Information Processing Letters 30 (1989) 21–26
North-Holland

16 January 1989

THE REFLECTED TREE HIERARCHY FOR PROTECTION AND SHARING

Ravinderpal Singh SANDHU

Department of Computer and Information Science, The Ohio State University, Columbus, OH 43210, U.S.A.

Communicated by Fred B. Schneider
Received 21 September 1987
Revised 8 July 1988

Consider a rooted tree with its leaves aligned on a horizontal line. The tree and its mirror image below the line give us a reflected tree. This is a natural structure for protection and sharing with nodes corresponding to protection groups, partially ordered by the subgroup relation. Reflected trees have an efficient representation which can be incrementally modified as the tree is changed.

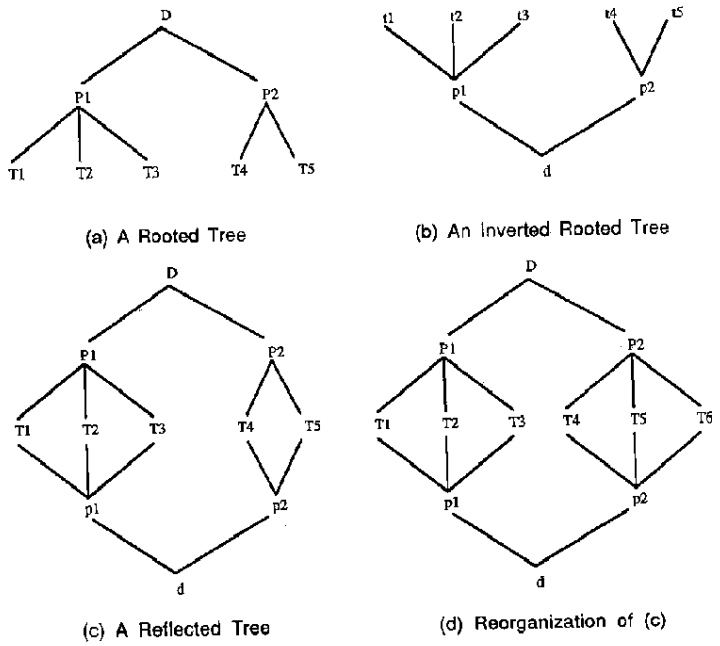
Keywords: Access control, groups, hierarchy, protection, tree

1. Introduction

It is a useful and common practice to provide facilities for protection and sharing with groups of users as unit. Membership in groups is presumably based on the need to share resources and information, so the group is an appropriate unit. System administrators can add and delete users from groups without concerning other users about these changes. Simpler systems typically allow access control only in terms of groups (e.g., [1]). Even the more sophisticated systems with access controls at the level of individual users usually support groups (e.g., [2]). It is almost inevitable that protection groups in an organization will be related by lines of authority and responsibility. We say U is a (proper) subgroup of V or $U \subset V$ if every member of U is thereby automatically a member of V but not vice versa. Note that members of U are more privileged than members of V. By definition, $\subset$ is asymmetric and irreflexive. It is natural to assume that $\subset$ is transitive. Then $\subset$ is a strict partial ordering or hierarchy on the groups. We emphasize that $\subset$ is a relation on groups and not on users of files. A user may belong to multiple incomparable groups as determined by his various roles in an organization. Similarly, a file or resource may be accessible by several incomparable groups.

The rooted tree is perhaps the simplest hierarchy in this context. Figure 1(a) shows an example for a department D with projects $P1$, $P2$ respectively with tasks $T1$, $T2$, $T3$ and $T4$, $T5$. The department managers are members of D and thereby automatically members of project and task groups within that department. Similarly, the supervisors of project $P1$ are members of $P1$ and thereby members of the task groups within $P1$. The rooted tree is a good structure for the purpose of oversight and separation. The group at the root of a subtree has an oversight of files available to the groups in that subtree. Siblings are mutually incomparable and thereby separated. However, a rooted tree does not support sharing among incomparable groups. We point out that oversight applies only to files which users have chosen to make accessible to various groups. Thus, a supervisor of $P1$ chooses which of his files he makes accessible to the group $P1$. These files are then accessible by members of $P1$ and D. The supervisors of $P1$ are, of course, at the same time entitled to keep private files not accessible by any group.

The projects in a department may need to



(a) A Rooted Tree

(b) An Inverted Rooted Tree

(c) A Reflected Tree

(d) Reorganization of (c)

Fig. 1.

access common resources and files. For this purpose an inverted rooted tree is the appropriate hierarchy. Figure 1(b) shows an inverted tree which is the mirror image of the tree of Fig. 1(a). In this case, resources available to group d are accessible, and hence shared, by all the task and project groups of that department. The resources available to $p1$ are accessible by tasks $t1$, $t2$ and $t3$ but not by tasks of project $p2$. The inverted tree also supports separation since "siblings" such as $p1$ and $p2$ are incomparable. However, the inverted tree does not support oversight.

We propose the *reflected tree* as a hierarchy to support separation, oversight and sharing in a uniform, simple and natural manner. A reflected tree is obtained by combining a tree whose leaves are aligned on a horizontal line [1] with its reflection below that line. Figure 1(c) shows the reflected tree corresponding to the tree of Fig. 1(a).

The $\subset$ relation represented by this Hasse diagram is obtained by orienting the edges downwards and taking the transitive closure of the resulting acyclic directed graph. We generally denote groups above and below the reflecting line in upper and lower case respectively so that mirror image groups get the same name, ignoring case. Files available to $P1$ are accessible by members of D and $P1$. On the other hand, files available to $p1$ are accessible by members of D, $P1$, $T1$, $T2$ and $T3$. Groups above the line facilitate oversight, while those below the line support sharing. Separation is supported both above and below the line. The relationship between a pair of groups in the upper half is turned around with respect to their mirror images, and vica versa. Let U and V be groups which are either both in the upper half or both in the lower half, with mirror images U' and V' respectively [2]. Then

[1] If the leaves are at different levels, we can stretch or shrink some branches to achieve this.

[2] A group on the line is its own mirror image.

(1) $U \subset V \Leftrightarrow V' \subset U'$,

(2) $U \sim V \Leftrightarrow U' \sim V'$

($\sim$ means incomparable).

The second property actually holds for all U, V whether or not they are on the same side of the line.

The reflected tree is a natural and simple structure for exchange of information and control of resources. In the context of Fig. 1(c), consider a disk which is paid for by the sponsor of project $P1$, who insists that it be used only for the purpose of project $P1$ and that there be some accountability for this fact. This is achieved by making the disk accessible by the group $p1$, so none of the groups under $P2$ have access to it. $P1$ and its task groups properly have access to the disk. The department managers as members of D also have access to the disk and are presumably trusted not to use the disk for work on project $P2$. However there is little incentive to violate this trust since this work will not be accessible by members of $P2$, $T4$ and $T5$. Alternately, we can implement a limited-access mode which provides access to the disk labeled $p1$ only to groups in the range $p1$ and its mirror image above the line. The same considerations apply to a piece of proprietary software which is licensed for use only with project $P1$.

As another example, consider a bulletin board system in which users post messages tagged with the group they are intended for. We could allow two modes of posting, say exclusive and shared, respectively indicating that the message is exclusively for the named group or for that group and its subgroups. In this way, a message tagged exclusively for $T1$ will be readable only by members of $T1$, while a message tagged as shared for $T1$ will be readable by members of $T1$, and $P1$ and D. A message tagged as shared for d will be readable by members of any group in the department. Going a step further we could distinguish two shared modes, say limited and unlimited, respectively indicating that the message is to be read by groups in the range between the named group and its mirror image or all subgroups. Then a message tagged as limited shared for $p1$ will be readable only by the members of $P1$, $T1$, $T2$ and $T3$. Whereas a message tagged as unlimited shared for $p1$ will in addition be readable by members of D.

In this paper we will show that reflected trees have a particularly elegant and maintainable representation. In Section 2 we show that a reflected tree can be represented by assigning a pair of integers called *lr-values* to each group. The subgroup relation between any two groups is easily determined by comparing their *lr*-values. Moreover, a subtree (and its image) can be replaced by a forest of subtrees (and their images) without affecting the *lr*-values of groups outside that subtree. Section 3 discusses some pragmatic issues in applying these ideas and some extensions to the representation developed in Section 2.

2. Representation by *lr*-values

Our technique for representing reflected trees is based on the familiar concept of pre-order traversal of a tree defined by the following recursive procedure:

(1) Visit the root.

(2) Traverse the subtrees, if any, of the tree in pre-order.

We perform two pre-order traversals of the upper half of the reflected tree, including groups on the reflecting line. In the left pre-order traversal, siblings are visited in left to right order, whereas in the right pre-order traversal, they are visited right to left. These traversals are respectively identified as L and R. For the reflected tree of Fig. 1(c), $L = (D\ P1\ T1\ T2\ T3\ P2\ T4\ T5)$ and $R = (D\ P2\ T5\ T4\ P1\ T3\ T2\ T1)$. The position of a group in L and R respectively defines the *lr*-values assigned to groups in the upper half. Groups below the line are assigned the same *lr*-values as

Table 1

group	D	$P1$	$T1$	$T2$	$T3$	$P2$	$T4$	$T5$	$p1$	$p2$	d
type	a	a	a	a	a	a	a	a	b	b	b
l	1	2	3	4	5	6	7	8	2	6	1
r	1	5	8	7	6	2	4	3	5	2	1

Table 2

$t(U)$	$t(V)$	Condition for $U \subset V$
a	a	$lr(U) < lr(V)$
a	b	$lr(U) < lr(V) \vee lr(V) < lr(U) \vee lr(V) = lr(U)$
b	a	false
b	b	$lr(V) < l(U)$

their mirror image. Each group G is also assigned a type $t(G)$ which is one of "a" or "b" respectively indicating that the group is above or below the line. Groups on the line are treated as being above the line for this purpose. This gives us the representation shown in Table 1 for the reflected tree of Fig. 1(c).

It is easily determined whether one group is a proper subgroup of another on basis of the *lr*-values and types of these two groups. Let the notation $lr(U) < lr(V)$ denote $l(U) < l(V)$ and $r(U) < r(V)$. Similarly let $lr(U) = lr(V)$ stand for $l(U) = l(V)$ and $r(U) = r(V)$. The rules of Table 2 then determine whether $U \subset V$.

Theorem. *The conditions of Table 2 are correct.*

Proof. Consider each case in turn.

(1) Let $t(U) = t(V) = 1$. If $U \subset V$, then U is at the root of a subtree in the upper half which includes V, so U precedes V in both L and R and $lr(U) < lr(V)$. If $V \subset U$, then, by the same argument, $lr(V) < lr(U)$ so $lr(U) \not< lr(V)$. Similarly, for $U = V$, $lr(U) = lr(V)$ and $lr(U) \not< lr(V)$. It remains to consider $U \sim V$. In the upper half of the reflected tree the path from the root to U is to the left or right of the path from the root to V. So U precedes V in L and follows V in R, or vice versa. So $lr(U) \not< lr(V)$.

(2) Let $t(U) = \mathrm{a}$, $t(V) = \mathrm{b}$. $U \subset V$ if and only if the mirror image V' of V is comparable with U, that is, $U \subset V'$ or $V' \subset U$ or $U = V'$. So the *lr*-values are related as shown in the table.

(3) Let $t(U) = \mathrm{b}$, $t(V) = \mathrm{a}$. It is obvious that $U \not\subset V$.

(4) Let $t(U) = t(V) = \mathrm{b}$. In this case, mirror images of U and V have the opposite relation than U and V do. So this condition follows from the first case applied to the mirror images. □

Modifications to the existing hierarchy are almost inevitable in the real world. For example, we may add task $T6$ to project $P2$ in Fig. 1(c) to obtain Fig. 1(d). This changes the *lr*-values of all groups other than D, $P2$ and their images. It is particularly awkward that *lr*-values of $P1$ and its tasks, which are unrelated to project $P2$, get affected in this manner. In a reflected tree the subtree is a natural unit for such modifications. In general, a subtree and its image may be replaced by a forest of subtrees and their images. By keeping gaps in the original *lr*-values we can reorganize without affecting *lr*-values of groups outside the reorganized subtree. For this purpose assign a quota to each group. The minimum quota is 1 and a group and its image have the same quota. The *lr*-values are assigned as follows.

$$l(A) = 1 + \sum \mathrm{quota}(X)$$

for all X preceding A in L

$$r(A) = 1 + \sum \mathrm{quota}(X)$$

for all X preceding A in R

Table 3

group	D	$P1$	$T1$	$T2$	$T3$	$P2$	$T4$	$T5$	$p1$	$p2$	d
type	a	a	a	a	a	a	a	a	b	b	b
quota	5	5	5	5	5	5	5	5	5	5	5
l	1	6	11	16	21	26	31	36	6	26	1
r	1	21	36	31	26	6	16	11	21	6	1

Table 4

group	D	$P1$	$T1$	$T2$	$T3$	$P2$	$T4$	$T5$	$T6$	$p1$	$p2$	d
type	a	a	a	a	a	a	a	a	a	b	b	b
quota	5	5	5	5	5	4	5	5	1	5	4	5
l	1	6	11	16	21	26	30	35	40	6	26	1
r	1	21	36	31	26	6	16	11	10	21	6	1

Since groups in a subtree occur consecutively in L and in R, groups outside the reorganized subtree will retain their *lr*-values if the sum of the quotas does not change during reorganization. For example, in Table 3 we show the *lr*-values for Fig. 1(c) with a quota of 5 for every group.

Adding $T6$ to obtain Fig. 1(d) amounts to reorganizing the subtree under $P2$. Let us reduce the quota of $P2$ to 4 and assign T6 a unit quota to obtain Table 4.

3. Discussion

In the example above, $T4$ and $T5$ are the only groups whose *lr*-values get changed after reorganization. If we anticipate the need to add new task groups below $P2$, it is possible to avoid this change also. The idea is to keep a pseudo-group as a place-holder for this purpose. The group PH in Fig. 2(a) would allow us to introduce $T6$ in Fig. 2(b) as a reorganization of PH into the forest consisting of $T6$ and PH. Then the *lr*-values of all existing groups other than PH will be unchanged. We can similarly keep place holders for new projects, departments, divisions and so on at appropriate levels of the tree.

The sum of the quotas of all groups in a subtree determines the maximum number of groups in a forest which may replace it, without modifying *lr*-values outside that subtree. We can be generous in assigning quotas, overestimating by a factor of say 32 at the cost of 5 additional bits per *lr*-value. Also note that if the quotas are insufficient to accommodate reorganization, we can back up one level in the reflected tree and reorganize there.

It is sometimes useful to recognize that group A is an immediate predecessor of B in a hierarchy, i.e., $A \subset B$ and there is no C such that $A \subset C \subset B$. For instance, consider the policy that a user can post information only to groups which he is a member of or groups which are immediate predecessors of these. In Fig. 1(c) this would allow members of $T1$ to post information to $P1$ but not to D. Each group in the upper half of a reflected tree, except the root, has a unique immediate predecessor. For such groups let $lr^+(A)$ be the *lr*-values of A's unique immediate predecessor. Similarly, each group except the root in the lower half has a unique immediate successor. In this case let lr^-(A) be the *lr*-values of A's unique immediate successor. Groups on the reflecting line have a unique immediate predecessor and a unique immediate successor. For these groups let $lr^+(A)$ and $lr^-(A)$ respectively be the *lr*-values of A's unique immediate predecessor and unique immediate successor. In all cases we can recognize whether or not A is an immediate predecessor of

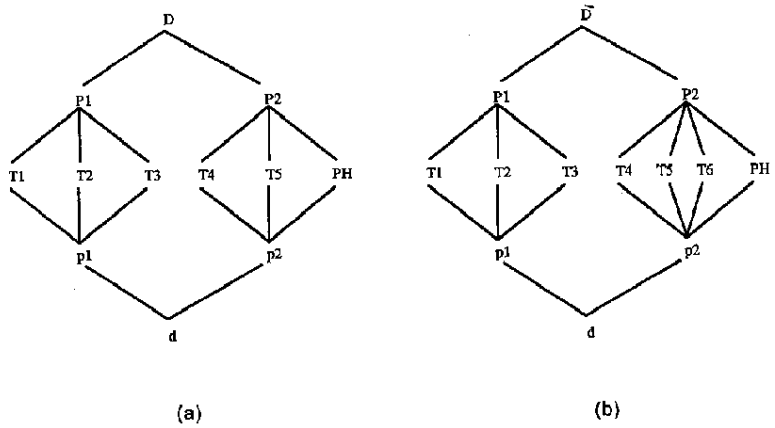


Fig. 2. Place-holders.

B by testing the condition $lr(A) = lr^+(B) \vee lr^-(A) = lr(B)$.

The distinction between exclusive, limited shared and unlimited shared access mentioned in Section 1 can of course be easily implemented in terms of *lr*-values. For limited shared access in particular, we can check if group B is in the range between A and A's mirror image by checking the condition $lr(A) \leqslant lr(B)$. There are other interesting possibilities for exploiting the fact that a group and its mirror image have identical *lr*-values. For instance, consider the policy that a member of group A above the line can add or delete users to groups in the subtree below A and its reflection. That is, in Fig. 1(c) members of $P1$ can modify membership of $T1$, $T2$, $T3$ and $p1$ but not d. We can easily check whether members of A are allowed to modify membership of B by testing whether or not

$$lr(A) < lr(B) \vee [lr(A) = lr(B) \wedge t(B) = \mathrm{b}].$$

As another example, consider the policy that only those documents approved by the supervisors of project $P1$ should be made accessible by group $p1$. More generally, that information can be posted to groups below the line only by (direct of indirect) members of the mirror images of these groups. Recall that groups below the line facilitate sharing, so this is a natural method to ensure the quality and suitability of information posted to these groups. If A and B are respectively above and below the line we can easily check whether members of A can post information to B by checking the condition $lr(A) < lr(B)$.

4. Conclusion

The reflected tree is actually a special case of the ntree hierarchy recently defined by this author [3]. Some properties of reflected trees developed here are related to our results for ntrees. However, there are significant properties specific to reflected trees. The subtree is a natural unit for reorganization of reflected trees. Ntrees have a more complicated structure and it is not clear what a suitable unit for reorganization might be. An appealing and useful aspect is that a group and its mirror image have the same *lr*-values. The methods used for ntrees would assign different *lr*-values for such groups. The technique for recognizing immediate predecessors and successors is also particular to reflected trees since it uses the property that groups above and below the line respectively have a unique immediate predecessor and successor.

References

[1] D.M. Ritchie and K. Thompson, The UNIX time-sharing system, *Comm. ACM* 17 (7) (1974) 365–375.

[2] J.H. Saltzer, Protection and the control of information sharing in MULTICS, *Comm. ACM* 17 (7) (1974) 388–402.

[3] R.S. Sandhu, The ntree: A two-dimension partial order for protection groups, *ACM Trans. on Computer Systems* 6 (2) (1988) 197–222.