psnmp

Maintainer manual

ENSEIRB

1 Overview

psnmp is a LAN management software. Its main feature is guessing the topology of an Ethernet network.

This is the maintainer manual, documenting how the source is divided and some tricky algorithms. Thoses algorithms can be found in treeb and psnmp programs and in the package common library. We will discuss about the algorithm for topology, the VLAN configurations gathering throught SNMP protocol. The module dealing with trees and the database will described for non trivial functionality like the merge of trees, tree rotation for changing its root. Futhermore module organization and relations will be analysed.

Please ejoy this manual.

2 Copying

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
   a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
   b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
   c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

   The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

   If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.
8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.
10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software

which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> *one line to give the program's name and an idea of what it does.*
> Copyright (C) 19*yy* *name of author*
>
> This program is free software; you can redistribute it and/or
> modify it under the terms of the GNU General Public License
> as published by the Free Software Foundation; either version 2
> of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful,
> but WITHOUT ANY WARRANTY; without even the implied warranty of
> MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
> GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License
> along with this program; if not, write to the Free Software
> Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

> Gnomovision version 69, Copyright (C) 19*yy* *name of author*
> Gnomovision comes with ABSOLUTELY NO WARRANTY; for details
> type 'show w'. This is free software, and you are welcome
> to redistribute it under certain conditions; type 'show c'
> for details.

The hypothetical commands '`show w`' and '`show c`' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than '`show w`' and '`show c`'; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright
> interest in the program 'Gnomovision'
> (which makes passes at compilers) written
> by James Hacker.
>
> *signature of Ty Coon*, 1 April 1989
> Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

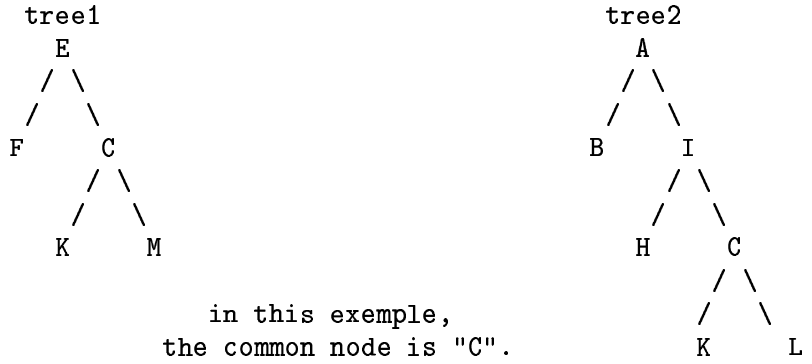
3 Tree merging

3.1 General presentation

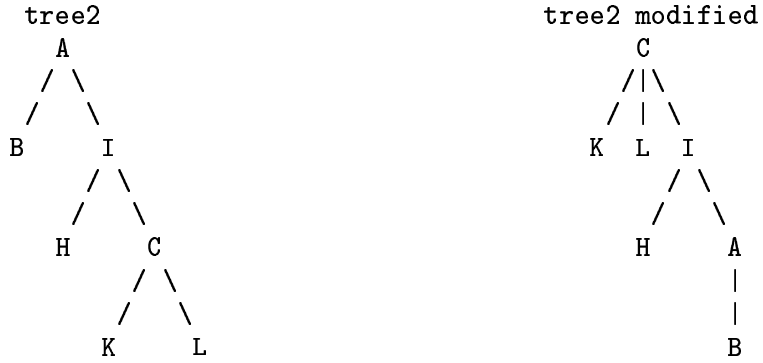
We have two trees as parameters of entry which we have to merge. We never modify tese both trees. Technically, we begin by making another copy of the first tree. Then, we add the branches of the second tree not yet present in the first tree. At the end, we return this copy modified. We call this copy the new tree. However, if both trees don't have any common nodes or if we obtain a bucle, the working of fusion is left and we return a pointer NULL

3.2 research of the common node

We start the working of fusion by copying the first tree entried as parameter. The second step consists in the research of the common node: we make a traversing in depth of the first tree in order to define the first node also present in the second tree.



After this work, we have to center the second tree on this common node: by this way, this first common node becomes the root of the second work.



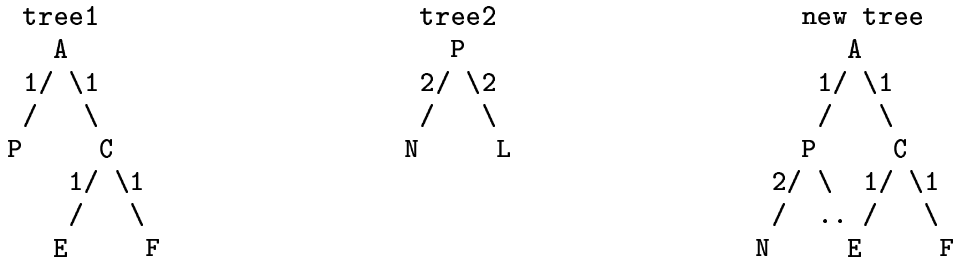
By now, we are going to work only on this modified version of the second tree.

3.3 The traversing of the second tree

Then we begin by traversing the modified version of the second tree in depth. For each node met, we make the adapted processing.

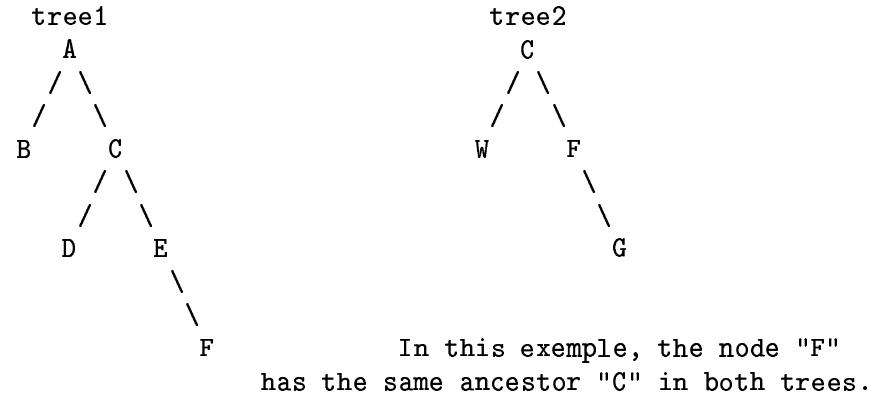
3.3.1 a node of the second tree not present in the first tree

If we meet a node "N" not present in the first tree, we add this node to the new tree without forgetting to give the name of the VLAN of this new branch. In this goal, we research in the new tree the father "P" of this new node in the second tree and add to this node "P" of the new tree a son named "N".



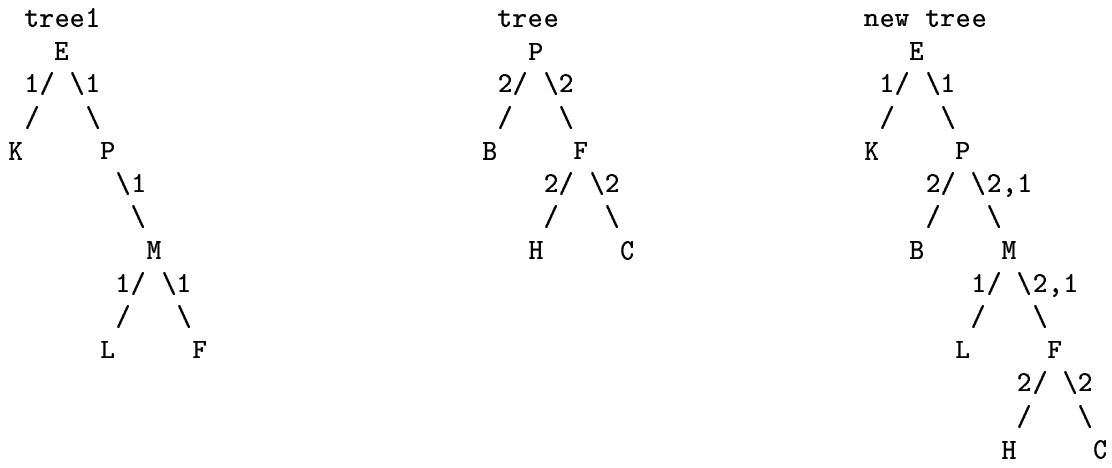
3.3.2 a node present in both trees

If we meet a node "F" yet present in the first tree. To begin, we verify that this node has the same first common ancestor in the first tree as in the second. We call first common ancestor of "F" the closest ancestor of "F" present in both trees.



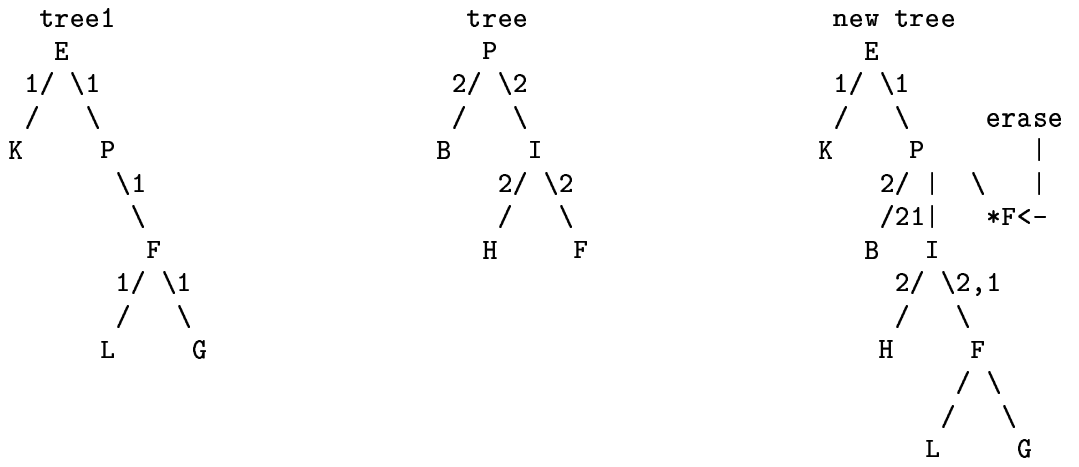
If it's not the case, we stop the algorithm because it involves that there is a buckle. If the node studied has the same ancestor in both trees, we distinguish tree cases. In fact, when both trees had been built, some nodes can have been forgotten.

- some nodes are forgotten in the second tree but not in the first:



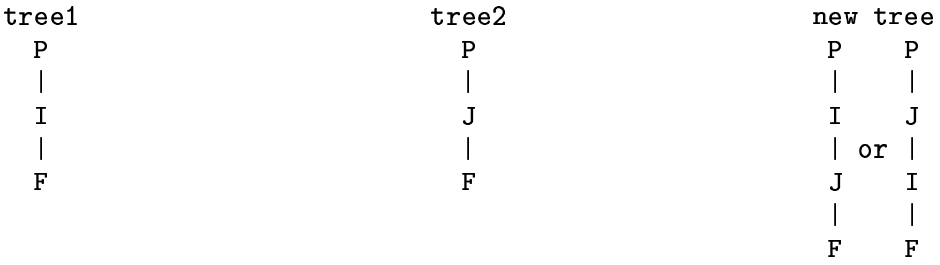
This case squares with the fact that the father of this node "F" in the second tree is a node present in the first tree: the father of "F" in the second tree is the common ancestor. In this case, we have to bring up to date the management of VLAN: we have to add the name of the VLAN of the branch PF of the second tree on the branches in the new tree which go from the node P to the node F (ie the branches PH, HI and IF.

- some nodes are forgotten in the first tree but not in the second:

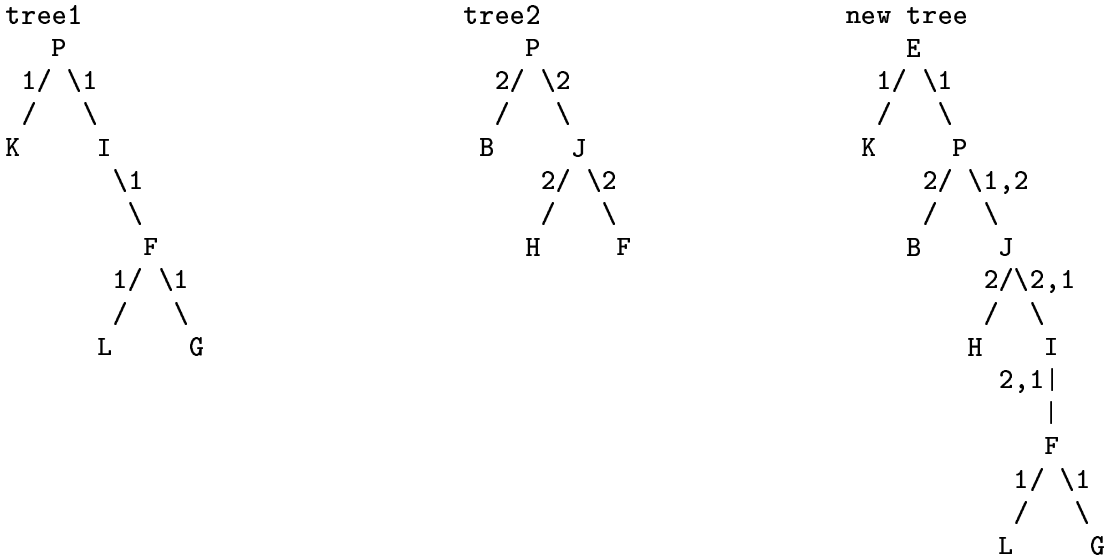


This case squares with the fact that the father I of this node F in the second tree is not a node of the first tree. The nodes H and I have yet been added to the first tree. We have to erase F from son of P and its whole arborescence from the new tree. Then we recopy this arborescence under I in the new tree. We have also to bring up to date the value of VLANS of the branches which go from the ancestor common P to the node studied F in the new tree. In this goal, we add, to the list of VLANs of these branches, the value of the branch PF (ie the branch "father of node studied - node studied")in the first tree.

- some nodes are forgotten in both trees:



The problem is that we can't place the node I and J because we don't know if I is the father of J or the contrary. That's why we decide to choose arbitrarily to place firstly the nodes of the second tree and after these of the first tree. With this choose, we obtain the result:



At the end of the algorithm, we return the new tree which is the merge of both trees.

4 VLANs

4.1 List of VLANs

Let's refer to RFCs for further information about OIDs' descriptions, and also MIBs' descriptions ('`*.mib`') given by constructors for each equipment.

In the following descriptions, ifIndex refers to the OID '`.1.3.6.1.2.1.2.2.1.1`', which is a index used to talk about interfaces .

The list of VLANs is available at the following OIDs:

- *dot1qVlan*
  '`.1.3.6.1.2.1.17.7.1.4.3.1.1.NNN`'
  '`iso. org. dot. internet. mgmt. mib-2. dot1dBridge. qBridgeMIB. qBridgeMIBObjects. dot1qVlan. dot1qVlanStaticTable. dot1qVlanStaticEntry. dot1qVlanStaticName. NNN`'
  '`NNN`' is a VLAN's number. The OID value is the VLAN's description.
- *a3VlanGM*
  '`.1.3.6.1.4.1.43.10.1.14.1.1.1.2.NNN`'
  '`iso. org. dot. internet. private. enterprises. a3Com. generic. genExperimental. genVirtual. a3ComVlanGroup. a3ComVlanGlobalMappingTable. a3ComVlanGlobalMappingEntry. a3ComVlanGlobalMappingIfIndex. NNN`'
  '`NNN`' is a VLAN's number. The OID value is the number of an virtual interface; this number of an virtual interface represents a VLAN and refers to the ifIndex.

4.2 Interface/VLAN table

The table of interfaces and VLANs assocations (i.e. for each port, the list of VLANs it is part of) is available at the following OIDs:

- *dot1qEgress*
  '`.1.3.6.1.2.1.17.7.1.4.3.1.2.NNN`'
  '`iso. org. dot. internet. mgmt. mib-2. dot1dBridge. qBridgeMIB. qBridgeMIBObjects. dot1qVlan. dot1qVlanStaticTable. dot1qVlanStaticEntry. dot1qStaticEgressPort. NNN`'
  '`NNN`' is a VLAN's number. The OID value is a bits' mask"; this mask (seen as a boolean list) coding if the VLAN is on a port (1) or not (0); the port 1 is the first bit, port 2 the second, etc.
- *ifStack*
  '`.1.3.6.1.2.1.31.1.2.1.3.MMM.NNN`'
  '`iso. org. dot. internet. mgmt. mib-2. ifMIB. ifMIBObjects. ifStackTable. ifStackEntry. ifStackStatus. MMM. NNN`'
  '`MMM`' is a ifStackHigher value and '`NNN`' a ifStackLower value; they refer to the ifIndex; it's a coding of a relation father/son in trees of interfaces (port or virtual interfaces) where the interaces representing the VLANs are the roots, and ports belonging to this VLAN are the leaves. The OID value is 1(active).

```
An example of a VLAN configuration and all
ifStackHigher/ifStackLower defined to code this
configuration:
+======================+======================+
|     VLAN Xface 9     |       VLAN Xface 11  |
+======================+======================+
+=======================+
|      ENCAPS Xface 10  |
+=======================+
+=====+=====+=====+=====+=====+=====+=====+=====+
|  1  |  2  |  3  |  4  |  5  |  6  |  7  |  8  |<=== Ports
+=====+=====+=====+=====+=====+=====+=====+=====+

ifStackTable Instances :

ifStackHigher ifStackLower
     0               9
     0               11
     1               0
     2               0
     3               0
     4               0
     9               10
    10               1
    10               2
    10               3
    10               4
    11               5
    11               6
    11               7
    11               8
```

A stack that contains a VLAN, encapsulation and a port interface, specifies:

- For packets received through the given port that use the given encapsulation scheme and contain the given tag, those packets are members of the given VLAN.
- For unencapsulated packets from the given VLAN that are to be transmitted out the given port, those packets must first be encapsulated using the given encapsulation algorithm and tag.

This table is implemented by all 3Com network devices that support the encapsulation of multiple VLANs over a single interface.

4.3 Tagging configuration

The behaviour for non tagged packets (actually, we want to know on which VLAN number the untagged packets are tagged for each port) is described in the following OIDs:

- *dot1qPvid*
  '`.1.3.6.1.2.1.17.7.1.4.5.1.1.NNN`'
  '`iso. org. dot. internet. mgmt. mib-2. dot1dBridge. qBridgeMIB.`

`qBridgeMIBObjects. dot1qVlan. dot1qPortVlanTable. dot1qPortVlanEntry. dot1qPvid. NNN`'
'`NNN`' is a interface's number and refers to the ifIndex. The OID value is a VLAN's number find in dot1qVlanStaticName and is the VLAN on which the interface tags.

- *a3VlanE*
  '`.1.3.6.1.4.1.43.10.1.14.4.1.1.3.NNN`'
  '`iso. org. dot. internet. private. enterprises. a3Com. generic. genExperimental. genVirtual. a3ComEncapsulationGroup. a3ComVlanEncapsIfTable. a3ComVlanEncapsIfEntry. a3ComVlanEncapsIfTag. NNN`'
  '`NNN`' refers to the ifIndex and more precisely to an virtual interface (called encapsulation's interface) which is associated to a port in ifStack to configure the tagging behaviour of this port. It's the tag used when encapsulating packets transmitted, or de-encapsulating packets received through this interface. The OID value is the tag value (a VLAN's number).

Encapsulation is the process of adding control information to a datagram so that it can be transmitted by a protocol or networking technology other than that for which it was originally intended.

4.4 Compatibility table

- 3Com CoreBuilder 3500
  - List of VLANs: a3VlanGM
  - Interface/VLAN table: ifStack
  - Tagging configuration: a3VlanE
- 3Com SuperStack II
  - List of VLANs: a3VlanGM
  - Interface/VLAN table: ifStack
  - Tagging configuration: a3VlanE
- Cabletron Smartstack ELS100-S24TX2M (now called Vertical Horizon by Enterasys)
  - List of VLANs: dot1qVlan
  - Interface/VLAN table: dot1qEgress
  - Tagging configuration: dot1qPvid
- Enterasys Matrix E5
  - List of VLANs: dot1qVlan
  - Interface/VLAN table: dot1qEgress
  - Tagging configuration: dot1qPvid
- Bay Networks 2216T
  - List of VLANs: (dot1qVlan)
  - Interface/VLAN table: (dot1qEgress)
  - Tagging configuration: (dot1qPvid)

4.5 Heuristic for forwarding lists dispatching by VLAN

For 3Com switches, it rules because we get the forwarding lists by VLAN using special community strings, suffixing the standard community name with `@` followed by the VLAN number.

For other switches, it is much more tricky. We have to guess on which VLANs mac addresses of the forwarding lists are seen by hosts. Suppose you have a host A that sees on one of its ports the MAC address of host B, without knowing on which VLANs B packets are tagged.

If at least one of the hosts A and B is VLAN-aware:

- If B knows the VLANs on which A packets are tagged, the problem becomes rather simple. We assume then that A sees B on the same VLANs as B sees A.
- If not, we have two methods:
  - (very slow) we try to collect the ports on which B sees A, and to make the union of the VLAN lists for all these ports. We also get the VLAN list for the port on which A sees the MAC address of B. Finally, we do the intersection the VLAN lists for A and B.
  - (quicker) we try to collect the VLAN list for host B. We also get the VLAN list for the port on which A sees the MAC address of B. Finally, we do the intersection the VLAN lists for A and B.

Otherwise, both A and B are not VLAN-aware:

We look for the host seeing B on one of its ports but with a minimum number of entries in the corresponding forwarding list. Then we fetch the tagging configuration for this port, and we assume this MAC is always seen on this VLAN.

5 Tree modules

5.1 List of nodes

The module '`nodesl.h`' implements functions dealing with list of nodes. These lists are used to help some tree-related functions. They are typically used to store a list of *host_id*.

- `belong_to` permits to know if a node belongs or not to a given list.
- `build_nodes_list` builds the list of nodes which belong to the tree that you give as parameter.
- `erase_list` erases a list and releases the memory became useless
- `give_same_node` returns the first node of a tree (met during the traversing in depth) which belongs also to the list given as parameter
- `check_ancestor` researches the first ancestor of a node in a tree which belongs to a list given as parameter.

5.2 Tree Manipulation

The module '`tree_traversing.h`' implements functions dealing with the manipulation of trees. These functions are usefull to implement the module `fusion`.

- `root_of_tree` gives the root of the tree given as parameter.
- `make_node_root` center the tree given as parameter on the node given. In fact, it change the tree by considering host as the new root and move all host's ancestors (father, grandfather...) by placing the father as a son to host, the grandfather as a son to the father and so on.
- `traverse_tree` draws the tree given as parameter.
- `copy` makes a copy of the tree given as parameter.
- `copy_part_of_tree` permits us to copy the arborescence under the node given as parameter in a tree given.
- `erase_tree` erases the tree given as parameter abd releases the memory given as parameter.
- `erase_treevlanl` erases the arborescence given as parameter.
- `delete_son` erases the son of a tree and also all the arborescence under this node.
- `look_for_father` researches the `host_id` of the father of the node given in a tree given.
- `add_part_of_tree` adds to a tree an arborescence given under a node given.
- `has_brother` precises if a node given in a tree given has a brother.
- `give_brother` returns the closest right brother of a node in a tree given.
- `has_son` precises if a node given in a tree given has a son.
- `give_son` returns the closest left son of a node in a tree.
- `stack_ancestor_host` returns a stack of all ancestors of a node given in a tree. First, there's the father in the stack, then the grandfather and so on.

- `give_vlan` researches the value of a VLAN of a branch of tree. It makes the copy of the structure VLAN given and returns this copy.
- `give_vlan_no_copy` precises the structure VLAN of a branch of tree without making a copy.
- `add_numvlan_at_end` adds a value of vlan at the end of vlan list.
- `add_list_vlan` adds a value of vlan at the end of a list given. if this value given doesn't belong to the list, it's added. Otherwise, the list given is not modified.

5.3 Fusion

This module '`fusion.h`' implements functions that builds the fusion of two trees. The function `fusion` doesn't modify the two trees which you want to merge. It builds a new tree which results of the fusion of thesetwo trees. If there isn't any common node between both trees, or if a buckle appears, the working of fusion has been given up. At the begining, this function copies the first tre given as parameter. By this way, we obtain a new tree. It researches, during the traversing in depth of the first tree, the first node which belongs also to the second tree. It centers the second tree on this node. Finally, it traverses in depth the second tree modified and adds to the new tree the branches which has not yet in the first tree. It 's this new tree the result of the fusion of two trees.

Warning The trees entered can not have yet information about VLANS. In this case, you have to precise the value of the VlAN of each trees thanks to the parameters num_vlan1 and num_vlan2. If the tree entried has information about VLANS, you give the value -1 to the parameter num_vlan.

6 Misc modules

6.1 Stack functions

The module '`stack.h`' implements a stack of integers.

- `push` and `pop` do the classical stack stuff.
- `copystack` duplicates a whole stack.
- `show_stack` is for debugging purpose, it dumps the stack on `stderr`.

6.2 Snmpz Module

This module encapsulate functionnalities linked to Snmp requests. A session is started thanks to `snmpz_open` and is finished with `snmpz_close`. The request can be classified by the type of the request result :

- `snmpz_getX` or `snmpz_getnX` for get or get next Snmp request
- X is the letter for the return type of the method.

A getnext sequence must be initialized using snmpz_walkX.

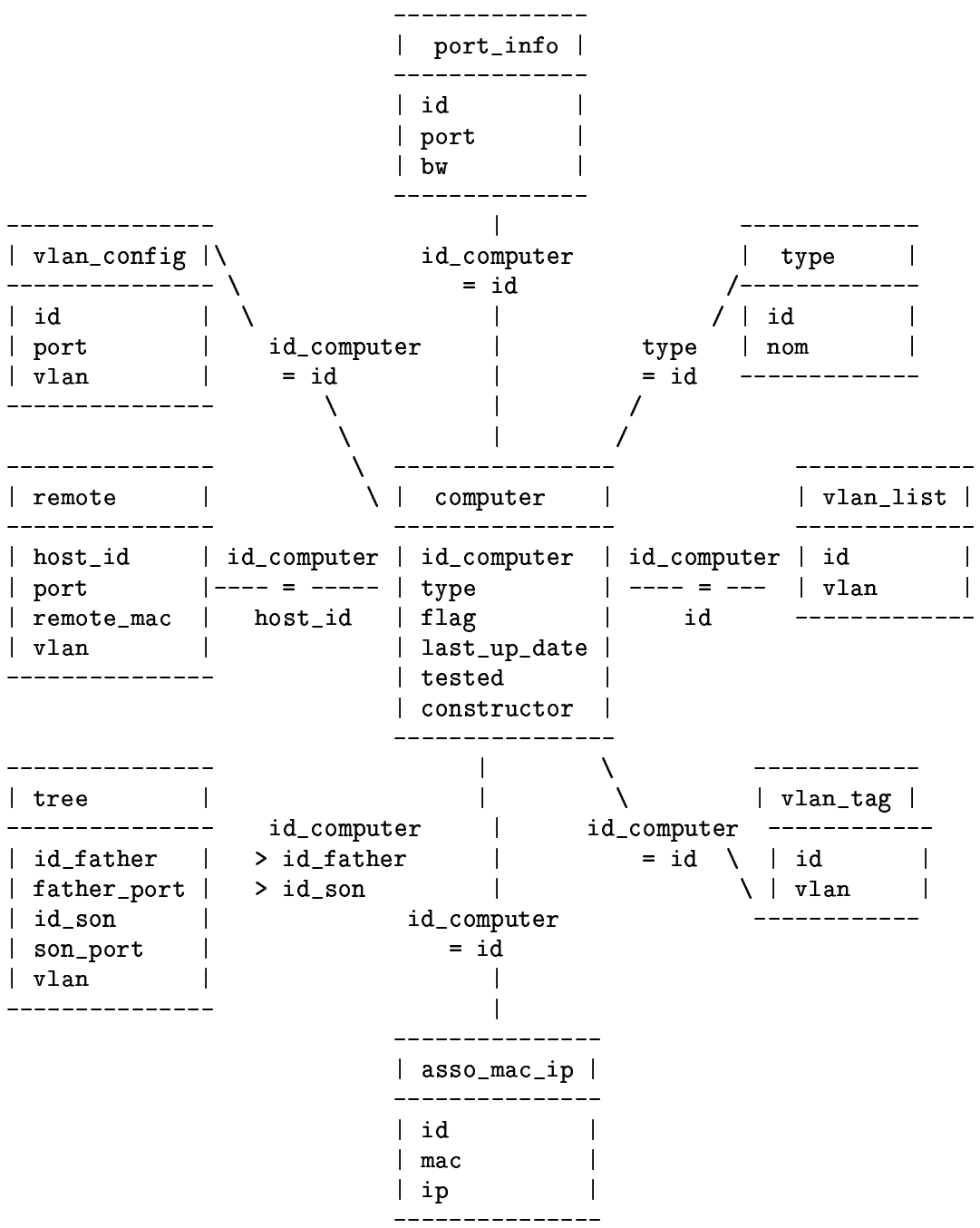
6.3 Module Organization

The package psnmp is organized with layered APIs. The lowest ones deal with the database and the network Snmp requests. On top of these, two modules, one in charge of the topology calculation and another of the graphic interface.

7 Database

7.1 Database Organisation

We have placed the datas in the different bases in nine different tables.



- `computer` : is the main table. It permits us to store the last date where the computer reponds to a SNMP query and the type of the computer. The field `tested` is an information used by the tree building algorithm.
- `type` : defines the different types of computer that you can met : switch, station, router...
- `port_info` : It permits to keep in memory band width of a given computer on a precise port.
- `asso_mac_ip` : makes the correspondence between the identifier of a computer, the MAC address, and the IP address.
- `vlan_config` : show which vlan goes through a port of an host.
- `vlan_list` : give all vlans which go through an host.
- `vlan_tag` : give, for a port of an host, which tag about vlan is made.
- `tree` : give all links between an host (the father) and all of its sons with a specified vlan.
- `remote` : store all adress MAC which are known by the host for a each port.

8 Topology algorithm

8.1 Description

This algorithm which builds the topology suppose there is no loop in the network, and '`cut`' the equipement which create the loop if necessary.

After collecting datas, the main operation is the determination of direct link between equipments. The link are non-oriented.

First we just look at internal nodes in order to build the tree, and after that we'll consider and place leaves such as terminal and computer. This allow us to reduce drastically the number of equipments during the building.

8.2 Placing internal nodes

We are talking about internal nodes which we are trying to place.

First, each equipment is on status '`non-tested`', and a temporary root is choosen (this is the location on which the software is running).

This algorithm place some equipment, and call itself recursively on each direct son of the node on which it has work after changes it has to do on the tree.

In fact, for a node called '`N`', a call of the algorithm on '`N`' will do the following operation:

- it collects
  - the list of the equipments that '`N`' see,
  - and the list of the equipments which see '`N`' and which are in the '`non-tested`' status.

  The equipments in these lists and '`N`' are put in '`tested`' status.
- Then it try to place each of them, with the function '`host\_place\_son`' we'll describe after.
- At last, we apply the algorithm on each direct son.

The arguments of '`host\_place\_son`' are 2 nodes, '`N`' and '`F`'; this function try to place '`F`' in the part of the tree (which represents the building in progress) where the root is '`N`'.

- It collects the direct son list of '`N`' on the port where '`F`' is seen if this port is known, otherwise on all the ports. This list can be empty.
- For each of the elements (called '`Fi`') of this list, '`host\_place\_son`' try to know the closest equipment ('`F`' or '`Fi`') to '`N`':
  - If it's '`F`', the link N~>~Fi is update to F~>~Fi and we create the link N~>~F
  - If it's '`Fi`', then '`F`' is under '`Fi`' and we apply '`host\_place\_son`' with '`Fi`' and '`F`' as arguments. We use a flag to be sure not to find two node '`Fi`' closer to '`N`' than '`F`' (i.e. '`F`' has two father), because we work with trees and this case is impossible. Otherwise we do nothing.
  - Else, we try to solve this problem with ancestors of '`N`'. At last, we do nothing.
- If there is no son closer to '`N`' than '`F`', then '`F`' is placed as a direct son of '`N`', and we give the port of this link if it's known.

When we want to know how to place two sons of a node, we compare on which ports each son see his father and the other son.

8.3 Placing leaves

For each node we place its leaves seen on free ports (i.e. all the ports where this node don't see another node of the builded tree). We create a link for each leaf.

9 Interface specification

9.1 Loading Tree for Vlans

When the interface is started, at first it initialize a lists of a structure containing a VLAN identifier and the related tree. This list is built by uploading the VLAN list from the database and issue a tree buildding function for each of them.

Then, controls of the interface are created based on these trees i.e the tree explorer and the topology map. If no VLAN can be retreive from the database or all the VLAN-trees are empty the interface will not start.

9.2 Topology Map

The topology map is a GTK control. It encapsulates the lists of host and links that are part of the map. Hosts are placed on a hash table accessed through their computer's id while links are stored in a pointer array.

Graphical hosts of the map are stored in GTK containers drawn on a layer. This choice limits the graphical possibility of the user-interface (like transparency, or data based drawing). However, the program doesn't need to bother drawing these hosts : the layer automaticaly redraws it.

9.3 Vlan Merging

The user interface allow to display the merging if it is possible of some Vlan trees choosed by the user. The Vlan trees are selected through the interface menu, nevertheless at least one of them must be selected. The merge can failed i.e a loop have been detected while the merging of trees. In this case, the first Vlan tree will be reloaded, the map wil be cleaned and finally the tree-like-explorer will be reinitialized.

Table of Contents