ISSN:0975-9646
Abdul Jabbar Khilji et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 2 (1) , 2011, 474-476

The Foot Step of mobile network - wireless network architecture

Abdul Jabbar Khilji [#1], Dr.Raghuraj Singh [*2] , Shruti Sandal [#3],,Shashi Shekhar Ranga [#4]
#Assistant professor
Department Of Computer Application
Engineering College Bikaner
**Head of Department of Computer Science & Engineering Department,*
H.B.T.I., Kanpur

***Abstract*—In this paper we discuss a modal of embedded Linux system support wireless network and its management. Wireless networks spread over a large physical area. Physical access of each node in other words signal sending place and accepting place is very difficult. To operates this type of networks there are some software are necessary and they are very difficulty to configure because of its cost. By using this system these type of problems are solved . The architecture is tailored to productive and extensive tested networks, in which reconfiguration is even more frequent. It is provides fallback solutions for configuration errors, and kernel panics. During the life time of a WMN it is necessary that all the communication is goes frequent.**

I INTRODUCTION

Wireless mesh networks (WMN) are embryonic to an significant access technology for broadband services. There are multiple deployments of WMN related to research, e.g. MIT Roofnet [1], [2], Orbit project [3], Microsoft Research [4], [5]. Furthermore, there are multiple cities which are currently deploying metropolitan area networks [6]. All these deployments cover geographically large areas. One can imagine that WMNs are deployed in hostile environments such as forests, deserts, or arctic regions. After deployment not all nodes may be physically accessible or the access may be very complicated and therefore costly

Updation of network according to the current requirement is very necessary .Some times these type of reconfiguration and update process is a cause t of failure of the network. The change of radio communication parameters can affect the physical topology of the network as well as cut off nodes from the network. It further provides the possibility to test configurations that are automatically reverted after a certain amount of time, in case of errors.

II SET UP OF NETWORK

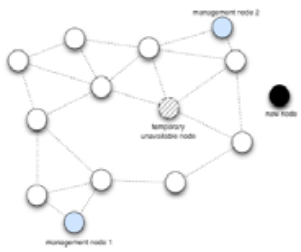

Fig 1Multiple nodes provide management functionalities for the network.

This type network consists number of node in other word multi-node. Each node receive same mode of frequency .But there is not necessary that each node receive signal. We can say some of node is switch off or out of range. when we reconfiguring network special attention must be pay for this type of node because this type of node create a problem. Functionalities of nodes can be accessed via a web interface. They could further provide tools, e.g., node image generators or a complete development environment. For this type configuration is done in following manner:-

(A)Configuration of neighbor node and software setup :- Each node is once in a while asking its neighbors for newer configurations and software. If updates are available, the node downloads them to its exchange storage. Neighbors of this node will download the updates from there. The downloaded configuration and software updates will be activated after a predefined time.

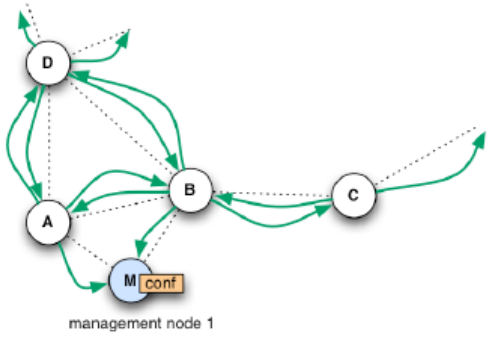

Fig2 Node check configuration of its neighbored.

A new configuration is injected at a management node (M) or a normal node. Nodes that have been down during the distribution of the updates will get the configurations and software updates from their neighbors as soon as they are up again.some time when we change the bandwidth or allotted range of transmission then if node is give the request to reconfiguration ,then it will create error. In order to guarantee the connectivity of the network after a reconfiguration, fall back solutions and checks are intended(which is shown in following figure)

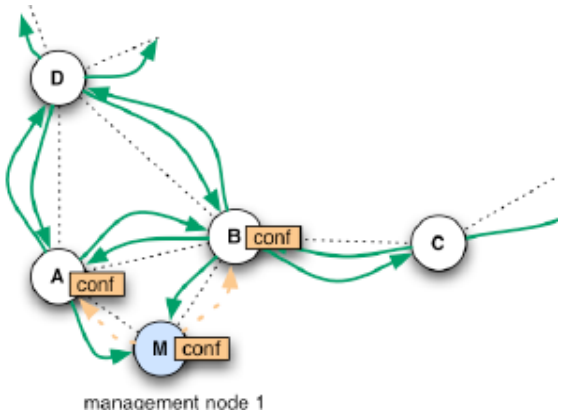

Fig 3 First nodes (A, B) get the update from node M.

If the user wishes to test a certain configuration, we introduce a temporary update feature in our architecture. The user generates and deploys a test configuration. He further defines a validity time for the new configuration. All nodes backup their configuration, before loading the new one. After the configuration has been fully distributed and set up in the network, a timer on each node is started. The user has now the possibility to check his test configuration. If it satisfies his needs, he can confirm it by sending a confirmation message to each node. The confirmation message stops the timer at the nodes. If the configuration is erroneous or the user did not confirm it, the old configuration will be loaded at the nodes. The network will operate in its last state again. Our architecture provides a safe way to upgrade the node's operating system. The update images are first checked for integrity by the help of hashes and checksums.. The system is now instructed to load the operating system only once from the update storage. On the next reboot it would load again from the default storage. If the software update succeeds and the node is up with the new operating system, the update can be made permanent by copying the updates to the default storage.

(B) *Incorporation of a New Node into the Network:-* A new node joins the network by first scanning for active communication channels. On the found channels it searches for IP networks, assigns itself an unused IP address and tries to load configurations from its neighbors. The node authenticates its communication peers with the help of the public keys in its storage. The same is done by the network nodes. They only provide configurations and software updates to known nodes.Therefore, the public key of the new node has to be distributed to all network nodes before the node can join the network All public keys are then loaded on all nodes at setup time.

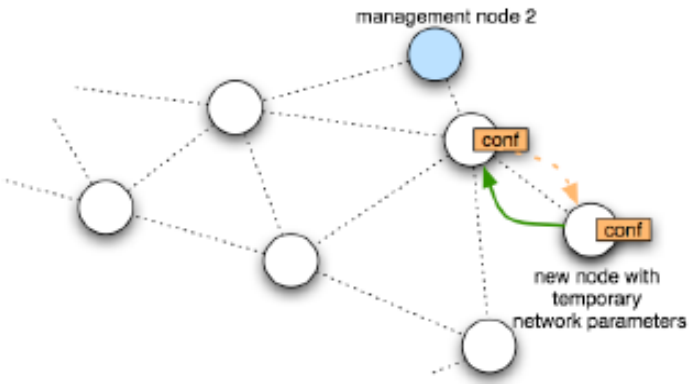

Fig5 Tempory Network for node from neighbors

III. HARDWARE REQUIRE FOR NETWOK CONNECTION

For our wireless mesh network we use the Wireless Router Application Platform (WRAP) from PCEngines [8]. Our nodes are WRAP2.C and its RoHS (EU restriction of the use of certain hazardous substances in electrical and electronic equipment) compliant successor board WRAP2.E. It is an embedded board with 233 MHz AMD Geode SC1100 CPU, 128MB RAM, Compact Flash card slot, one Ethernet port, two miniPCI sockets and one serial port. We have preferred WRAP to any Linksys Router based solution with OpenWrt [9] because of its ability to carry two wireless miniPCI cards. This enables multi-radio/multi-channel communication. Our nodes are equipped with two Atheros 802.11a/b/g miniPCI cards. We have further added a 3V Lithium coin cell as battery backup for the real time clock of the node.

IV LINUX FOR NETWORK PLATE FORM

In wireless networking ,current solutions are not sufficient .It needs more ,so its result is Linux. Our motive is to give the best performance of each node either it is connected or not .In other word it switch on or off. We have achieved this by using special software written for embedded systems. Our selection includes busybox [8] as a replacement of common UNIX utilities and uClibc [9] as small C library. Busybox is a well-known tool for small or embedded devices. It combines tiny versions of many common UNIX utilities

Busybox and uCube these s/w make the image easily extensible and customizable. Our image includes the following security features that are also described in Hardened Linux From Scratch (HLFS) book [10]:

- Position Independent Executable (PIE) [11]
- PaX [12]
- Grsecurity [13]
- Stack Smashing Protector (SSP) [14]

All above mention security features is the power of any network PIE uses the shared libraryand normal execution .It place Grsecurity very near to the high secure version of network.SSP is protect from the virus attack.

(a)Linux follow a path in network:

Linux follows a particular path in networks in which :-

(1)first booting process is going on.

(2)then individual node configure specify its state in networks.

(3)Update kernel through grub boot loader

V MANAGEN\MENT AND SAFE ATTACHMENT OF NODE

If a new network configuration for a certain node is desired, the user creates the configuration, e.g., with the management console. The new configuration file is copied by the management console (or manually) to the *exchange files* directory. Further, the user defines the wait cycles (intervall between two *cfagent* runs, in our case two minutes) until the configuration is permanent.

Node is classify according to the network state .If it is static state then first it change its state from dynamic to static and receive its IP address. After this the individual nodes save their current configuration and remove the *user interaction* file from previous updates. Further, they read the number of wait cycles to keep the new configuration before falling back to the old configuration.

A user can check the state of the network on every single node or over the web interface. If the network satisfies the user's requirements, he confirms the network to keep the current configuration. The confirmation message has to reach

all of the nodes before they counted down their own wait cycles (timer). If confirmed, the nodes set the current configuration to default, disable the timer, and remove the old configuration.

New node is attaches in plug and play system. and is automatically configured, if it has a working base image with the necessary keys. Image and keys can be generated by the management console. There is no configuration needed at the creation time of the image.

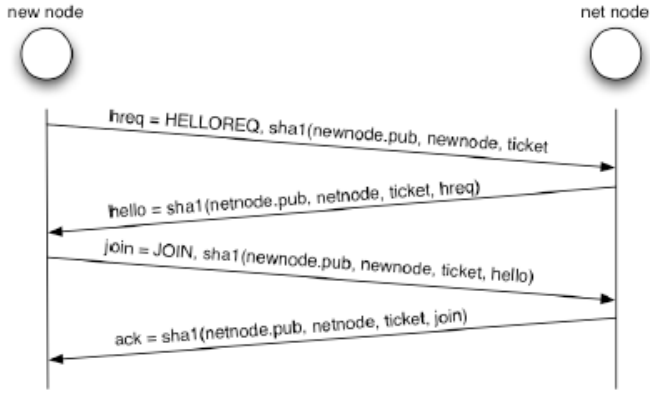

Fig 6 communication b/w node

VI CONCULSION

In above we discuss the architecture of wireless network and implementation of individual node in networks. .when any user is out of order for any reason like switch off or out of reach can easily attach to network and it is reconfigure at dynamically. The whole configuration is done in-band. It offers timed updates. A configuration can be tested and in case of errors the node reverts to the old configuration after a certain amount of time. A configuration can be tested and in case of errors the node reverts to the old configuration after a certain amount of time. Also show how we can use Linux to implement the network and monitor the change after applying it.

REFERENCE

[1] J. Bicket, D. Aguayo, S. Biswas, and R. Morris, "Architecture and evaluation of an unplanned 802.11b mesh network," in MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking. Cologne, Germany: ACM Press, August 2005, pp. 31–42.

[2] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris, "Linklevel measurements from an 802.11b mesh network," in International Conferences on Broadband Networks (BroadNets), 2004.

[3] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh, "Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network protocols," in WCNC 2005 IEEE Wireless Communications and Networking Conference, vol. 3, March 2005, pp. 1664 – 1669.

[4] R. Draves, J. Padhye, and B. Zill, "Routing in multi-radio, multi-hop wireless mesh networks," in 10th annual international conference on Mobile computing and networking MobiCom '04. Philadelphia, PA, USA: ACM Press, 2004, pp. 114–128.

[5] ——, "Comparison of routing metrics for static multi-hop wireless networks," in Conference on Applications, technologies, architectures, and protocols for computer communications SIGCOMM '04. Portland, Oregon, USA: ACM Press, August 2004, pp. 133–144.

[6] R. Karrer, A. Sabharwal, and E. Knightly, "Enabling large-scale wireless broadband: The case for taps." in 2nd Workshop on Hot Topics in Networks (Hot-Nets II, Cambridge, MA, November 2003.

[7] Research paper of Secure Remote Management and Software Distribution for Wireless Mesh Networks Thomas Staub, Daniel Balsiger, Michael Lustenberger and Torsten Braun.

[8] R. Landley, "Busybox," http://www.busybox.net, 2006.

[9] E. Andersen, "uclibc," http://www.ulibc.org, 2006.

[10] HLFS Development Team, "Hardened Linux From Scratch (HLFS)," http://www.linuxfromscratch.org/hlfs, 2006.

[11] J. Jelinek, "Position Independent Executable (PIE)," http://gcc.gnu.org/ml/gcc-patches/2003-06/msg00140.html, June 2003.

[12] PaX Project, "Pax," http://pax.grsecurity.net/, 2006.

[13] B. Spengler, "Grsecurity," http://www.grsecurity.net/, 2006.

[14] H. Etoh, "Stack Smashing Protector (SSP)," http://www.trl.ibm.com/ projects/security/ssp/, August 2005.