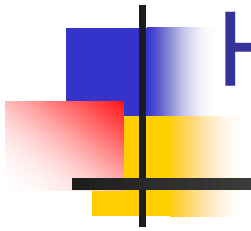
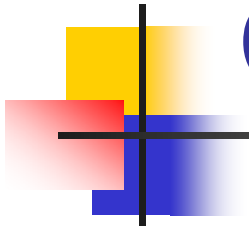


How to implement protocol in NS2

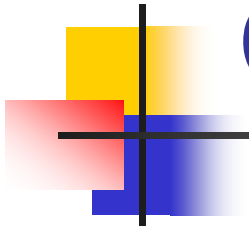
Shi Feng Yan

Department of Computer Science and Information Engineering
Chung-Hua University



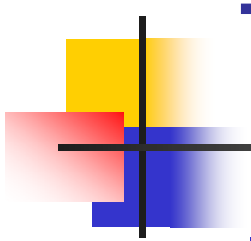
Outline

- Our Proposed Protocol
- How to implement protocol



Our Proposed Protocol

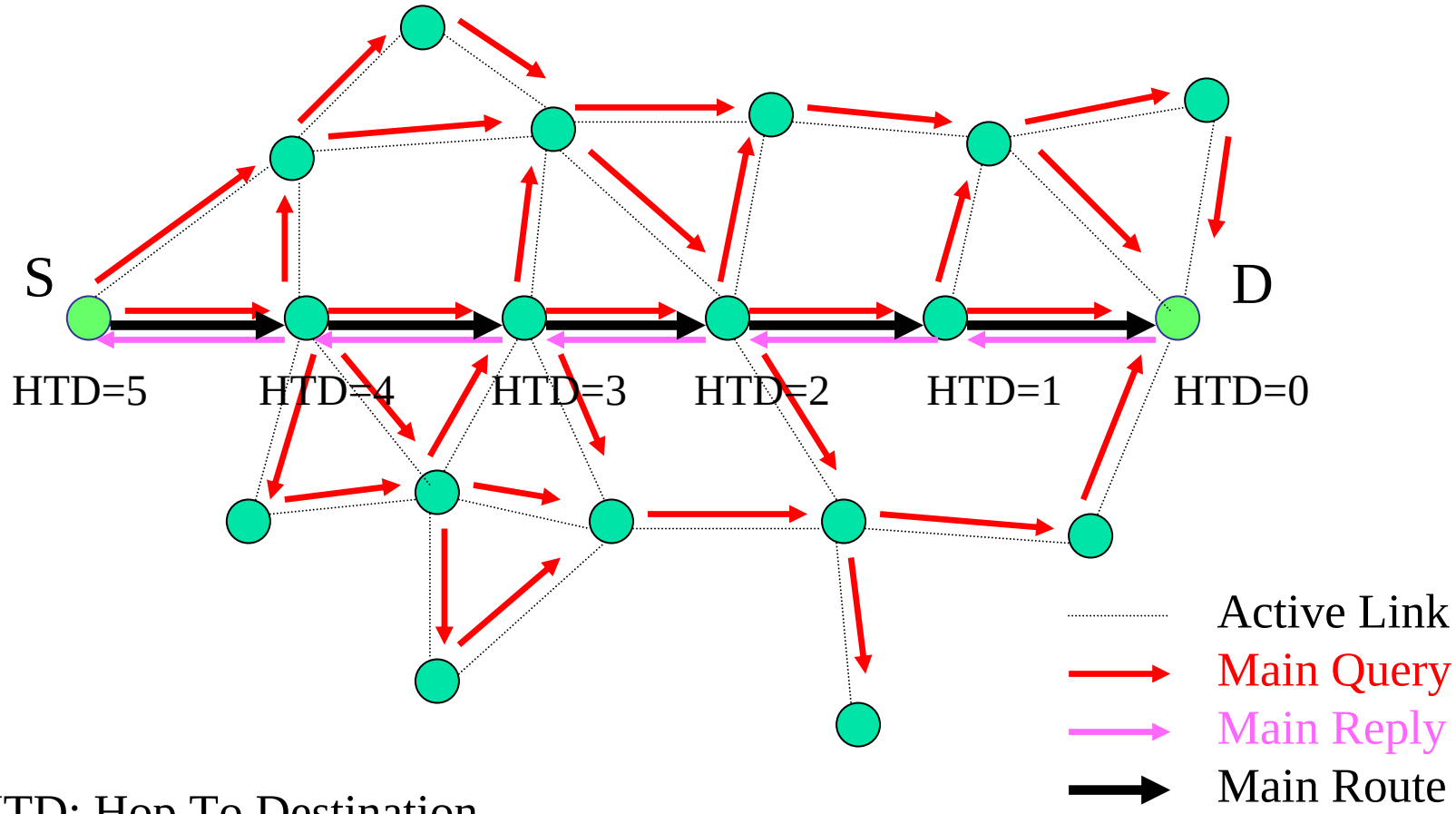
- The Objectives
- Route construction
- Repairing



The Objectives

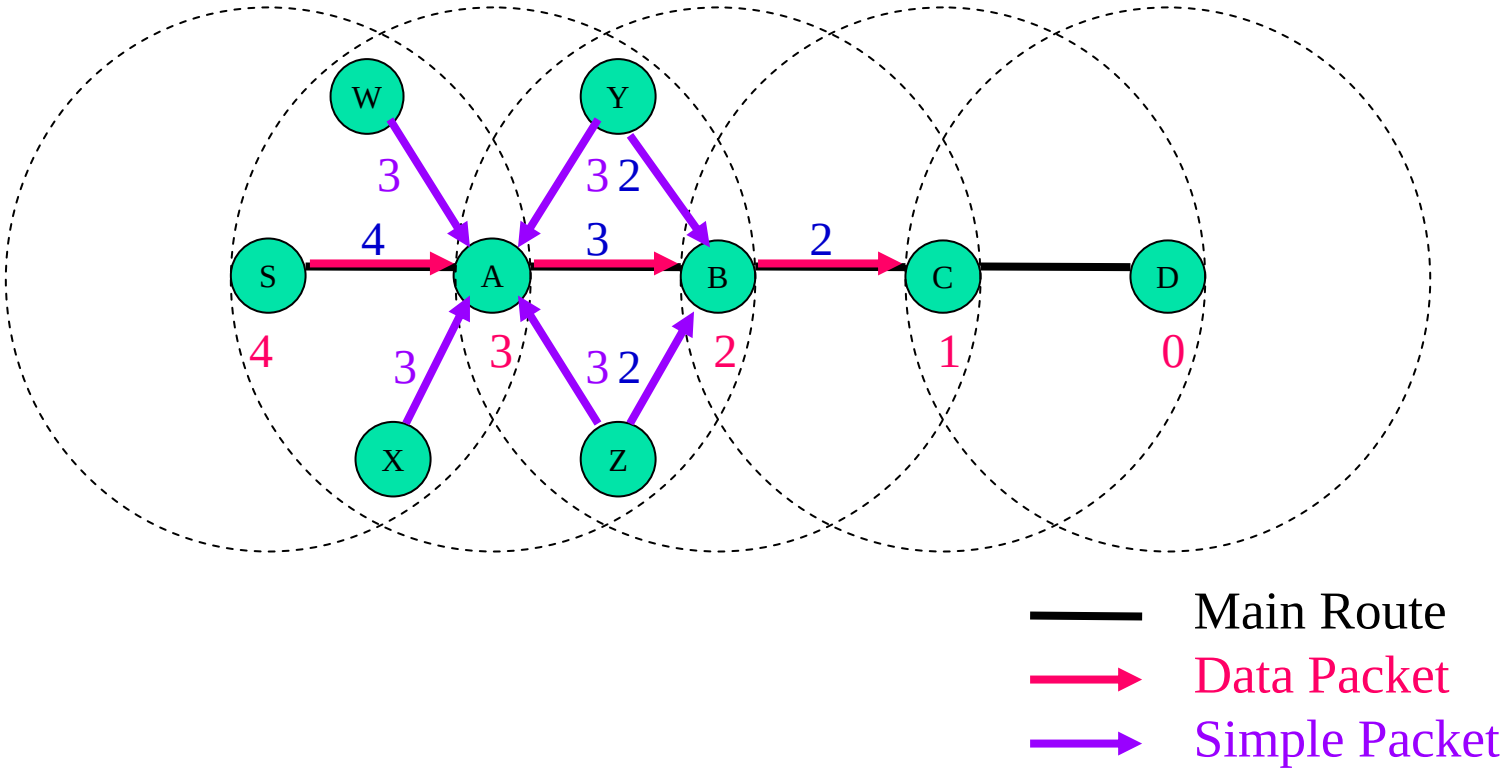
- Build some backup routes surround the main route
- Repair the route quickly when a link breaks
- Decrease maintenance cost

Main route construction

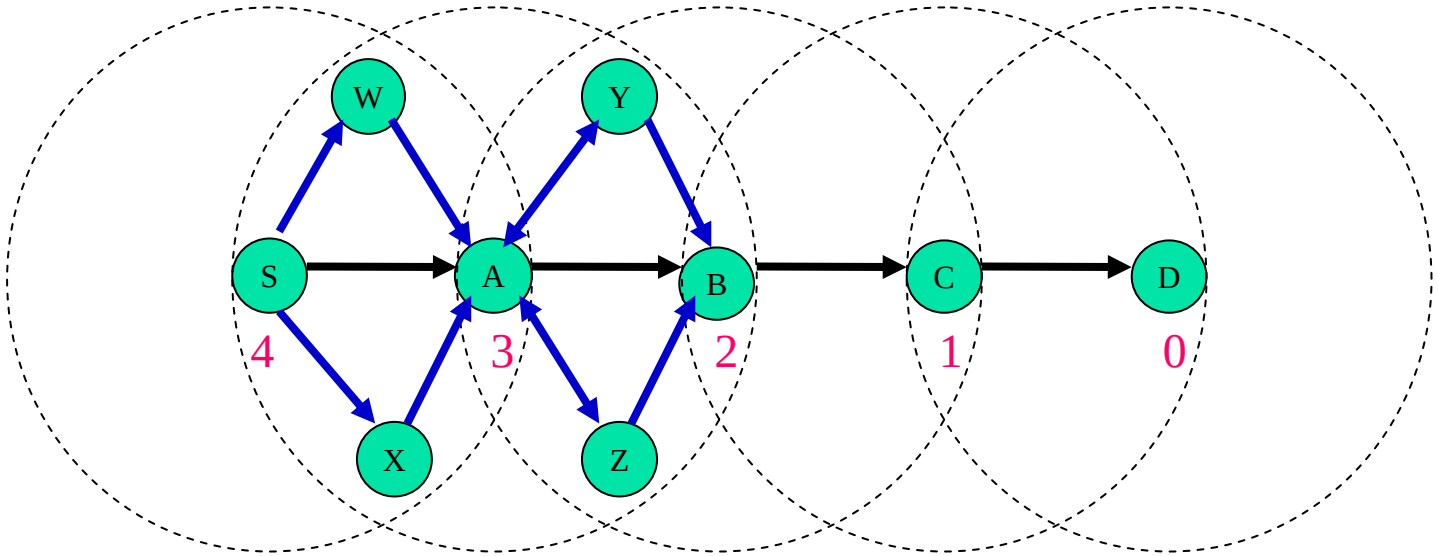


HTD: Hop To Destination

Backup route construction



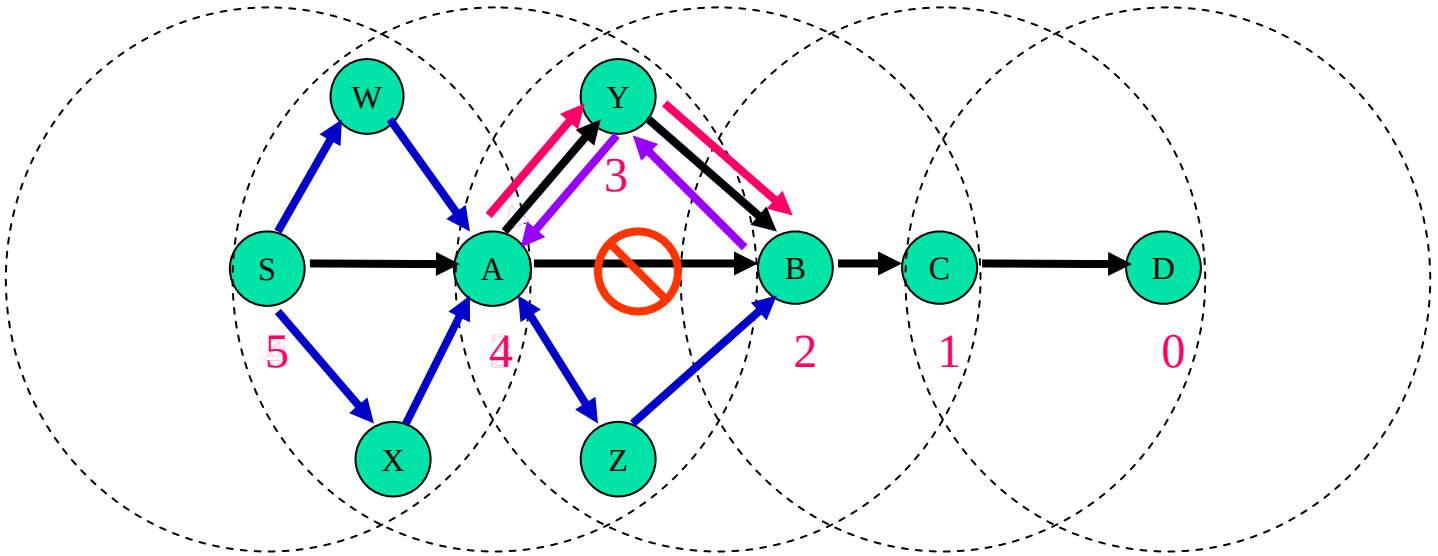
Backup route construction



Main Route

Backup Route

Main route failure in case 1



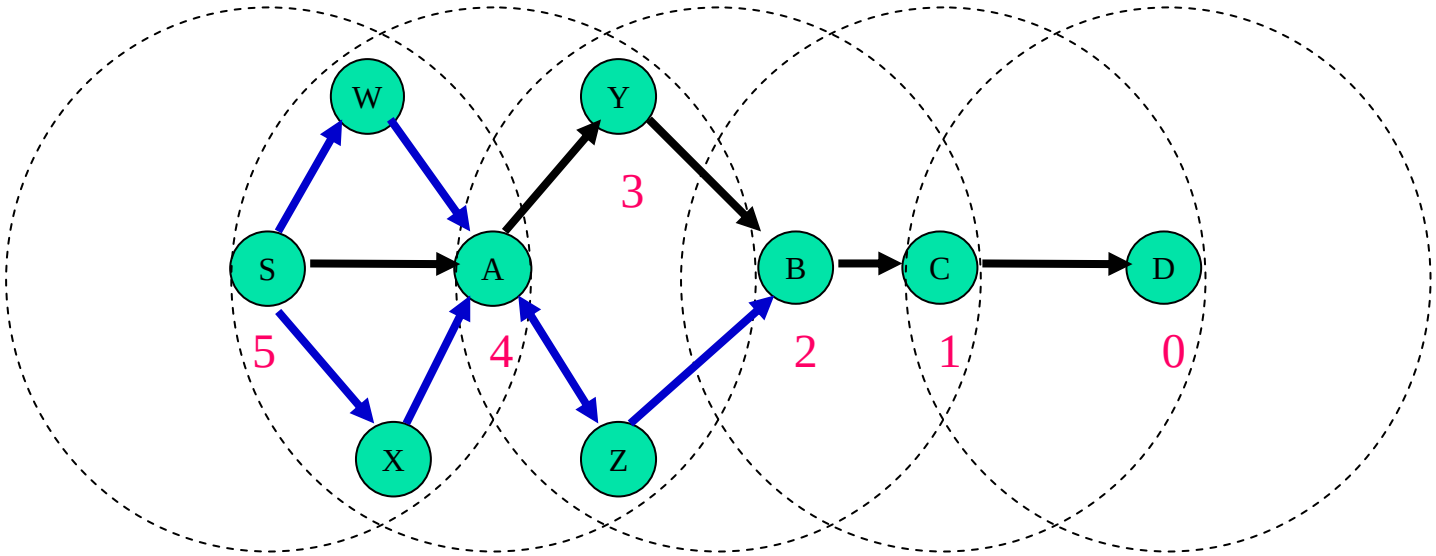
Main Route

Backup Route

Main Failure Query

Failure Reply

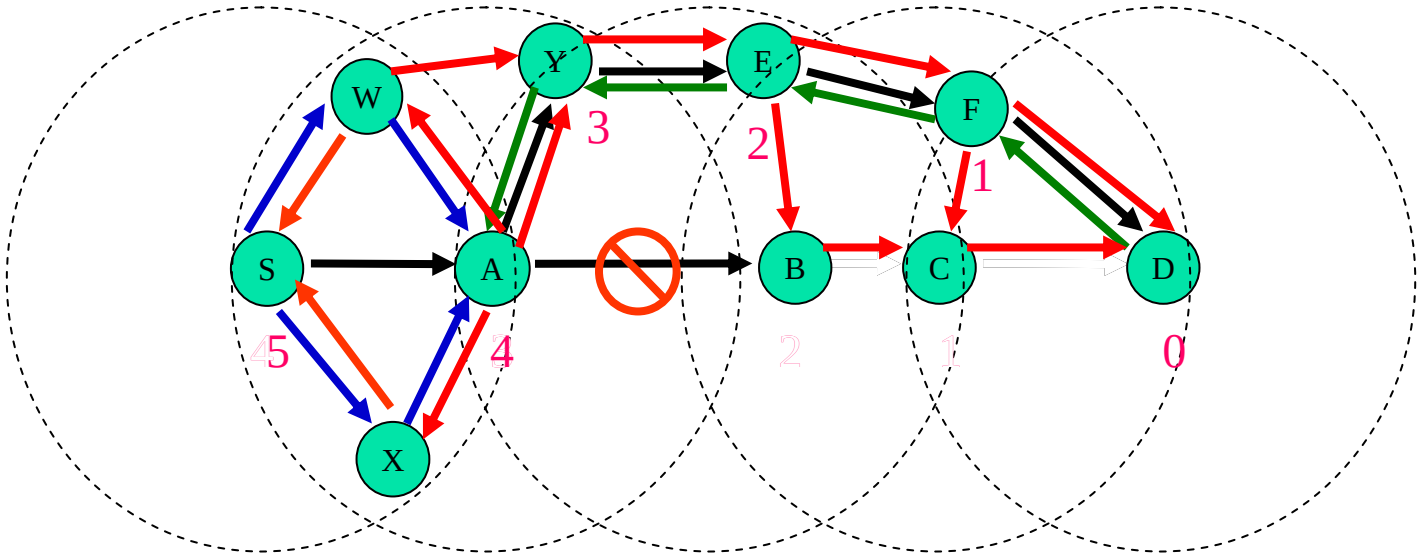
After repairing in case 1



Main Route

Backup Route

Main route failure in case 2



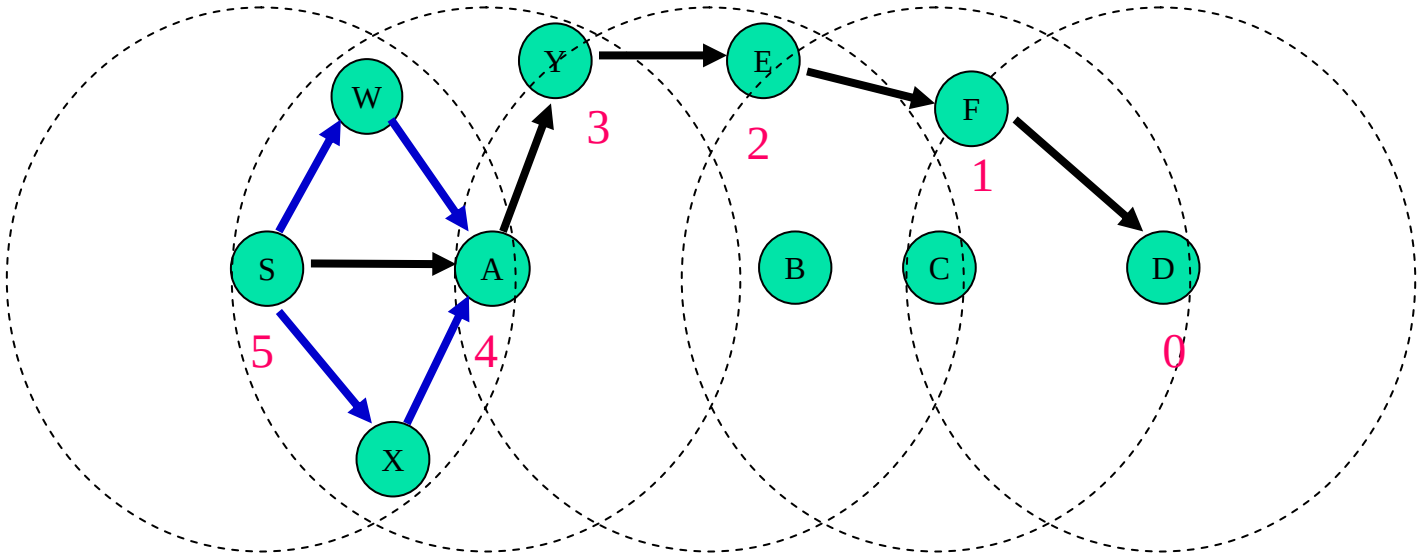
Main Route

Backup Route

Backup Query

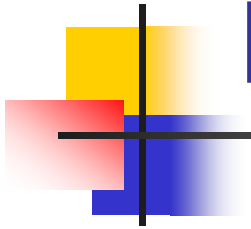
Backup Reply

After repairing in case 2



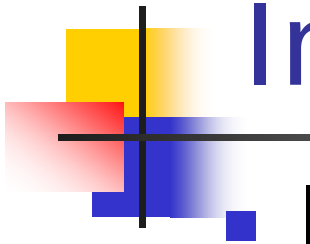
Main Route

Backup Route



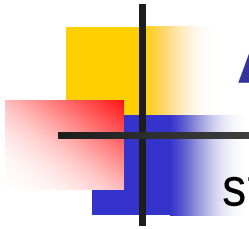
How to implement protocol

- In C++
- In TCL



In C++

- Insert
  - Apacket.h
  - Armbr.cc / Armbr.h
  - Armbrrttable.cc / Armbrrttable.h
- Modify
  - Packet.h
  - CMU-Trace.cc / CMU-Trace.h



Apacket.h

```
struct Apacket {
    u_int8_t type__;
    nsaddr_t sid__;
    nsaddr_t did__;
    unsigned int seq__;
    int count__;
    // Header access methods
    static int offset_; // required by PacketHeaderManager
    inline static int& offset() { return offset_; }
    inline static Apacket* access(const Packet* p) {
        return (Apacket*) p->access(offset_);
    }
};
```

Packet.h(1)

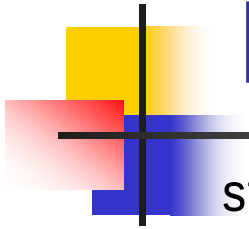
```
enum packet_t {
    ...
    PT_ARMBR,
    ^ ^ ^ ^ ^ ^ ^
    We add our protocol header type here
    PT_NTYPE    // This MUST be the LAST one
};
```

Packet.h(2)

```
class p_info {
public:
    p_info() {
            ...
            name_[PT_ARMBR] = "ARMBR";
           ^^^^^^^^^^^^^^^^^^^^^^^^^^
```

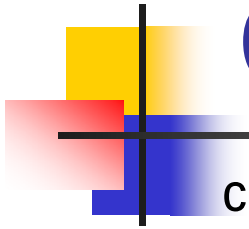
We add our control packet header name here

```
            name_[PT_NTYPE]= "undefined";
    };
```



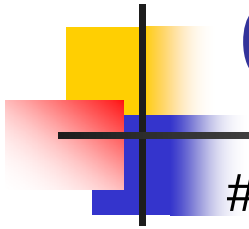
Packet.h(3)

```
struct hdr_cmn {
    ...
    int    ref_count_;       // free the pkt until count to 0
    int htd;
    ^ ^ ^ ^ ^ ^
    We add a new field in data packet
};
```



CMU-Trace.h

```
class CMUTrace : public Trace {
...
private:
    ...
void      format_aodv(Packet *p, int offset);
void      format_armbr(Packet *p, int offset);
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
        we add this line here
};
```



CMU-Trace.cc(1)

```
#include <Apacket.h>
^^^^^^^^^^^^^^^^^^^^
```

include our control packet header file

...

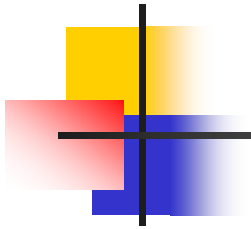
CMU-Trace.cc(2)

```
void CMUTrace::format(Packet* p, const char *why)
{   ...
    switch(ch->ptype()) {
            ...
    default:
            switch(ch->ptype()) {
            case PT_ARMBR:
            ^^^^^^^^^^^^^^
                    format_armbr(p, offset);
                    ^^^^^^^^^^^^^^^^^^^^^^^
                    break;
                    ^^^^^^
            add these three lines
            ...
}}}
```

CMU-Trace.cc(3)

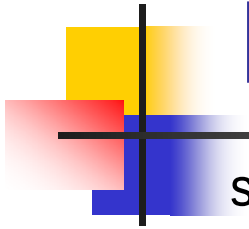
Add this function in CMU-Trace.cc

```
void CMUTrace::format_armbr(Packet *p, int offset)
 {  struct Apacket *ah = Apacket(p);
    if (newtrace_) {
      sprintf(wrk_ + offset, "-P armbr -Pt 0x%x -Ps %d -Pd %d
                -Pq %d -Pc %d -Pc CLEAR ",ah->type__, ah->si
                d__, ah->did__, ah->seq__, ah->count__);
    } else {
      sprintf(wrk_ + offset, "[0x%x %d %d %d %d] (CLEAR)",
                ah->type__, ah->sid__, ah->did__, ah->seq__, ah->
                count__);
    }
}
```



In TCL

- NS\Ns2\TCL\Lib
  - Ns-lib.tcl
  - Ns-Packet.tcl
  - Ns-Mobilenode.tcl

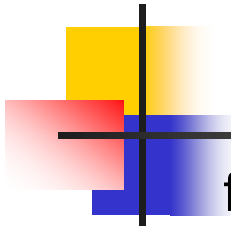


Ns-lib.tcl(1)

```
switch -exact $routingAgent_ {
        ...
        AODV {
                set ragent [$self create-aodv-agent $node]
        }
        TORA {
                Simulator set IMEPFlag_ ON
                set ragent [$self create-tora-agent $node]
        }
        ARMBR {
                set ragent [$self create-armbr-agent $node]
        }
        ...
}
```

Ns-lib.tcl(2)

```
Simulator instproc create-armbr-agent { node } {
    set ragent [new Agent/ARMBR [$node id]]     ;
# create a new routing agent
    $node set ragent_ $ragent     ;
# attach routing agent to node
    $self at 0.0 "$ragent start"      ;
# start BEACON/HELLO Messages
    return $ragent
}
```



Ns-Packet.tcl

```
foreach prot {
    ...
    ARMBR
} {add-packet-header $prot  }
```

Ns-Mobilenode.tcl

```
set armbronly [string first “ARMBR" [$agent info class]]
if {$armbronly != -1 } {
        $agent set-ll [$self set ll_(0)]
        $agent if-queue [$self set ifq_(0)]    ;
                # ifq between LL and MAC
        $agent install-tap $mac_(0)
}
```

ARMBR.cc(1)

```
int ARMBRAgent::command(int argc, const char*const* argv) {
    ...
    else if(argc == 3) {
        if (strcasecmp (argv[1], "addr") == 0) {
        ...
        else if(strcasecmp(argv[1], "install-tap") == 0) {
                obj = TclObject::lookup(argv[2]);
                if(obj == 0) {
                        fprintf(stderr, “Armbr: 'install-tap' failed (%s)
                                (%s)\n", argv[1], argv[2]);
                        return (TCL_ERROR);
                }
                Mac *m = (Mac*) obj;
                m->installTap(this);
                return TCL_OK;
        }
```

ARMBR.cc(2)

```
void ArmbrAgent::tap(const Packet *p)
{
    struct hdr_cmn *ch = HDR_CMN(p);
    ...
    // don't trouble me with my own packets
    if(ih->saddr()==myaddr||ih->saddr()==0||ih->daddr()==0) return;
    // don't trouble me with packets I'm about to receive anyway
    if(ih->daddr()==(nsaddr_t)IP_BROADCAST||ih->daddr()==myaddr)
    return;
    ...
    Packet* pkt = p->copy();
    recvSIMPLE(pkt);
}
```