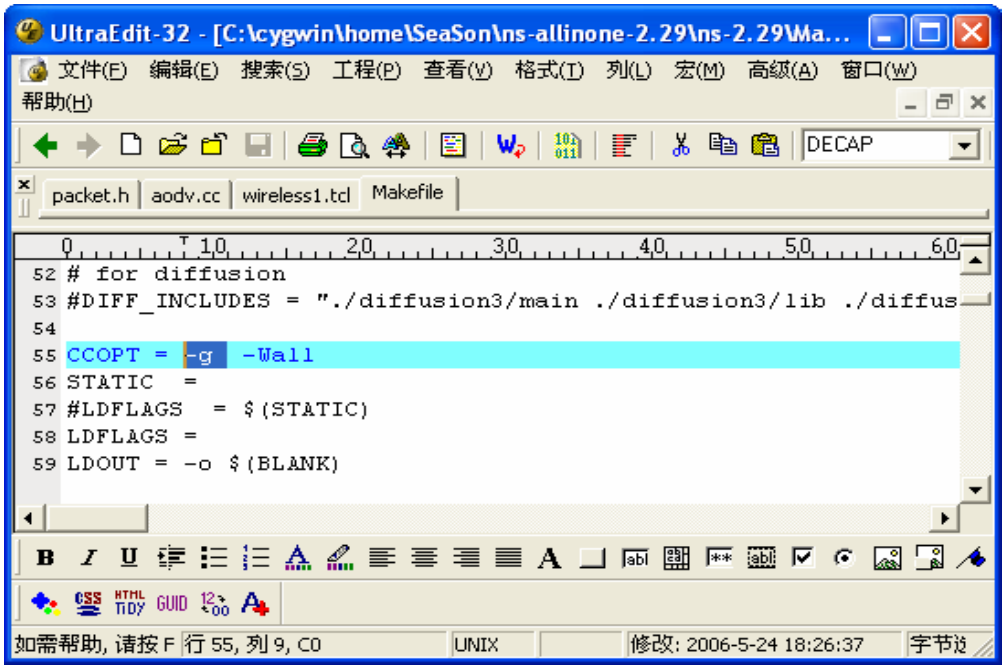
1. 重新运行 cygwin 的 setup 文件，选择界面中的 keep（必须!!! 否则会死的很惨的），从列表中选择 gdb 组件，然后选择安装即可。
2. 修改 Makefile，添加调试信息

   修改 Makefile（注意：修改的是没有任何后缀的文件，不是 Makefile.vc 或者 Makefile.in）

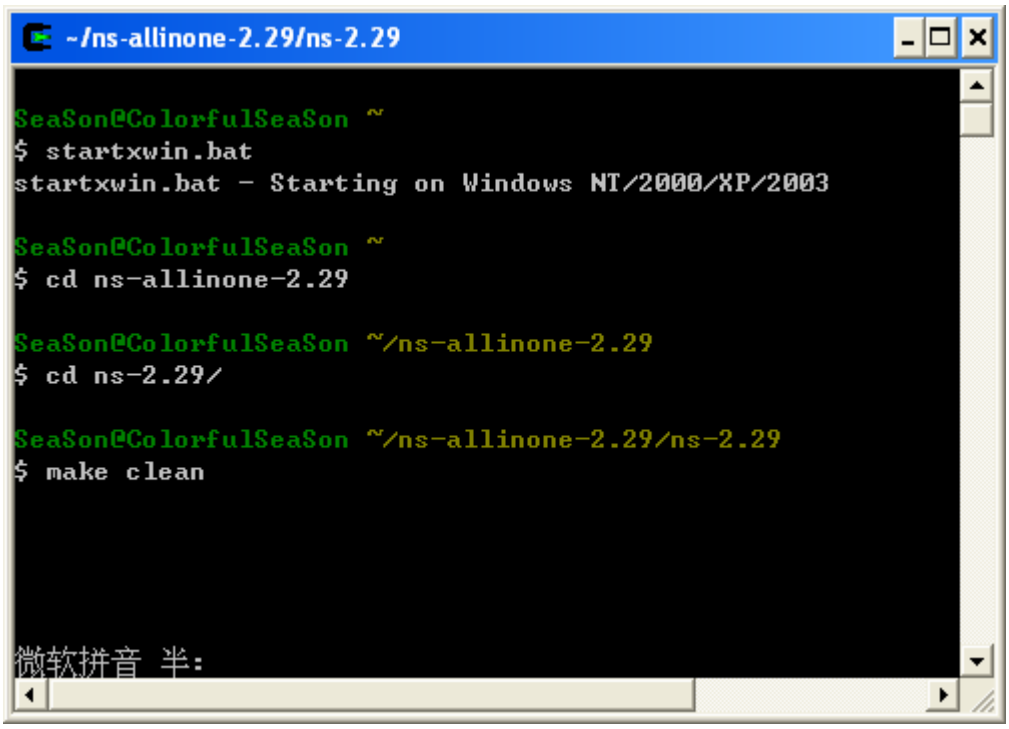


在下面位置添加-g

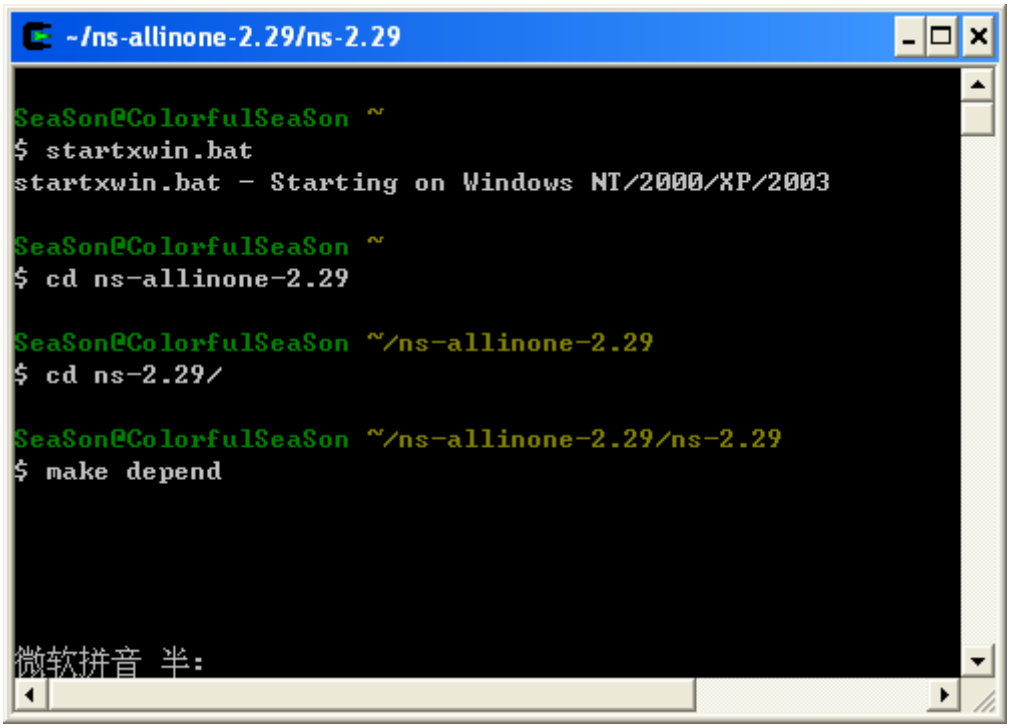
CCOPT = -g //这后面可能还有其他参数，保留即可



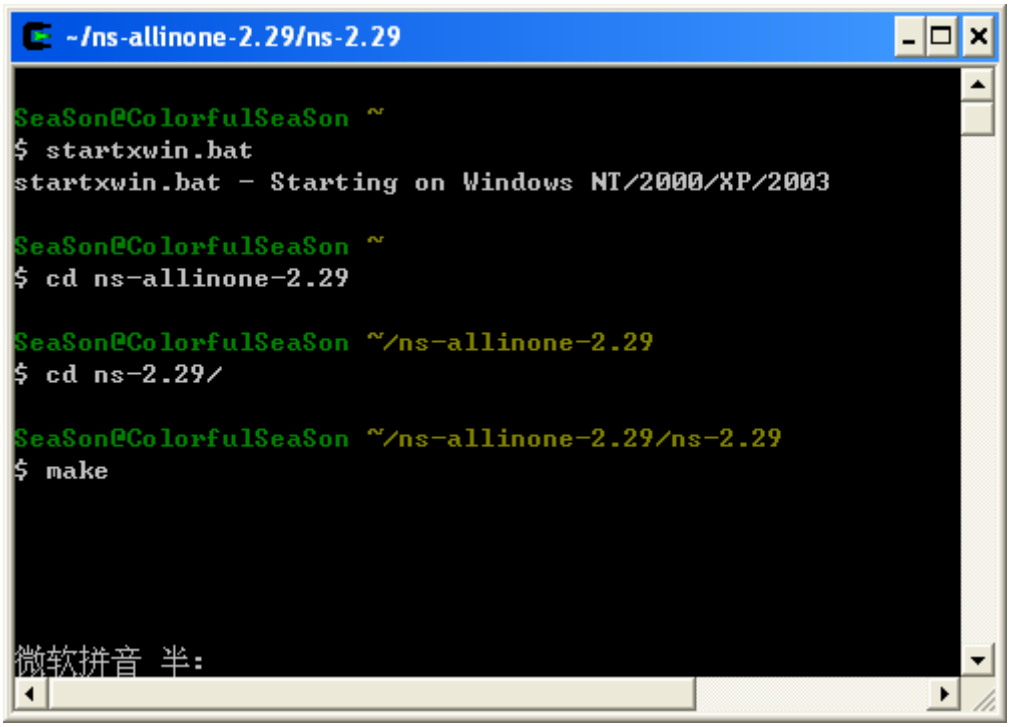
3. 接着需要重新编译 NS2
进入 ns-2.2* 目录下面执行
make clean



接着 make depend # 执行这个命令过程中如出错，不用管！



然后 make

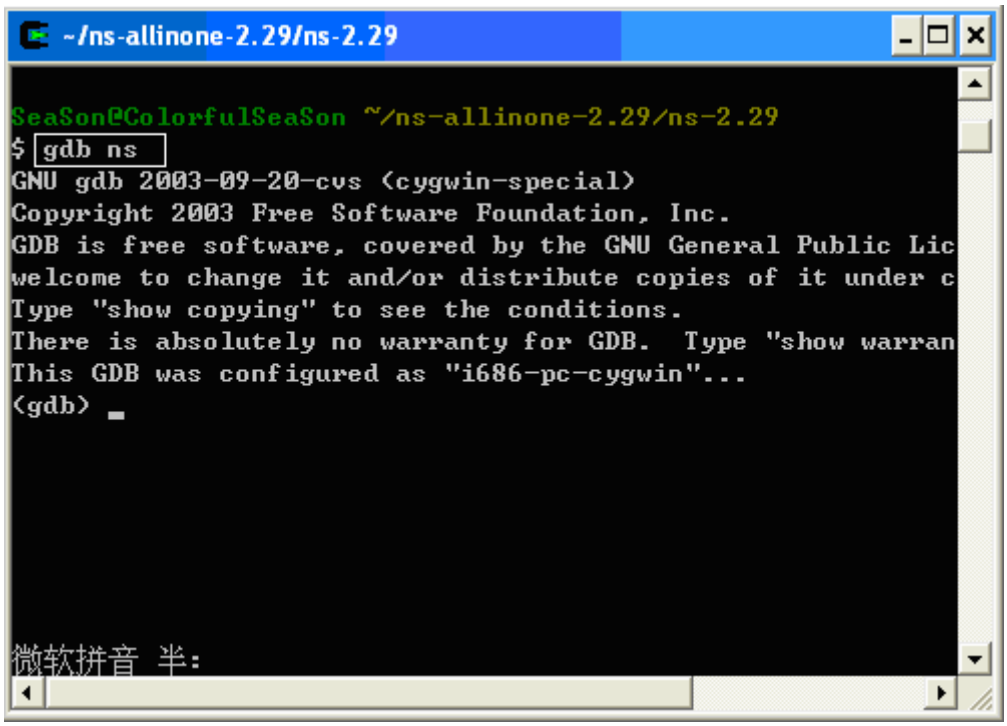


如果上面过程中没有错误，则恭喜你安装成功:）

如果要使用图形界面，则安装过程中需要安装 tcltk 库，然后对于 2003(以前的有些版本页可以)以后的默认都回安装图形界面调试工具 insight。后面的说明都是基于命令行界面的，适用比较稳定、方便。

常用命令

1. 进入 gdb 调试状态，在 Cygwin 窗口或者 Xwin 窗口输入命令 **gdb ns**，如下图所示：



2.设置断点

命令格式 <gdb> b file.cc:112，下图即在 aodv.cc 的第 112 行设置一个断点

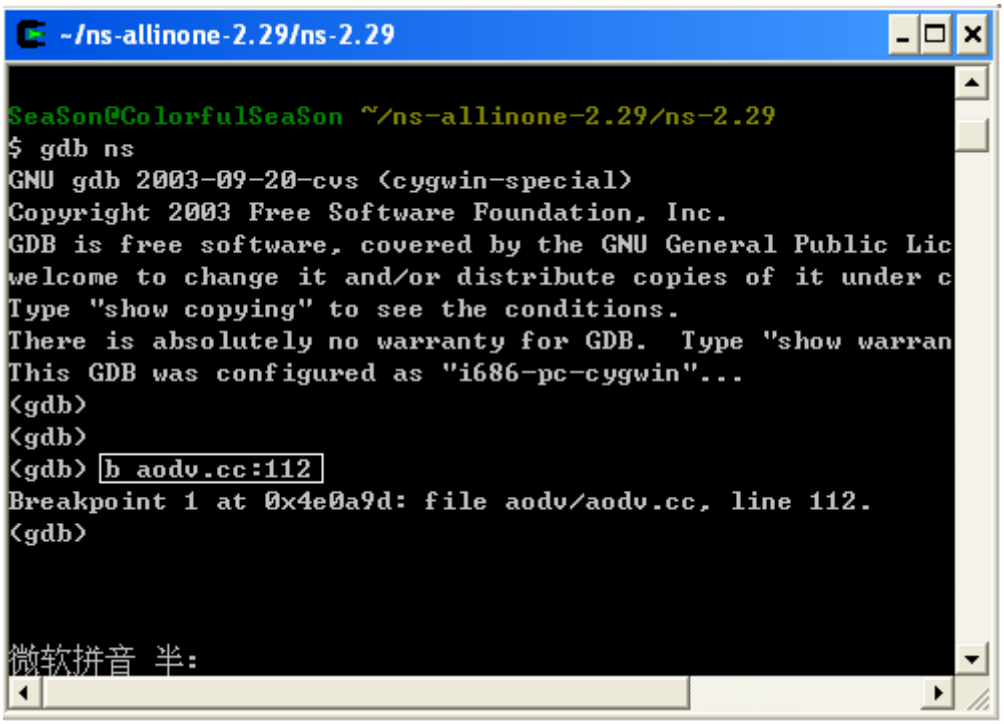
其中<gdb> 为命令提示符

b 设置端点命令

file.cc 调试的文件，Ns2 中任何一个 C++文件都可以进行调试

“:”行数指示符

112 为行号



当然设置断点的格式还有其他，具体参见手册。

3. 删除断点

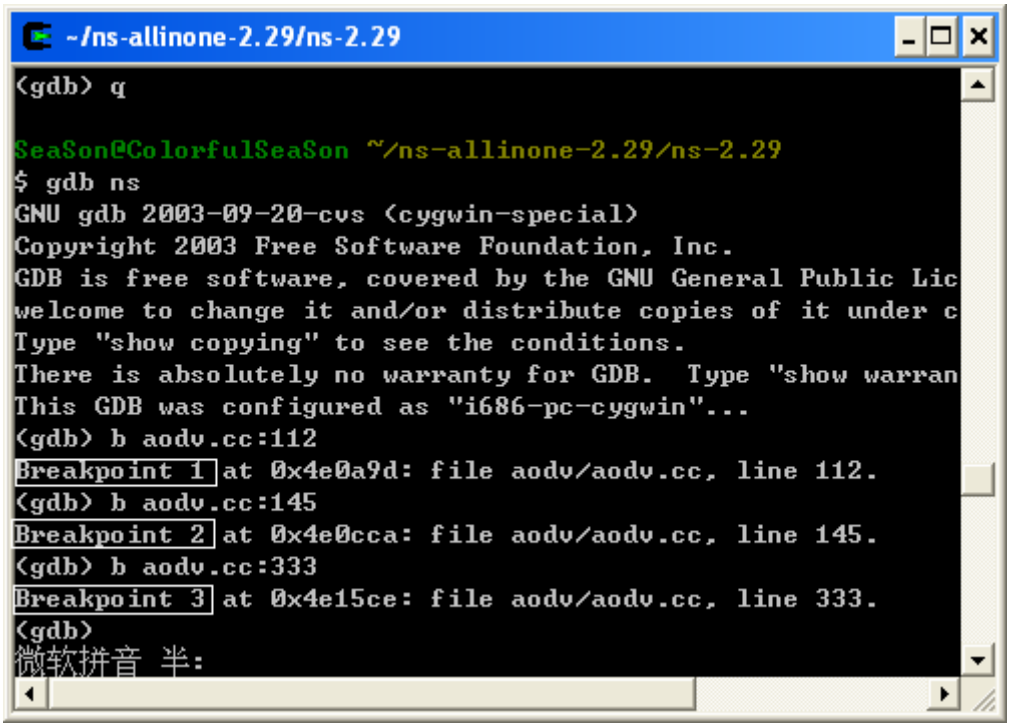
命令格式：<gdb>d b 2

其中 **d** 为 delete

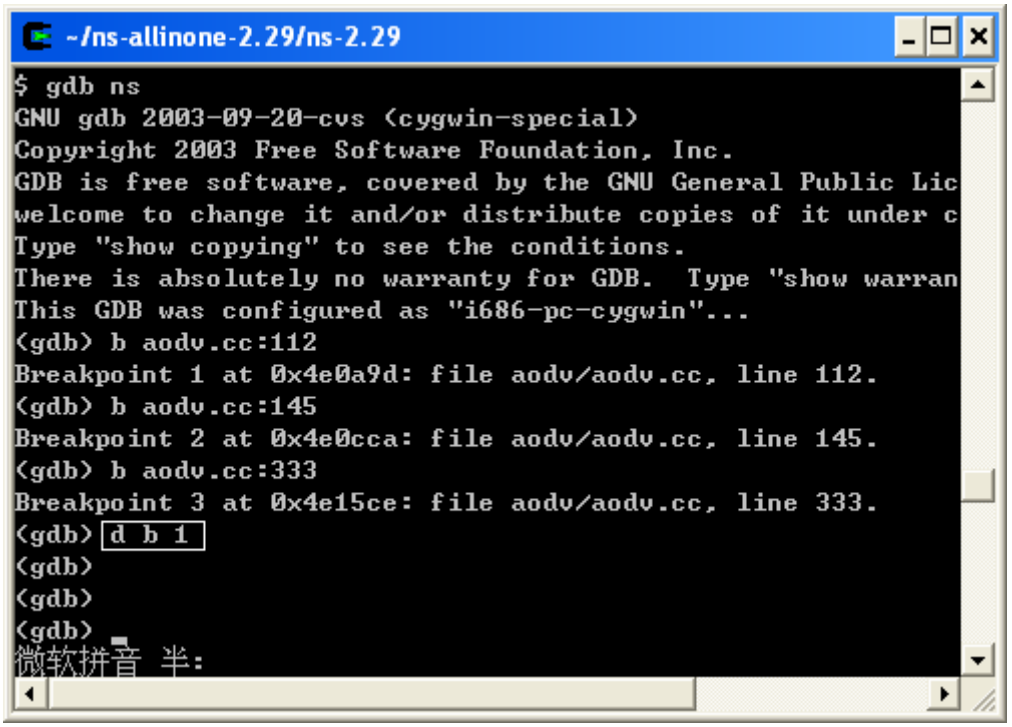
b 为 breakpoints

2 为断点的编号

利用 2.中的方法继续创建断点 2、3。



利用命令 d b 1 即删除第一个断点（Breakpoint 1 at 0x4e0a9d: file aodv/aodv.cc, line 112.），如下图所示



4.运行脚本

命令格式：r scrip.tcl

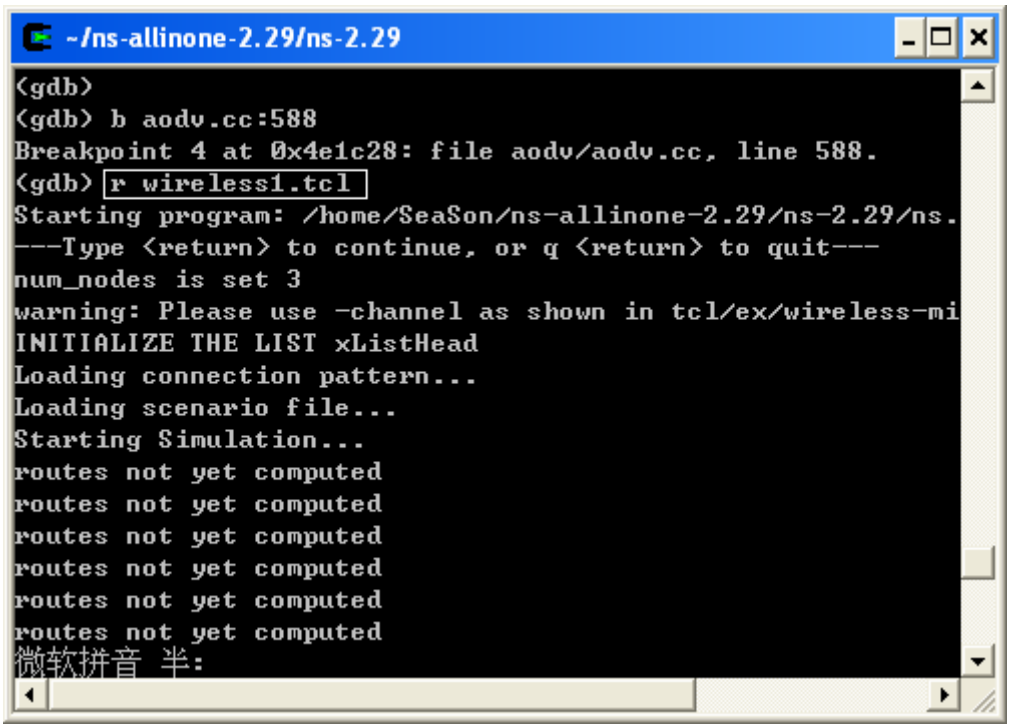
其中 **r** 为命令

scrip.tcl 为脚本

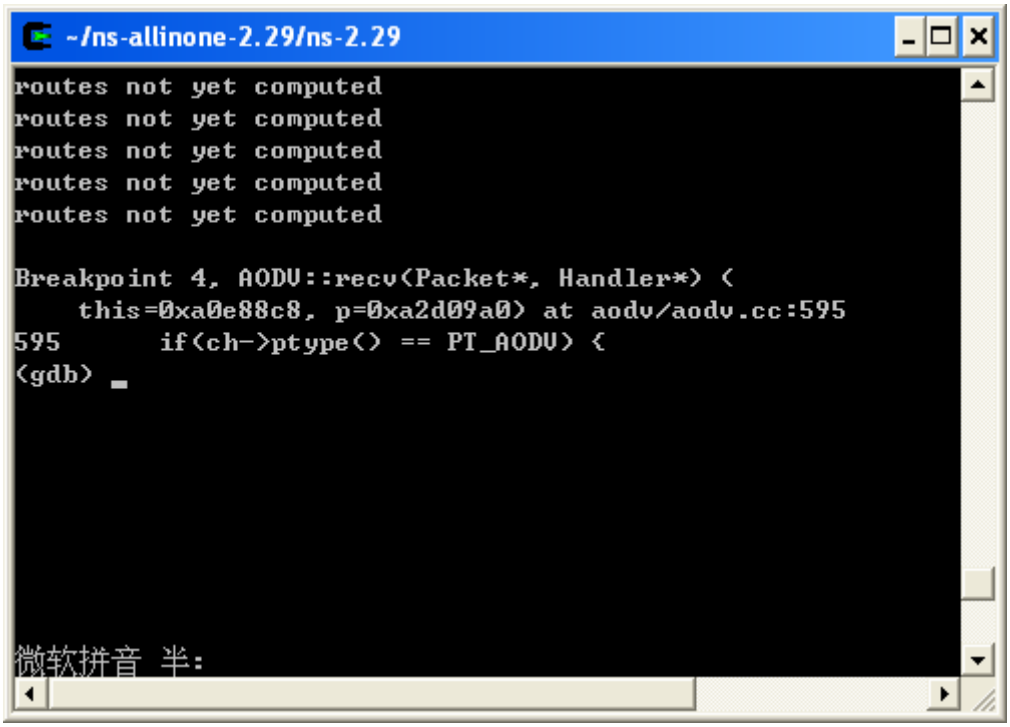
以 ns 自带的 wireless1.tcl 为例，我们首先在 aodv.cc 的 recv 函数开始设置端点如下图所示：

```
void
AODV::recv(Packet *p, Handler*) {
struct hdr_cmn *ch = HDR_CMN(p);
struct hdr_ip *ih = HDR_IP(p);
//add by season
//struct hdr_aodv *ah = HDR_AODV(p);

 assert(initialized());
 //assert(p->incoming == 0);
 // XXXXX NOTE: use of incoming flag has been depracated; In order to track dir
                设置断点在此
 if(ch->ptype() == PT_AODV) {
   ih->ttl_ -= 1;
   recvAODV(p);
   return;
 }


 /*
  * Must be a packet I'm originating...
```

运行测试脚本 wireless.tcl，如下图所示：



然后程序在断点位置停止，如下图所示：



5. 显示变量或函数值

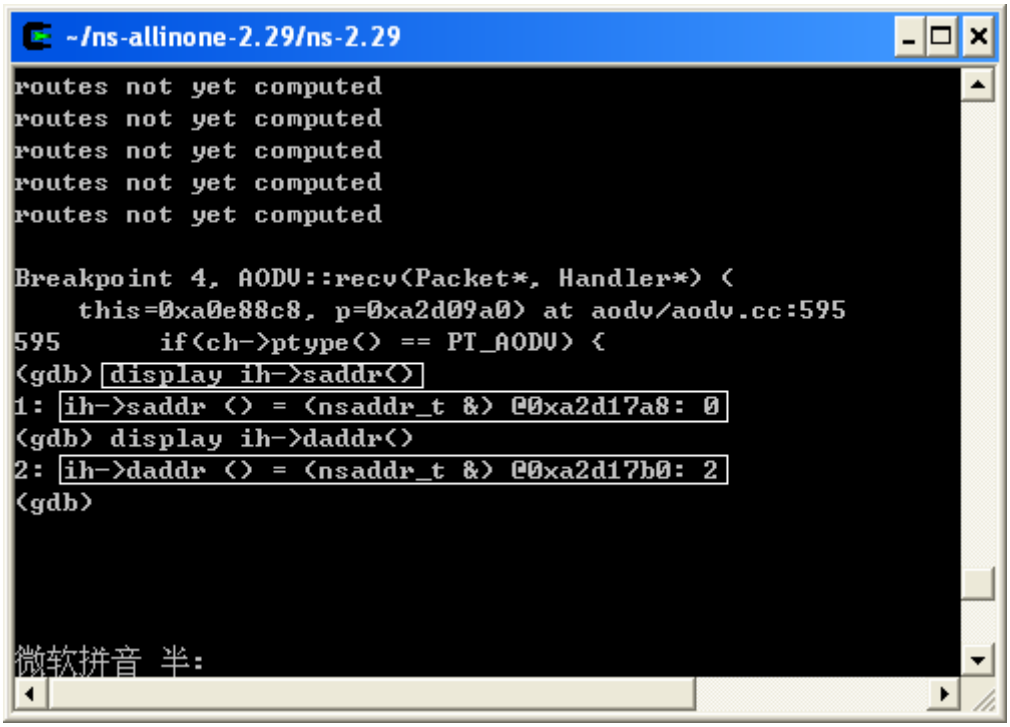
命令格式：display var

其中 **var** 可以为变量名或者函数名

在 recv（）函数的端点处我们想要查看数据包的源地址，即利用下列命令

display ih->saddr()

结果如下图所示，即数据包源地址为 0,目的地址为 2.



6. 删除变量或函数值显示

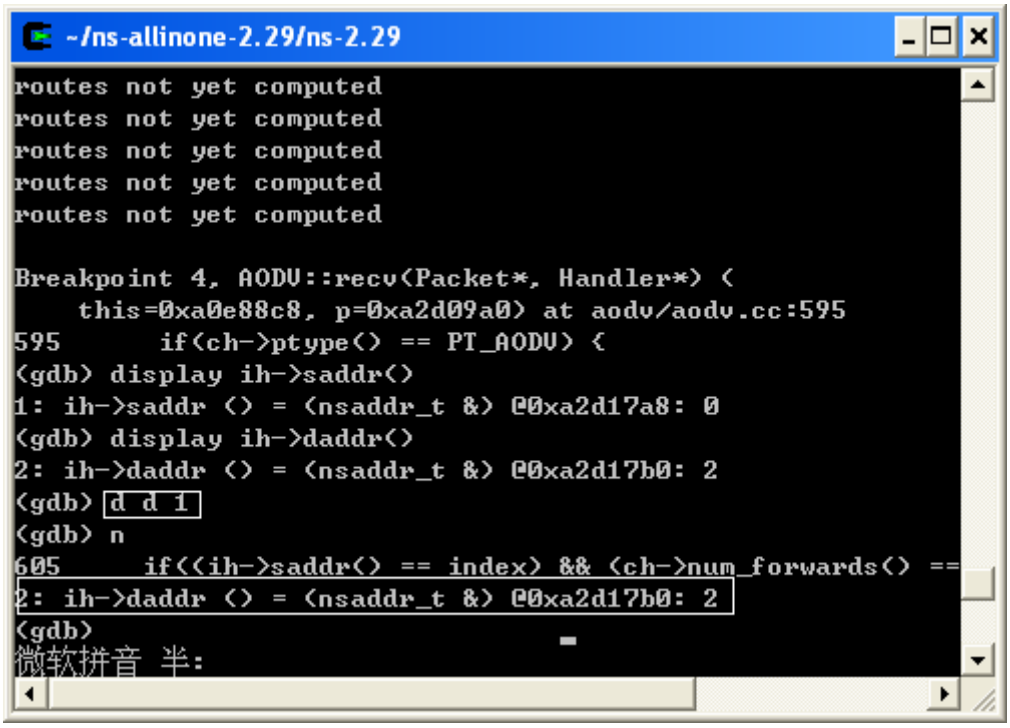
命令格式：d d 1

其中 d---delete

d--- display

1--- 变量编号

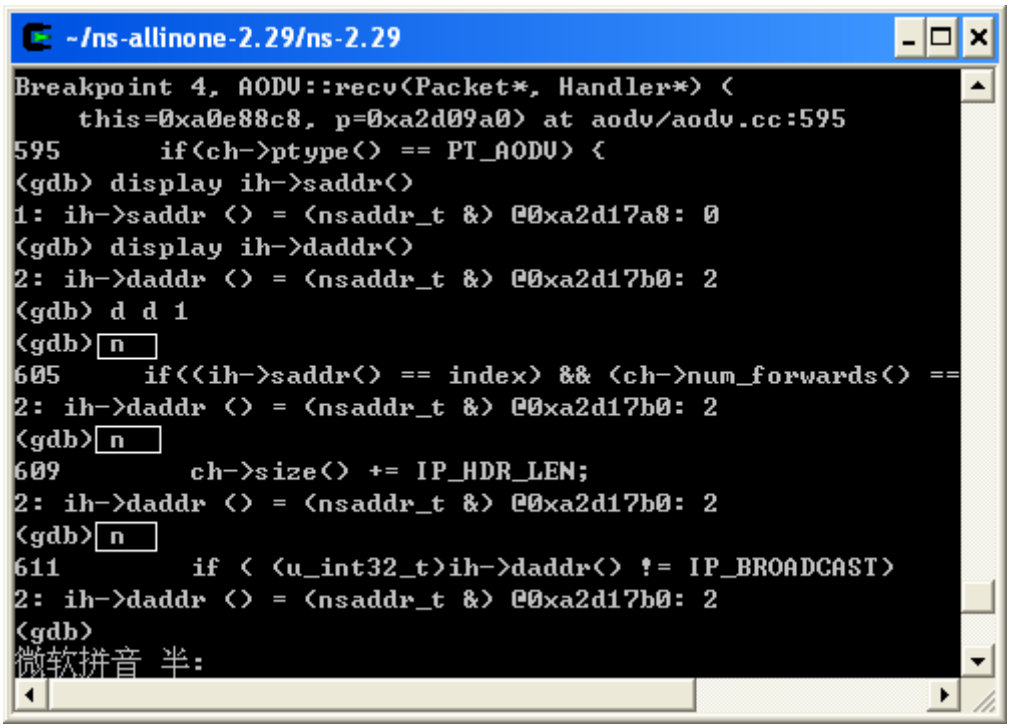
使用命令”d d 1”即删除第一个变量显示。这是在单步调试的时候将不再显示 ih->saddr(),否则如果不删除将显示所有的。



7. 单步执行

命令格式：n

即 next



8. 单步跳入

命令格式：s

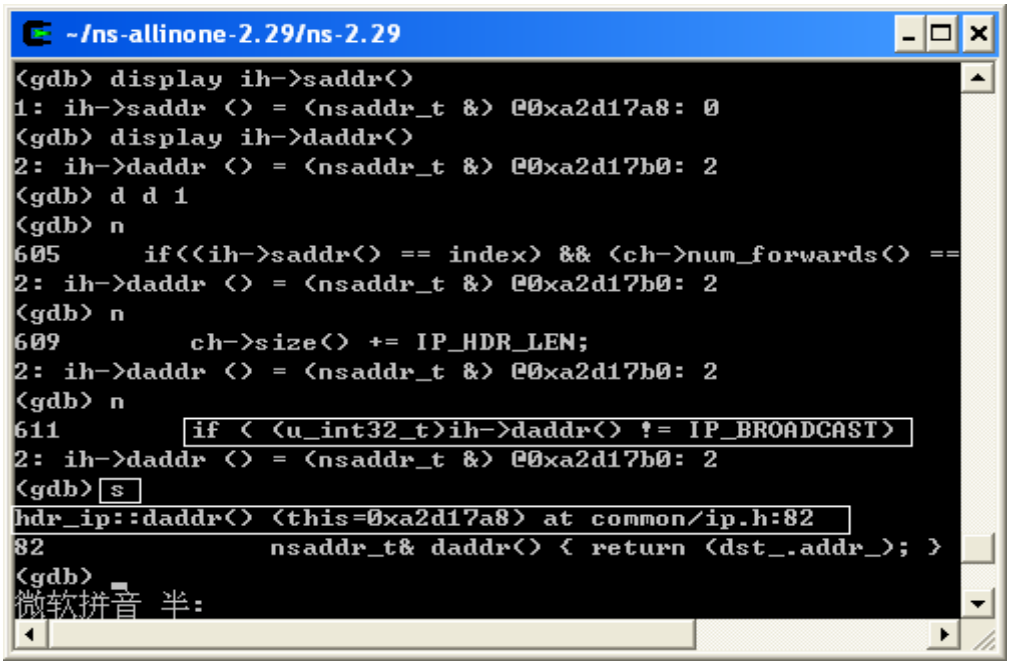
即 step

如下图所示，在执行到下面代码的时候，执行 s 命令

```
00611:    if ( (u_int32_t)ih->daddr() != IP_BROADCAST)
```

则跳转到 daddr()函数

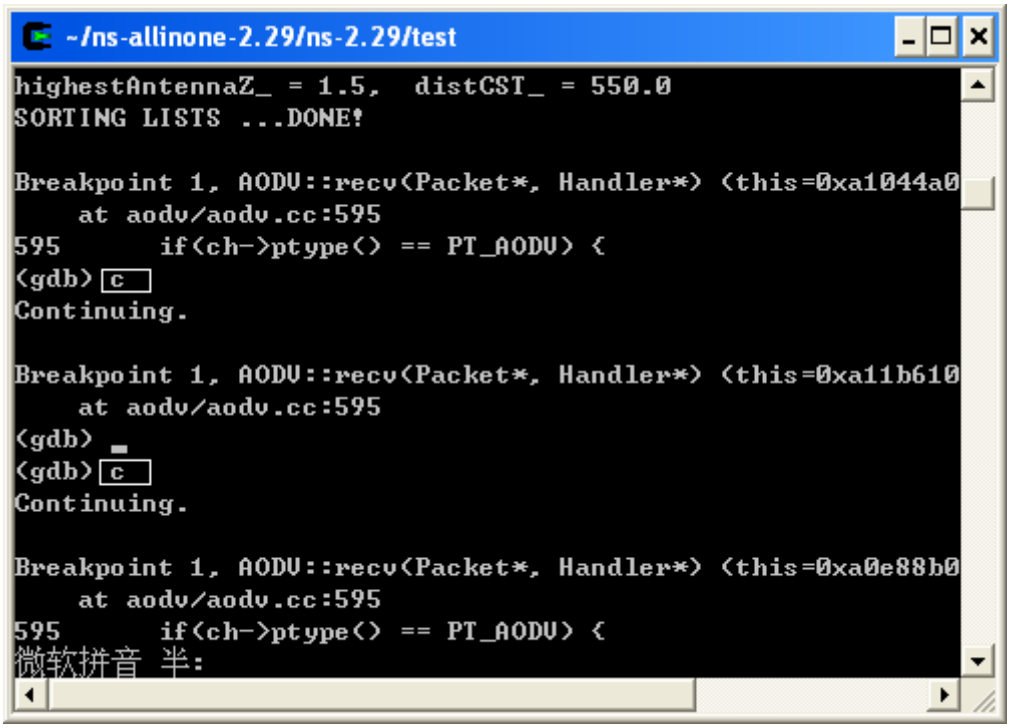
```
hdr_ip::daddr() (this=0xa2d17a8) at common/ip.h:82
```



9.循环执行

命令格式：c

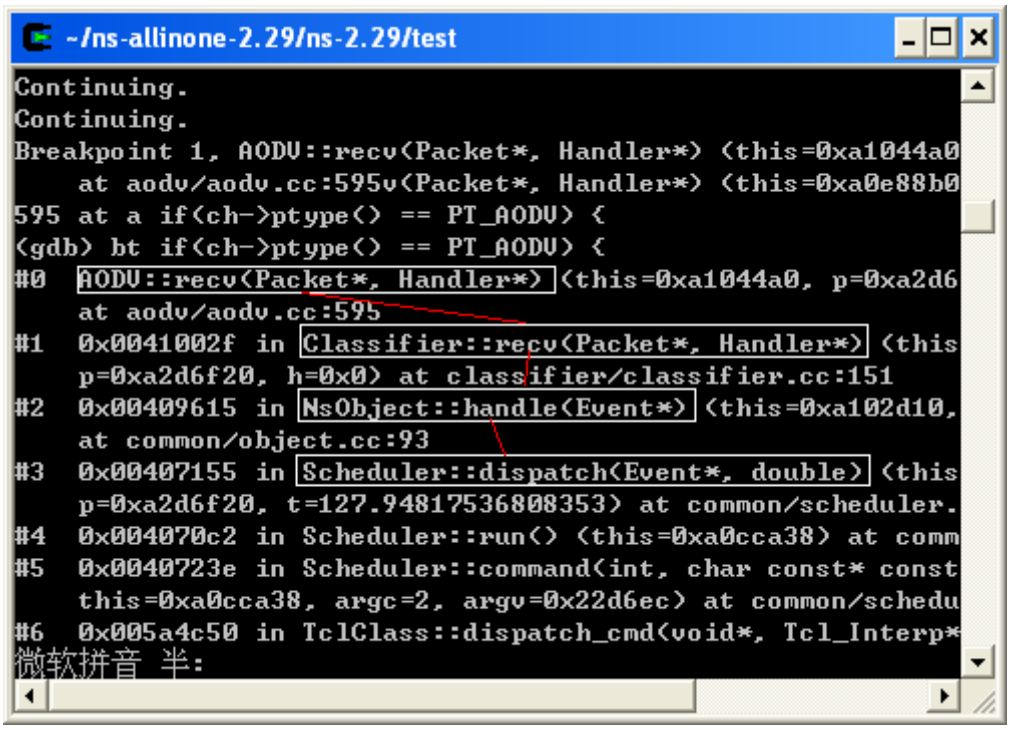
即 continue



10.下面介绍的命令是非常有用的，列出运行栈的内容。

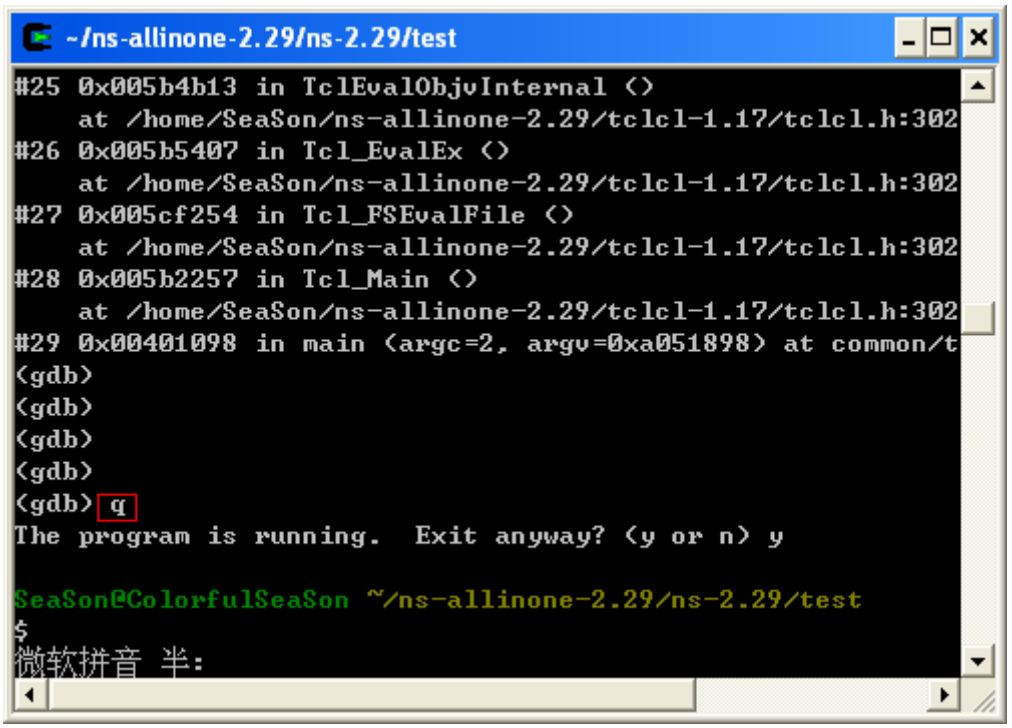
命令格式 bt

主要针对的是如果你遇到 segment fault 的时候，你可以用以上命令，确定在那个为止出问题，以及函数之间的调用关系，后面会具体说明的。



11. 退出调试

命令格式 q



其他的相关命令可以参看手册，不过调试 NS2 以上的命令基本上已经够用了。

调试示例

示例一、调试 segmentation fault

为了具有普遍性，我特意在 aodv.cc 添加了一个 segmentation fault

添加方法：

1. 打开 aodv.cc,添加头文件
   #include "mac-802_11.h"
   如下图所示：

```
#include <aodv/aodv.h>
#include <aodv/aodv_packet.h>
#include <random.h>
#include <cmu-trace.h>
//#include <energy-model.h>
//add by season
#include "../common/mobilenode.h"

//add by season 2006-7-8
#include "mac-802_11.h"
#define max(a,b)        ( (a) > (b) ? (a) : (b) )
#define CURRENT_TIME    Scheduler::instance().clock()
```

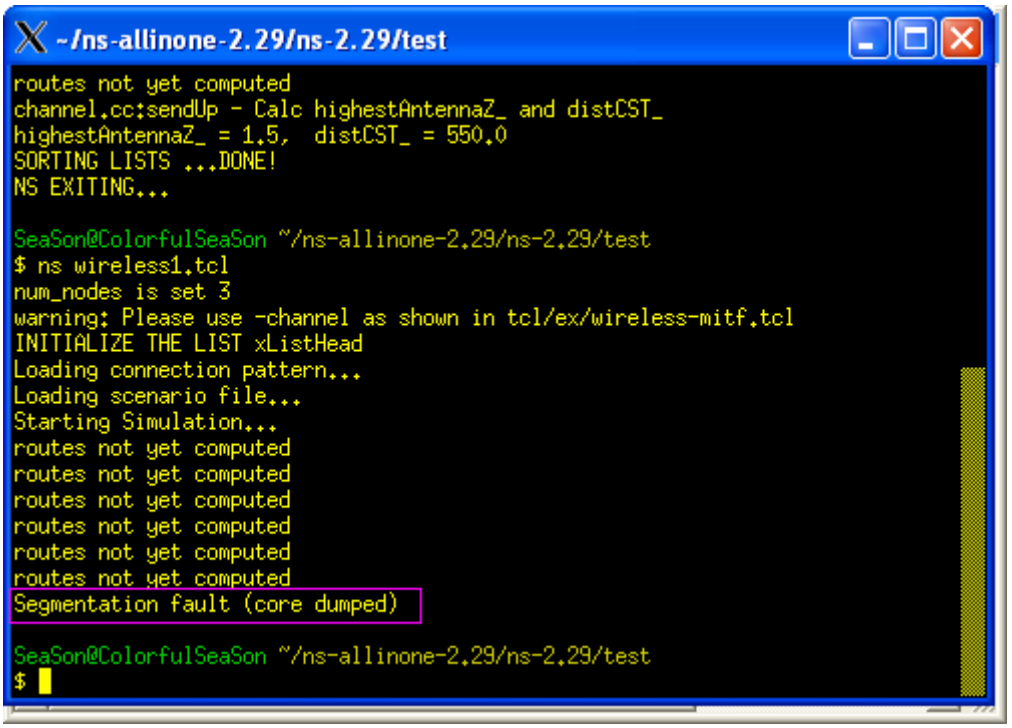
2.在 recv()函数中开头添加下面的代码引入 **segmentation fault** 错误：

```
00585:
00586: void
00587: AODV::recv(Packet *p, Handler*) {
00588: struct hdr_cmn *ch = HDR_CMN(p);
00589: struct hdr_ip *ih = HDR_IP(p);
00590:
00591: //add by season
00592: struct hdr_mac802_11 *mh;
00593: int    test = mh->dh_duration;      这句话产生错误
00594:
00595:
```

3. 重新编译 NS2，是错误生效。在 ns-2.2*/目录下输入 make

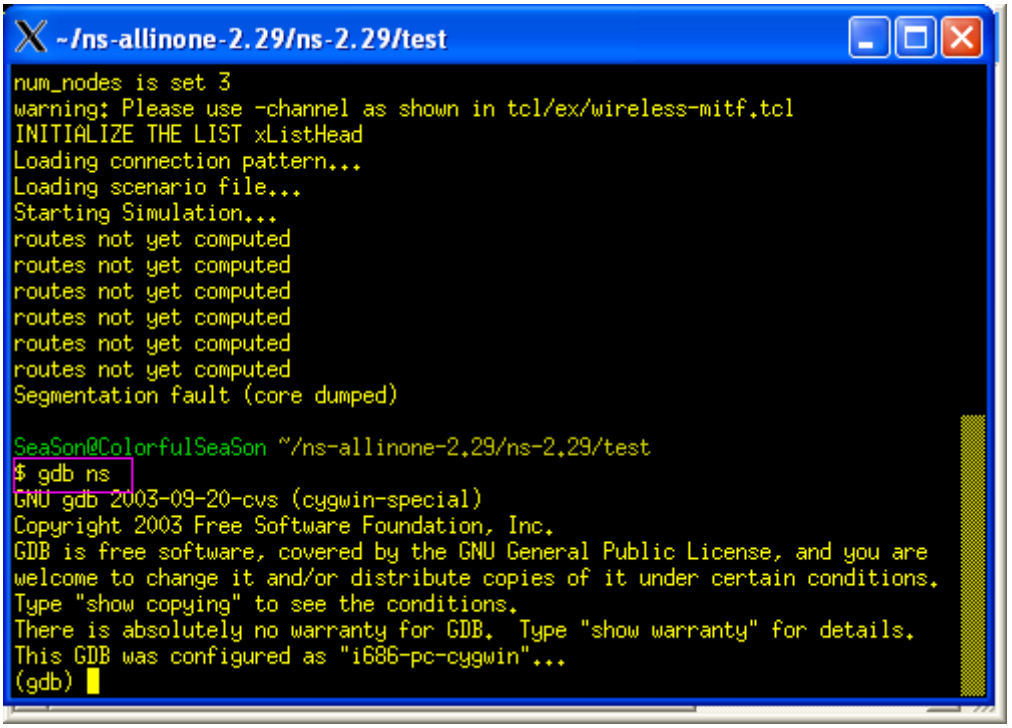
```
~/ns-allinone-2.29/ns-2.29
   expressions
aodv/aodv.cc: In member function `void AODV::send(Packet*,
aodv/aodv.cc:1461: warning: comparison between signed and u
   expressions
aodv/aodv.cc:1489:2: warning: no newline at end of file
make: *** [aodv/aodv.o] Error 1

SeaSon@ColorfulSeaSon ~/ns-allinone-2.29/ns-2.29
$ make
g++ -c -g  -Wall  -DTCP_DELAY_BIND_ALL -DNO_TK -DTCLCL_CLAS
USE_SHM -DHAVE_LIBTCLCL -DHAVE_TCLCL_H -DHAVE_LIBOTCL1_11 -
IBTK8_4 -DHAVE_TK_H -DHAVE_LIBTCL8_4 -DHAVE_TCL_H  -DHAVE_C
N -DSMAC_NO_SYNC -DCPP_NAMESPACE=std -DUSE_SINGLE_ADDRESS_S
I/home/SeaSon/ns-allinone-2.29/tclcl-1.17 -I/home/SeaSon/ns
.11 -I/home/SeaSon/ns-allinone-2.29/include -I/home/SeaSon/
ude -I/usr/include/pcap -I./tcp -I./sctp -I./common -I./lin
I./apps -I./mac -I./mobile -I./trace -I./routing -I./tools
ast -I./diffusion3/lib/main -I./diffusion3/lib -I./diffusio
微软拼音 半:
```

4. 在 xwin 窗口中运行脚本，路由协议必须为 AODV,以 wireless1.tcl 为例。运行后产生，下列错误：

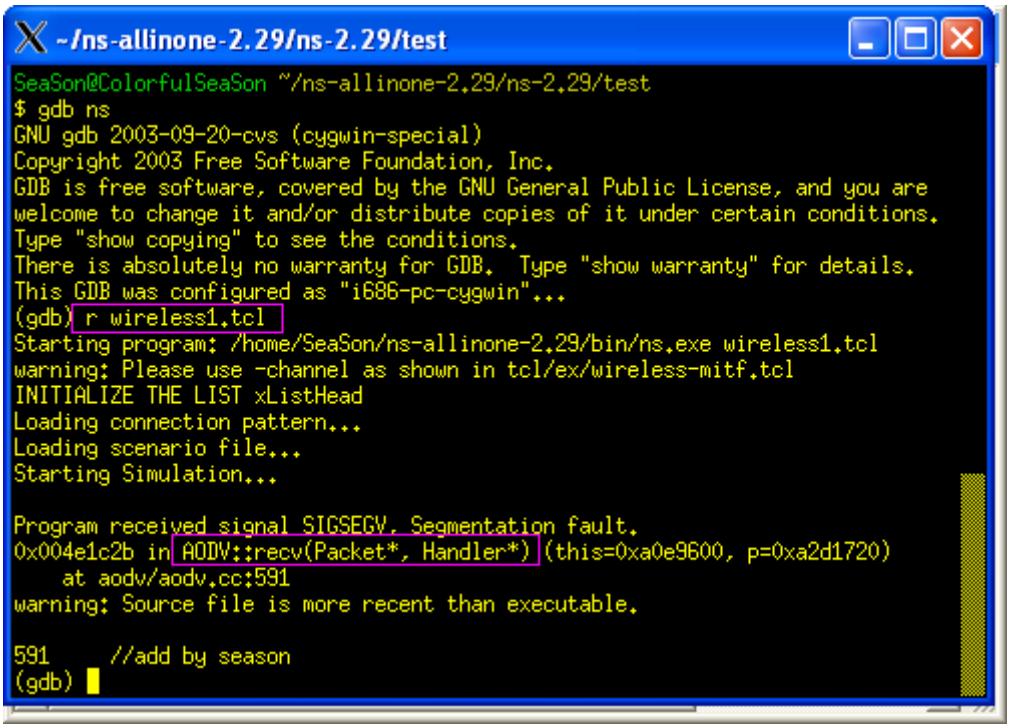


5. 利用 gdb 确定错误的位置

1) (在 xwin 窗口)输入 gdb ns，进入调试状态。

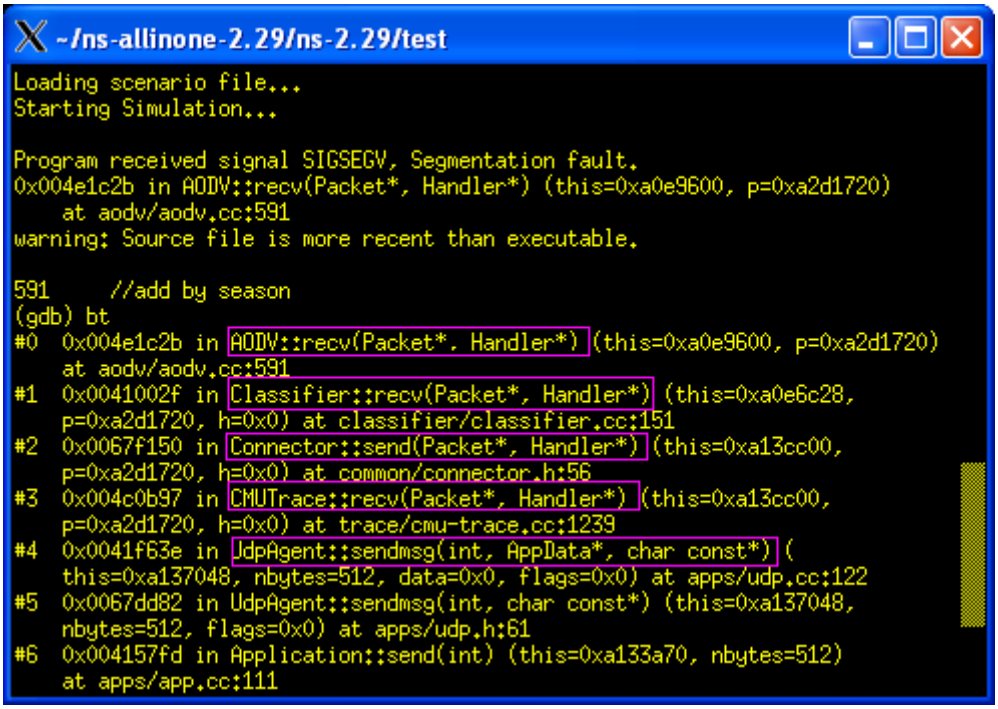


2) 在 gdb 中运行脚本，便会在出错的地方停止程序，如下图所示：



3）如果你想进一步知道在什么地方调用这个函数出错的，可以使用 bt 命令，具体结果如下图所示。

当然有些错误没这么明显找到的，可能显示的错误是系统文件，这时候你就需要仔细的分析运行栈中的内容，一般的方法是观察运行栈中的函数调用，看看哪一个是你修改的函数，然后将断点设在相应位置，然后重新开始调试，一般都可以找出错误的原因，这就需要你足够的耐性了。



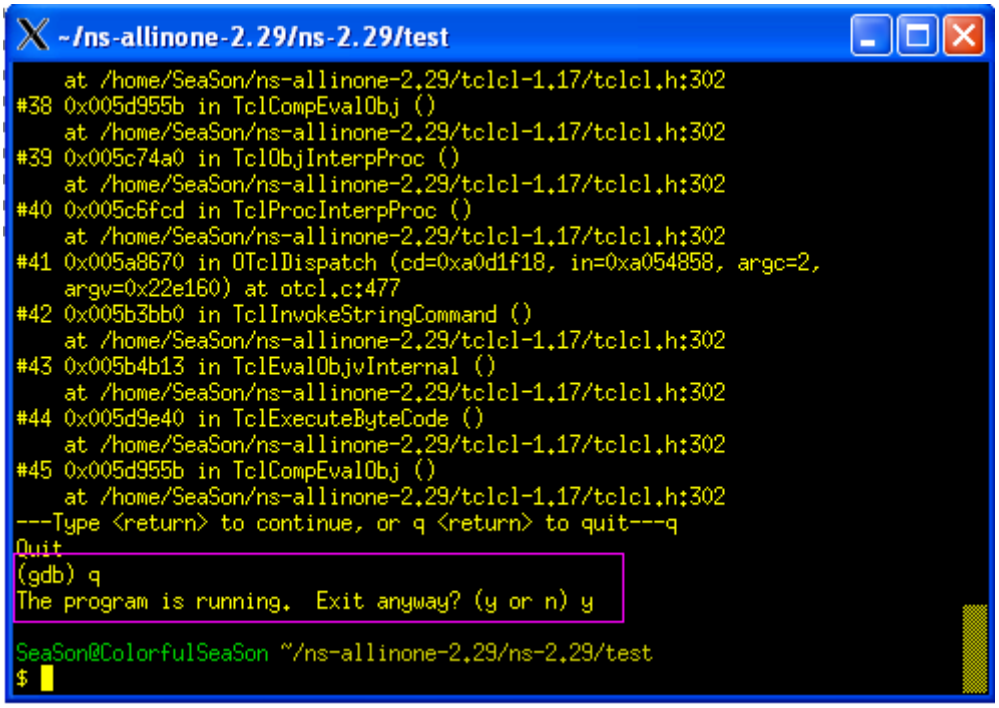
4) 通过上面的步骤我们就确定了错误的位置，然后将最开始我们修改的去掉就 OK 了。

注释掉我们引入的错误：

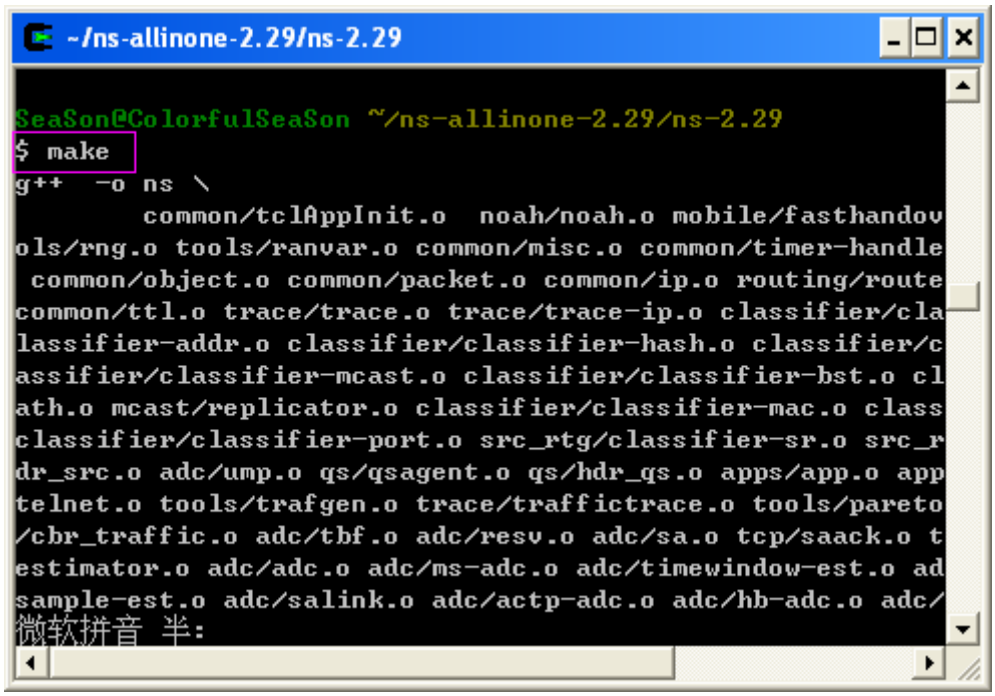
```
00586: void
00587: AODV::recv(Packet *p, Handler*) {
00588: struct hdr_cmn *ch = HDR_CMN(p);
00589: struct hdr_ip *ih = HDR_IP(p);
00590:
00591: //add by season
00592: //struct hdr_mac802_11 *mh;
00593: //int       test = mh->dh_duration;
00594:
```

重新编译，（注意，一定要退出 gdb,否则编译会出错的）

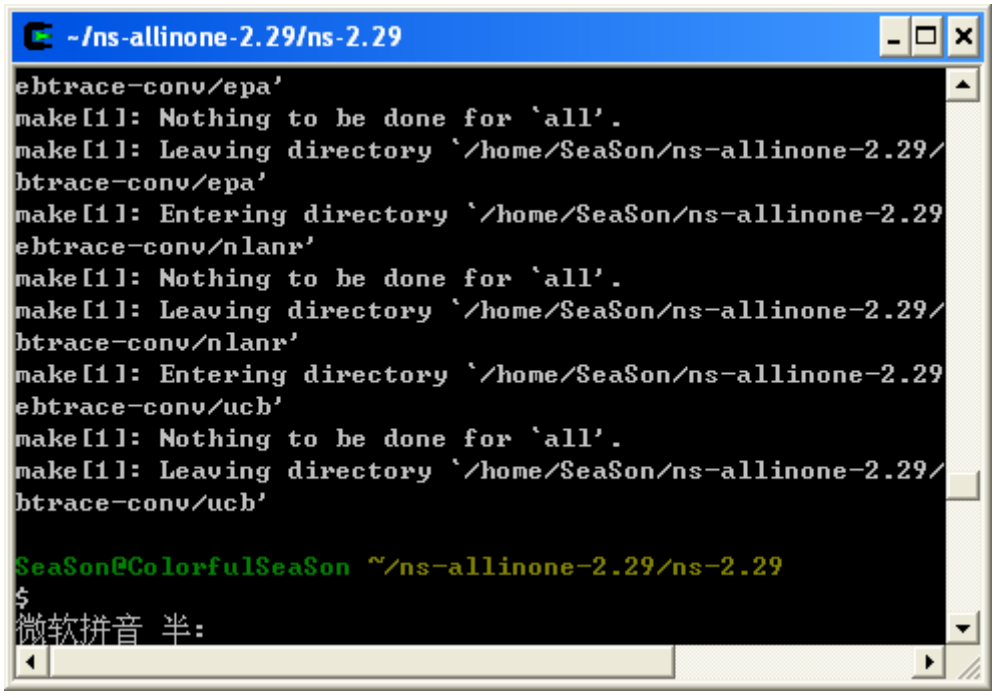
首先退出 gdb 调试状态



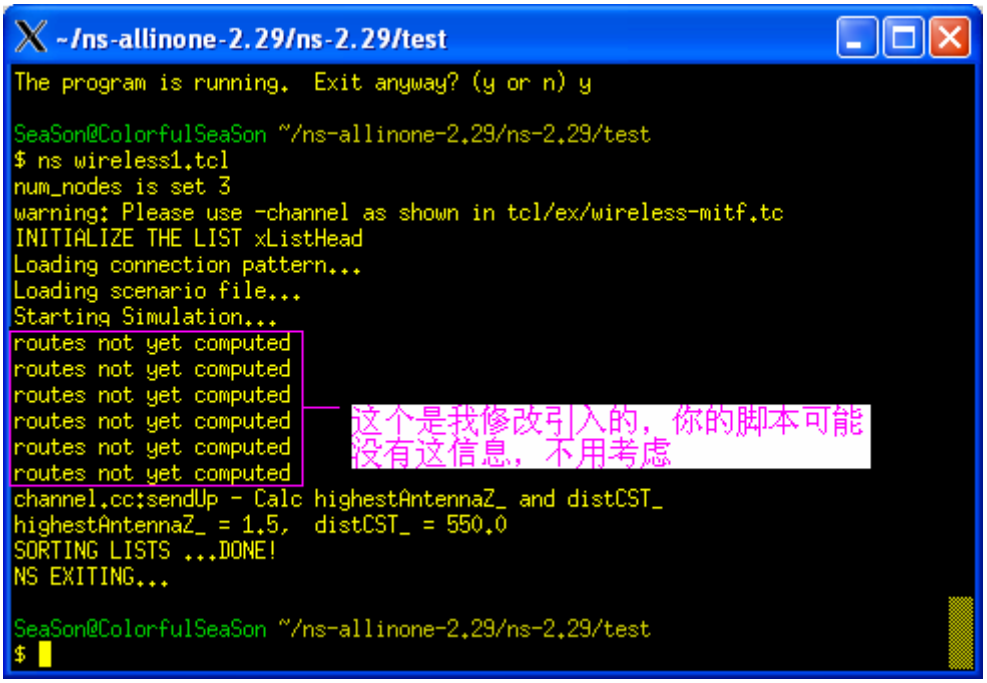
然后重新编译 make



如果出现下面界面，即编译成功。

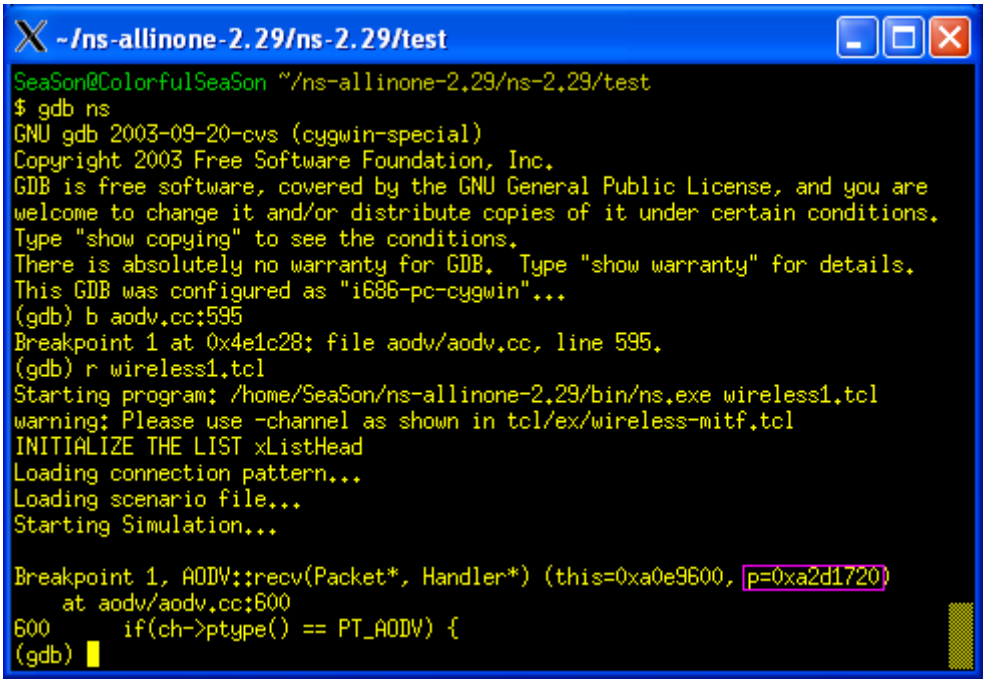


重新运行我们的脚本 wireless1.tcl，就不会出现刚才的错误了。如下图所示：



示例二、逻辑错误的调试

说明，一般遇到的都是 **segmentation fault** 错误，但并非是这个错误没有了你的程序就完全 **OK** 了，还需要调试逻辑错误，就是跟踪某个数据包，看他是否按照你设计的流程去走，这个一般我是通过跟踪数据包的地址实现的。即下图所示。



具体的步骤我就不写了，太多了，我介绍一下思路：

1. 在节点内各协议之间的发送，这个需要你熟悉下图的结构，一般断点设在各

层协议的 recv()函数之内，然后你逐层的跟踪就可以确定问题出现在那个协议上了，这个是体力活，呵呵。

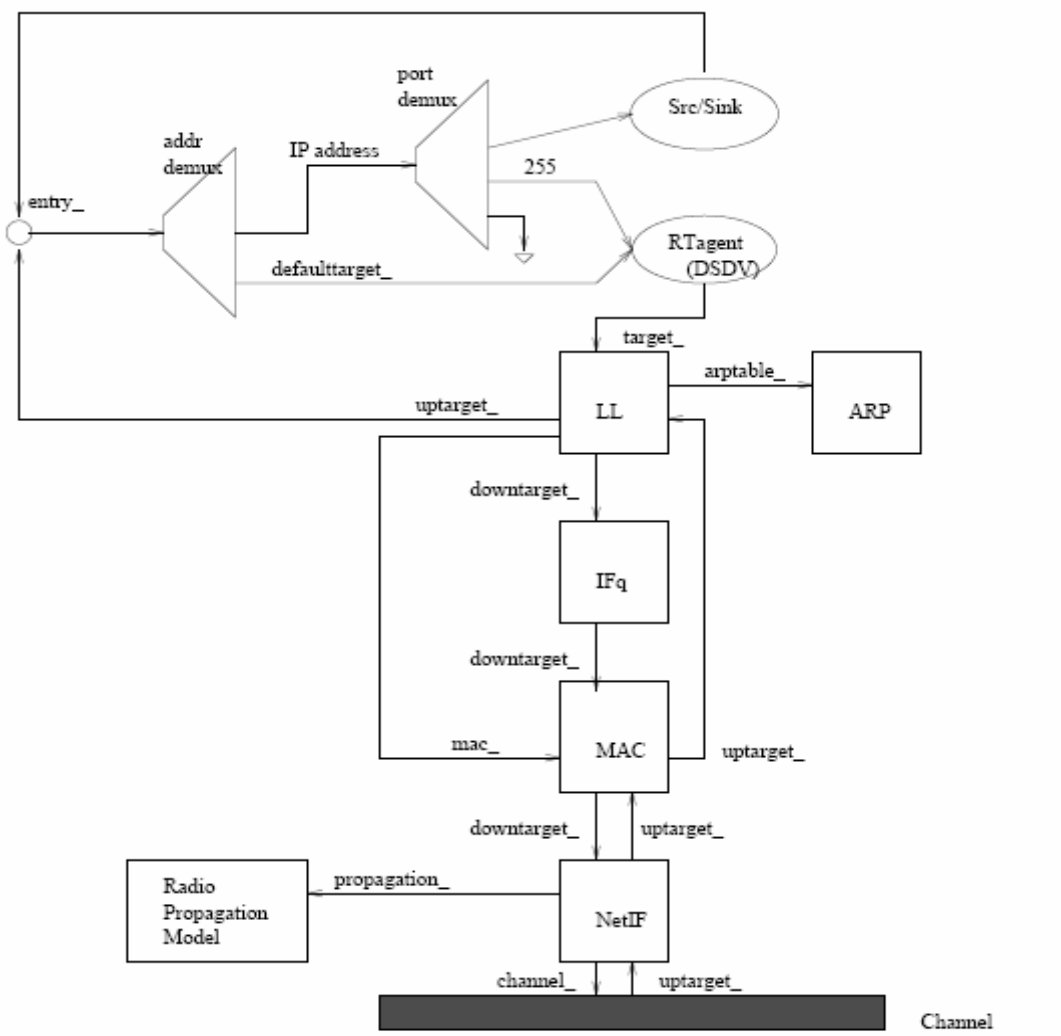


Figure 16.1: Schematic of a mobilenode under the CMU monarch's wireless extensions to *ns*

2. 节点之间的发送

这个你仔细分析一下 channel.cc 文件中的 sendUp()函数就明白了，注意有两个同名的函数，都看看。节点之间发送数据就是通过这个函数，我一般设置端点都在这个函数中。

```
00327: WirelessChannel::sendUp(Packet* p, Phy *tifp)
00328: {
00329:      Scheduler &s = Scheduler::instance();
00330:      Phy *rifp = ifhead_.lh_first;
00331:      Node *tnode = tifp->node();
00332:      Node *rnode = 0;
00333:      Packet *newp;
00334:      double propdelay = 0.0;
00335:      struct hdr_cmn *hdr = HDR_CMN(p);
00336:
00337:       /* list-based improvement */
00338:       if(highestAntennaZ_ == -1) {
00339:            fprintf(stdout, "channel.cc:sendUp - Calc highestAntennaZ_ a
00340:            calcHighestAntennaZ(tifp);
00341:            fprintf(stdout, "highestAntennaZ_ = %0.1f,  distCST_ = %0.1f
00342:       }
00343:
```