An Efficient and Robust Addressing Protocol for Node Auto configuration in Ad Hoc Networks

S.KIRUBA CONSTANCE,
M.E II YEAR,
Department Of Computer Science And Engineering,
GOVERNMENT COLLEGE OF ENGINEERING,
TIRUNELVELI-627007

G. ANITHA,
Assistant Professor,
Department Of Computer Science And Engineering,
GOVERNMENT COLLEGE OF ENGINEERING,
TIRUNELVELI-627007

ABSTRACT:

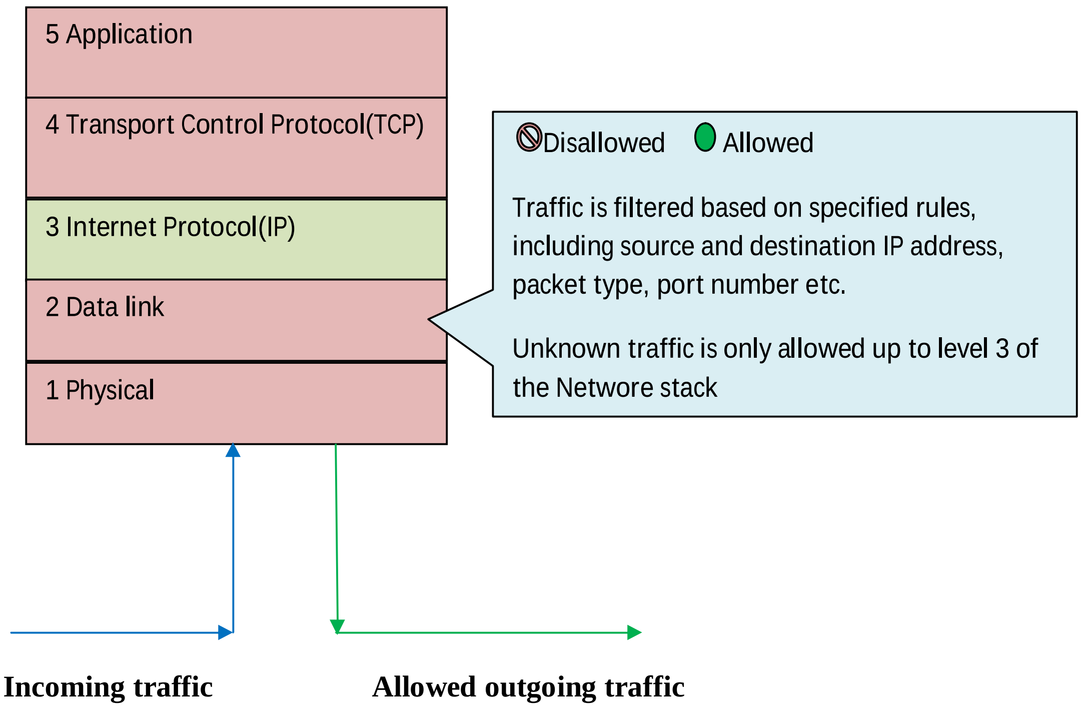
Due to the lack of infrastructure in ad hoc network address assignment is a key challenge. Autonomous addressing protocols are required to provide the address. To avoid address collisions in a dynamic network with fading channels, frequency partitions, and joining/leaving nodes, it requires a distributed and self-managed mechanism. This paper proposes and analyzes a lightweight protocol, which helps in configuring mobile ad hoc nodes based on a distributed address database stored in filters. It reduces the control load and makes the proposal robust to packet losses and network partitions.

Index terms: Ad-hoc networks, Addressing mechanisms, IP Address configuration

INTRODUCTION:

Due to the lack of a centralized administration in adhoc network, it makes these networks attractive for several distributed applications, such as sensing, Internet access to deprived communities, and disaster recovering. Network partitions, caused by node mobility, fading channels [1], and nodes joining and leaving the network, can disrupt the distributed network control. Usually unaddressed issue of ad hoc networks is the frequent network partitions. Address assignment in ad hoc networks, is even more challenging due to the self-organized nature of these environments. This paper, propose and analyze an efficient approach called Filter-based Addressing Protocol (FAP). Maintains a distributed database stored in filters containing the currently allocated addresses in a compact fashion.

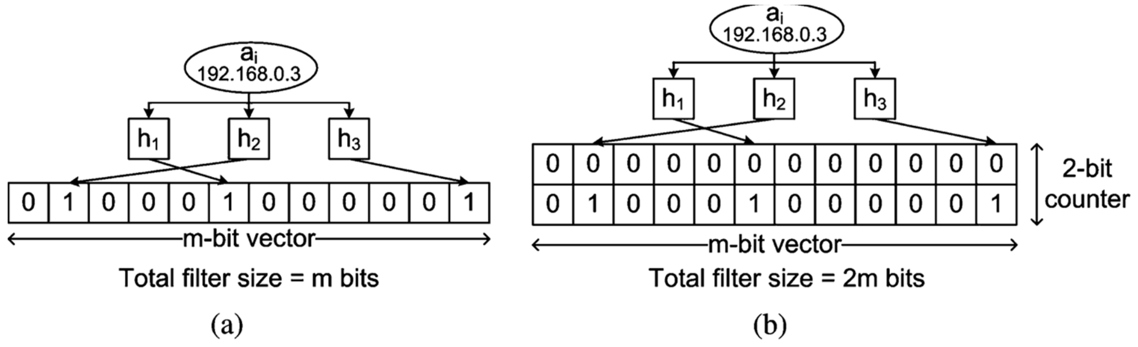
FAP:
Its aim is to reduce the control load and to improve partition merging detections without requiring high storage capacity and dynamically auto-configure network addresses. These objectives are achieved through small filters and an accurate distributed mechanism to update the states in nodes. It uses the filter signature (i.e., a hash of the filter) as a partition identifier instead of random numbers. The filter signature represents the set of all the nodes within the partition. Therefore, if the set of assigned address changes, the filter signature also changes.
This filter is preseon at every node to simplify frequent node joining events and reduce the control overhead required to solve address collisions inherent in random assignments.


5 Application
4 Transport Control Protocol(TCP)
3 Internet Protocol(IP)
2 Data link
1 Physical
Disallowed
Allowed
Traffic is filtered based on specified rules, including source and destination IP address, packet type, port number etc.
Unknown traffic is only allowed up to level 3 of the Networe stack
Incoming traffic
Allowed outgoing traffic

It maintains a distributed database stored in filters containing the currently allocated addresses in a compact fashion. Two different filters, depending on the scenario: the Bloom filter- based on hash functions, the Sequence filter- compresses data based on the address sequence. Filter signature, which is the hash of the address filter, as a partition identifier. The filter signature is an important feature for easily detecting network, merging events, in which address conflicts may occur

Bloom Filters

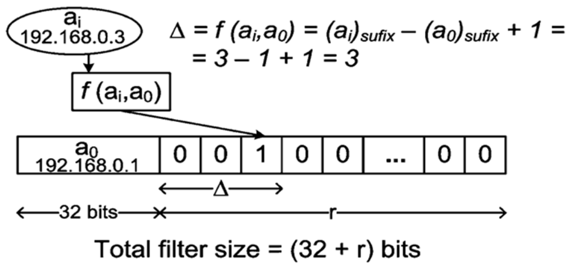
The Bloom filter is composed of a n -bit vector that represents a set composed of elements. The elements are inserted into the filter through a set of independent hash functions, whose outputs are uniformly distributed over the m bits. First, all the bits of the vector are set to zero. If at least one bit is set to 0, then, is not on the filter. Otherwise, it is assumed that the element belongs to. There is, a false-positive probability that an element be recognized as being in. The false-positive probability is given by the probability of these bits not being 0, which means that $P_{FP}= (1-P_0)^{k}$ k minimizes the false-positive probability, which is given by $k=\frac{m.ln2}{n}$Bloom filters do not present false negatives, which means that a membership test of an element that was inserted into the filter is always positive.



(a) Bloom filter with k=3 hash functions (b) 2 bit counter bloom filter

Sequence Filters

It stores and compacts addresses based on the sequence of addresses. This filter is created by the concatenation of the first address of the address sequence, which we call initial element (a_0), with a r-bit vector, where r is the address range size.


a_i
192.168.0.3
$\Delta = f(a_i,a_0) = (a_i)_{sufix} - (a_0)_{sufix} + 1 =$
$= 3 - 1 + 1 = 3$
$f(a_i,a_0)$
a_0
192.168.0.1
0 0 1 0 0 ... 0 0
Δ
32 bits
r
Total filter size = (32 + r) bits

In this filter, each address suffix is represented by one bit, indexed by Δ, which gives the distance between the initial element suffix ($a_{0\text{suffix}}$) and the current element suffix (suffix). If a bit is in 1, then the address with the given suffix is considered as inserted into the filter, otherwise, the bit in 0 indicates that the address does not belong to the filter. Therefore, there are neither false positives nor false negatives in the Sequence filter because each available address is represented by its respective bit. Sequence filter is deterministic and, as a consequence, neither false positives nor false negatives are created. The size of the Sequence filter depends only on the size of the address range, r,and on the address sizeA_s

MODULES:

Network Initialization:

Two different scenarios can happen in the initialization:

Joining nodes arrive one after the other with a long enough interval between them, called gradual initialization, or all the nodes arrive at the same time, called abrupt initialization. FAP fits well for both gradual and abrupt initialization scenarios, using Hello and AREQ messages. The first node is alone to choose a partition identifier. The following joining nodes are handled by the first node through the joining node procedure. If all nodes join the network approximately at the same time, it may cause inconsistencies in the address allocation procedure. The Hello message is used by a node to advertise its current association status and partition identifier. The AREQ message is used to advertise that a previously available address is now allocated. Each AREQ has an identifier number, which is used to differentiate AREQ messages generated by different nodes, but with the same address.

Node Joining:

In FAP, a node trying to join the network listens to the medium for a period. If the node does not receive a Hello message within this period, then it starts the network, acting as the initiator node. If the node receives a Hello message, then the network already exists and the node acts as a joining node.

Address Assignment:

An initiator node randomly chooses an address, considering the address range defined by the bits of the network prefix, creates an empty address filter, and starts the network initialization phase. In this phase, the node floods the network N_f times with AREQ messages to make all initiator nodes to receive the AREQ message. After waiting a period T_w the node leaves the initialization phase and inserts in the address filter all the addresses received with AREQs**.** Then the node starts to send Hello messages with the address filter signature, which is a hash of the filter. This signature identifies the network and is used to detect partitions. If the initiator node receives any AREQ with the same address that it has chosen, but with a different identifier number, then there is an address collision, the node waits for a T_c period and then chooses another available address and sends another AREQ. After T_c probability of choosing a used address reduces which decreases the probability of collisions and, consequently, reduces network control load.

Network Merging Events:

After receiving Hello, neighbors evaluate whether the signature in the message is the same as its own signature to detect merging events. The joining node, then asks for the source of the first listened Hello message (the host node) to send the address filter of the network using an Address Filter (AF) message. When the host node receives the AF, it checks bit I, which indicates whether the message is being used for a node-joining procedure or a partition-merging procedure. If I=1, the message came from a joining node. Then, the host node answers the request with another AF with bit R set to 1, indicating that the AF is an answer to a previous filter request. When the joining node receives the AF reply message, it stores the address filter, chooses a random available address, and floods the network with an AREQ to allocate the new address which in turn updates the filter message of the other nodes.

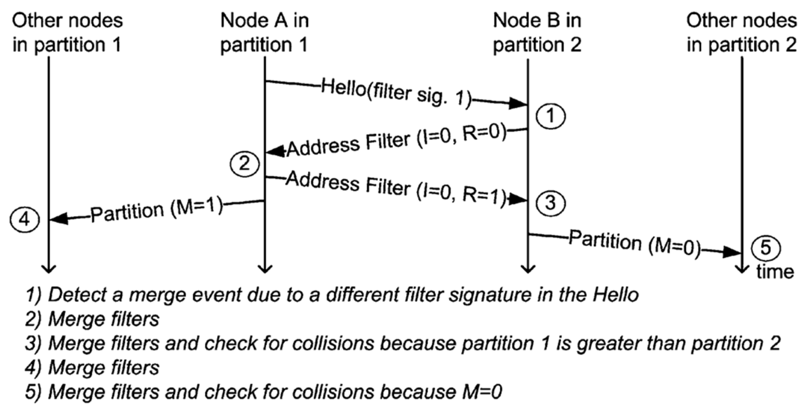
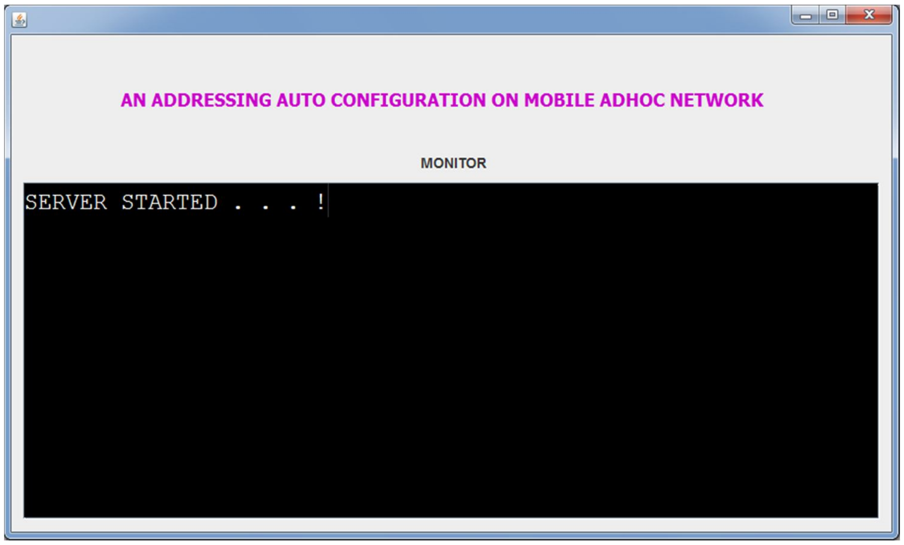

Other nodes in partition 1
Node A in partition 1
Node B in partition 2
Other nodes in partition 2
Hello(filter sig. 1)
Address Filter (I=0, R=0)
Address Filter (I=0, R=1)
Partition (M=1)
Partition (M=0)
time
1) Detect a merge event due to a different filter signature in the Hello
2) Merge filters
3) Merge filters and check for collisions because partition 1 is greater than partition 2
4) Merge filters
5) Merge filters and check for collisions because M=0

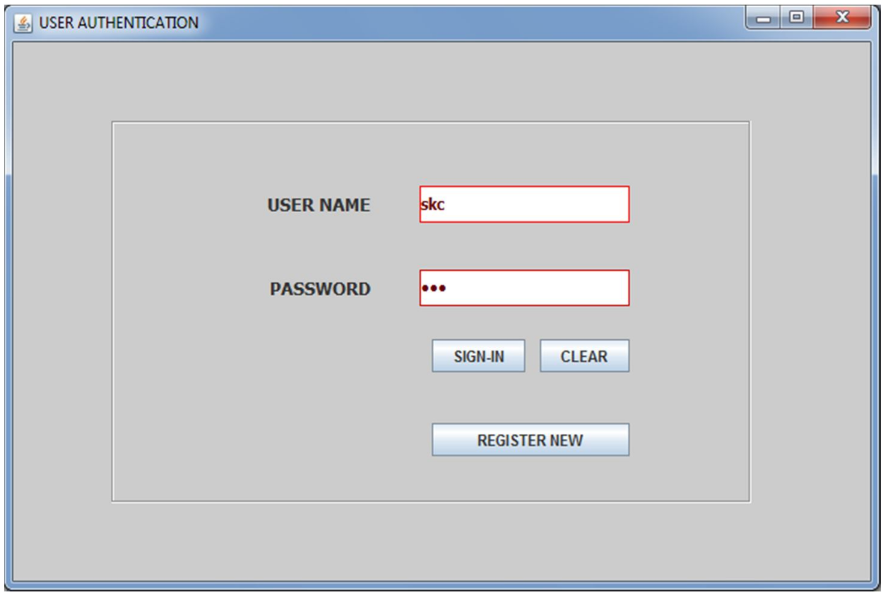
Fig: FAP scheme for node ingress

When a node leaves the network, its address should become available to the other nodes. If it is correctly shut down, it floods the network with a notification to remove its address from the address filter if not, the address remains allocated in the filters, causing scarcity of address. This can be identified in the address filter by the fraction of bits set to 1 in the Bloom and in the Sequence filter and by the fraction of counters greater than one in the Counter Bloom Filter. Every node verifies this fraction in their address filters every time the filter is updated. If this fraction reaches a threshold that indicates that the filter is full or almost full, all the nodes reset their address filters and returns to the network initialization. Instead of choosing a new address, the node uses its current address, which is not collided, to reduce message overhead and to avoid breaking active data connections. To avoid frequent address filter renews, there is a minimum period between filter renews, defined as $T_{M..}$ FAP uses timers to avoid that a leaving node inconsistency in the address filter.

RESULTS:

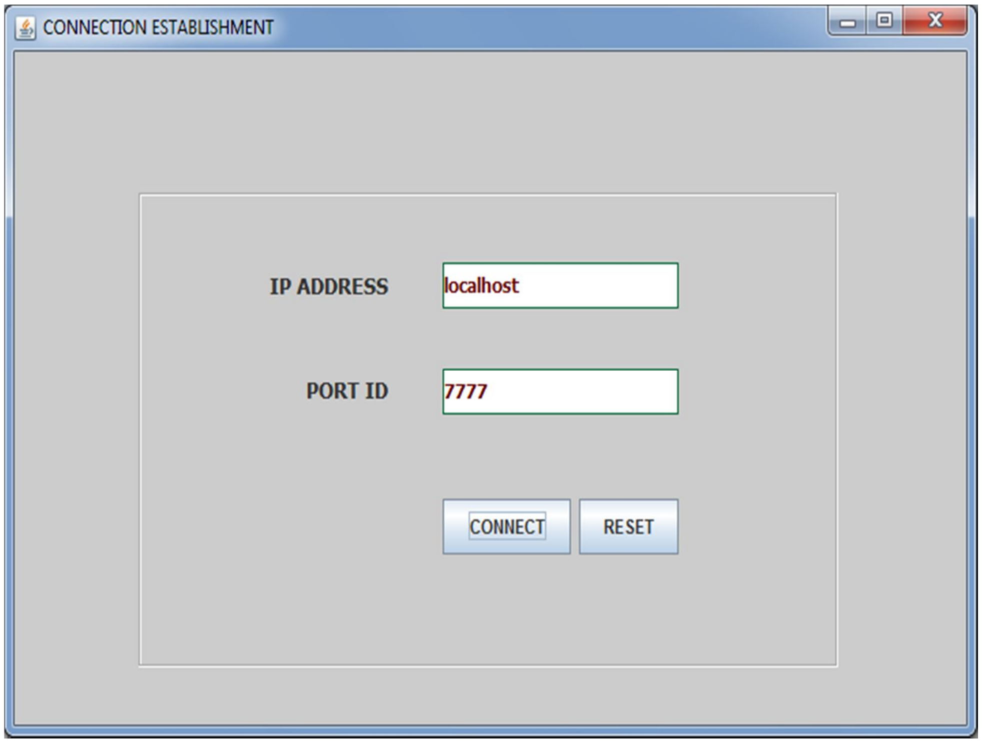


```
SERVER STARTED . . . !
```



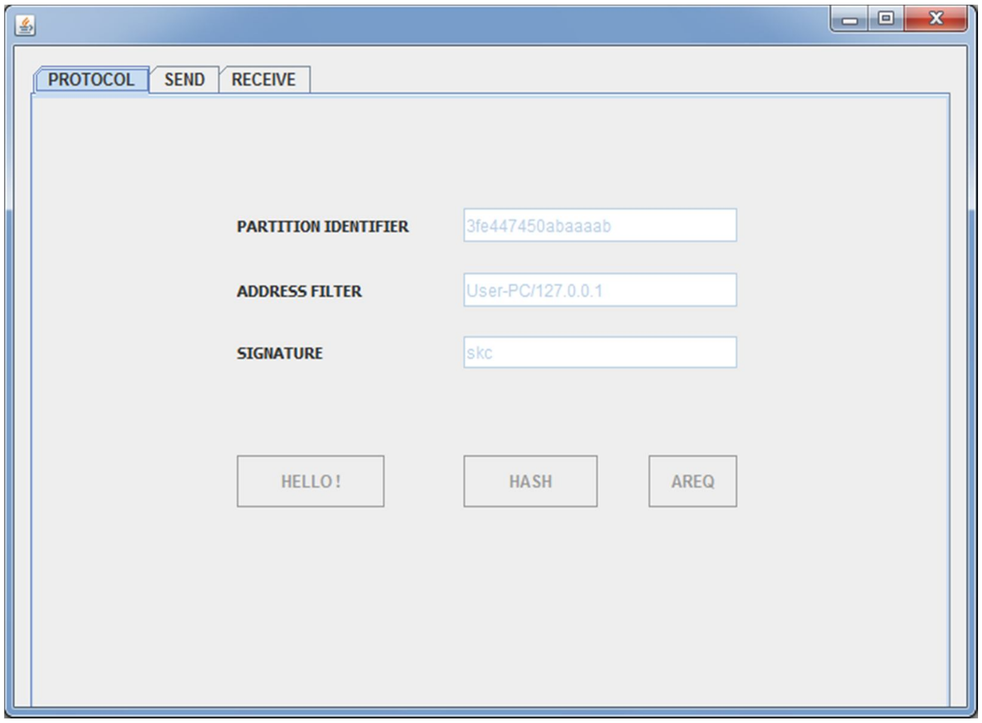
USER NAME skc

PASSWORD •••



IP ADDRESS localhost

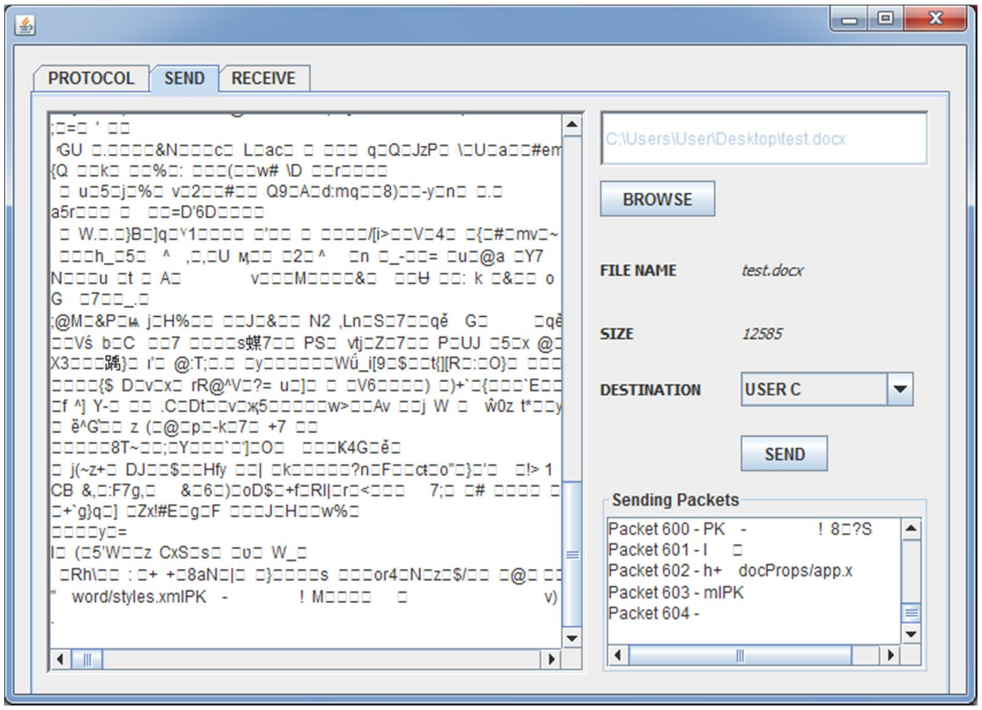
PORT ID 7777



PARTITION IDENTIFIER 3fe447450abaaaab

ADDRESS FILTER User-PC/127.0.0.1

SIGNATURE skc



```
;□=□ ' □□
 'GU □.□□□□&N□□□c□ L□ac□ □ □□□ q□Q□JzP□ \□U□a□□#em
{Q □□k□ □□%□: □□□(□□w# \D □□r□□□□
 □ u□5□j□%□ v□2□□#□□ Q9□A□d:mq□□8)□□-y□n□ □.□
a5r□□□ □  □□=D'6D□□□□
 □ W.□.□}B□]q□Y1□□□□ □'□□ □ □□□□/[i>□□V□4□ □{□#□mv□~
 □□□h_□5□  ^  ,□,□U M□□ □2□^  □n □_-□□= □u□@a □Y7
N□□□u □t □ A□        v□□□M□□□□&□  □□U □□: k □&□□ o
G  □7□□_.□
;@M□&P□м j□H%□□ □□J□&□□ N2 ,Ln□S□7□□qě  G□    □qě
□□Vś b□C  □□7 □□□□s蝶7□□ PS□ vtj□Z□7□□ P□UJ □5□x @□
X3□□□鶸}□ r'□ @:T;□.□ □y□□□□□□Wú_i[9□$□□t{][R□:□O}□ □□□
□□□□{$ D□v□x□ rR@^V□?= u□]□ □ □V6□□□□) □)+`□{□□□`E□□
□f ^] Y-□ □□ .C□Dt□□v□ж5□□□□□w>□□Av □□j W □  ŵ0z t*□□y
□ ě^G□□ z (□@□p□-k□7□ +7 □□
□□□□□8T~□□;□Y□□□`□']□O□  □□□K4G□ě□
□ j(~z+□ DJ□□$□□Hfy □□| □k□□□□□?n□F□□ct□o"□}□'□  □!> 1
CB &,□:F7g,□   &□6□)□oD$□+f□Rl|□r□<□□□    7;□ □# □□□□ □
□+`g}q□] □Zx!#E□g□F □□□J□H□□w%□
□□□□y□=
I□ (□5'W□□z CxS□s□ □o□ W_□
 □Rh\□□ :□+ +□8aN□|□ □}□□□□s □□□or4□N□z□$/□□ □@□ □□
"  word/styles.xmlPK  -         ! M□□□□   □              v)
```

C:\Users\User\Desktop\test.docx

FILE NAME test.docx

SIZE 12585

DESTINATION USER C

```
Packet 600 - PK  -          ! 8□?S
Packet 601 - l   □
Packet 602 - h+   docProps/app.x
Packet 603 - mlPK
Packet 604 -
```

REFERENCES

[1] R. Droms, “Dynamic Host Configuration Protocol”, Network Working Group, IETF RFC 2131, March 1997.

[2] S. Cheshire, B. Aboba and E. Guttman, “Dynamic Configuration of IPv4 Link-Local Addresses”, Network Working Group, IETF RFC 3927, March 2005.

[3] M. Günes and J. Reibel, “An IP Address Configuration Algorithm for Zeroconf Mobile Multihop Ad Hoc Networks,” *Proc. Int’l. Wksp. Broadband Wireless Ad Hoc Networks and Services*, Sophia Antipolis, France, Sept. 2002.

[4] S. Nesargi and R. Prakash, “MANETconf: Configuration of Hosts in a Mobile Ad Hoc Network,” *Proc. IEEE INFOCOM 2002*, New York, NY, June 2002.

[5] H. Zhou, L. M. Ni, and M. W. Mutka, “Prophet Address Allocation for Large Scale Manets,” *Proc. IEEE INFOCOM 2003*, San Francisco, CA, Mar. 2003.

[6] M. Mohsin and R. Prakash, "IP Address Assignment in a Mobile Ad Hoc Network," *Proc. IEEE MILCOM 2002*, Anaheim, CA, Oct. 2002.

[7] C. Perkins Charles Perkins, Jari Malinen, Ryuji Wakikawa, Yuan Sun and Elizabeth M. Belding-Royer, "IP Address Autoconfiguration for Ad Hoc Networks," IETF draft, 2001.

[8] N. H. Vaidya, "Weak Duplicate Address Detection in Mobile Ad Hoc Networks," *Proc. ACM MobiHoc 2002*, Lausanne, Switzerland, June 2002, pp. 206–16.

[9] K. Weniger, "Passive Duplicate Address Detection in Mobile Ad Hoc Networks," *Proc. IEEE WCNC 2003*, New Orleans, LA, Mar. 2003.

[10] J. Jeong, J. Park, H. Kim, H. Jeong and D. Kim, "Ad Hoc IP Address Autoconfiguration," IETF draft, August 2005 (work in progress).