

This work has been submitted to **NECTAR**, the **Northampton Electronic Collection of Theses and Research**.

**Patent**

**Title:** Method and process for routing and node addressing in wireless mesh networks

**Creators:** Al-Sherbaz, A., Jasim, S., Lami, I. and Adams, C.

**Example citation:** Al-Sherbaz, A., Jasim, S., Lami, I. and Adams, C. (2011) *Method and process for routing and node addressing in wireless mesh networks*. UK. 5 October 2011. Patent number GB2479136 (A).

**Version:** Published version

**Official URL:** <https://www.ipo.gov.uk/p-ipsum/Case/PublicationNumber/GB2479136>

<http://nectar.northampton.ac.uk/4226/>



(21) Application No: **1005254.6**  
 (22) Date of Filing: **30.03.2010**

(51) INT CL:  
**H04W 8/26** (2009.01) **H04L 12/56** (2006.01)  
**H04L 29/12** (2006.01) **H04W 40/24** (2009.01)

(71) Applicant(s):  
**Ali Habib Al-Jasmin Al-Sherbaz**  
**Hunter Street, Applied Computing Department,**  
**The University of Buckingham, BUCKINGHAM,**  
**MK18 1EG, United Kingdom**  
  
**Sabah Jasim**  
**Hunter Street, Applied Computing Department,**  
**The University of Buckingham, BUCKINGHAM,**  
**MK1 8EG, United Kingdom**  
  
**Ihsan Alshahib Lami**  
**Hunter Street, Applied Computing Department,**  
**The University of Buckingham, BUCKINGHAM,**  
**Buckinghamshire, MK18 1EG, United Kingdom**  
  
**Chris Adams**  
**Hunter Street, Applied Computing Department,**  
**The University of Buckingham, BUCKINGHAM,**  
**MK18 1EG, United Kingdom**  
  
**University of Buckingham**  
**Applied Computing Department, Yeomanry House,**  
**Hunter Street, BUCKINGHAM, MK18 1EG,**  
**United Kingdom**

(56) Documents Cited:  
**US 20100031356 A1** **US 20090003356 A1**  
**US 20060198320 A1**

(58) Field of Search:  
 INT CL **H04L, H04W**  
 Other: **WPI EPODOC**

(continued on next page)

(54) Title of the Invention: **Method and process for routing and node addressing in wireless mesh networks**  
 Abstract Title: **Routing and node addressing in wireless mesh networks**

(57) The algorithm proposed in this patent enables any node in a Wireless Mesh Network (WMN) to have knowledge of all intermediate nodes, in all the possible-routes towards the destination node. The algorithm uses a novel recursive algorithm to accumulate knowledge beyond the neighbouring nodes, as well as the sequence of all the intermediate nodes used to form these routes. Prime numbers are used in the address of the host portion of the WMN node IP address to identify the optimum route.

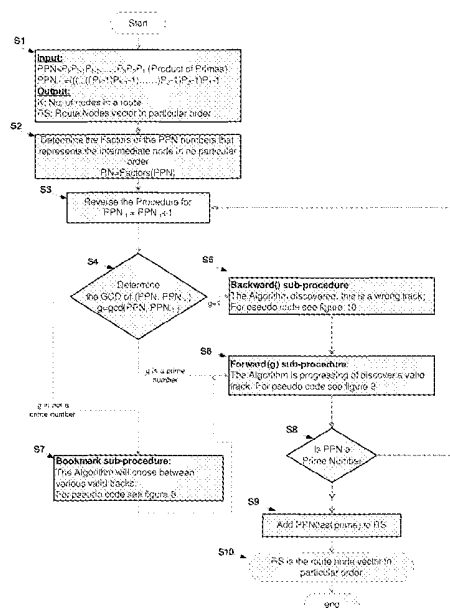


Figure 7

**GB 2479136 A continuation**

(72) Inventor(s):

**Ali Habib Al-Jasmin Al-Sherbaz  
Sabah Jasim  
Ihsan Alshahib Lami  
Chris Adams**

(74) Agent and/or Address for Service:

**Ali Habib Al-Jasmin Al-Sherbaz  
Hunter Street, Applied Computing Department,  
The University of Buckingham, BUCKINGHAM,  
MK18 1EG, United Kingdom**

## Method and Process for Routing and Node Addressing in Wireless Mesh Networks

### **Description:**

The invention proposed in this patent is named Prime-IP. Prime-IP is a novel routing algorithm for Wireless Mesh Networks (WMN).

Prime-IP achieves higher Quality of Service than standard WMN implementations because it will always choose the optimum route path between the source node and the destination node. i.e. Prime-IP achieves more reliable routes, less traffic processing overhead, higher security level, increased data throughput, and reduced error rate.

Furthermore, Prime-IP produces unique routing path where each individual node are identified, and where the route can be classified by each of these individual nodes. i.e. with Prime-IP, each node will have knowledge of not only the neighbouring nodes, but also nodes that are beyond their neighbours nodes. Consequently, Prime-IP builds, at each node level, a dynamic knowledge database (or map) of all other nodes in the WMN.

Prime-IP achieves this by:

- (a) Assigning a unique prime number in the host-portion of the IP-Address of each individual node.
- (b) Packs two extra number fields in the Route Reply Packet (RREP) named  $PPN_1$  and  $PPN_2$ . The value of these two fields will be calculated dynamically during the route reply discovery stage.
- (c) The values of  $PPN_1$  and  $PPN_2$  are calculated from the prime numbers allocated to the nodes in the WMN, starting with the destination node (the initial value of  $PPN_1 =$  "destination node prime number", while  $PPN_2 =$  "destination node prime number" - 1). Thereafter, as RREP get forwarded by the destination node to the neighbouring nodes,  $PPN_1$  and  $PPN_2$  values change to ( $newPPN_1 =$

previousPPN<sub>1</sub> x CurrentNodePrimeNumber) and (newPPN<sub>2</sub>= (previousPPN<sub>2</sub> x CurrentNodePrimeNumber) - 1). This process continues for the next intermediate nodes until the routes reach the source node.

- (d) Based on the values of received PPN<sub>1</sub> and PPN<sub>2</sub> from the various possible paths between the destination and the source nodes, the source node then uses a backtrack procedure to construct the intermediate nodes vector in a particular order for each of the received RREPs. This then is used to select the optimum available route out of all the possible path options.

Typical WMN are classified to three categories as Infrastructure WMNs, Client WMNs and Hybrid WMNs. The intermediate nodes in all these categories, using current conventional routing algorithms, do not accumulate knowledge beyond their nearest neighbouring nodes. Prime-IP can be applied to all of these categories of WMNs and therefore all nodes shall have knowledge beyond their neighbouring nodes.

### **Examples:**

The following 3 examples are designed to illustrate the working (method and process) of Prime-IP in (a) assigning the Prime Number, (b) Calculating PPN<sub>1</sub> and PPN<sub>2</sub>, and (c) construct the optimum route using the backtrack procedure. In these examples the following Diagrams and Tables are used:

**List of Figures:**

Figure 1 shows a diagram of a general WMN topology,

Figure 2 shows a diagram of a general Client WMN topology (sometimes called Mobile Ad-Hoc Networks (MANET)),

Figure 3 shows a diagram of a random WMN topology with a prime number address ( $P_1, P_2, \dots, P_i, \dots, P_k$ ) assigned to every node.

Figure 4a shows the host portions (H bits) and the network portions in the IPv4 and IPv6 address format,

Figure 4b shows how prime numbers are embedded in the IP addresses (Ipv4 and Ipv6) with different host portion sizes (H bits),

Figure 5a, 5b and 5c shows the diagram from Figure 3, but with actual Prime number assigned to all node address ( $P_1 \dots P_k$ ), and highlighting 3 route examples,

Figure 5d shows a tree diagram of how the “backtrack procedure” is applied to determine the track which represents the intermediate nodes vector in that particular order,

Figures 6a, 6b and 6c show examples of constructing and deconstructing of the “Product of the Prime Numbers” ( $PPN_1$  and  $PPN_2$ ) for the routes highlighted in figures 5a, 5b and 5c respectively,

Figure 7 shows a flow chart of the overall Backtrack procedure,

Figure 8 shows a pseudo-code description of the Bookmark sub-procedure,

Figure 9 shows a pseudo-code description of the Forward sub-procedure, and

Figure 10 shows a pseudo-code description of the Backward sub-procedure.

### **Figures Explanation:**

In figure 1, a general WMN has been construct by using three different domains; Internet 1, Mesh Route Domain (MRD) 2 and Mesh Client Domain (MCD) 3. The MRD, contains “mesh routers” nodes 4 which is equipped with high processing and memory capabilities. Some of the “mesh routers” are also called gateways 5, which are special wireless routers with high-bandwidth wired connection to the Internet. In the MCD, the “mesh clients” 6 are mobile nodes. The links between the Internet 1 and the MRD through the gateways 5 are wired connections 7. The links between the MRD and MCD are wireless connections 8.

In figure 2, a general Client Wireless Mesh Network (also called Mobile Ad-Hoc Network (MANET)) 9 has been illustrated; it is a number of mobile nodes in random topology without base-station or access point. The mobile nodes can be classified to senders 6a, destinations 6b and routers 4 which are dynamically changed upon their instant functionality.

In Figure 3, assuming there is a WMN network with random topology, each circle represents an individual wireless node 5, 6a and 6b. The lines between these circles are the bi-directional links between two nodes 8. Finally, all nodes have been assigned a unique prime number as described below.

In Figure 4a, both IP address versions (IPv4 and IPv6) are illustrated with their host portion 10 and network portions 11, 12 respectively.

## Prime-IP Description

The Prime-IP is designed for a dynamic network topology (WMN) where the topology and membership (nodes' association/re-association) may change at any time. It is based on the "*Fundamental theory of Arithmetic*" that states:

- (a) Every positive integer (except the number 1) can be represented in one way only apart from rearrangement as a product of one or more primes.
- (b) Every natural number is either prime or can be uniquely factored as a product of primes in a unique way.

To devise a general formula for calculating  $PPN_1$  and  $PPN_2$  in Prime-IP, it is assumed that any route is represented by a number of nodes starting in sequence from source node being  $P_1$  till the destination node  $P_d$  with any variable number of nodes in between  $P_1, P_2, \dots, P_{d-1}, P_d$ .

NB, the use of the term "route" signify a definitive physical intermediate nodes between the source and destination nodes, while term "path" is used to signify any possible route via any combination of physical intermediate nodes.

Also, assume  $P_i \in RN$ ,

Where:  $P_i$  is an arbitrary prime number,

$RN$  is a set of prime numbers

where  $RN = \{P_1, P_2, P_3, \dots, P_i, \dots, P_{k-2}, P_{k-1}, P_k\}$ , in ascending order,

$RS$  is an arbitrary set of prime,

where  $RS \subseteq RN$ ,

$P_k$  = largest prime number in  $RN$ ,

$1 \leq i \leq k$ ,  $i$  is an integer number,

$k$  = total of prime numbers in  $RN$  set,

$RS = \{P_j\}$ , where  $1 \leq j \leq d$ ,  $d \leq k$ , and



$d$  = total of prime numbers in RS set (intermediate nodes)

Then

$PPN_1 = \text{Product } (P_d P_{d-1} P_{d-2} \dots P_3 P_2 P_1)$ , and

Factors  $(PPN_1) \Leftrightarrow \{ P_d P_{d-1} P_{d-2} \dots P_3 P_2 P_1 \}$

And

$PPN_2 = (((\dots ((P_d - 1) P_{d-1} - 1) \dots) P_3 - 1) P_2 - 1) P_1 - 1$ ,

Or, put in a series format,

$$PPN_2 = (P_d P_{d-1} P_{d-2} \dots P_3 P_2 P_1) - (P_{d-1} P_{d-2} \dots P_3 P_2 P_1) - \dots - (P_3 P_2 P_1) - (P_2 P_1) - (P_1) - 1$$

i.e. in general, and substituting for the value of  $PPN_1$  in  $PPN_2$ ,

$$PPN_2 = PPN_1 - \sum_{i=d-1}^1 [\prod_{j=1}^i (P_j)] - 1$$

This shows that  $PPN_1$  will always be greater than  $PPN_2$

In conclusion, therefore, there is one and only one:

1. Factors-set (RS) for each  $PPN_1$ .
2.  $PPN_1$  is a product of the RS elements set.

However, these operations do not produce the list of prime factors in any particular order. Prime-IP produces the list of prime factors in a particular order, which is the same as the sequence of the factors order produced by the constructing process.

Figure 3 shows that all nodes have been assigned a unique prime number as an address  $P_i$  {where  $P_i = P_1 \dots P_k$ }. Therefore, in order to have a route from a source node to a destination node through intermediate routing nodes, the source node issues “route request packet” (RREQ). A flooding process is then ensued in various paths until the destination node is reached. As soon as the destination node gets this

request, a “route reply packet” (RREP) shall be returned to the source node via various path options. Finally, the source node establishes the optimum route from these available route options. i.e. in Figure 3, for example, a source node can be  $P_{16}$  which issues a RREQ to reach  $P_1$  as a destination node. There are various routes that could be selected to do this, such as

$$RS_1 = \{P_1, P_4, P_6, P_5, P_9\},$$

$$RS_2 = \{P_1, P_4, P_3, P_8, P_{k-6}\},$$

$$RS_3 = \{P_1, P_{k-2}, P_{15}, P_{10}, P_9\},$$

Etc.

i.e. the  $P_1$  responds by an RREP and puts its prime number address in the reserved fields  $PPN_1$  and  $PPN_2$ . So, all the intermediate nodes between  $P_1$  and  $P_{16}$  shall replace the value of  $PPN_1$  and  $PPN_2$  by (a) multiplying their prime number address with the existing value of the  $PPN_1$ , and (b) multiplying their prime number address with the existing value of the  $PPN_2$  then subtracting 1 from  $PPN_2$ . Finally,  $P_{16}$  will receive various values of  $PPN_1$  and  $PPN_2$  dependent on the nodes that RREP path passes through. i.e. the value of  $PPN_1$  &  $PPN_2$  in the RREP for  $RS_1$  shall be

$$PNN_1 = P_1 \times P_4 \times P_6 \times P_5 \times P_9$$

$$PPN_2 = (((((P_1-1) \times P_4-1) \times P_6-1) \times P_5-1) \times P_9-1), \text{ or}$$

$$= (P_1 \times P_4 \times P_6 \times P_5 \times P_9) - (P_4 \times P_6 \times P_5 \times P_9) - (P_6 \times P_5 \times P_9) - (P_5 \times P_9) - (P_9) - 1$$

As this example demonstrates, the use of prime numbers as an address is unique, and shall provide unique identification of all the nodes, in all the possible paths in any routing discovery process (include both node-IP's and sequence).

NB. The possible prime number assignment is however limited to, for example, 54 prime numbers in any 8-bit addressable field (where all possible numbers are 255). This should not limit Prime-IP because the host portion of IP address field can be extended to 16, 24 and up to 64 bits in IPv6.

Furthermore, the source node will accumulate information about all possible intermediate nodes to the destination node (generated by Prime-IP backtrack procedure, using only two variables  $PPN_1$  and  $PPN_2$ ). Therefore, Prime-IP potentially can generate a dynamic map of the entire WMN.

### **Prime- IP and IP Addresses:**

Figure 4a shows both IP address versions (IPv4 and IPv6) with their host portion and network portions. The length of the IPv4 is 32 bits (The host portion of the IP address is 2-24 bits, while the remainder bits are used for the network portion). The length of the IPv6 address is 128 bits allowing for the host portion to be up to 64 bits.

Figure 4b illustrates an example of arbitrary prime numbers selection. These are chosen for different host portion lengths for both IPv4 and IPv6. i.e.

- For 8 bits host portion, there are only 54 prime numbers that are possible (total numbers = 256 or  $2^8$ ). For instance, 5 and 239 are prime numbers are converted to 8 bits binary as (00000101) and (11101111) respectively. i.e. to generate the Prime-IP addresses (x.x.x.5 and x.x.x.239) for IPv4 and (xx...5 and xx...EF) for IPv6. Note that “x” in the above IP addresses represents the network portion number.
- For 16 bits host portion, there are about 6000 possible prime numbers, out of total numbers of 65,536 ( $2^{16}$ ). For instance, 313 and 51,449 are prime numbers are converted to 16 bits binary as (00000001-00111001) and (11001000-11111001) respectively. This is to generate the Prime’s IP

addresses (x.x.1.57 and x.x.200.249) for IPv4 and (xx...0139 and xx...C8F9) for IPv6.

- For host portion using 24 bits, there are around one million prime numbers that can be used ( $2^{24} = 16,777,216$ ). For instant, 2,051,773 and 12,004,991 are prime numbers and are converted to 24 bits binary as (00011111-01001110-10111101) and (10110111-00101110-01111111) respectively. Thus generating the Prime's IP addresses (x.31.78.189 and x.183.46.127) for IPv4 and (xx...1F4EBD and xx...B72E7F) for IPv6.
- In the host portion of 48 bits, there are around eight trillion prime numbers out of ( $2^{48}$ ) numbers. For instant, 9,990,454,997 and 281,474,076,384,103 are prime numbers which are converted to 48 bits in hexadecimal as (0002537A3ED5)<sub>hex</sub> and (FFFFCA561B67)<sub>hex</sub> , respectively, to generate the Prime's IP addresses (xx... 0002537A3ED5 and xx... FFFCA561B67) for IPv6. There is no entry for IPv4 in table 6b because it is 32 bits length, and so it is not applicable in the case.
- For 64 bits host portion, there are around  $4 \cdot 10^{17}$  prime numbers out of ( $2^{64}$ ) . For instant, 9,007,199,254,740,991 and 2,305,843,009,213,693,951 are prime numbers which have being converted to 64 bits in hexadecimal as (001FFFFFFFFFFFFFFF)<sub>hex</sub> and (1FFFFFFFFFFFFFFF)<sub>hex</sub> respectively to generate the Prime- IP addresses (xx...001FFFFFFFFFFFFFFF and xx...1FFFFFFFFFFFFFFF) for IPv6.

### Prime- IP Backtrack Procedure

For every available route from the source node to the destination node, Prime-IP backtrack procedure generates the vector containing the intermediate node addresses in a particular order.  $PPN_1$  and  $PPN_2$  numbers are used as input to the backtrack procedure. The “source node” gets a route replay packets containing the  $PPN_1$  and  $PPN_2$  numbers. As shown in figure 5a and figure 5b, two different routes are highlighted between the source node-71 and the destination node-73. Figure 5c shows a highlighted route between source node-7 and the destination node-11.

Figures 6a, 6b and 6c demonstrate the values of  $PPN_1$  and  $PPN_2$  in the highlighted routes that have been illustrated in Figures 5a, 5b and 5c respectively. In each of these examples, the source node gets a set of  $PPN_1$  and  $PPN_2$  numbers which is classified as following:

Route shown in figure 5a and 6a:

$$PPN_1 = 622,26,766,372,853,959$$

$$PPN_2 = 61,363,623,565,807,294$$

Route shown in figure 5b and 6b:

$$PPN_1 = 322,120,106,673$$

$$PPN_2 = 317,689,113,736$$

Route shown in figure 5c and 6c:

$$PPN_1 = 72,930$$

$$PPN_2 = 64,503$$

After the route replay packet has been received by the source node, the backtrack procedure will start to generate the vector (RS). Figure7 shows a flowchart of this procedure, which includes the iterations performed to consider all possibilities in forming the route vectors.

### Main Backtrack Procedure Flowchart using Figure 5a and 6a examples

To illustrate the backtrack procedure, the  $PPN_1$  and  $PPN_2$  numbers for the highlighted route in Figure 5a will be used to explain the flowchart of Figure 7:

1. In  $S_1$ , five variables have been defined:

Input Variables:

$$PPN_1 = 62,226,766,372,853,959$$

$$PPN_2 = 61,363,623,565,807,294$$

Output Variables:

K: Number of intermediate nodes in a route

RN: Intermediate Route Nodes vector in no particular order

RS: Intermediate Route Nodes vector in a particular order

Local Variables:

INX: Index

2. In  $S_2$ , determine the RN vectors by factorising the  $PPN_1$  number that represents the intermediate nodes in a no-particular order.

$$RN = \text{Factors}(PPN_1) = [23, 29, 41, 59, 73, 83, 97, 211, 311]$$

3. In  $S_3$ , add one to the  $PPN_2$ :

$$PPN_2 = PP_2 + 1 = 61,363,623,565,807,295$$

4. In  $S_4$ , determine the GCD of  $PPN_1$  and  $PPN_2$ :

$$g = \text{GCD}(PPN_1, PP_2), \text{GCD is Greater Common Division}$$

$$= \text{GCD}(62226766372853959, 61363623565807295) = [41]$$

5. Also in  $S_4$ ,

- if  $g=1$ : then the procedure is tracking the wrong track; therefore, the backward sub-procedure is invoked (described in Figure 10) to backtrack the procedure to the last benchmark in  $S_5$ .
- If  $g$  not a prime number (in this example,  $g=41$ ): then the procedure will chose between various valid tracks.
- if  $g$  a prime number: then  $g=41$  and we progress to discover a valid track.

6. In  $S_6$ , the forward sub-procedure is invoked as described in Figure 9. This moves the process forward by calculating the values to discover node 41:

$$PPN_1 = PPN_1/g = 62,226,766,372,853,959/41 = 1,517,726,009,093,999$$

$$PPN_2 = PPN_2/g = 61,363,623,565,807,295/41 = 1,496,673,745,507,495$$

Remove the first prime number [41] from RN and add it to RS

$$RN = [23,29,59,73,83,97,211,311]$$

$$RS = [41], INX = INX+1 = 1$$

7. In  $S_8$ , if  $PPN_1$  is not a prime number, then go to  $S_3$ .

8. Repeat (3-7) above as many times as necessary in order to obtain the final RS that represents all the intermediate route nodes vector in a particular order. The following is to discover node 311:

$$PPN_2 = PPN_2 + 1 = 1,496,673,745,507,495 + 1 = 1,496,673,745,507,496$$

$$g = GCS(PPN_1, PPN_2)$$

$$= GCD(1517726009093999, 1496673745507496) = [311]$$

$$PPN_1 = PPN_1/g = 1,517,726,009,093,999/311 = 4,880,147,939,209$$

$$PPN_2 = PPN_2/g = 1,496,673,745,507,496/311 = 4,812,455,773,336$$

$$RN = [23,29,59,73,83,97,211]$$

Remove the prime number [311] from RN and add it to RS. i.e.

$$RS = [41,311], INX = INX+1 = 2$$

Again, for node 211:

$$PPN_2 = PPN_2 + 1 = 4,812,455,773,336 + 1 = 4,812,455,773,337$$

$$g = \text{GCS}(PPN_1, PPN_2) = \text{GCD}(4880147939209, 4812455773337) = [211]$$

$$PPN_1 = PPN_1 / g = 4,880,147,939,209 / 211 = 23,128,663,219$$

$$PPN_2 = PPN_2 / g = 4,812,455,773,337 / 211 = 22,807,847,267$$

$$RN = [23, 29, 59, 73, 83, 97],$$

Remove the prime number [211] from RN and add it to RS

$$RS = [41, 311, 211], \text{ INX} = \text{INX} + 1 = 3$$

Again for node 29:

$$PPN_2 = PPN_2 + 1 = 22,807,847,267 + 1 = 22,807,847,268$$

$$g = \text{GCS}(PPN_1, PPN_2) = \text{GCD}(23128663219, 22807847268) = [29]$$

$$PPN_1 = PPN_1 / g = 23,128,663,219 / 29 = 797,540,111$$

$$PPN_2 = PPN_2 / g = 22,807,847,268 / 29 = 786,477,492$$

$$RN = [23, 59, 73, 83, 97],$$

Remove the prime number [29] from RN and add it to RS

$$RS = [41, 311, 211, 29], \text{ INX} = \text{INX} + 1 = 4$$

Again for node 59:

$$PPN_2 = PPN_2 + 1 = 786,477,492 + 1 = 786,477,493$$

$$g = \text{GCS}(PPN_1, PPN_2) = \text{GCD}(797540111, 786477493) = [59]$$

$$PPN_1 = PPN_1 / g = 797,540,111 / 59 = 13,517,629$$

$$PPN_2 = PPN_2 / g = 786,477,493 / 59 = 13,330,127$$

$$RN = [23, 73, 83, 97],$$

Remove the prime number [59] from RN and add it to RS

$$RS = [41, 311, 211, 29, 59], \text{ INX} = \text{INX} + 1 = 5$$



Again node 97:

$$PPN_2 = PPN_2 + 1 = 13,330,127 + 1 = 13,330,128$$

$$g = \text{GCS}(PPN_1, PPN_2) = \text{GCD}(13517629, 13330128) = [97]$$

$$PPN_1 = PPN_1 / g = 13,517,629 / 97 = 139,357$$

$$PPN_2 = PPN_2 / g = 13,330,128 / 97 = 137,424$$

$$RN = [23, 73, 83],$$

Remove the prime number [97] from RN and add it to RS

$$RS = [41, 311, 211, 29, 59, 97], \text{INX} = \text{INX} + 1 = 6$$

Again, this time for node 23:

$$PPN_2 = PPN_2 + 1 = 137,424 + 1 = 137,425$$

$$g = \text{GCS}(PPN_1, PPN_2) = \text{GCD}(139357, 137425) = [23]$$

$$PPN_1 = PPN_1 / g = 139,357 / 23 = 6,059$$

$$PPN_2 = PPN_2 / g = 137,425 / 23 = 5,975$$

$$RN = [73, 83],$$

Remove the prime number [23] from RN and add it to RS

$$RS = [41, 311, 211, 29, 59, 97, 23], \text{INX} = \text{INX} + 1 = 7$$

Again, finally for node 83:

$$PPN_2 = PPN_2 + 1 = 5,975 + 1 = 5,976$$

$$g = \text{GCS}(PPN_1, PPN_2) = \text{GCD}(6059, 5976) = [83]$$

$$PPN_1 = PPN_1 / g = 6,059 / 83 = 73$$

$$PPN_2 = PPN_2 / g = 5,976 / 83 = 72$$

$$RN = [73]$$

Remove the first prime number [83] from RN and add it to RS

$$RS = [41, 311, 211, 29, 59, 97, 23, 83], \text{INX} = \text{INX} + 1 = 8$$

9. In  $S_8$ , if  $PPN_1$  is a prime number then this is also the destination node, (in this example,  $PPN_1 = 73$ ), and so go to  $S_9$
10. In  $S_9$ , add (73) to  $RS = [41, 311, 211, 29, 59, 97, 23, 83, 73]$
11. In  $S_{10}$ , Finally,  $RS$  represents the highlighted route in figure 5a and 6a in this particular order.

**Another Backtrack Procedure Example, but also invoking the backward-sub-procedure as shown in figure 5c and 6c example:**

To illustrate the backtrack procedure further, the  $PPN_1$  and  $PPN_2$  numbers for the highlighted route in Figures 5c/6c will be used to explain the flowchart of Figure 7, but when the backward-sub-procedure is also invoked, as follows:

1.  $S_1$ , 5 variables have been defined:

Input Variables:

$PPN_1 = 72,930$

$PPN_2 = 64,503$

Output Variables:

K: Number of intermediate nodes in a route

RN: Intermediate Route Nodes vector in no-particular order

RS: Intermediate Route Nodes vector in particular order

Local Variables:

INX: Index

2.  $S_2$ , determines the RN vectors by factorising the  $PPN_1$  number that represents the intermediate nodes in no-particular order.

$RN = \text{Factors}(PPN_1) = [2, 3, 5, 11, 13, 17]$

3.  $S_3$ , add one to  $PPN_2$ :

$$PPN_2 = PPN_2 + 1 = 64,504$$

4.  $S_4$ , determine the GCD of  $PPN_1$  and  $PPN_2$ :

$$g = \text{GCS}(PPN_1, PPN_2) = \text{GCD}(72930, 64504) = [22]$$

- if  $g = 1$ : then the procedure is tracking the wrong track; therefore, the backward sub-procedure is invoked (described in Figure 10) to backtrack the procedure to the last benchmark in  $S_5$ .
- if  $g$  is a prime number: No
- If  $g =$  not a prime number (in this example,  $g = 22$ ): then the procedure will choose between various valid tracks, then,  $S_7$ .

5.  $S_7$  (more details in Figure 8), Bookmark Sub-procedure,

$gx$ : Factors of  $g$  in descending order (  $g$  is not prime number).

Bmark: Bookmark array used for the forward and backward sub-procedures as described figure 9 and figure 10.

The Bookmark Structure is:

Bmark(INX,1) = Branch :0 only one prime, : $i^{\text{th}}$  Multiple of  $(n-1)$  Prime number

Bmark(INX,2) = Number of Factors of the GCD at this point (number of branches)

Bmark(INX,3) =  $PPN_1$  at this point

Bmark(INX,4) =  $PPN_2$  at this point

Bmark(INX,5) = GCD at this point

Bmark(INX,6) = LB is the Previous Bookmark

Figure 5d illustrates the behaviour of the backtrack procedure (showing “depth-first search” algorithm when in a tree structure). Prime-IP assigns a bookmark (Bmark array) for every branch, when more than one track is possible. For instant, at this point,  $gx = \text{factors}(22) = [11, 2]$  in descending order. The procedure shall choose between two tracks, either [11] or [2]. While  $gx$  vector has been sorted in descending order, selecting the prime number will also be in descending order. As shown in figure 5d, node [11] is selected as the next node is track(1). If the procedure discovers that track(1) is the wrong track selection, then node [2] shall be selected as a next node in track(2).

Bmark(1,1) = 1, track(1)

Bmark(1,2) = 2 , represents the number of various valid tracks at this bookmark point

Bmark(1,3) = 72,930 ,  $PPN_1$  at this point

Bmark(1,4) = 64,504 ,  $PPN_2$  at this point

Bmark(1,5) = 22 ,  $g$  at this point

Bmark(1,6) = 1 , LB represents a Previous Bookmark

Forward( $gx(\text{Bmark}(1,1))$ )  $\rightarrow$  Forward(11) ,move the procedure forward.

6.  $S_6$ , the forward sub-procedure is invoked as described in Figure 9. This moves the process forward by calculating the values to discover node 11

$$PPN_1 = PPN_1/g = 72,930/11 = 6,630$$

$$PPN_2 = PPN_2/g = 64,504/11 = 5,864$$

$$RN = [2, 3, 5, 13, 17]$$

Remove the prime number [11] from RN and add it to RS

$$RS = [11], INX = INX + 1 = 2$$

7.  $S_8$ , if  $PPN_1$  is not a prime number:  $PPN_1 = 6,630$ .

8.  $S_3$ , add one to the  $PPN_2$ :  $PPN_2 = PPN_2 + 1 = 5,865$

9.  $S_4$ , determine the GCD of  $PPN_1$  and  $PPN_2$ :

- $g = \text{GCS}(PPN_1, PPN_2) = \text{GCD}(6630, 5865) = [255]$
- If  $g$  is not  $= 1 \rightarrow g = 255$
- if  $g$  is not a prime number:  $g = 255$ , then  $S_7$

10.  $S_7$ , this is the start of the “Bookmark Sub-procedure”, as shown in Figure 8.

At this point,  $gx = \text{factors}(255) = [17, 5, 3]$ . The process wants to find the right track by performing depth-first search. The procedure will consequently test the following tracks:  $\text{track}(1,1)$ ,  $\text{track}(1,2)$  and  $\text{track}(1,3)$ . Figure 5d shows the possible three tracks:

- Selecting  $[17]$  as a next node is track  $(1,1)$ .
- Selecting  $[5]$  as a next node is track  $(1,2)$ .
- Selecting  $[3]$  as a next node is track  $(1,3)$ .

11. Next, the process moves forward following track  $(1,1)$  in order to find all the intermediate nodes in a particular order. Once  $S_4$  detects that the process is in a wrong track ( $g = 1$ ), the backtrack procedure will stop and chooses the next track (e.g. track  $(1,2)$ ). Figure 8 and figure 5d illustrates the testing of this track as in the following steps:

$Bmark(2,1) = 1$ , track  $(1,1)$

$Bmark(2,2) = 3$ , represents the number of various valid tracks at this bookmark point

$Bmark(2,3) = 6,630$ ,  $PPN_1$  at this point

$Bmark(2,4) = 5,865$ ,  $PPN_2$  at this point

$Bmark(2,5) = 255$  , g at this point

$Bmark(2,6) = 1$  , LB represents a Previous Bookmark

$Forward(gx(Bmark(2,1))) \rightarrow Forward(17)$  ,move the procedure forward

12. Move forward in the track (1,2) in order to find the intermediate node in particular order. Once  $S_4$  detects that is a wrong track ( $g=1$  is true), backtrack procedure will stop and undertake the next track (e.g. track (1,3)). Figure 8 and figure 5d illustrates the testing of the track and how the sub-procedures behaviours will be.

$Bmark(2,1) = 2$  , track(1,2)

$Bmark(2,2) = 3$  , represents the number of various valid tracks at this bookmark point

$Bmark(2,3) = 6,630$  ,  $PPN_1$  at this point

$Bmark(2,4) = 5,865$  ,  $PPN_2$  at this point

$Bmark(2,5) = 255$  , g at this point

$Bmark(2,6) = 1$  , LB represents a Previous Bookmark

$Forward(gx(Bmark(2,1))) \rightarrow Forward(5)$  ,move the procedure forward

13. Move forward in the track (1,3) in order to find the intermediate node in particular order. Once  $S_4$  detects that is a wrong track ( $g=1$  is true), backtrack procedure will stop and undertake a backward track (track(1)), because it is the last track at this bookmark. Figure 8 and figure 5d illustrates the testing of the track and how the sub-procedures behaviours will be.

$Bmark(2,1) = 3$  , track(1,3)

$Bmark(2,2) = 3$  , represents the number of various valid tracks at this bookmark point

$Bmark(2,3) = 6,630$  ,  $PPN_1$  at this point

$Bmark(2,4) = 5,865$  ,  $PPN_2$  at this point

$Bmark(2,5) = 255$  , g at this point

$Bmark(2,6) = 1$  , LB represents a Previous Bookmark

$Forward(gx(Bmark(2,1))) \rightarrow Forward(3)$  ,move the procedure forward

14. In figure 5d, Now that the GCD of  $(PPN_1, PPN_2)$  is equal to one in all track(1) branches, i.e. the procedure decides it is the wrong track, and so chooses to progress along track(2). i.e. The right track will never give this result (GCD= 1) in S4, because at S8 the  $PPN_1$  will be tested whether it is a prime number or not. If it is a prime number, then the backtrack procedure progress in the right track and it knows that this is the last prime number (“destination node”). However, if the  $PPN_1$ , in S8, is not a prime number, then the procedure will move forward. At this point, there is no indication if it is following a wrong or correct track, until S4 will test is reached again

15. The backtrack procedure will now chose the next possible track as follows, in this example, it is track(2):

$Bmark(1,1) = 2$ , track(2)

$Bmark(1,2) = 2$  , represents the number of various valid tracks at this bookmark point

$Bmark(1,3) = 72,930$  ,  $PPN_1$  at this point

$Bmark(1,4) = 64,504$  ,  $PPN_2$  at this point

$Bmark(1,5) = 22$  , g at this point

$Bmark(1,6) = 1$  , LB represents a Previous Bookmark

$Forward(gx(Bmark(1,1))) \rightarrow Forward(2)$  ,move the procedure forward.

16.  $S_6$ , the forward sub-procedure figure 9 moves forward by the following calculations:

$$PPN_1 = PPN_1/g = 72,930/2 = 36,465$$

$$PPN_2 = PPN_2 / g = 64,504 / 2 = 32,252$$

$$RN = [2, 3, 5, 13, 17]$$

Remove the prime number [2] from RN and add it to RS

$$RS = [2], INX = INX + 1 = 2$$

17.  $S_8$ , if  $PPN_1$  is prime: No,  $PPN_1 = 36,465$ .

18.  $S_3$ , add one to the  $PPN_2$ :  $PPN_2 = PPN_2 + 1 = 32,253$

19.  $S_4$ , determine the GCD of  $PPN_1$  and  $PPN_2$ :

$$g = GCS(PPN_1, PPN_2) = GCD(36465, 32253) = [39]$$

20. In  $S_4$ ,

- If  $g = 1 \rightarrow$  NO, ( $g = 39$ ).
- if  $g$  is a prime number: No,  $g = 39$ ,
- if  $g$  is not a prime number:,  $g = 39$ , then  $S_7$

21.  $S_7$  the Bookmark Sub-procedure and as shown in figure 8,

At this point,  $gx = \text{factors}(39) = [13, 3]$ . The procedure will consequently test the following tracks: track(2,1) and track(2,2). Figure 5d shows the procedure that should have chosen between two tracks.

- Selecting [13] as a next node is track (2,1).
- Selecting [3] as a next node is track (2,2).

22. The procedure should have chosen between two tracks, track(2,1) or track(2,2) while  $gx$  vector has been sorted in descending order, selecting the prime number will be in descending order also. As shown in figure 5d, selecting [13] as a next



node is track(2,1) while selecting [3] as a next node is track(2,2) if the first track(2,1) is wrong.

23.  $S_6$ , the forward sub-procedure figure 9 moves forward by doing these calculations:

$$PPN_1 = PPN_1/g = 36,465/13 = 2,805$$

$$PPN_2 = PPN_2/g = 32,253/13 = 2,481$$

$$RN = [3,5,13, 17]$$

Remove the prime number [13] from RN and add it to RS

$$RS = [2,13], INX = INX+1 = 3$$

24. In  $S_8$ , if  $PPN_1$  is prime: No,  $PPN_1 = 2,805$ .

25. In  $S_3$ , add one to the  $PPN_2$ :  $PPN_2 = PPN_2 + 1 = 2,482$

26. In  $S_4$ , determine the GCD of  $PPN_1$  and  $PPN_2$ :

$$g = GCS (PPN_1, PPN_2) = GCD(2805, 2482) = [17]$$

27. In  $S_4$ ,

- If  $g = 1 \rightarrow$  NO, ( $g = 17$ ).
- if  $g$  is not a prime number, ( $g = 17$ ).
- if  $g$  is a prime number: Yes,  $g = 17$ , then  $S_6$

28. In  $S_6$ , the forward sub-procedure, figure 9, moves forward by doing these calculations:

$$PPN_1 = PPN_1/g = 2,805/17 = 165$$

$$PPN_2 = PPN_2/g = 2,482/17 = 146$$

$$RN = [3,5,17]$$

Remove the prime number [17] from RN and add it to RS

$$RS = [2,13,17], INX = INX+1 = 4$$

29. Repeat above steps many times in order to get the final RS that represents Intermediate Route Nodes vector in particular order

$$PPN_1 = PPN_1/g = 165/3 = 55$$

$$PPN_2 = PPN_2 /g = 147/3 = 49$$

$$RN = [3,5]$$

Remove the prime number [3] from RN and add it to RS

$$RS = [2,13,17,3], INX = INX+1 = 5$$

30. Repeat it again,

$$PPN_1 = PPN_1/g = 55/5 = 11$$

$$PPN_2 = PPN_2 /g = 50/5 = 10$$

$$RN = [5]$$

Remove the prime number [5] from RN and add it to RS

$$RS = [2,13,17,3,5], INX = INX+1 = 6$$

31. In  $S_8$ , if  $PPN_1$  prime number, YES  $PPN_1 = 11$ ,

32. In  $S_9$ , add (11) to  $RS = [2,13,17,3,5,11]$ ,

33. In  $S_{10}$ , RS represents the highlighted route in figure 5c,6c in particular order.

**Conclusion:**

The examples described in this patent demonstrate the novelties of this new algorithm. The claims made are focused on using “Prime numbers” to be the address of the host portion of WMN node IP and accumulating “node knowledge” of the entire WMN. In this patent, the resultant “Prime-IP” claims have been proved mathematically and shown to work for multi-hop dynamic topology WMNs.

**Claims:**

1. This algorithm, named “Prime-IP”, is designed to uniquely identify a path in a Wireless Mesh Networks-WMN, without impacting the routing protocol.
2. Prime-IP is designed to allow each individual node to uniquely identify a path based on the node’s location within the mesh.
3. The host portion of any Node’s IP address is assigned a unique prime number.
4. Prime-IP is designed to detect, and survive-the-loss-of, any deactivated nodes in the route.
5. Further to claim 2, and upon a “source node” issue a request to discover a route to a “destination node”, then, route reply packets received by the “source node” shall contain full path knowledge about their routes.
6. The route reply packet shall have two extra fields appended to the existing packet format.
7. Further to claim 6, these two new fields will contain a value related to the “Product of Prime Numbers”, see claim 3, and these two fields shall be named PPN1 and PPN2.
8. PPN1 shall have an initial value equal to the “destination route” prime number, and PPN2 shall have an initial value equal to the “destination node” prime number minus one.
9. Further to claims 3, 5 and 8, each intermediate node shall multiply its own prime number by the PPN1 value present in the route reply packet ( $\text{newPPN1} = \text{previousPPN1} \times \text{NodePrimeNumber}$ )
10. Further to claim 3, 5 and 8, each intermediate node shall multiply its own prime number by the PPN2 value in the packet and subtract 1 from the resultant product ( $\text{newPPN2} = \text{previousPPN2} \times \text{NodePrimeNumber} - 1$ ).
11. The accumulated operations of claims 9 and 10 all through the route reply process from the “destination node” to the “source node” shall results in unique PPN<sub>1</sub> and PPN<sub>2</sub> numbers.

12. Further to claim 11, Prime-IP shall produce a vector containing full path knowledge (intermediate nodes and their sequence in the route) from PPN1 and PPN2.
13. Prime-IP, using backtrack procedure, shall generate a vector of all intermediate nodes, of any route, in a particular order based on the value contained in PPN1 and PPN2.
14. Each individual node shall acquire a knowledge of what other nodes exist in the route that are beyond their nearest neighbouring nodes.
15. Prime-IP shall apply for both Ipv4 and Ipv6 address types.
16. Prime-IP applies to all types of wireless mesh networks.
17. Prime-IP applies to all types of multi-hop wireless networks
18. Prime-IP works with both fixed and mobile nodes.

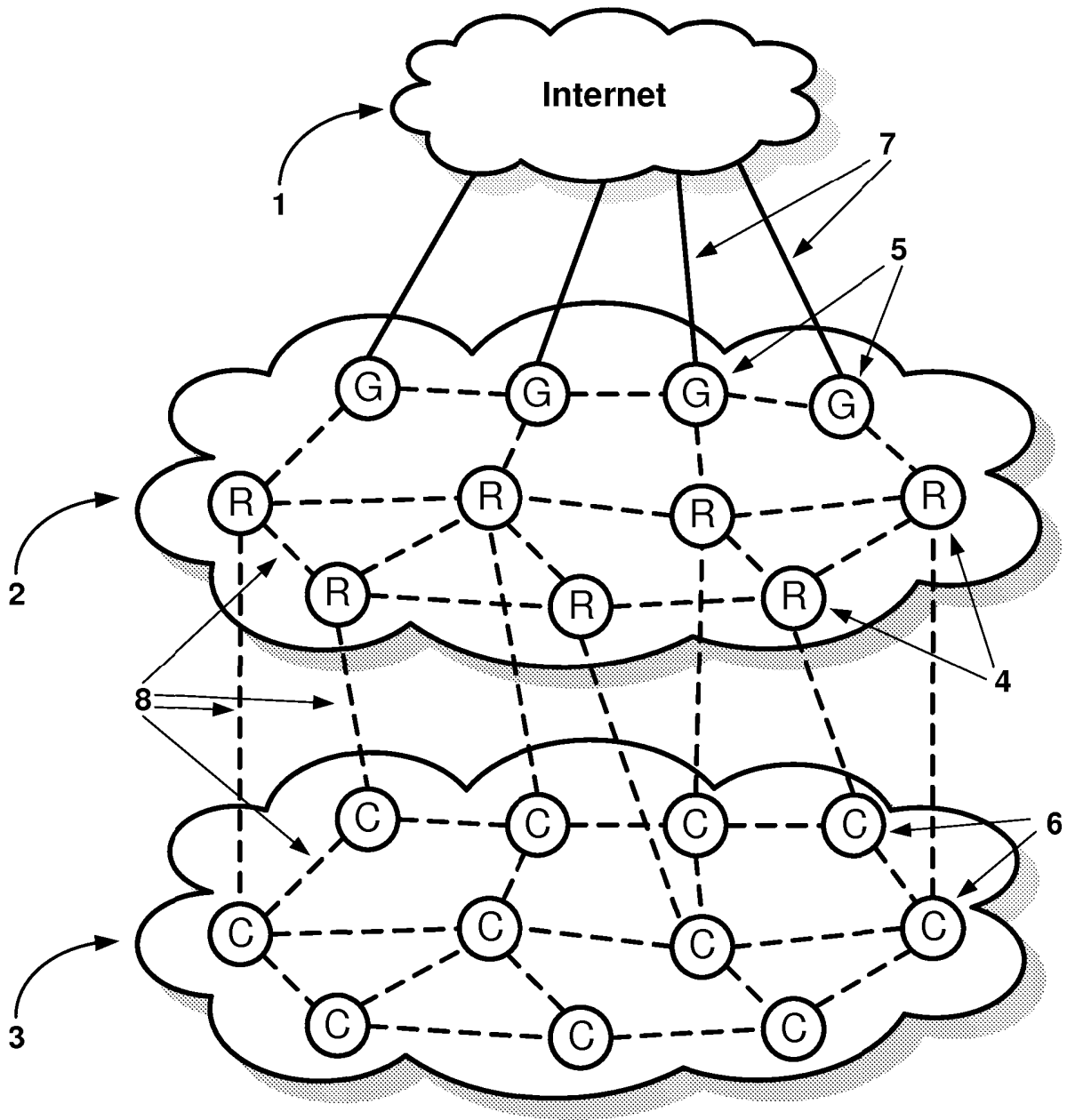


Figure 1

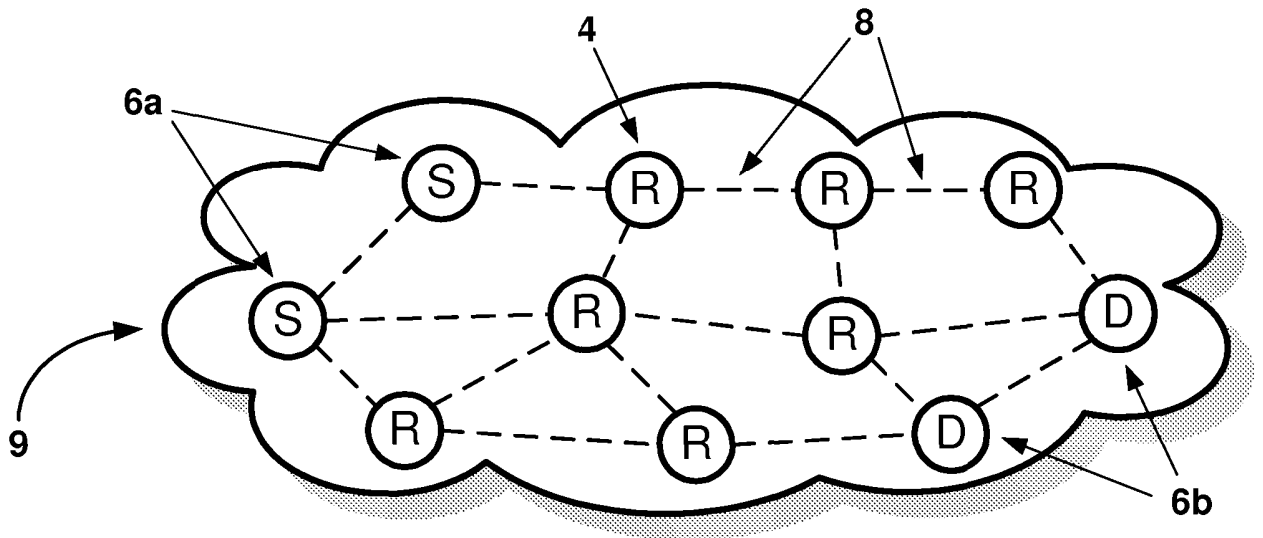


Figure 2

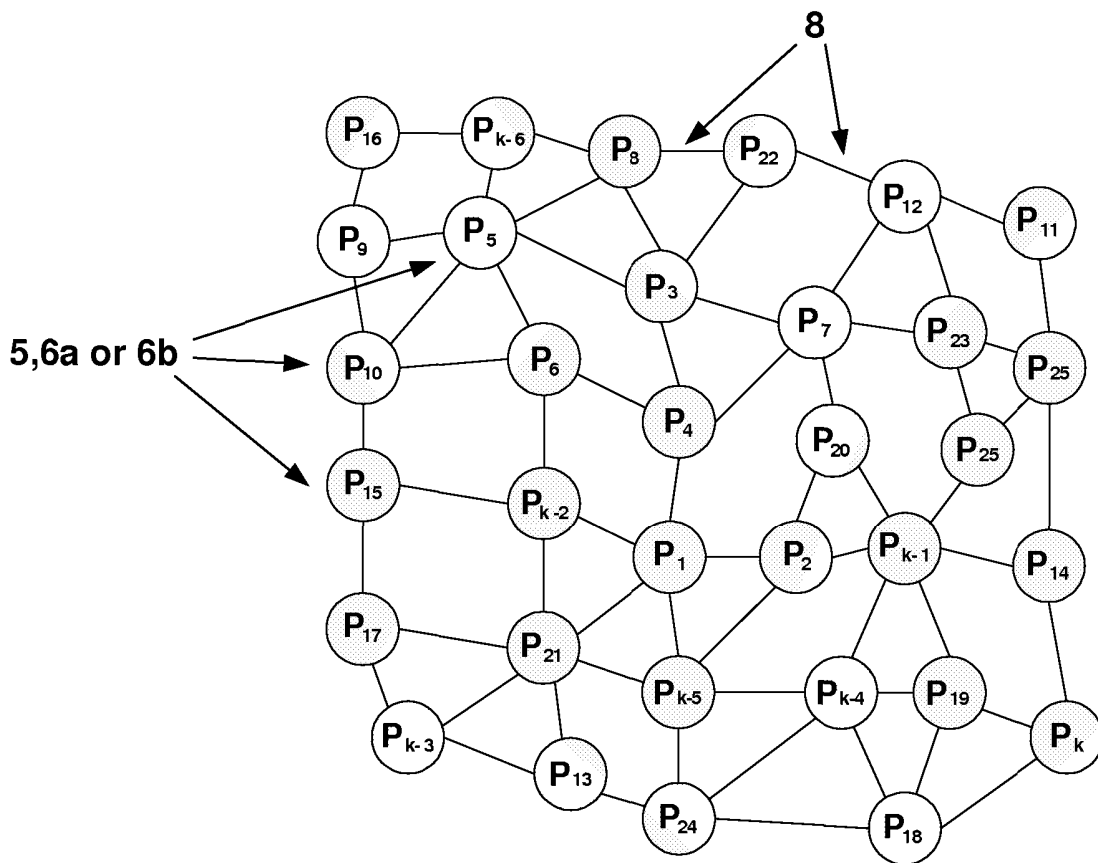


Figure 3

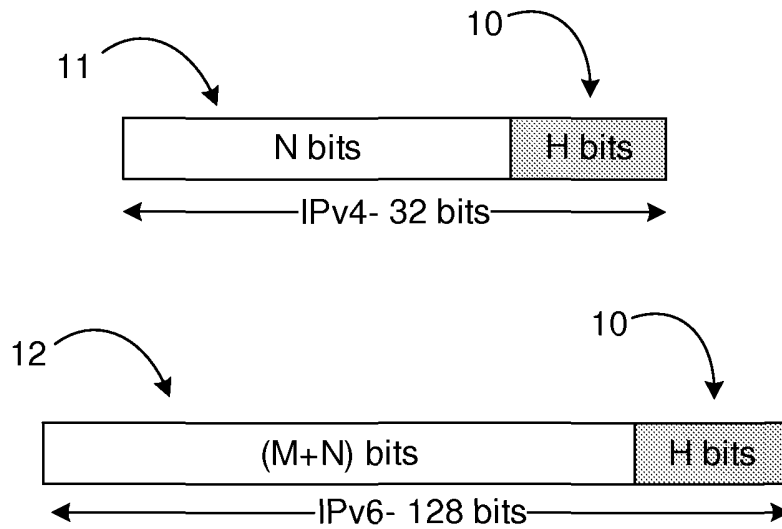


Figure 4a

Bits/Host	PN-Prime Number	Prime Number (Binary Representation)	Prime-IP IP-Address (IPv4-32 bits)	Prime-IP IP-Address (IPv6-128 bits)
8	5	00000101	x.x.x. <b>5</b>	xx... <b>05</b>
8	239	11101111	x.x.x. <b>239</b>	xx... <b>EF</b>
16	313	00000001 00111001	x.x. <b>1.57</b>	xx... <b>0139</b>
16	51449	11001000 11111001	x.x. <b>200.249</b>	xx... <b>C8F9</b>
24	2051773	00011111 01001110 10111101	x. <b>31.78.189</b>	xx... <b>1F4EBD</b>
24	12004991	10110111 00101110 01111111	x. <b>183.46.127</b>	xx... <b>B72E7F</b>
48	9990454997	( 0002537A3ED5 ) <sub>hex</sub>	Not Applicable	XX... <b>0002537A3ED5</b>
48	281474076384103	( FFFFCA561B67 ) <sub>hex</sub>	Not Applicable	XX... <b>FFFFCA561B67</b>
64	9007199254740991	( 001FFFFFFFFFFFFFFF ) <sub>hex</sub>	Not Applicable	XX... <b>001FFFFFFFFFFFFFFF</b>
64	2305843009213693951	( 1FFFFFFFFFFFFFFF ) <sub>hex</sub>	Not Applicable	XX... <b>1FFFFFFFFFFFFFFF</b>

Figure 4b



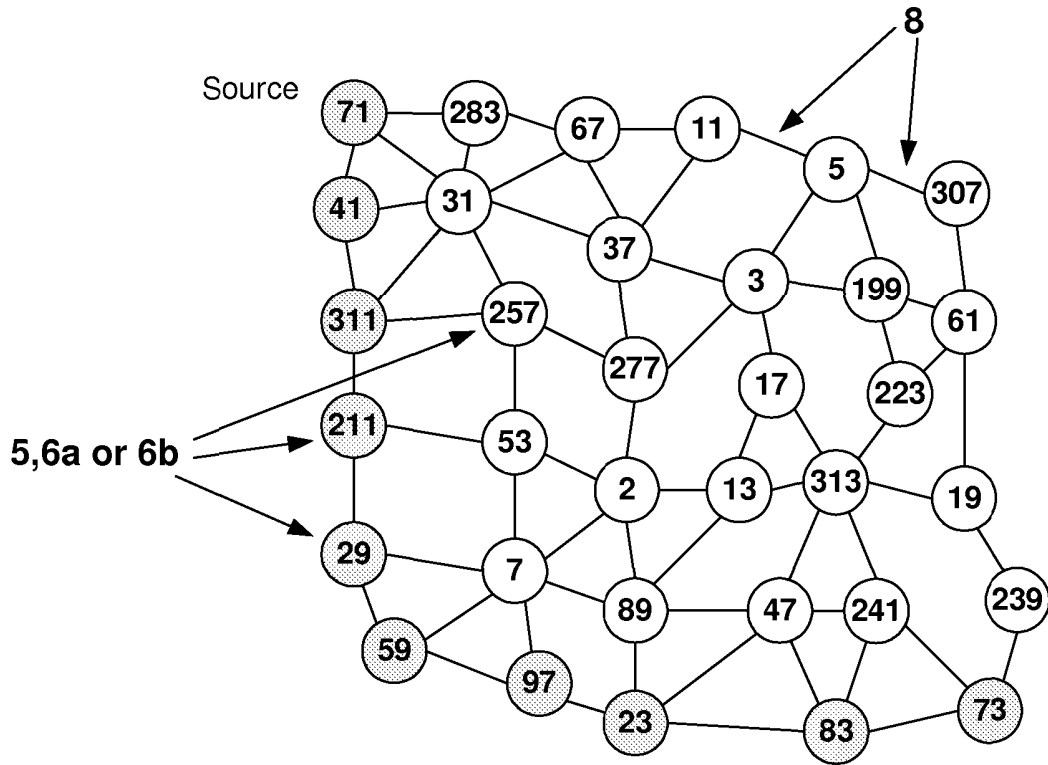


Figure 5a

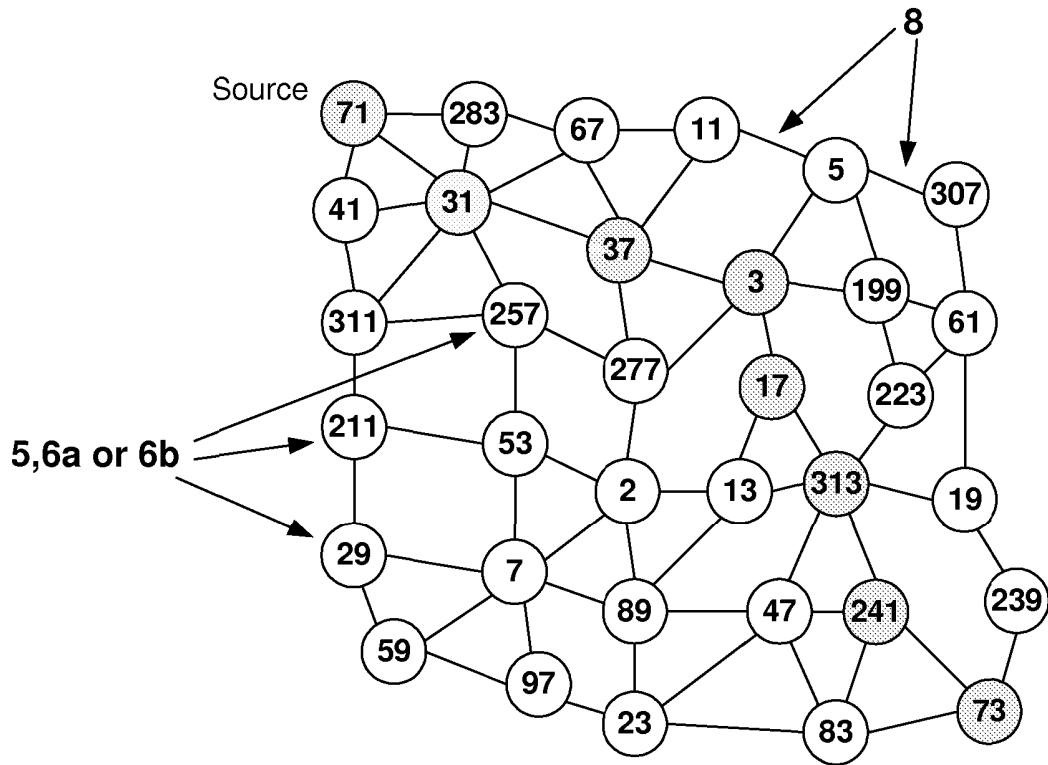


Figure 5b

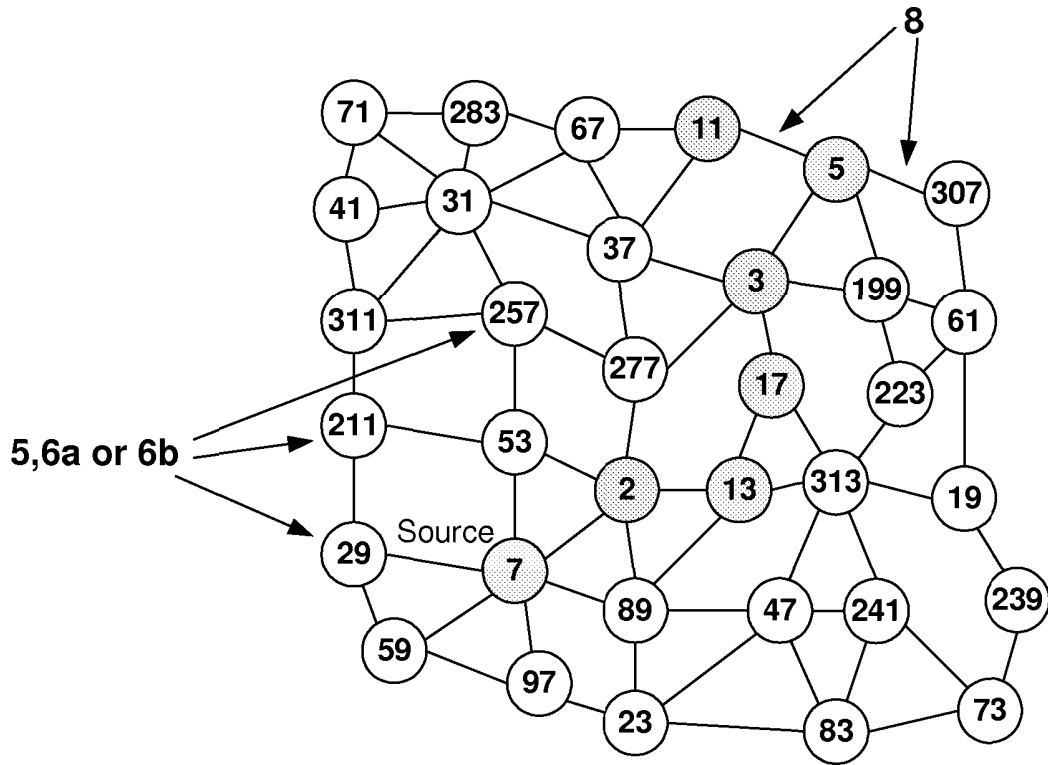


Figure 5c

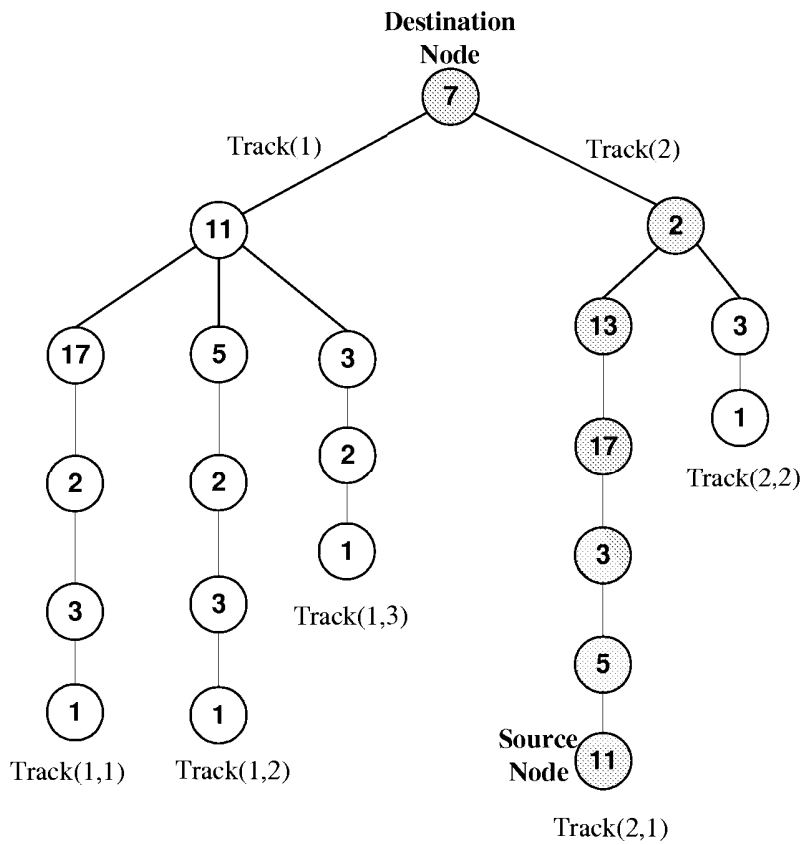


Figure 5d

<b>PPN<sub>1</sub> and PPN<sub>2</sub> value calculations from highlighted route in figure 5a</b>		
<b>PN</b>	<b>PPN<sub>1</sub></b>	<b>PPN<sub>2</sub></b>
73	73-Destination	72
83	6059	5975
23	139357	137424
97	13517629	13330127
59	797540111	786477492
29	23128663219	22807847267
211	4880147939209	4812455773336
311	1517726009093999	1496673745507495
41	62226766372853959	61363623565807294
71	Source Node	Source Node

Figure 6a

<b>PPN<sub>1</sub> and PPN<sub>2</sub> value calculations from highlighted route in figure 5b</b>		
<b>PN</b>	<b>PPN<sub>1</sub></b>	<b>PPN<sub>2</sub></b>
73	73-Destination	72
241	17593	17351
313	5506609	5430862
17	93612353	92324653
3	280837059	276973958
37	10390971183	10248036445
31	322120106673	317689129794
71	Source Node	Source Node

Figure 6b

<b>PPN<sub>1</sub> and PPN<sub>2</sub> value calculations from highlighted route in figure 5c</b>		
<b>PN</b>	<b>PPN<sub>1</sub></b>	<b>PPN<sub>2</sub></b>
11	11-Destination	10
5	55	49
3	165	146
17	2805	2481
13	36465	32252
2	72930	64503
7	Source Node	Source Node

Figure 6c

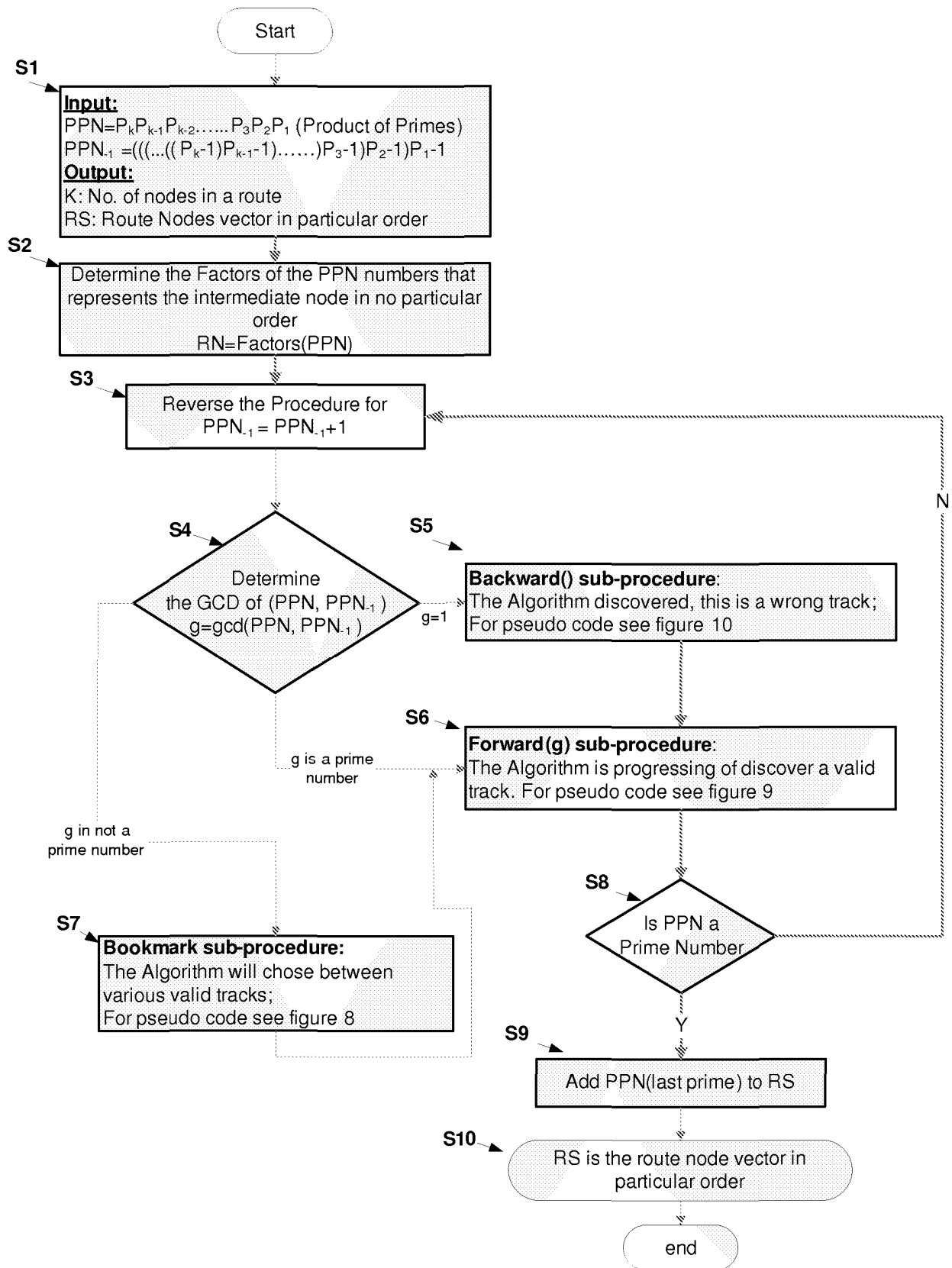


Figure 7

```

Bookmark sub-procedure:
{
// Whilst g is multiple of Prime Numbers
// gx: is factors of the g
// Bmark: Bookmark array which will be used for the
// backward/forward functions in the Backtrack procedure.
// The Bookmark Structure is:
// Bmark(INX,1) = 0, only one prime
//           > 0, Multiple of Primes
//Bmark(INX,2) = Number of Factors of the GCD at this point
//Bmark(INX,3) = PPN at this point
//Bmark(INX,4) = PPN-1 at this point
//Bmark(INX,5) = GCD at this point
//Bmark(INX,6) = LB, Previous Benchmark

1:      gx=sort(factor(g),'descend');
2:      Bmark(INX,1)=Bmark(INX,1)+1;
3:      Bmark(INX,2)=length(gx);
4:      Bmark(INX,3)=PPN;
5:      Bmark(INX,4)=PPN1;
6:      Bmark(INX,5)=g;
7:      Bmark(INX,6)=LB;
8:      LB=INX;
9:      Forward(gx(Bmark(INX,1)));
}

```

Figure 8

```

Forward(g) sub-procedure
{
1:      PPN=PPN/g;
2:      PPN1=PPN1/g;
3:      Remove g from RN vector;
4:      Add g to the end of RS vector;
5:      INX=INX+1;
}

```

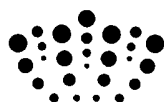
Figure 9

```

Backward() sub-procedure
{
//Bsteps : How many backward steps
//LB: Last Bookmark
1:      Bsteps=INX-LB-1;
2:      INX=LB;
3:      Bmark(INX,1)=Bmark(INX,1)+1;
4:      PPN=Bmark(INX,3);
5:      PPN1=Bmark(INX,4);
6:      GD=Bmark(INX,5);
7:      bgx=factor(GD);
// Add the wrong track prime numbers to the RN vector again
8:      RN=[RN RS(end-Bsteps:end)];
// Remove the wrong track prime numbers from the end of the RS vector
9:      RS(end-Bsteps:end)=[];
10:     if Bmark(INX,1)<=Bmark(INX,2)
11:         PR=bgx(Bmark(INX,1));
12:     else
13:         Bmark(INX,1)=0;
13:         LB=Bmark(INX,6);
14:         PR=Backward();
15:     end
Return PR
}

```

Figure 10



**Application No:** GB1005254.6

**Examiner:** Mrs Hannah Sylvester

**Claims searched:** 1-18

**Date of search:** 6 July 2010

**Patents Act 1977: Search Report under Section 17**

**Documents considered to be relevant:**

Category	Relevant to claims	Identity of document and passage or figure of particular relevance
X	at least 1, 3, 7, 16, 17 and 18	US2006/198320 A1 (HSU YUAN-YING [TW] et al) see whole document
A	-	US2010/031356 A1 (CHOI HYOUNG KEE [KR])
A	-	US2009/003356 A1 (VASWANI RAJ [US] et al)

**Categories:**

X Document indicating lack of novelty or inventive step	A Document indicating technological background and/or state of the art.
Y Document indicating lack of inventive step if combined with one or more other documents of same category.	P Document published on or after the declared priority date but before the filing date of this invention.
& Member of the same patent family	E Patent document published on or after, but with priority date earlier than, the filing date of this application.

**Field of Search:**

Search of GB, EP, WO & US patent documents classified in the following areas of the UKC<sup>X</sup> :

--

Worldwide search of patent documents classified in the following areas of the IPC

H04L; H04W
------------

The following online and other databases have been used in the preparation of this search report

WPI EPODOC
------------

**International Classification:**

Subclass	Subgroup	Valid From
H04W	0008/26	01/01/2009
H04L	0012/56	01/01/2006
H04L	0029/12	01/01/2006
H04W	0040/24	01/01/2009