Proceedings of National Conference on Cutting Edge Technologies in Computing & Communications
held at Sriram Engineering College, Chennai on 6th March 2020

A NOVEL APPROACH TO DETECT COLLABORATIVE ATTACKS IN MOBILE AD-HOC NETWORKS

[1]Vinodhini.A, [2]Mr. P.RethinaSabapathi
[1]M.E Student, [2] Associate Professor, Dept. of Computer Science & Engineering,
Sri Venkateswara College of Engineering and Technology,
Thiruvallur, Thirupachur, Tamil Nadu 631203

ABSTRACT

The presence of malevolent nodes, this requirement may lead to serious security concerns; for instance, such nodes may disrupt the routing process. In this context, preventing or detecting malicious nodes launching grayholeor collaborative black hole attacks is a challenge. This project attempts to resolve this issue by designing a dynamic source routing based routing mechanism, which is referred to as the cooperative bait detection scheme that integrates the advantages of both proactive and reactive defense architectures. Our CBDS method implements a reverse tracing technique to help in achieving the stated goal.

Keywords: MANET, CPNS, OTCL, Grey Hole, Black Hole, Malicious

INTRODUCTION

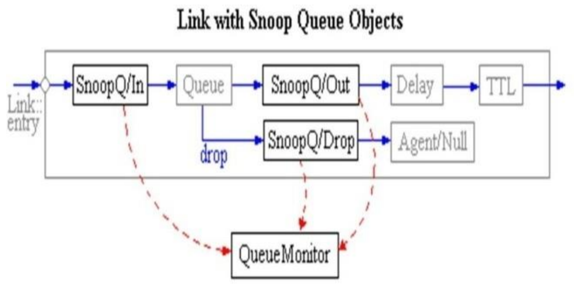
Due to the widespread availability of mobile devices, mobile ad hoc networks (MANETs) have been widely used for various important applications such as military crisis operations and emergency preparedness and response operations. This is primarily due to their infrastructure less property. In a MANET, each node not only works as a host but can also act as a router. While receiving data, nodes also need cooperation with each other to forward the data packets, thereby forming a wireless local area network.

PROJECT SCOPE

Monitoring and controlling physical systems through geographically distributed sensors and actuators have become an important task in numerous environment and infrastructure applications. These applications have received a renewed attention because of the advances in sensor network technologies and new development in Cyber-Physical Networked Systems (CPNS) .Typical CPNS cover a wide range of applications including transportation networks, vehicular networks, networks of unmanned vehicles and so on. Unlike more traditional embedded systems, CPNS is natural and engineered physical systems, which are integrated, monitored and controlled by an intelligent computational core. In CPNS, sensor nodes obtain the measurement from the physical components, process the measurements and send measured data to the controller through networks.

RELATED WORK

Ms. Neha B. Bhoyar, Prof. Poonam P. Borkar, "Acknowledgement Based Multipath Routing Scheme ForDetecting Mallicious Nodes In MANET," International Journal of Computer Science and Mobile Computing, Vol.4, February- 2015. Tariq Siddiqui,TanveerFarooqui,"A Survey on Malicious Node Detection in MANET,"Vol. 4, Issue12, December 2014. NavdeepKaur and Mouli Joshi "Implementing MANET Security using CBDS for Combating Sleep Deprivation & DOS Attack," International Journal of Science and Emerging Technologies with Latest Trends 16(1): 6- 12 (2014). Rakesh Kulkarni, MakarandJadhav,"Collision Avoidance Algorithm with Jitter in Multi-hop Cognitive Radio," international journal of engineering sciences & research technology, december,



2013. Ankita Khanna, P.U.Dere," A Review on Intrusion Detection and Security of Wormhole Attacks in MANET," International Journal of Science and Research (IJSR) Volume 3 Issue 12, December2014.

SYSTEM DESIGN

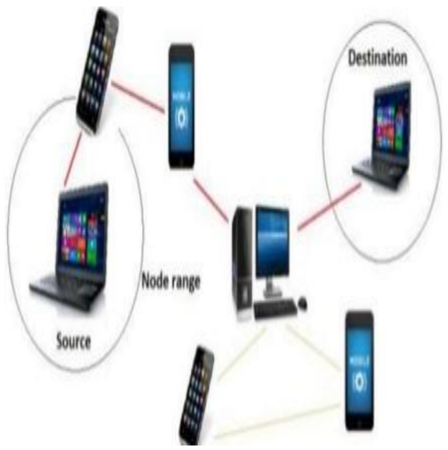
EXISTING SYSTEM

The detection mechanisms that have been proposed so far can be grouped into two broad categories 1) Proactive detection schemes are schemes that need to constantly detect or monitor nearby nodes. In these schemes, regardless of the existence of malicious nodes, the overhead of detection is constantly created. However, one of the advantages of these types of schemes is that it can help in preventing or avoiding an attack in its initial stage.2) Reactive detection mechanisms that have been proposed so far can be grouped into two broadcategories.

PROPOSED SYSTEM

Malicious nodes are thereby detected and prevented from participating in the routing operation, using a reverse tracing technique. In this setting, it is assumed that when a significant drop occurs in the packet delivery ratio, an alarm is sent by the destination node back to the source node to trigger the detection mechanismagain.

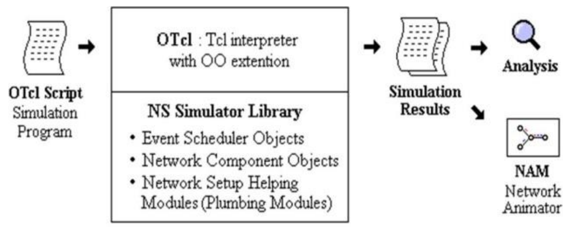
SYSTEMARCHITECTURE



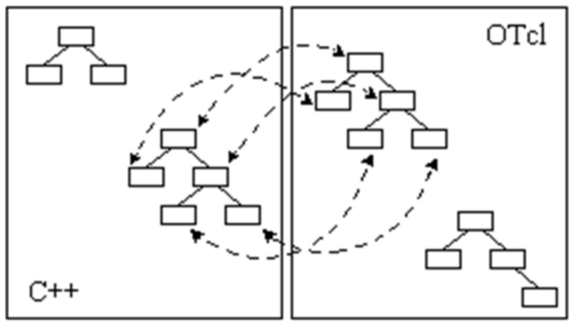
IMPLEMENTATION DETAILS

TOOL DESCRIPTION

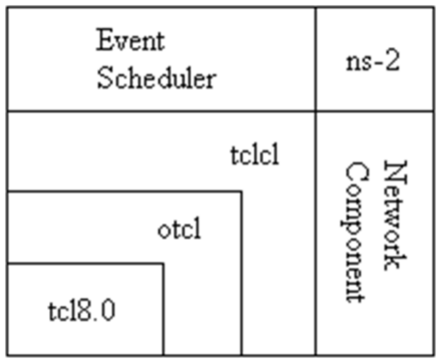
NS (version 2) is an object-oriented, discrete event driven network simulator developed at UC Berkely written in C++ and OTcl. Although all the usage of the simulator or possible network simulation setups may not be covered in this project, the project should help a new user to get started quickly.



Simplified User's View of NS in a simplified user's view, NS is Object-oriented Tcl (OTcl) script interpreter that has a simulation event scheduler and network component object libraries, and network setup (plumbing) module libraries (actually, plumbing modules are implemented as member functions of the base simulator object).an object hierarchy example in C++ and OTcl. One thing to note in the figure is that for C++ objects that have an OTcl linkage forming a hierarchy, there is a matching OTclobject hierarchy very similar to that of C++.



C++ and OTcl: The Duality

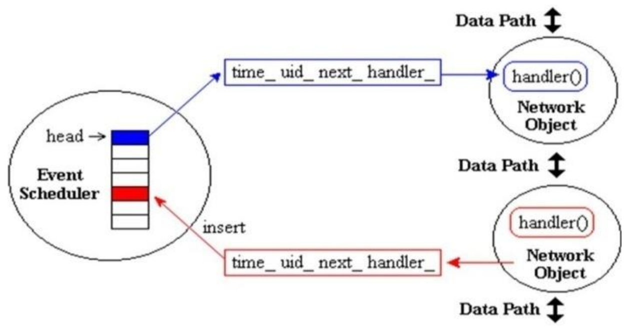


Architectural View of NS

The general architecture of NS. In this figure a general user (not an NS developer) can be thoughtof standing at the left bottom corner, designing and running simulations in Tcl using the simulator objects in the OTcl library.

Event Scheduler

This section talks about the discrete event schedulers of NS. As described in the Overview section, the main users of an event scheduler are network components that simulate packet-handling delay or that need timers. Figure 4.2.4 shows each network object using an event scheduler. Note that a network object that issues an event is the one who handles the event later at scheduled time. Also note that the data path between network objects is different from the event path. Actually, packets are handed from one network object to another using send(Packet*p) {target_->recv(p)}; method of the sender and recv(Packet*, Handler* h = 0) method of the receiver.



Discrete Event Scheduler

Working of NS2

OTcl: The User Language As mentioned in the overview section, NS is basically an OTclinterpreter with network simulation object libraries. It is very useful to know how to program in OTcl to use NS.

NS has two different types of event schedulers implemented. These are real-time and non-real-time schedulers. For a non-real-time scheduler, three implementations (List, Heap and Calendar) are available, even though they are all logically perform the same. This is because of backward compatibility: some early implementation of network components added by a user (not the original ones included in a package) may use a specific type of scheduler not through public functions but hacking around the internals. The Calendar non-real-time scheduler is set as the default. The real-time scheduler is for emulation, which allow the simulator to interact with a realnetwork.

In Tcl, the keyword proc is used to define a procedure, followed by an procedure name and arguments in curly brackets. The keyword set is used to assign a value to a variable. [expr ...] is to make the interpreter calculate the value of expression within the bracket after the keyword. One thing to note is that to get the value assigned to a variable, $ is used with the variable name. The keyword puts prints out the following string within double quotation marks. The following shows the result of Example1.

```
k < 5, pow = 1.0
k < 5, pow = 1120.0
k < 5, pow = 1254400.0
k < 5, pow = 1404928000.0
k < 5, pow = 1573519360000.0
k >= 5, mod = 0
k >= 5, mod = 4
k >= 5, mod = 0
k >= 5, mod = 0
k >= 5, mod = 4
```

The next example is an object-oriented programming example in OTcl. This example is very simple, but shows the way which an object is created and used in OTcl. As an ordinary NS user, the chances that you will write your own object might be rare. However, since all of the NS objects that you will use in a NS simulation programming, whether or not they are written in C++ and made available to OTcl via the linkage or written only in OTcl, are essentially OTcl objects, understanding OTcl object is helpful.

Node and Routing

A node is a compound object composed of a node entry object and classifiers. There are two types of nodes in NS. A unicast node has an address classifier that does unicast routing and a port classifier. A multicast node, in addition, has a classifier that classify multicast packets from unicast packets and a multicast classifier that performs multicast routing.

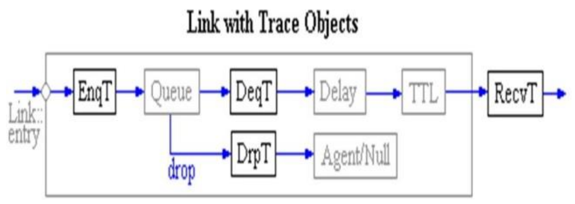
Node (Unicast and Multicast)

In NS, Unicast nodes are the default nodes. To create Multicast nodes the user must explicitly notify in the input OTcl script, right after creating a scheduler object, that all the nodes that will be created are multicast nodes. After specifying the node type, the user can also select a specific routing protocol other than using a default one.

There are two types of nodes in NS. A unicast node has an address classifier that does unicast routing and a port classifier. A multicast node, in addition, has a classifier that classify multicast packets from unicast packets and a multicast classifier that performs multicast routing.

Tracing

In NS, network activities are traced around simplex links. If the simulator is directed to trace network activities (specified using $ns trace-all fileor
$ns namtrace-all file), the links created after the command will have the following trace objects inserted as shown in Figure 4.2.5 Users can also specifically create a trace object of type type between the givensrc and dst nodes using thecreate-trace
{type file srcdst} command.



Inserting Trace Objects

When each inserted trace object (i.e. EnqT, DeqT, DrpT and RecvT) receives a packet, it writes to the specified trace file without consuming any simulation time, and passes the packet to the next network object. The trace format will be examined in the General Analysis Examplesection.

Queue Monitor

Basically, tracing objects are designed to record packet arrival time at which they are located. Although a user gets enough information from the trace, he or she might be interested in what is going on inside a specific output queue. For example, a user interested in RED queue behavior may want to measure the dynamics of average queue size and current queue size of a specific RED queue (i.e. need for queue monitoring). Queue monitoring can be achieved using queue monitor objects and snoop queue objects as shown in Figure 4.2.6

Monitoring Queue

When a packet arrives, a snoop queue object notifies the queue monitor object of this event. The queue monitor using this information monitors the queue. A RED queue monitoring example is shown in the RED Queue Monitor Example section.

Packet

A NS packet is composed of a stack of headers, and an optional data space. As briefly mentioned in the "Simple Simulation Example" section, a packet header format is initialized when a Simulator objectis created, where a stack of all registered (or possibly useable) headers, such as the common header that is commonly used by any objects as needed, IP header, TCP header, RTP header (UDP uses RTP header) and trace header, is defined, and the offset of each header in the stack isrecorded.

Trace Format

Each trace line starts with an event (+, -, d, r) descriptor followed by the simulation time (in seconds) of that event, and from and to node, which identify the link on which the event occurred. In the "Network Components" section to see where in a link each type of event is traced. The next information in the line before flags (appeared as "------" since no flag is set) is packet type and size (in Bytes). Currently, NS implements only the Explicit Congestion Notification (ECN) bit, and the remaining bits are not used. The next field is flow id (fid) of IPv6 that a user can set for each flow at the input OTclscript.

CONCLUSION

In this project, proposed a new mechanism called hidden terminal communication, with mesh and without mesh technique to avoid and detect gray hole or collaborative attacks. These techniques were also used to broadcast the message and to avoid collision during traffic in the network. By preventing the black hole and gray hole attack, minimizing the delay in the network.

FUTURE WORK

For the future work, aim to extending the throughput high in frequency range increase and then packet delivery ratio also increase. In a mesh network changes means delay will be extends, mesh breaking means delay can be minimize.

REFERENCES

[1] Ankita Khanna, P.U.Dere, "A Review on Intrusion Detection and Security of Wormhole Attacks in MANET," International Journal of Science and Research (IJSR) Volume 3 Issue 12, December 2014.

[2] A.Baadache and A. Belmehdi, "Avoiding blackhole and cooperative blackhole attacks in wireless ad hoc networks," Intl. J. Comput. Sci. Inf. Security, vol. 7, no. 1,2010.

[3] Fan-Hsun Tseng, Li-Der Chou and Han-ChiehChao,"A survey of black hole attacks in wireless mobile ad hoc networks," Tseng et al. Human-centric Computing and Information Sciences 2011.

[4] S. Marti, T.J.Giuli, K.Lai, and M.Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in Proc. 6th Annu. Intl. Conf. MobiCom, 2000, pp. 255–265.

[5] NavdeepKaur and Mouli Joshi, "Implementing MANET Security using CBDS for Combating Sleep Deprivation & DOS Attack," International Journal of Science and Emerging Technologies with Latest Trends" 16(1): 6- 12(2014).

[6] Ms. Neha B. Bhoyar, Prof. Poonam P. Borkar, "Acknowledgement Based Multipath Routing Scheme ForDetecting Mallicious Nodes In MANET," International Journal of Computer Science and Mobile Computing, Vol.4, February-2015.

[7] Rakesh Kulkarni, MakarandJadha, Sanjay Koli,"Collision Avoidance Algorithm with Jitter in Multi-hop Cognitive Radio Network," international journal of engineering sciences & research technology, December,2013.

[8] Ramandeep Kaur and JaswinderSingh,"Towards Security against Malicious Node Attack in Mobile Ad Hoc Network," International Journal of Advanced Research in Computer Science and Software Engineering,
Volume 3, Issue 7, July 2013.

[9] Ms.S.Shalini and Mrs.T.Manjula,"Security Enhancement using Trust Management in MANETs," international journal for trends in engineering & technology volume 3 issue 3 – march 2015.

[10] Tariq Siddiqui,TanveerFarooqui,"A Survey on Malicious Node Detection in MANET,"Vol. 4, Issue12, December2014.

[11] Y. Xue and K. Nahrstedt, "Providing fault-tolerant ad hoc routing service in adversarial environments," Wireless Pers.Commun., vol. 29, pp. 367.2004