

Implementation of Cloning in MANETS

¹Vandana, ²Suchitra V

^{1,2}Dept. of Electronics and Telecommunication, Siddaganga Institute of Technology, Tumakuru, India

Abstract

A mobile Adhoc network (MANET) is formed by a group of autonomous mobile nodes connected by wireless link without centralized control or established infrastructure. To clone protocol we need to modify different C++ and TCL files. Even if one component is modified then the entire Network Simulator-2 (NS-2) suite must be reconfigured. Cloning the protocol manually takes a lot of time and prone to error. Our objective is to ease the work of developers and researchers by showing the procedure to clone the AODV protocol automatically using a script. Methodology: In this study, a shell script is developed that will clone the AODV protocol by modifying 18 C++ and TCL files. The comparison of cloned and uncloned AODV protocol is done and the result generated are exactly the same for both the protocols. Proposed script is clone the protocol much faster and it will save the time and help the developers or research to focus more on their study on the protocol.

Keywords

MANETs, AODV Protocol, Shell Script, Clone AODV Protocol

I. Introduction

MANET is an autonomous system where two or more wireless devices or has the capability to communicate with each other without any centralized administrator or fixed network infrastructure [1].

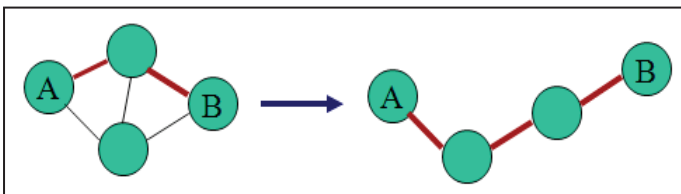


Fig 1: Frequent change in the topology of MANET

In MANETs the host movements and topology change is frequent, no cellular infrastructure and multi-hop wireless links. In MANETs data must be routed via intermediate nodes.

There are several types of cloning like profile cloning, code cloning, audio cloning and protocol cloning.

A. Profile cloning

Profile cloning is the identity theft of existing users profile and create a fake profile. Same site profile cloning and cross site profile cloning are the two types of profile cloning [2].

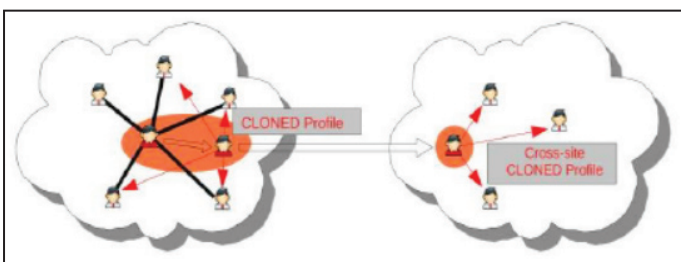


Fig. 2: Example of Same site and Cross Site Profile Cloning

B. Code cloning

Code cloning is the practice of duplicating existing source code for use elsewhere within a software system. Replicating code with or without making minor changes it may reduce production time, it increase productivity and it may increase maintenance cost [3].

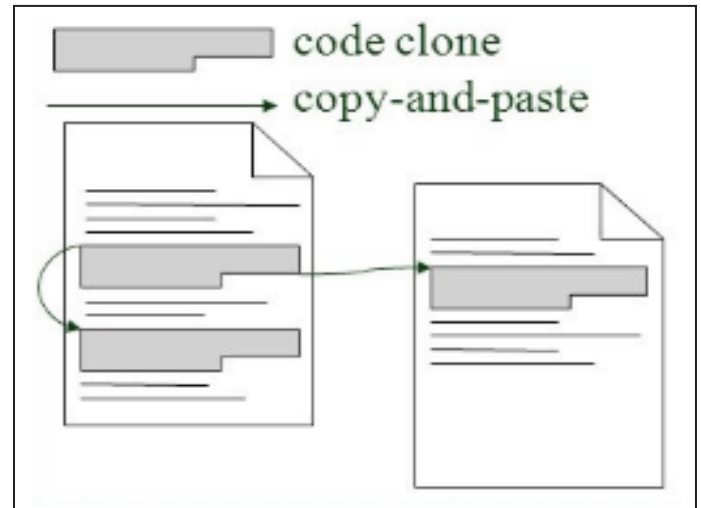


Fig. 3: Example of Code Cloning

C. Audio cloning

Cloning a voice typically requires collecting hours of recorded speech to build a dataset then using the dataset to train a new voice model [4].



Fig. 4: Example of Voice Cloning

Ad-Hoc On-Demand Distance Vector (AODV) Routing Protocol

In Ad-Hoc On Demand Distance Vector Routing Protocol (AODV routing protocol) routes are maintained only between nodes which need to communicate [5]. Route Requests (RREQ) are forwarded to the nodes. When a node re-broadcasts a Route Request, it sets up a reverse path pointing towards the source. When the intended destination receives a Route Request, it replies by sending a Route Reply (RREP) Route Reply travels along the reverse path set-up when Route Request is forwarded.

II. Network Simulator 2

Network simulator 2 (NS2) is an open-source event-driven simulator designed specifically for research in computer communication networks, MANETs, WSNs (wireless sensor networks). NS2 consists of two key languages such as C++ and Object-oriented Tool Command Language (OTcl). Both the languages have its own features but if any changes made in C++ is need to be recompiled and files to run a protocol are many and it's easy task. To implement the components in C++ is very difficult, so it can be defined in Otcl file, modifications are easily done in Tcl script than C++.

III. Implementation

AODV protocol is an important routing protocol and the improvisation of this protocol can be done in many fields like energy efficiency, reducing the complexity in finding shortest path and many other. In C++, if a small changes is made it will take more time and will give errors. To avoid these issue the protocol is cloned and make the necessary changes. But manually protocol cloning is very difficult it required to modify 18 C++ and TCL files. If one small code is missing then the cloning will not work. So, shell script is developed which will automatically clone the protocol.

Shell script procedure to clone AODV protocol: In this the script the cloned protocol is named as TAODV and ns-allinone-2.35 version is used.

Step 1: Copy the file of aodv in the different folder name as taodv.

Step 2: Rename all the file names and inside document. Rename all files inside taodv by prefixing "t".

Step 3: Replace aodv to taodv and AODV to TAODV.

Step 4: Go to #ns-allinone-2.35 \ns2.35 \common\ packet.h

```
.....
.....
static const packet_t PT_TAODV = 73;
.....
type == PT_AODV ||
type == PT_TAODV ||
type == PT_MDART)
.....
name_[PT_DCCP_RESET]="DCCP_Reset";
name_[PT_TAODV]= "TAODV"; name_[PT_NTTYPE]=
"undefined";
}
```

Step 5: #ns-allinone-2.35\ ns-2.35\ trace\ cmu-trace.h

```
void format_aodv(Packet *p, int offset);
void format_taodv(Packet *p, int offset);
```

Step 6: #ns-allinone-2.35\ ns-2.35\ trace\ cmu-trace.cc
#include <taodv/taodv_packet.h>

copy from line 862 to 965

```
void
CMUTrace::format_taodv(Packet *p, int offset)
```

```
.....
.....
abort();
}
}
case PT_AODV:
format_aodv(p, offset);
break;
case PT_TAODV:
format_taodv(p, offset);
break;
break;
```

Step 7: #ns-allinone-2.35\ ns-2.35\ tcl\ lib\ ns-packet.tcl

```
AODV # routing protocol for ad-hoc networks
TAODV # routing protocol for ad-hoc networks
```

Step 8: #ns-allinone-2.35\ ns-2.35\ tcl\ lib\ ns-lib.tcl

```
AODV {
set ragent [$self create-aodv-agent $node]
}
TAODV {
set ragent [$self create-taodv-agent $node] }
Simulator instproc create-taodv-agent { node }
{
# Create TAODV routing agent
set ragent [new Agent/TAODV [$node node-addr]]
$self at 0.0 "$ragent start" ;# start BEACON/HELLO
messages
$node set ragent_ $ragent
return $ragent
}
```

Step 9: #ns-allinone-2.35\ ns-2.35\ queue\ priqueue.cc

```
case PT_AODV:
case PT_TAODV
```

Step 10: #ns-allinone-2.35\ ns-2.35\ Makefile

```
aodv/aodv_logs.o aodv/aodv.o \
aodv/aodv_rtable.o aodv/aodv_rqueue.o \
taodv/taodv_logs.o taodv/taodv.o \
taodv/taodv_rtable.o taodv/taodv_rqueue.o \
```

(Making the same changes in makefile.vc and Makefile.in otherwise after ./configure)

Step 11: #ns-allinone-2.35\ ns-2.35\ tcl\ lib\ ns-agent.tcl

```
Agent/AODV set sport_ 0
Agent/AODV set dport_ 0

Agent/TAODV instproc init args {
$self next $args
}
Agent/TAODV set sport_ 0
Agent/TAODV set dport_ 0
```

Step 12: #ns-allinone-2.35 \ ns-2.35 \ tcl \ lib \ ns-mobilenode.tcl

```
set aodvonly [string first "AODV" [$agent info class]]
if {$aodvonly != -1} {
$agent if-queue [$self set ifq_0] ;
```

```
set taodvonly [string first "TAODV" [$agent info class]]
if {$taodvonly != -1} {
$agent if-queue [$self set ifq_0] ;
```

Step 13: #ns-allinone-2.35 \ ns-2.35 \ queue \ rtqueue.cc

Step 14: #ns-allinone-2.35 \ ns-2.35 \ routing \ rtable.h

```
friend class AODV;
friend class TAODV;
.....
friend class AODV;
friend class TAODV;
friend class TAODVLocalRepairTimer; .....
```

Step 15: #ns-allinone-2.35 \ ns-2.35 \ wpan \ p802_15_4nam.cc

```
(strcmp(packet_info.name(PT_AODV),name) == 0)?PT_AODV:
(strcmp(packet_info.name(PT_TAODV),name) == 0)?PT_TAODV:
```

Step 16: #ns-allinone-2.35 \ dei80211mr-1.1.4 \ src \ InitTCL.cc

```
tab_(PacketHeader/AODV) 1\n\
PacketHeaderManager set tab_(PacketHeader/TAODV) 1\n\
```

Step 17: Rename every aodv file with taodv name inside taodv folder

Step 18: Open each file and rename aodv to taodv and AODV to TAODV

Step 19: Rename every timer class in taodv.h and taodv.cc

Step 20: Edit taodv_rtable.h

```
class taodv_rt_entry {
friend class taodv_rtable;
friend class TAODV;
friend class LocalRepairTimer;
friend class TAODVLocalRepairTimer;
```

Step 21: Recompilation:

- Step 1: We should recompiled "packet.cc" as the "packet.h" is modified. This can be done by : touch common / packet.cc"
- Step 2: ./configure (if this fails go to step 22)
- Step 3: make clean
- Step 4: make
- Step 5: make install

Step 22: If ./configure fails then run ./install

```
$cd ns-allinone-2.35
$./install
$cd ns-allinone-2.35/ns-2.35
$sudo make install
```

Table 1: Simulation Parameters

Parameter	Value
Simulator	NS-2 (Version 2.35)
Channel Type	Wireless Channel
Radio Propagation Model	TwoRayGround
Network Interface Type	WirelessPhy
MAC Type	802_11
Interface Queue Type	DropTail/PriQueue
Link Layer Type	LL
Antenna	OmniAntenna
Maximum Packet in ifq	50
Area (M*M))	900*900
Number of Mobile Nodes	40-60
Source Type	UDP
Routing Protocol	AODV, TAODV

IV. Results

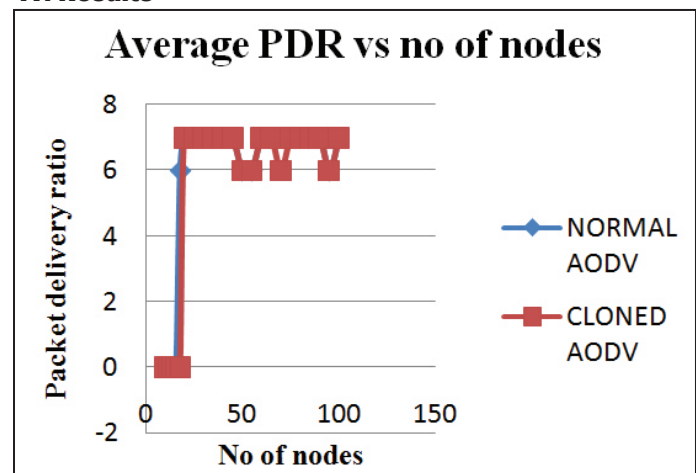


Fig. 5: Packet delivery ratio of normal AODV and cloned AODV (TAODV)

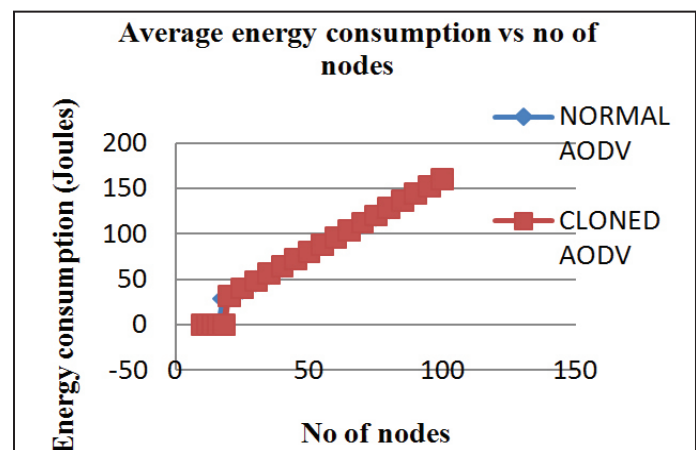


Fig. 6: Energy consumption of normal AODV and cloned AODV (TAODV)

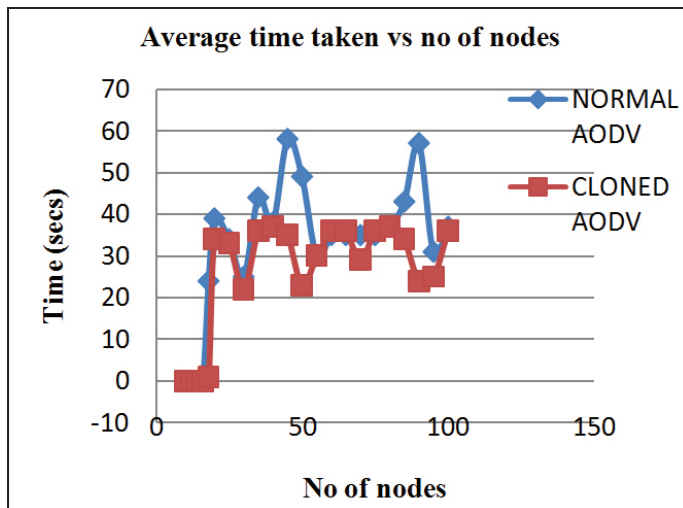


Fig. 7: Time delay of normal AODV and cloned AODV (TAODV)

The above graphs Figs. 5, 6 and 7 shows that the comparison of performance metrics between AODV and cloned AODV protocol. In Fig. 5 shows that the Packet Delivery Ratio (PDR) is same in both the protocols and in Fig 6 shows that the energy consumption is also same in both the protocols at 40 nodes it is 64 and the energy increases due to increase the number of nodes. In Fig 7 the execution time of normal AODV (5sec) is greater than the cloned AODV (4sec), delay increases while increasing the number of nodes.

V. Conclusion

Improvising the AODV protocol performance is difficult in NS-2. Even if a single C++ code is modified then entire network simulator package must be reconfigured. If the reconfiguration fails then the NS-2 package must be reinstall. This can be avoided using a AODV cloned protocol. In AODV cloned protocol performance is same as normal AODV protocol so it allowing the researchers or developers to experiment their ideas of improvement in this protocol.

References

- [1] El-Sayed M. El-Rabaiea, Nancy A. Al-Shaerb, "A Survey on Ad Hoc Networks", Article on 20- November 2016.
- [2] M.A.Devamane, Dr. M.K.Rana, "Detection and Prevention of Profile Cloning in Online Social Networks", IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAI E – 2014), May 09-11, 2014, Jaipur.
- [3] Dethe Tukaram, Uma Maheswari B, "Design and development of software tool for code clone search, detection, and analysis ", International Conference on Electronics Communication and Aerospace Technology [ICECA 2019] IEEE.
- [4] Hafiz Malik, "Securing Voice-driven Interfaces against Fake (Cloned) Audio Attacks", 2019 IEEE.
- [5] S. Taruna, G.N. Purohit, "Scenario Based Performance Analysis of AODV and DSDV in Mobile Adhoc Network", N. Meghanathan et al. (Eds.): CCSIT 2011, Part II, CCIS 132, pp. 10–19, 2011.



Karnataka, India.

Vandana received her bachelor's degree in Electronics and Communication Engineering from Kalpataru Institute of Technology, Tiptur, recognized by Visvesvaraya Technological University, Belagavi, Karnataka, India, in 2018 and pursuing Master's degree in Digital Electronics and Communication system from Siddaganga Institute of Technology (Autonomous), Tumkur, Karnataka, India recognized by Visvesvaraya Technological University, Belagavi,



Siddaganga Institute of Technology (Autonomous), Tumkur, Karnataka, India since 2015 till date. Her research interests include Wireless Networking, MANETs, Wireless Sensor network, and antenna designing technique.

Mrs. Suchitra V received her B.E degree in Electronics and Communication Engineering from Basaveshwara Engineering College (Autonomous), Bagalkot, Karnataka, India in 2011, the MTech degree in Digital Communication from Acharya Institute of Technology (Under VTU), Bangalore, Karnataka, India in 2013. Presently, working Assistant Professor in Department of Electronics and Telecommunication Engineering,