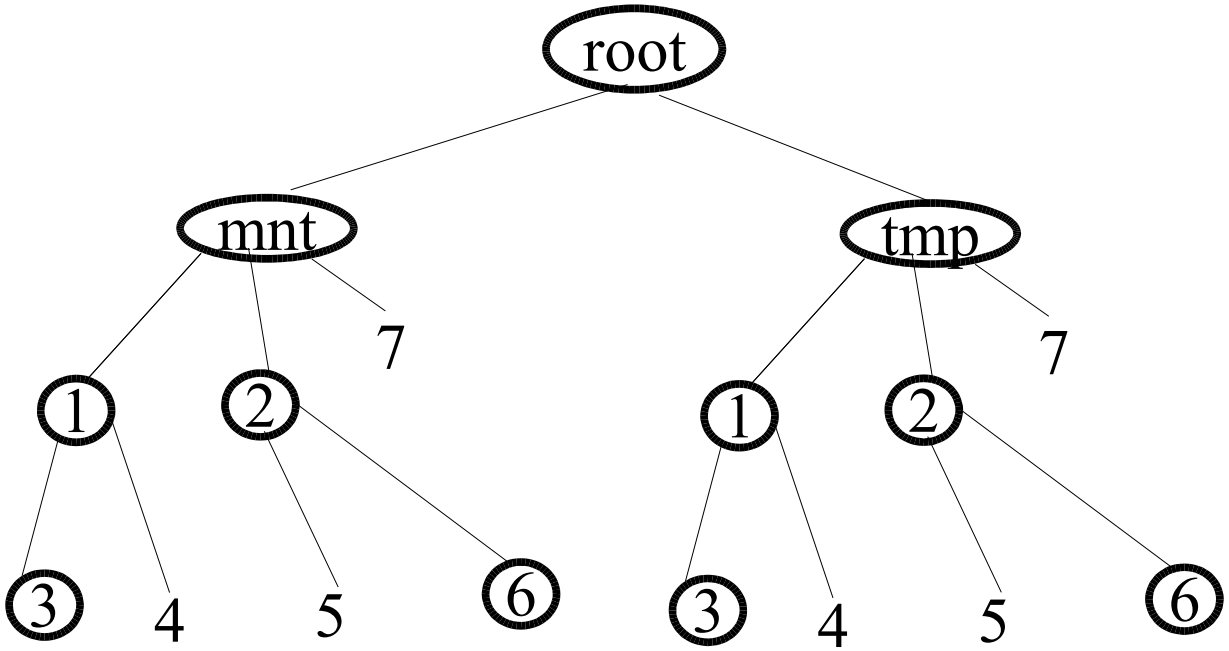
OTTAWA LINUX SYMPOSIUM 2006

Shared Subtree Concept and Implementation and Applications in the Linux Kernel



Ram Pai
linuxram@us.ibm.com

Al Viro
viro@ftp.linux.org.uk

Agenda

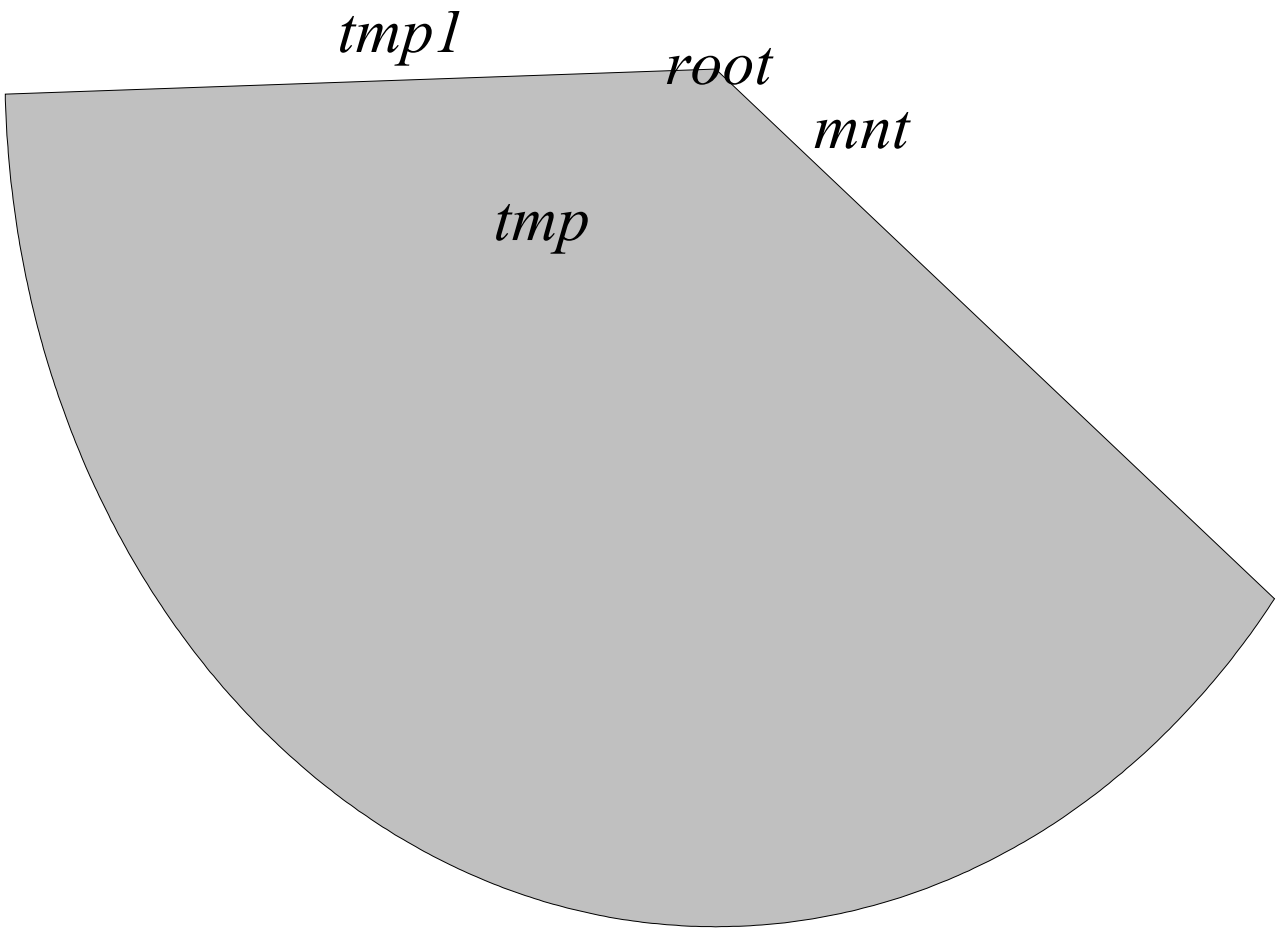
- Background.
- Requirement/Applications.
- Shared subtree solution.
- Shared subtree semantics.
- Implementation detail.
- Future work.

Background

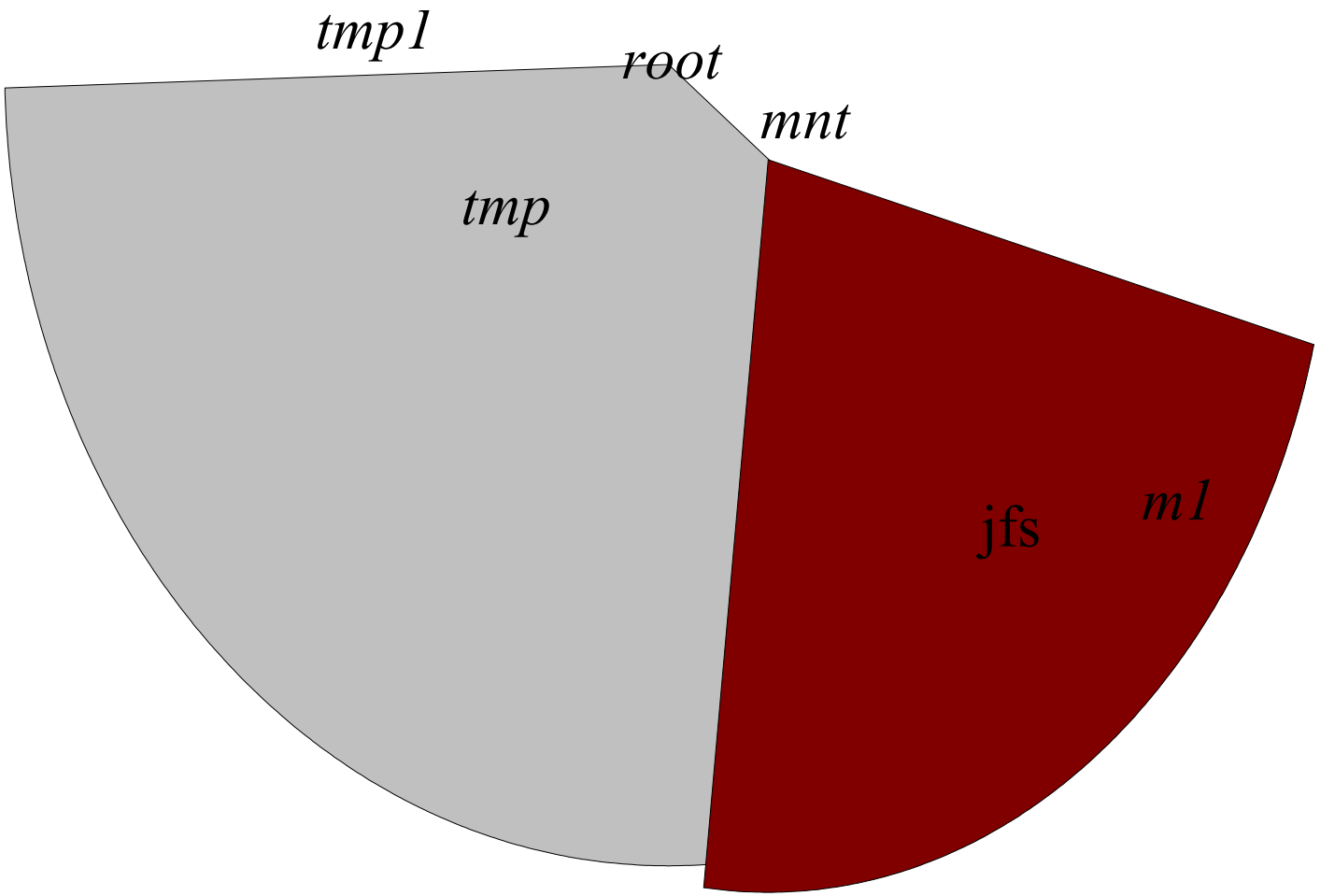
mount semantics in Linux® VFS

- normal device mount (mount /dev/sda0 /mnt)
- bind mount (mount –bind /mnt /tmp)
- rbind mount (mount –rbind /mnt /tmp)
- move mount (mount –move /mnt /tmp)
- namespaces (*CLONE_NS* flag for *sys_clone()*)
- unmount (umount [-l] /mnt)

Background

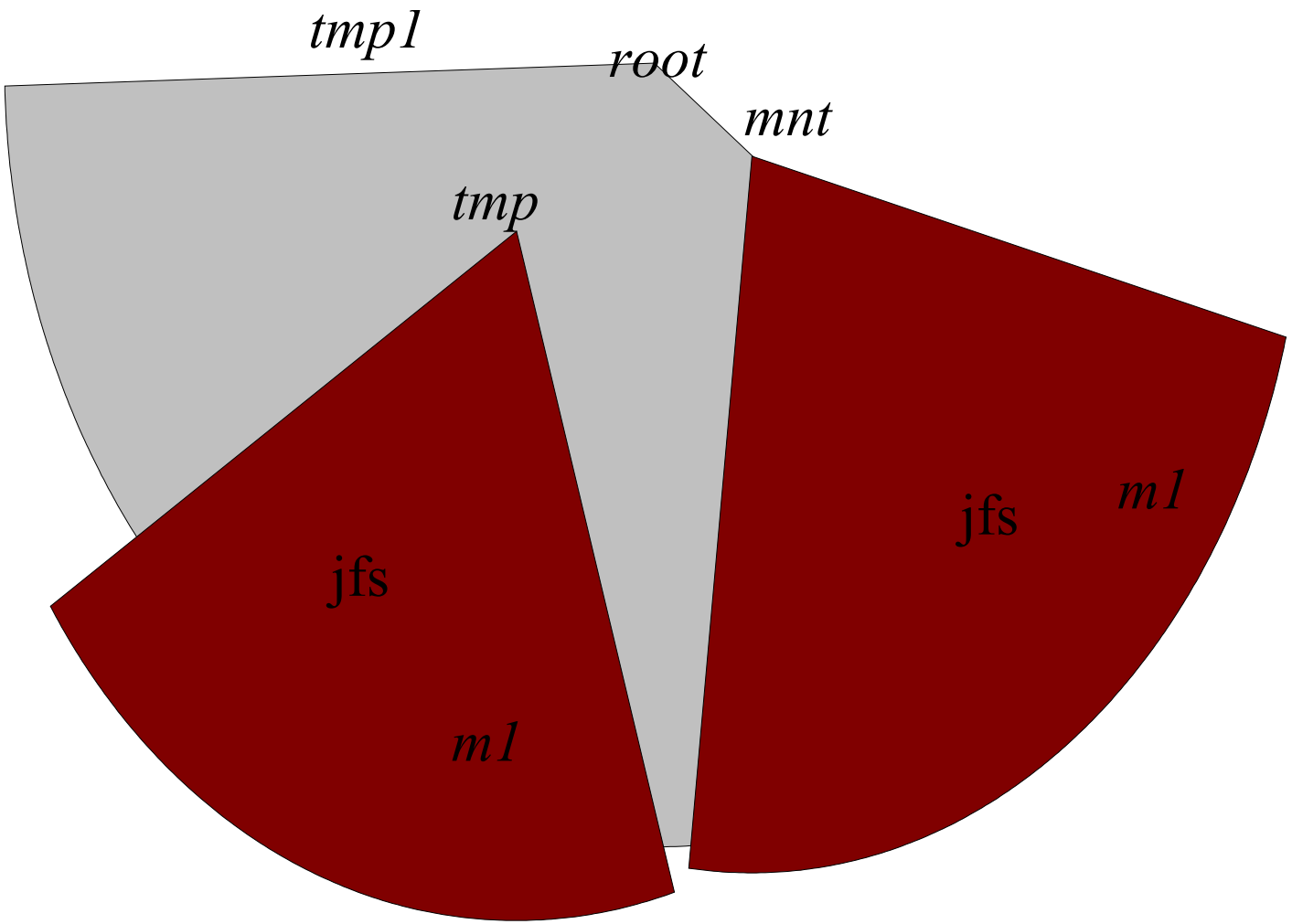


Background



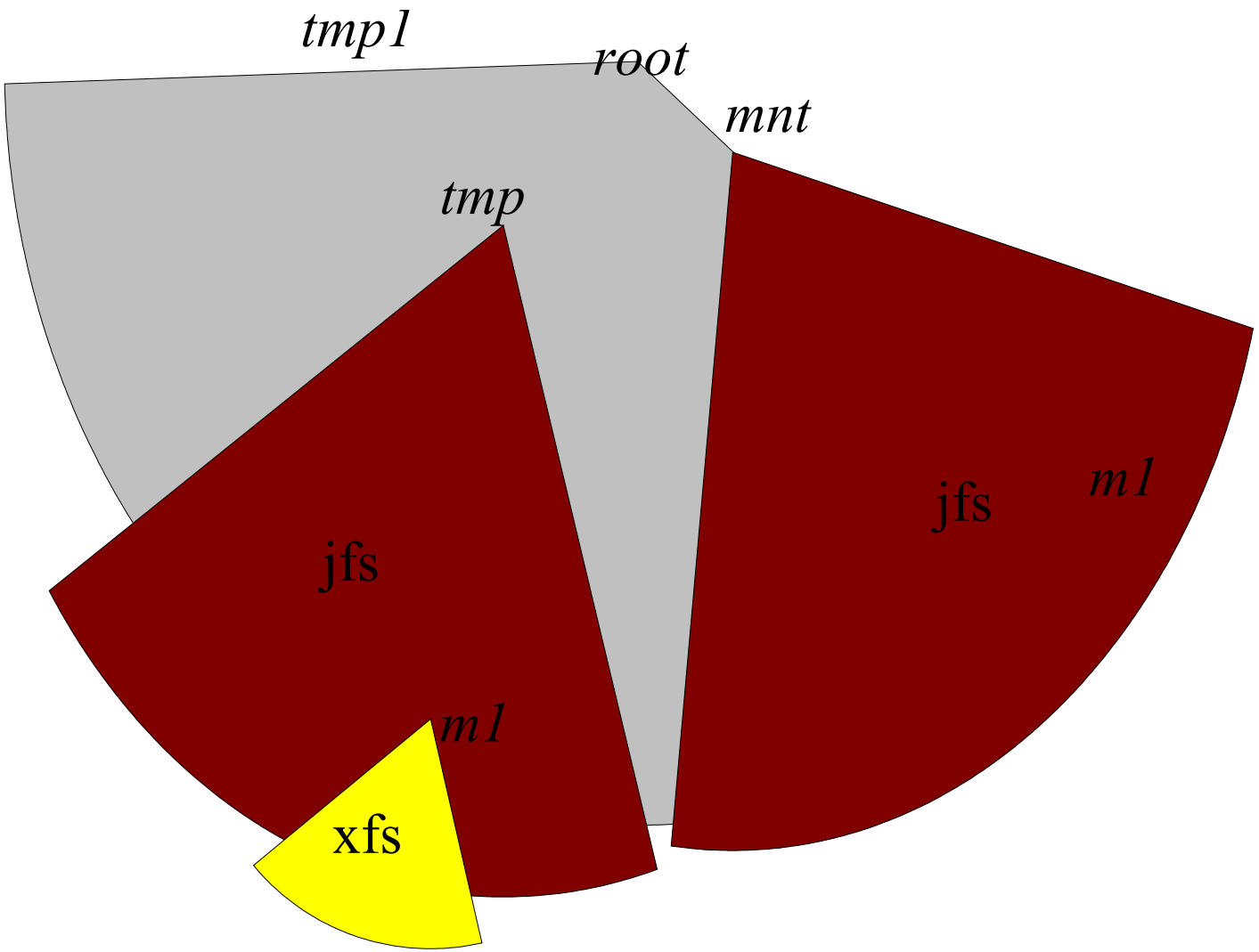
- normal device mount (mount /dev/sda0 /mnt)

Background



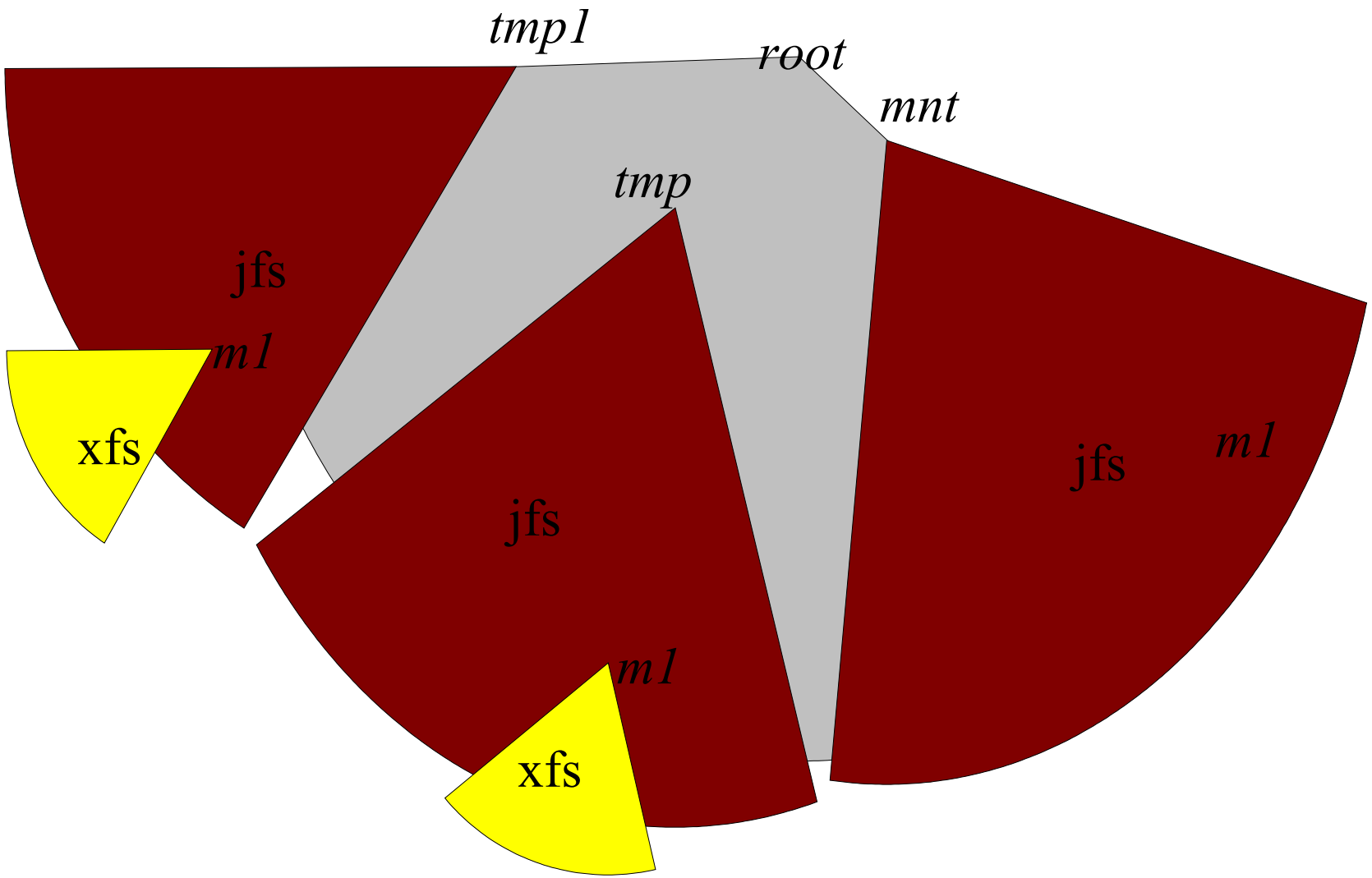
- bind mount (mount –bind /mnt /tmp)

Background



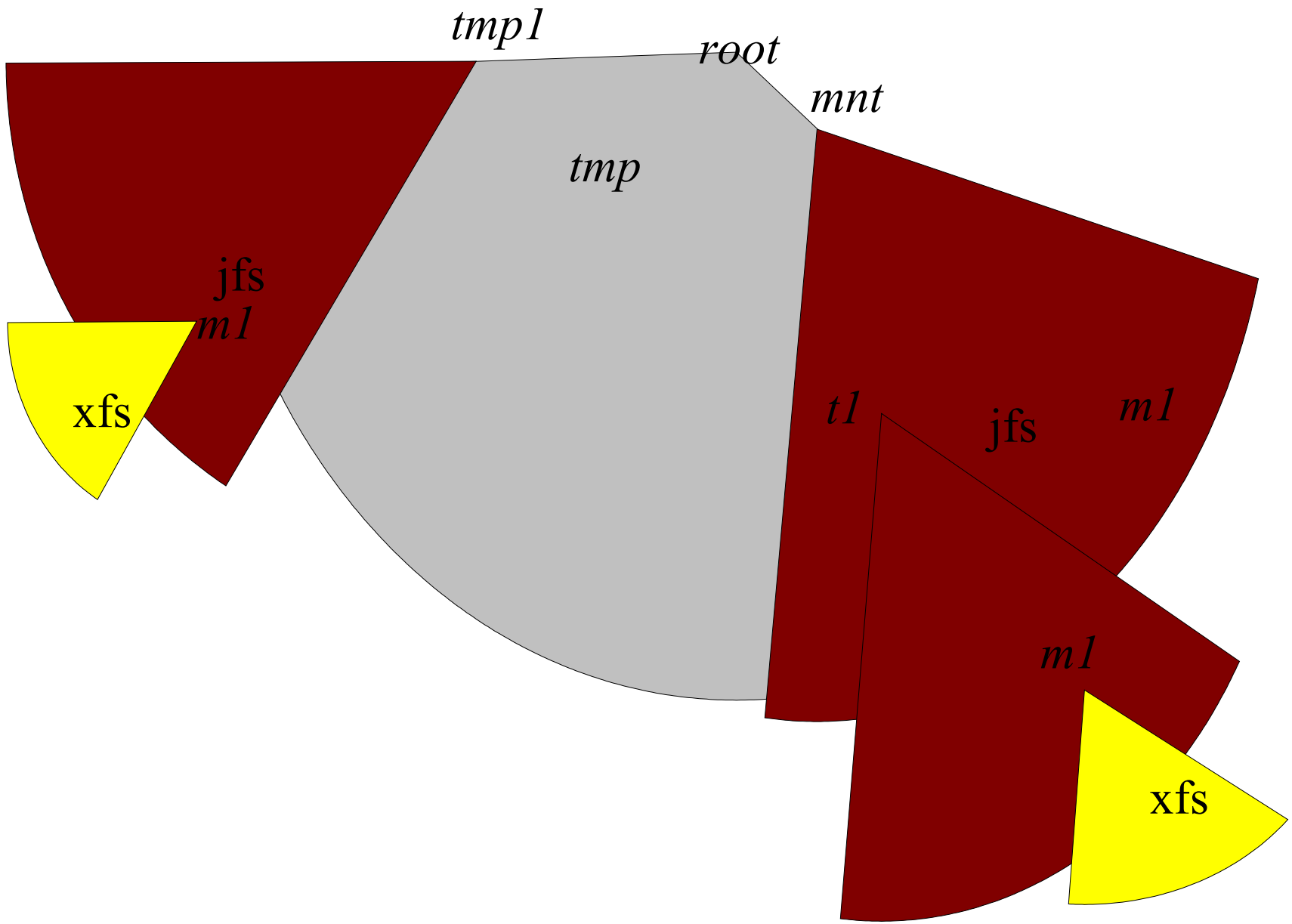
- normal device mount (mount /dev/sda1 /tmp/m1)

Background



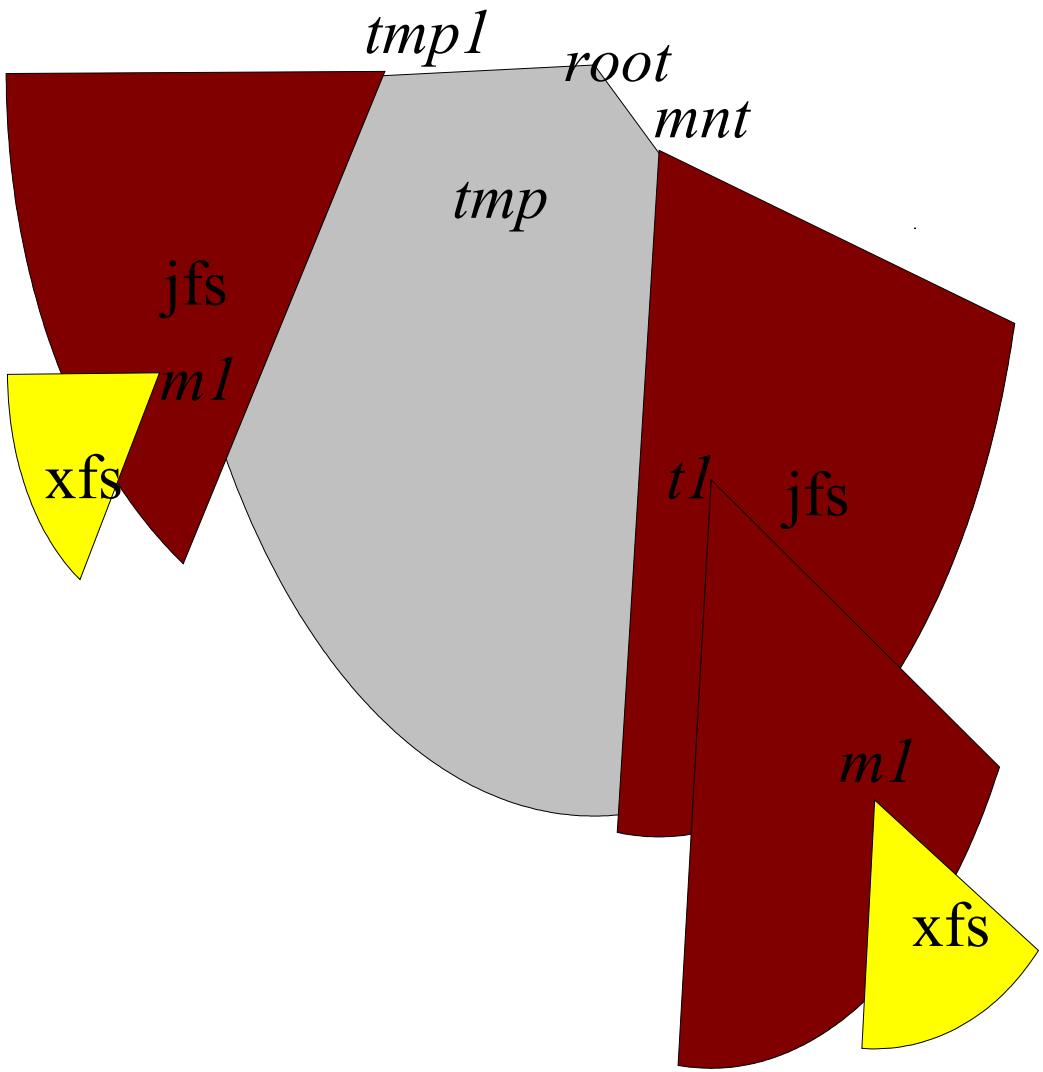
- rbind mount (mount –rbind /tmp /tmp1)

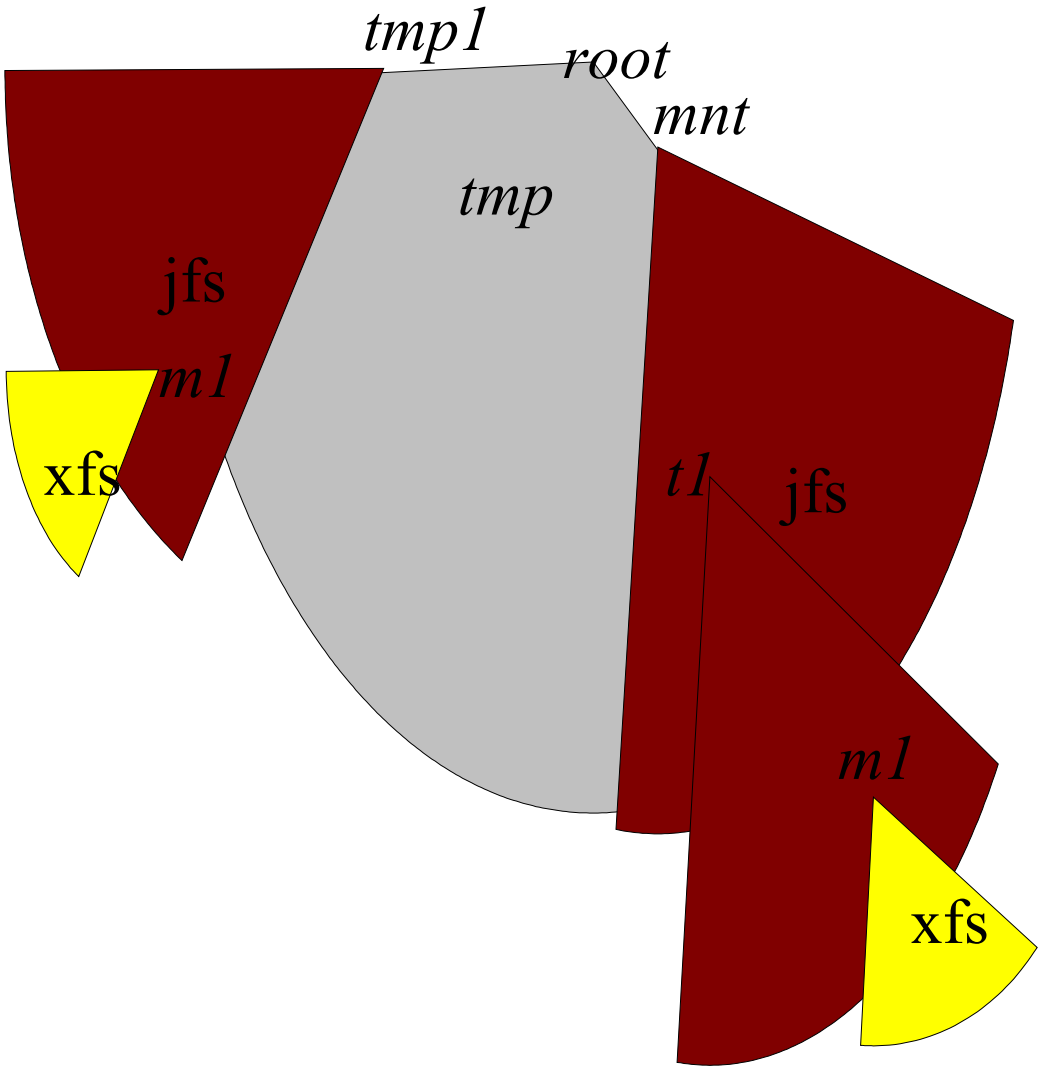
Background



- rbind mount (mount –move /tmp /mnt/t1)

Background





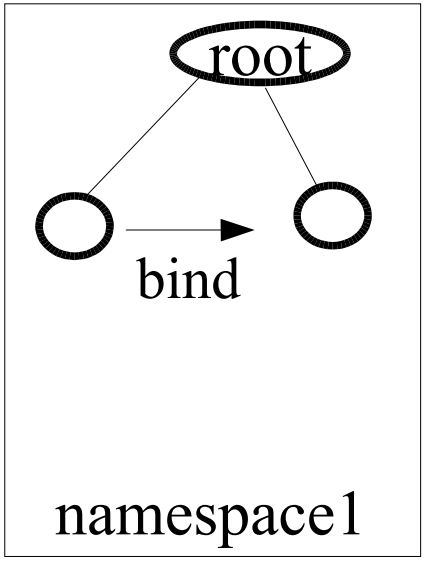
- clone namespace

Requirement

- share mount trees.
  - containers: all containers share the same mount tree
  - MVFS: all views share the same mount tree
  - automounter: mount automatically visible on all filesystem-namespaces
- Private changes to a mount-subtree.
  - FUSE: mount invisible to anybody else.
  - SeLinux LSPP: mount invisible to anybody else.
  - Containers: private mounts not visible to other containers.
- How?
  - clone-namespace (*CLONE_NEWNS in clone()*).
  - rbind (mount –rbind src dest)

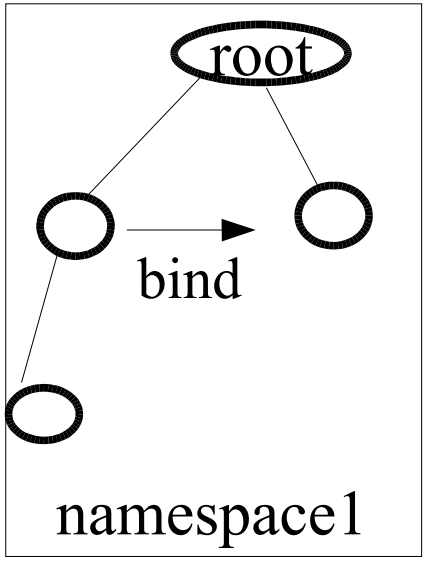
Problem

- bind mounts are static.



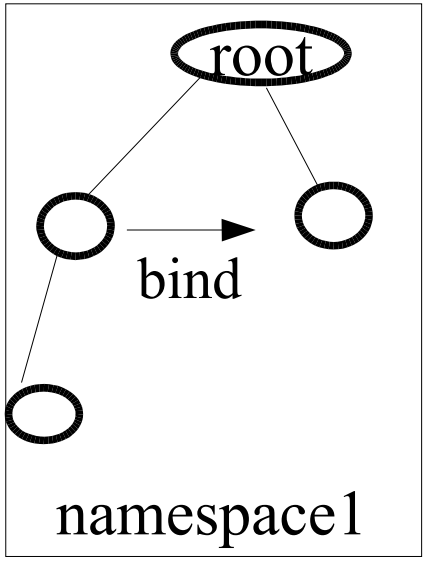
Problem

- bind mounts are static.
  - submounts in one mount instance do not reflect in the other mount instance.

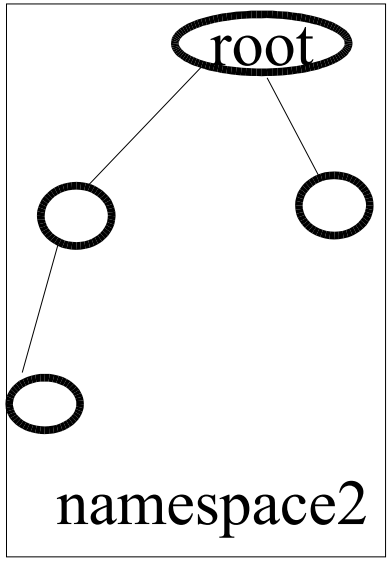


Problem

- bind mounts are static.
  - submounts in one mount instance do not reflect in the other mount instance.

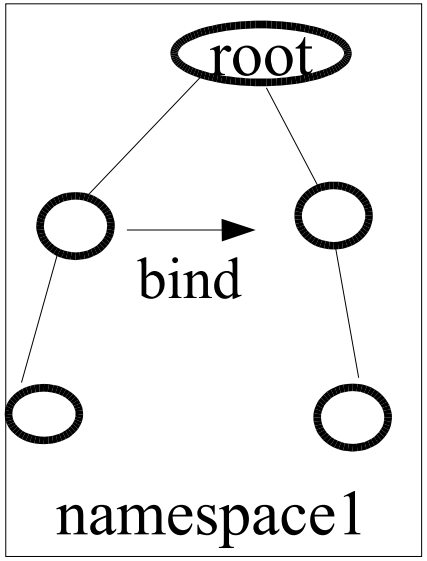


- filesystem-namespaces are isolated.
  - mounts in system namespace are invisible to cloned namespace.

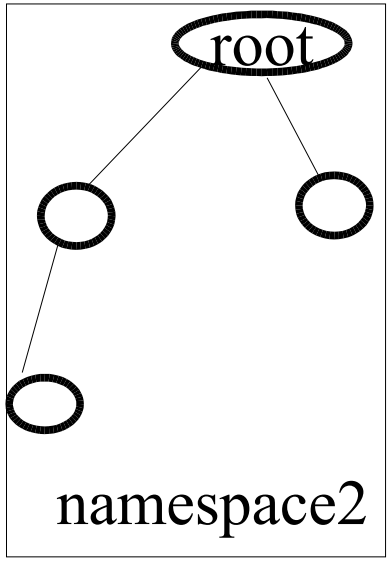


Problem

- bind mounts are static.
  - submounts in one mount instance do not reflect in the other mount instance.



- filesystem-namespaces are isolated.
  - mounts in system namespace are invisible to cloned namespace.



Shared subtree solution

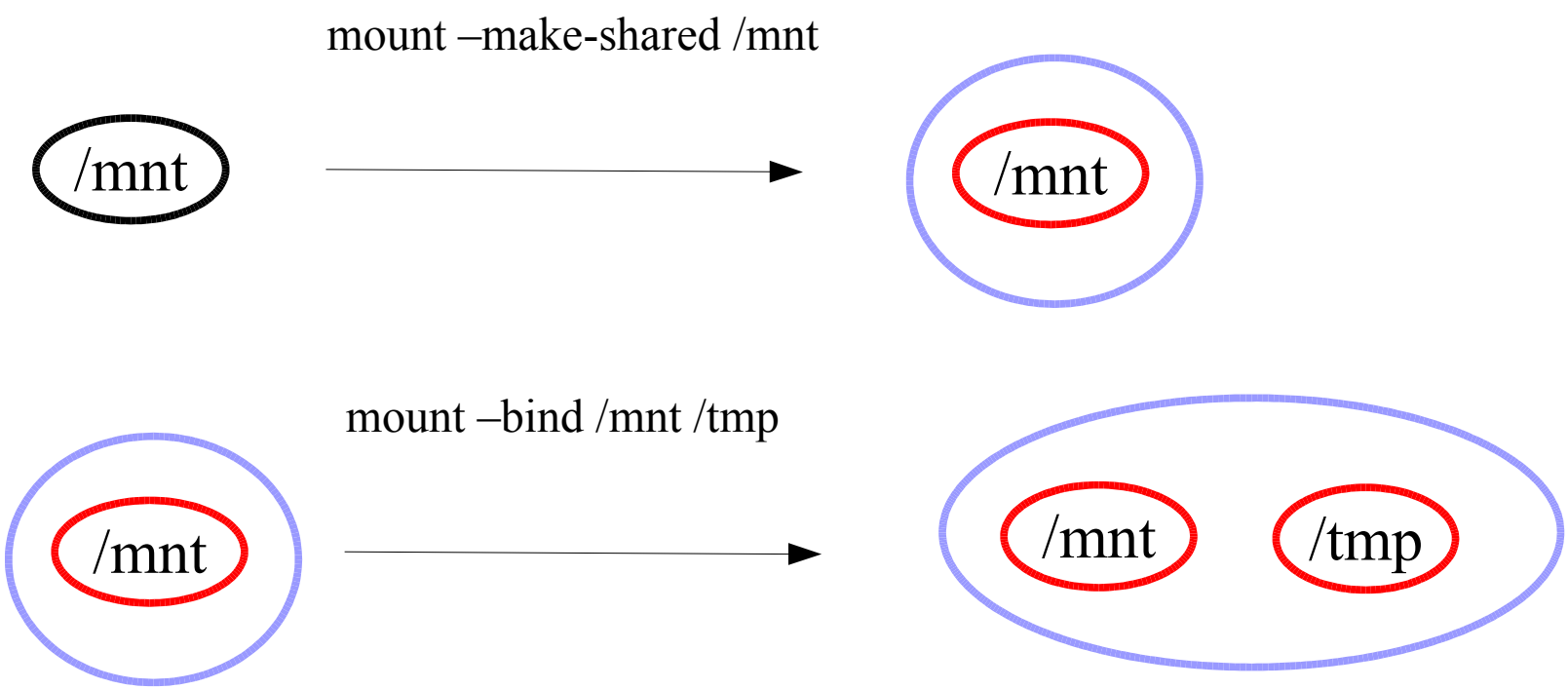
RFC proposed by Al Viro in Jan 2005

- http://lwn.net/Articles/119232/

Ram Pai provided the Linux implementation

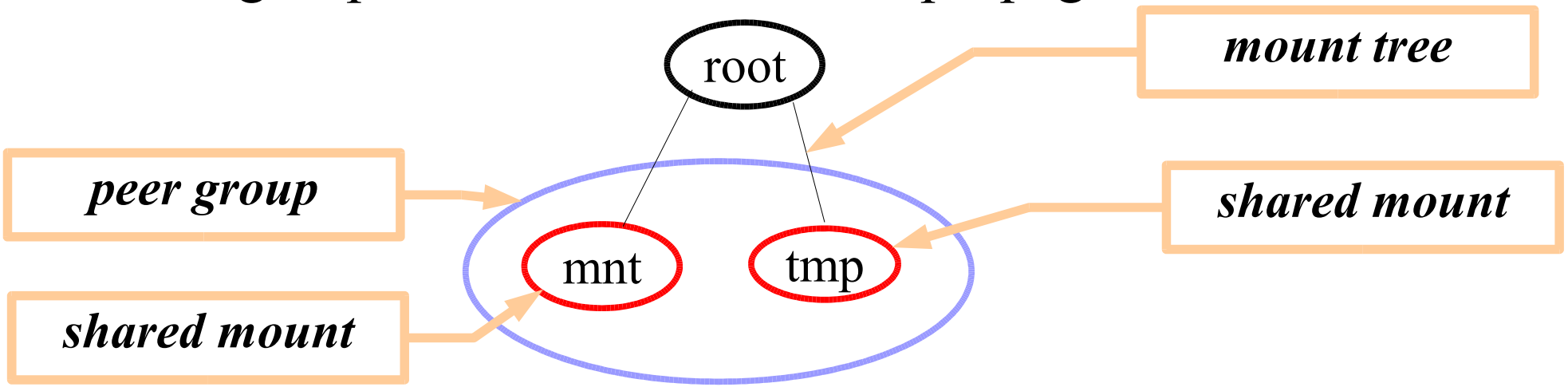
- feature accepted for 2.6.15 Linux kernel

Shared mount



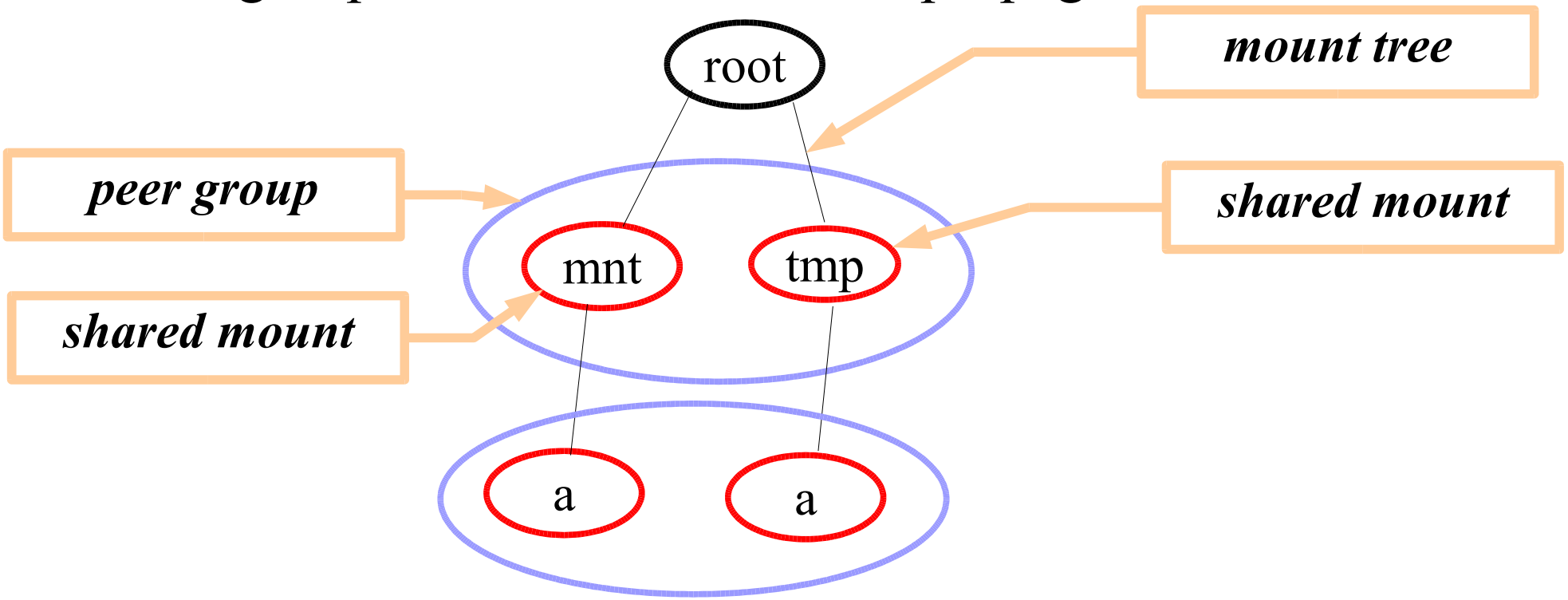
Shared mount

- shared mount
  - mount, unmount events propagate to each other
- peer group
  - group of shared mounts that propagate to each other

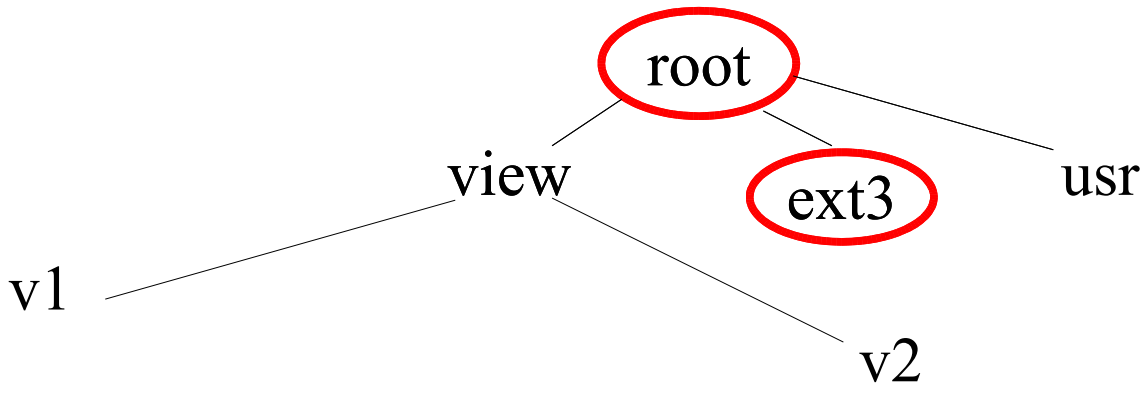


Shared mount

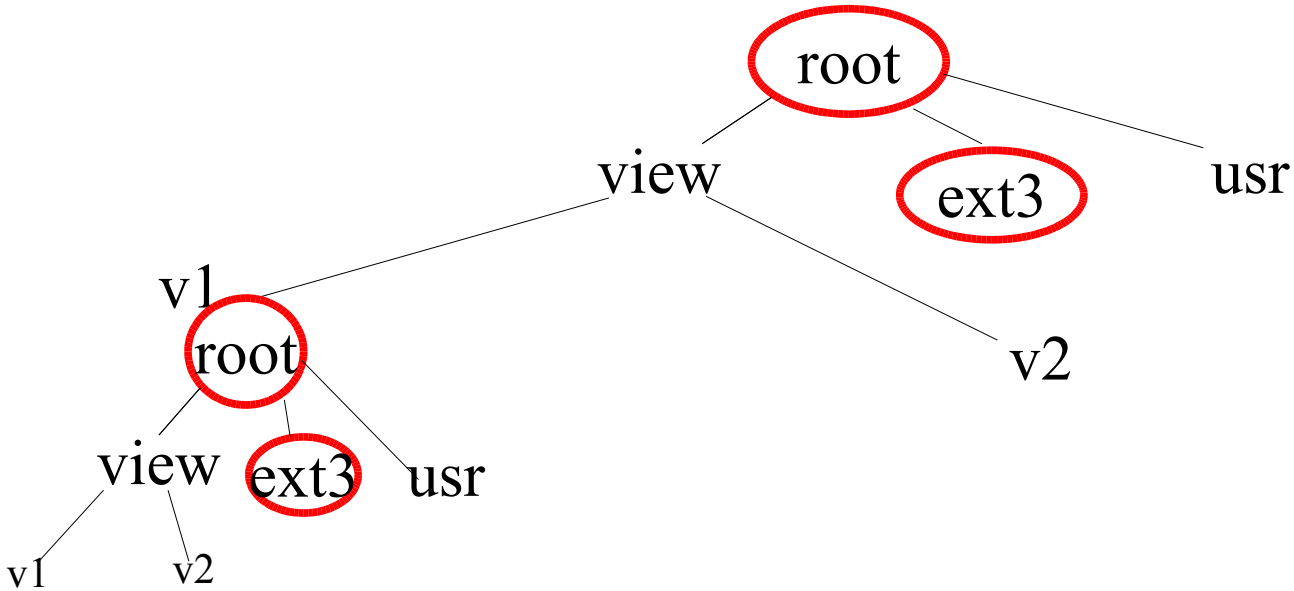
- shared mount
  - mount, unmount events propagate to each other
- peer group
  - group of shared mounts that propagate to each other



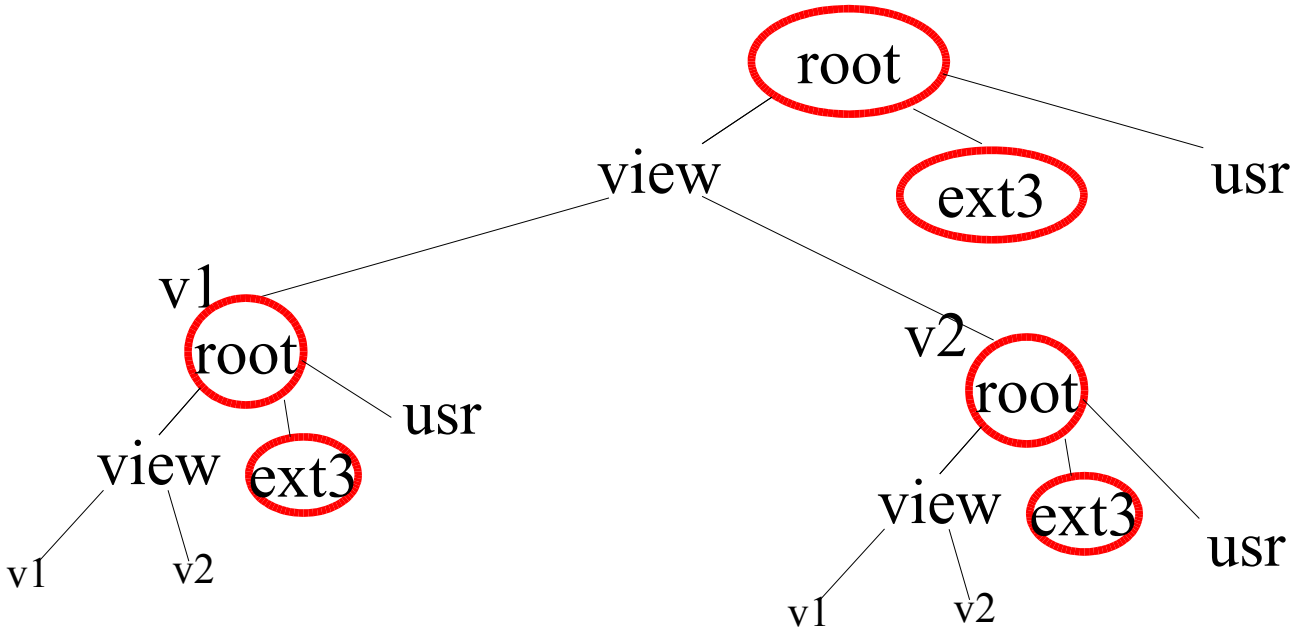
Shared mount application in MVFS



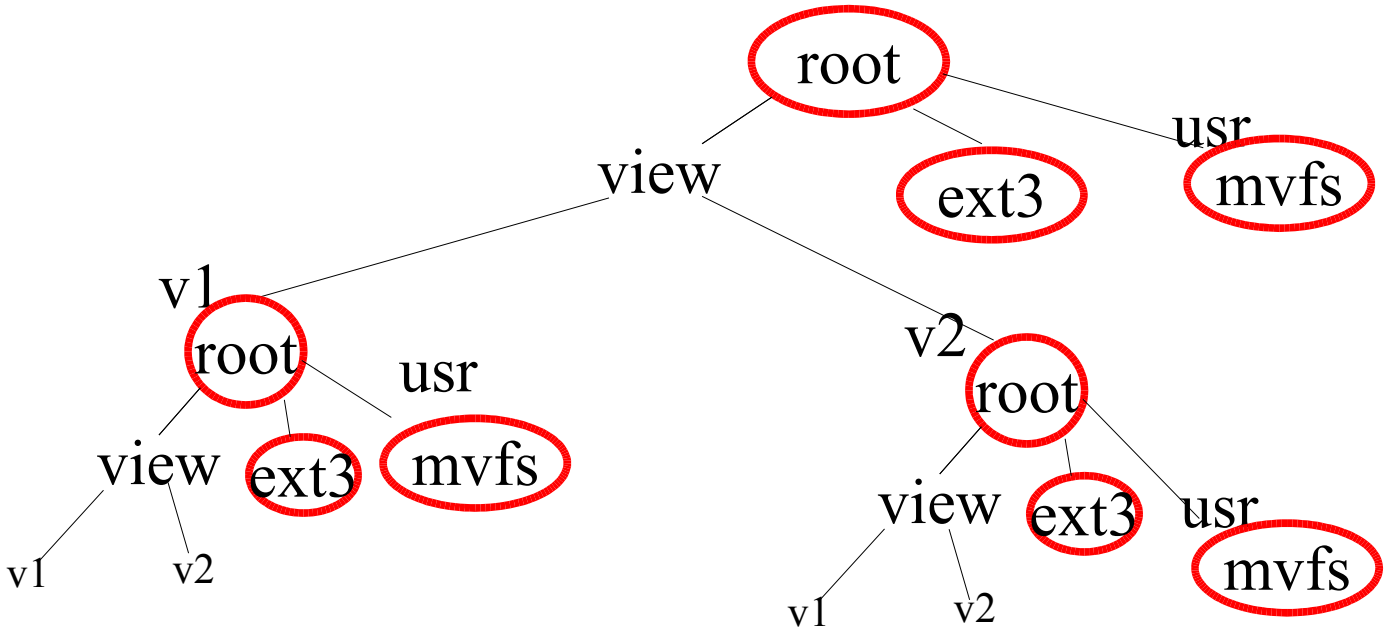
Shared mount application in MVFS



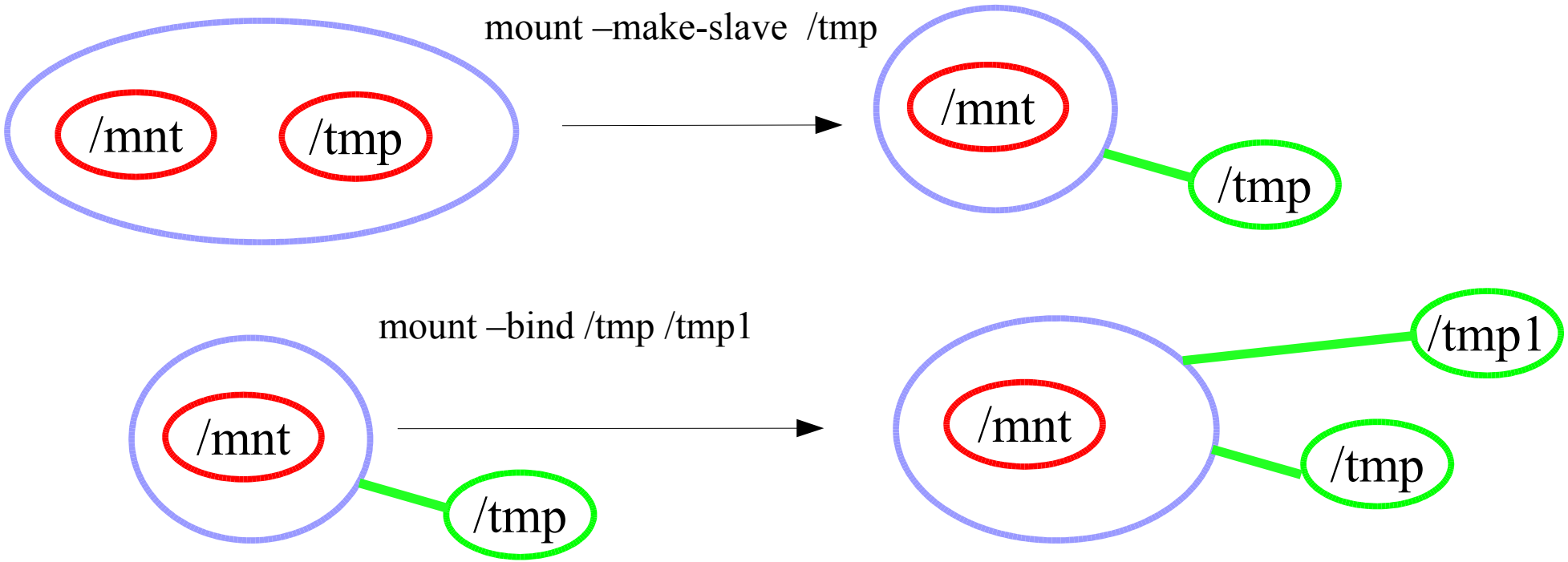
Shared mount application in MVFS



Shared mount application in MVFS

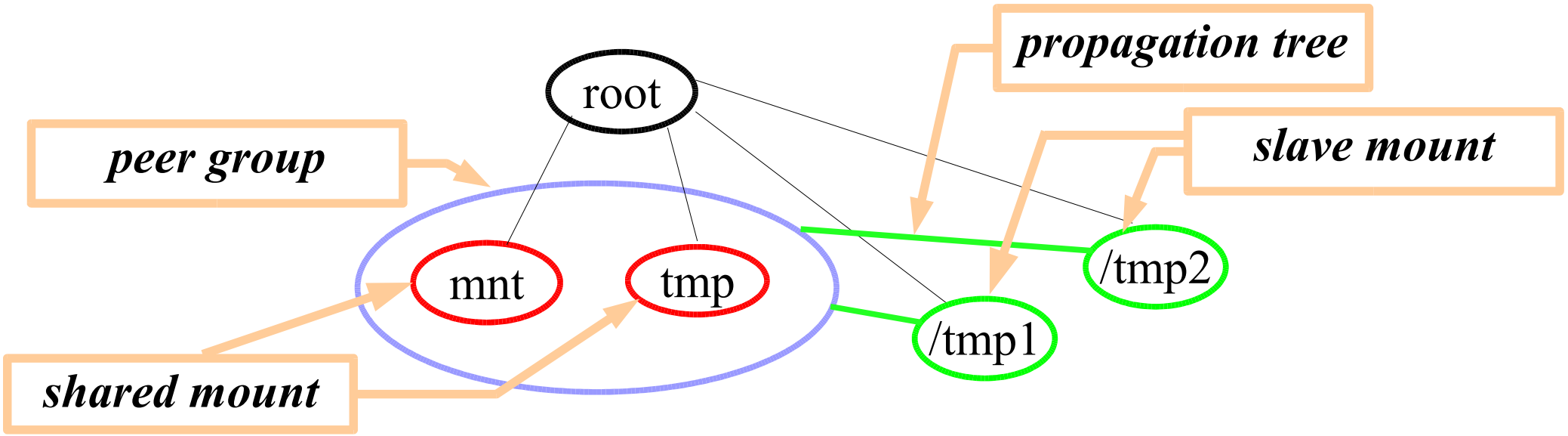


Slave mount


mount –make-slave /tmp
/mnt
/tmp
/mnt
/tmp
mount –bind /tmp /tmp1
/mnt
/tmp
/mnt
/tmp1
/tmp

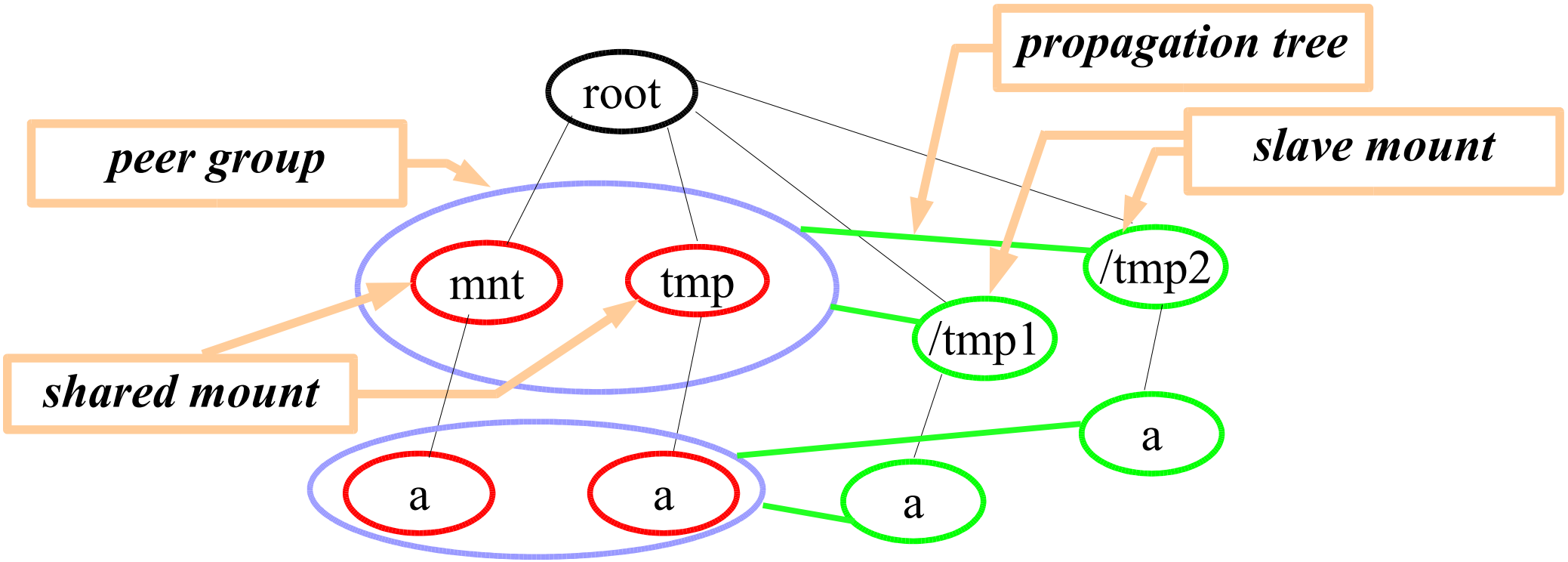
Slave mount

- slave mount
  - mount,unmount events propagate towards it from master not vice-versa.
- propagation tree
  - dictates the flow of mount and unmount events.



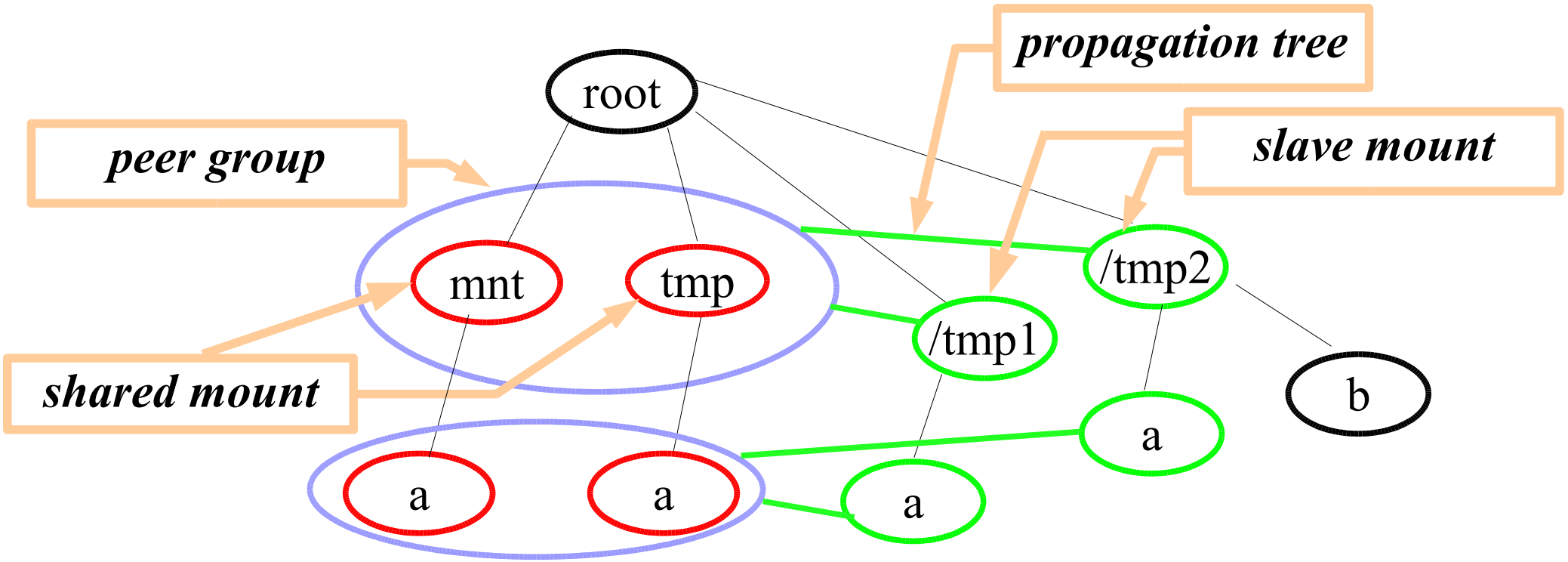
Slave mount

- slave mount.
  - mount,unmount events propagate towards it from master not vice-versa.
- propagation tree.
  - dictates the flow of mount and unmount events.

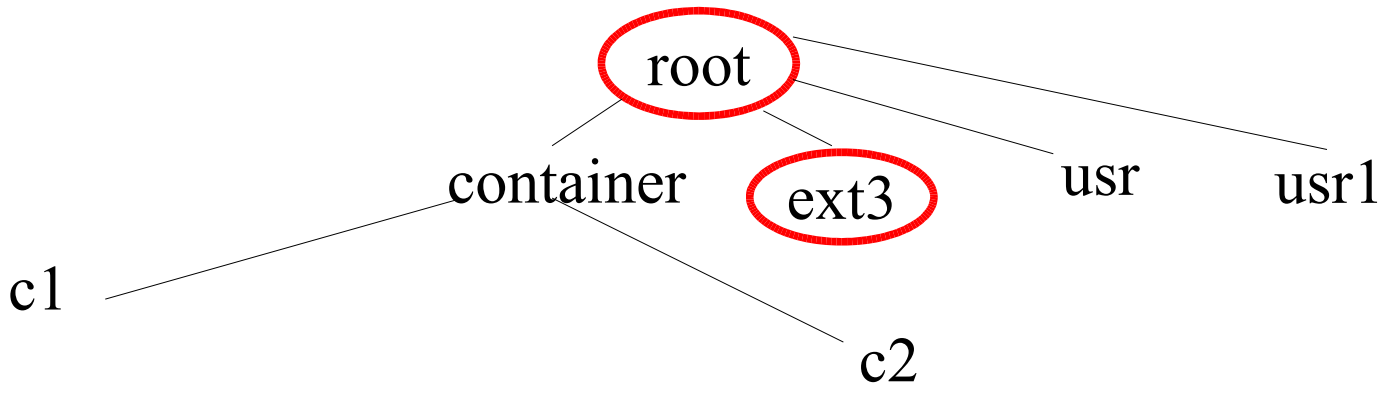


Slave mount

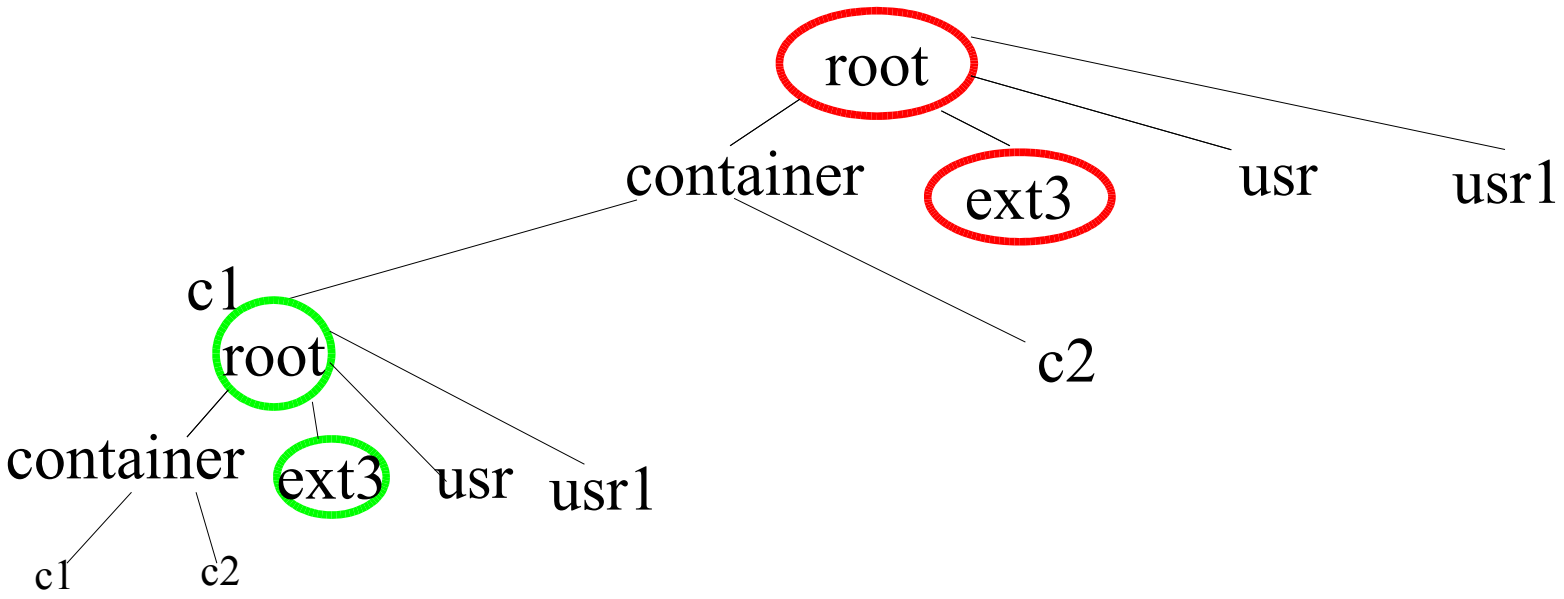
- slave mount.
  - mount,unmount events propagate towards it from master not vice-versa.
- propagation tree.
  - dictates the flow of mount and unmount events.



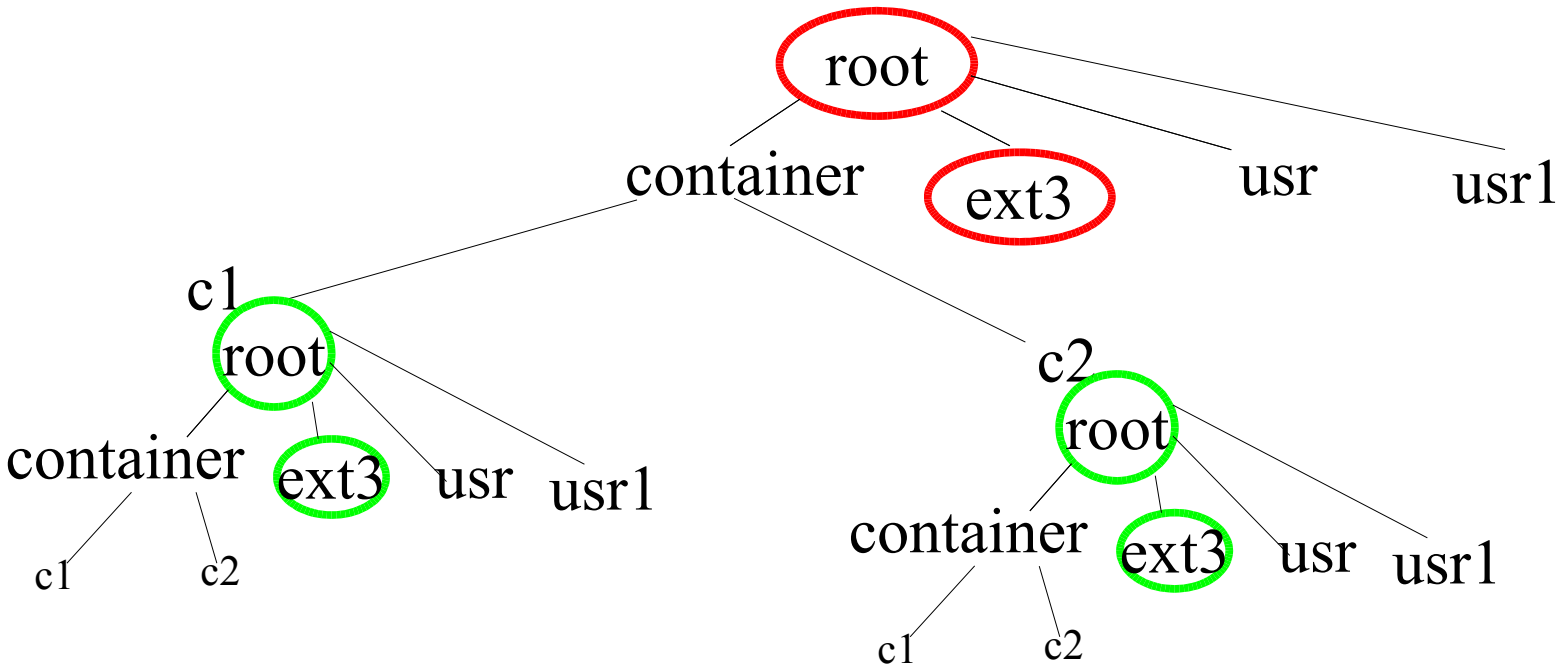
Slave mount application in Containers.



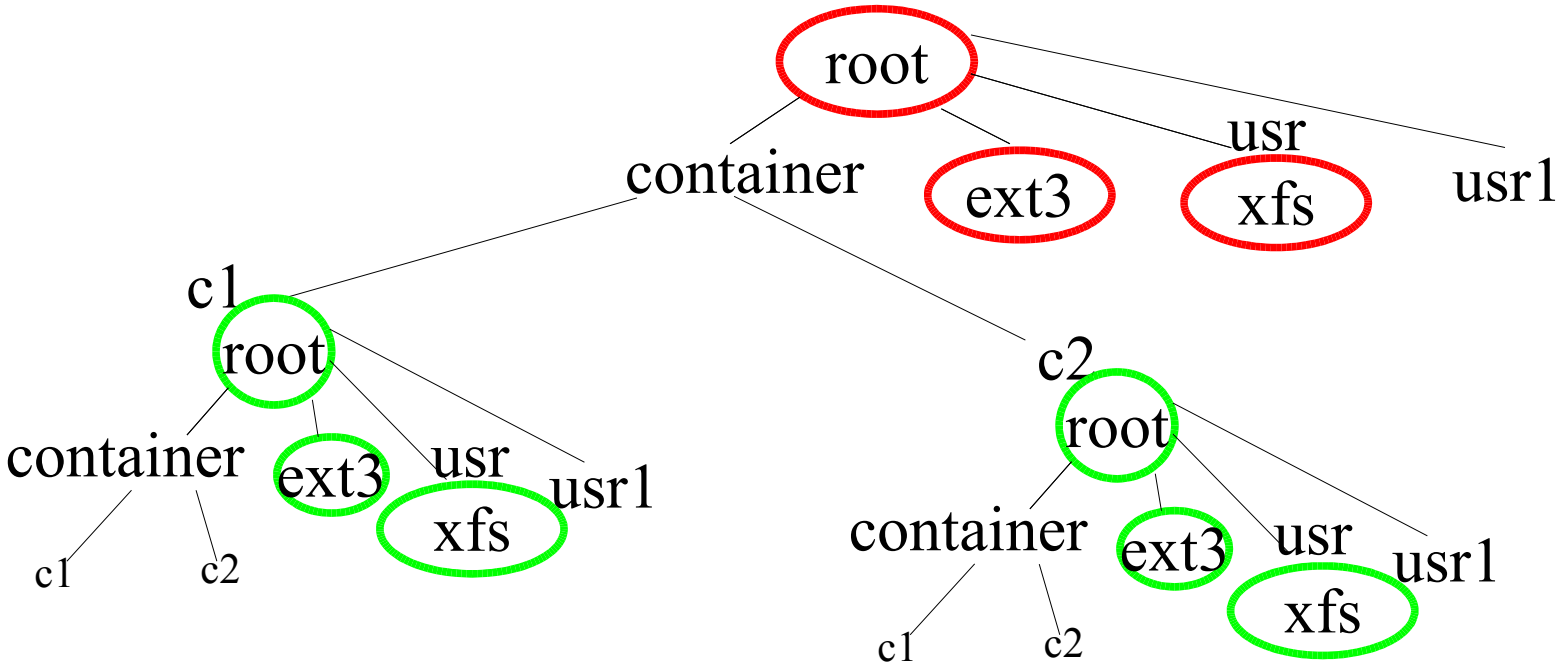
Slave mount application in Containers.



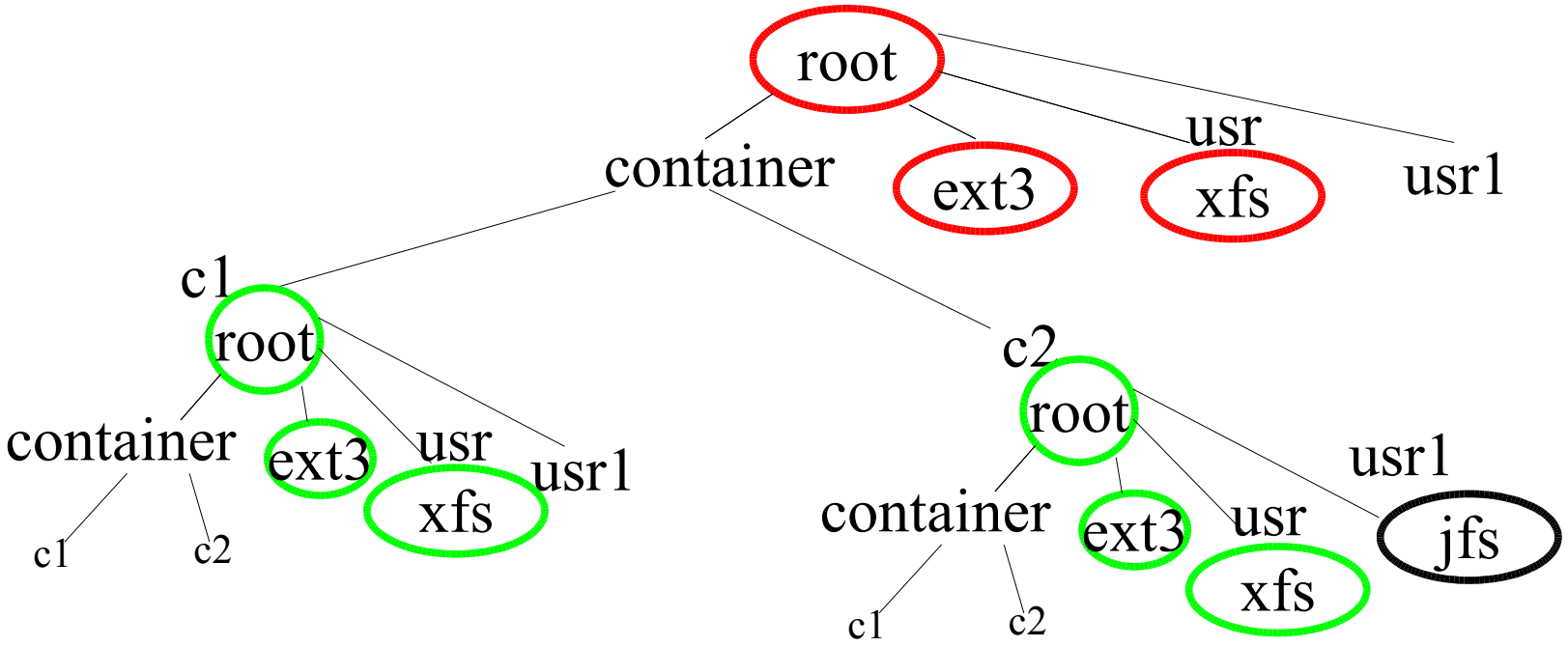
Slave mount application in Containers.



Slave mount application in Containers.

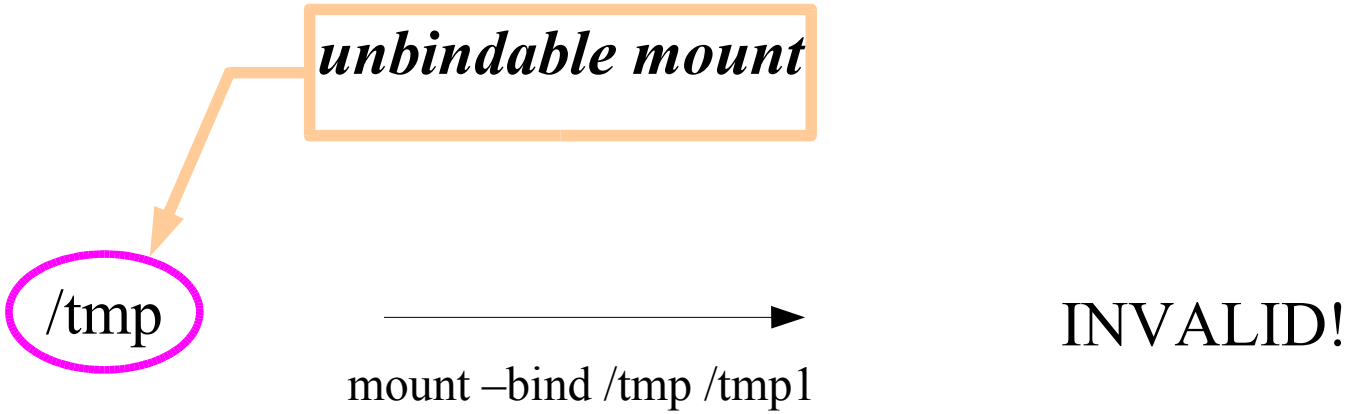


Slave mount application in Containers.

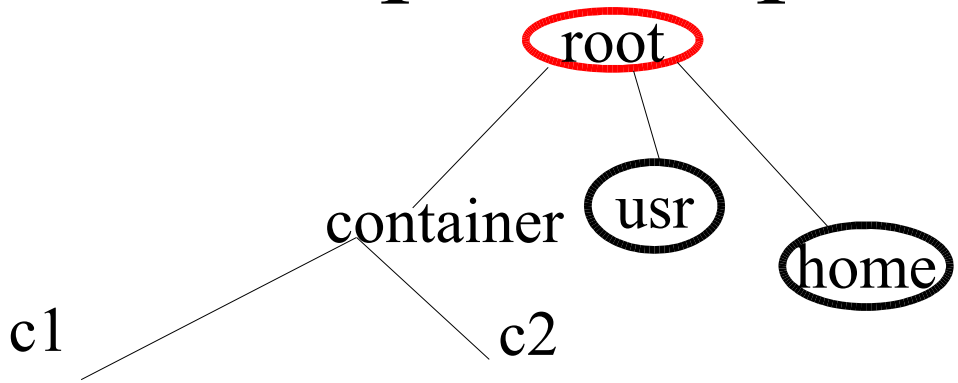


Unbindable mount

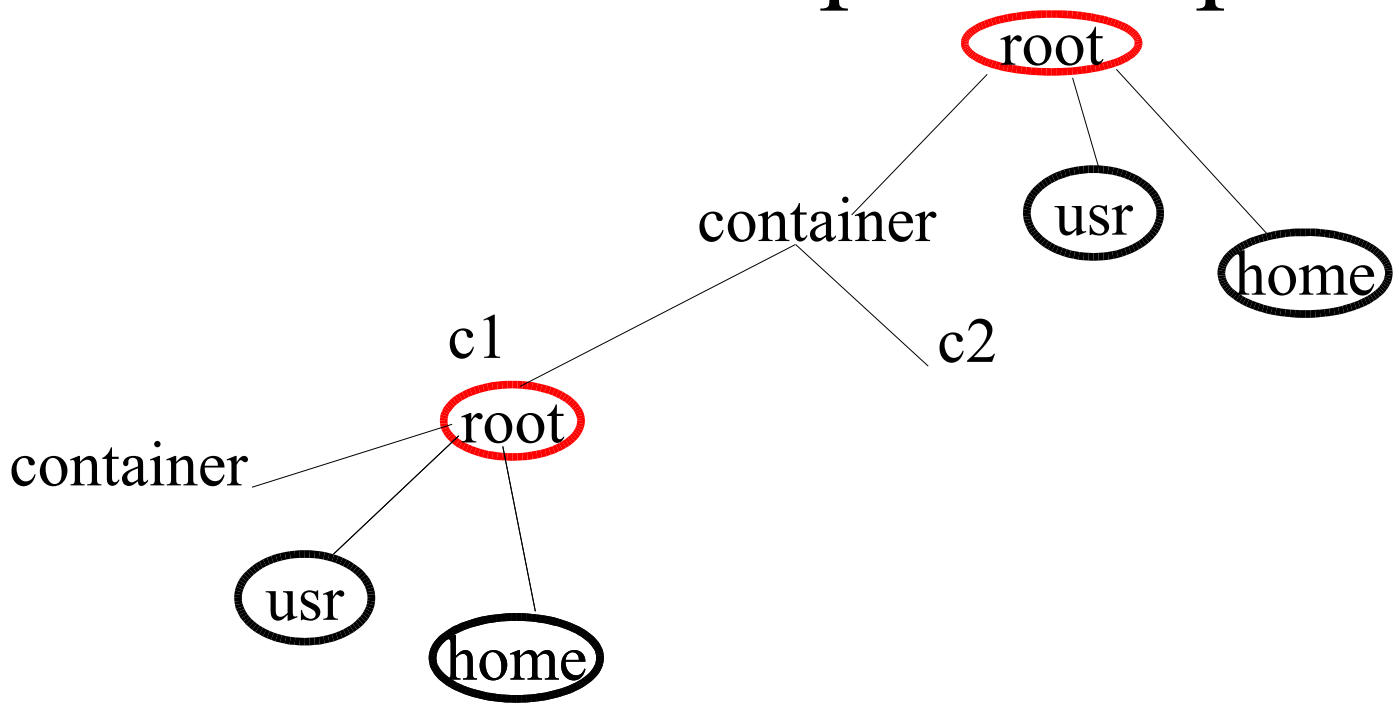
- unbindable mounts
  - no propagation.
  - not bindable.



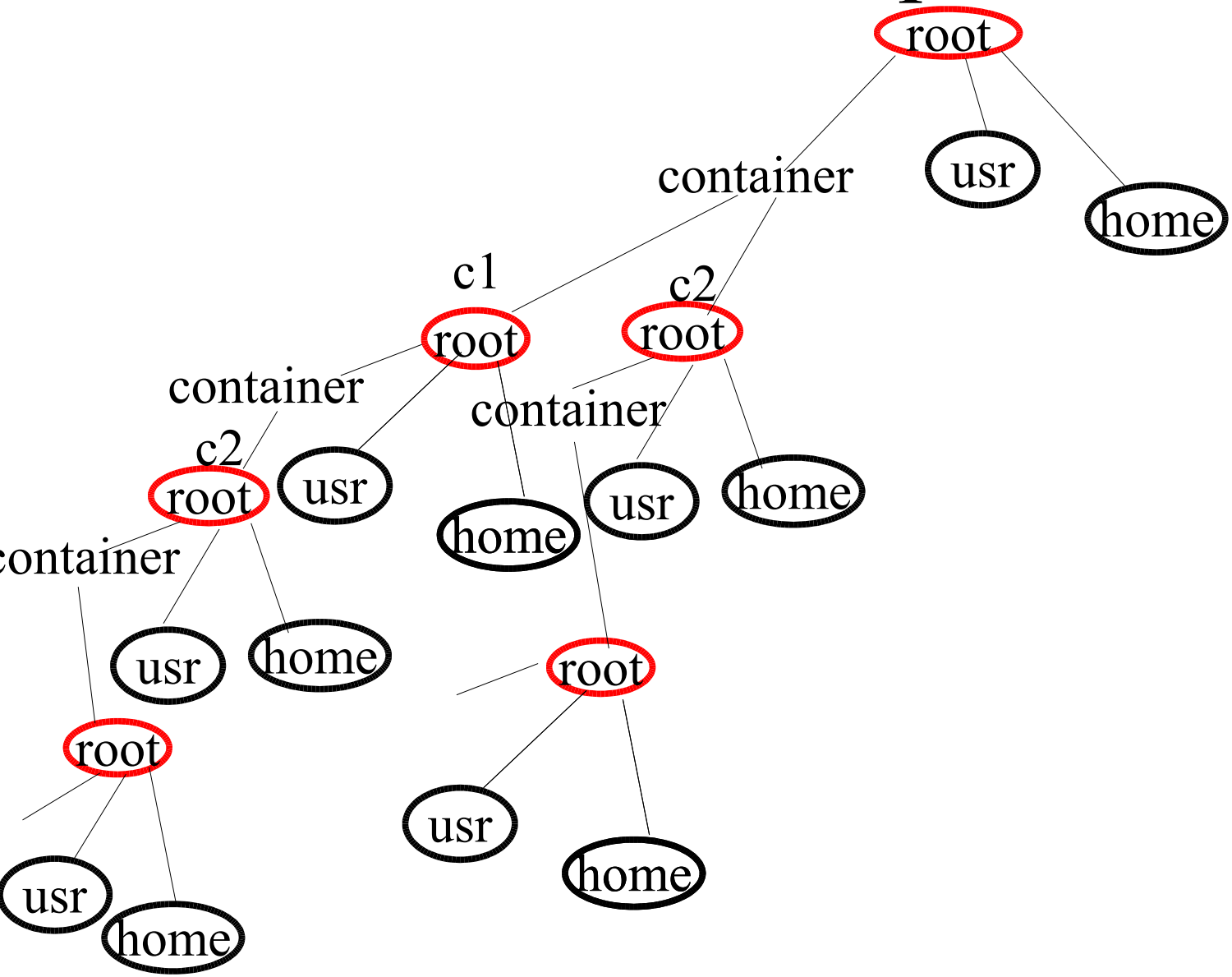
Mount explosion problem



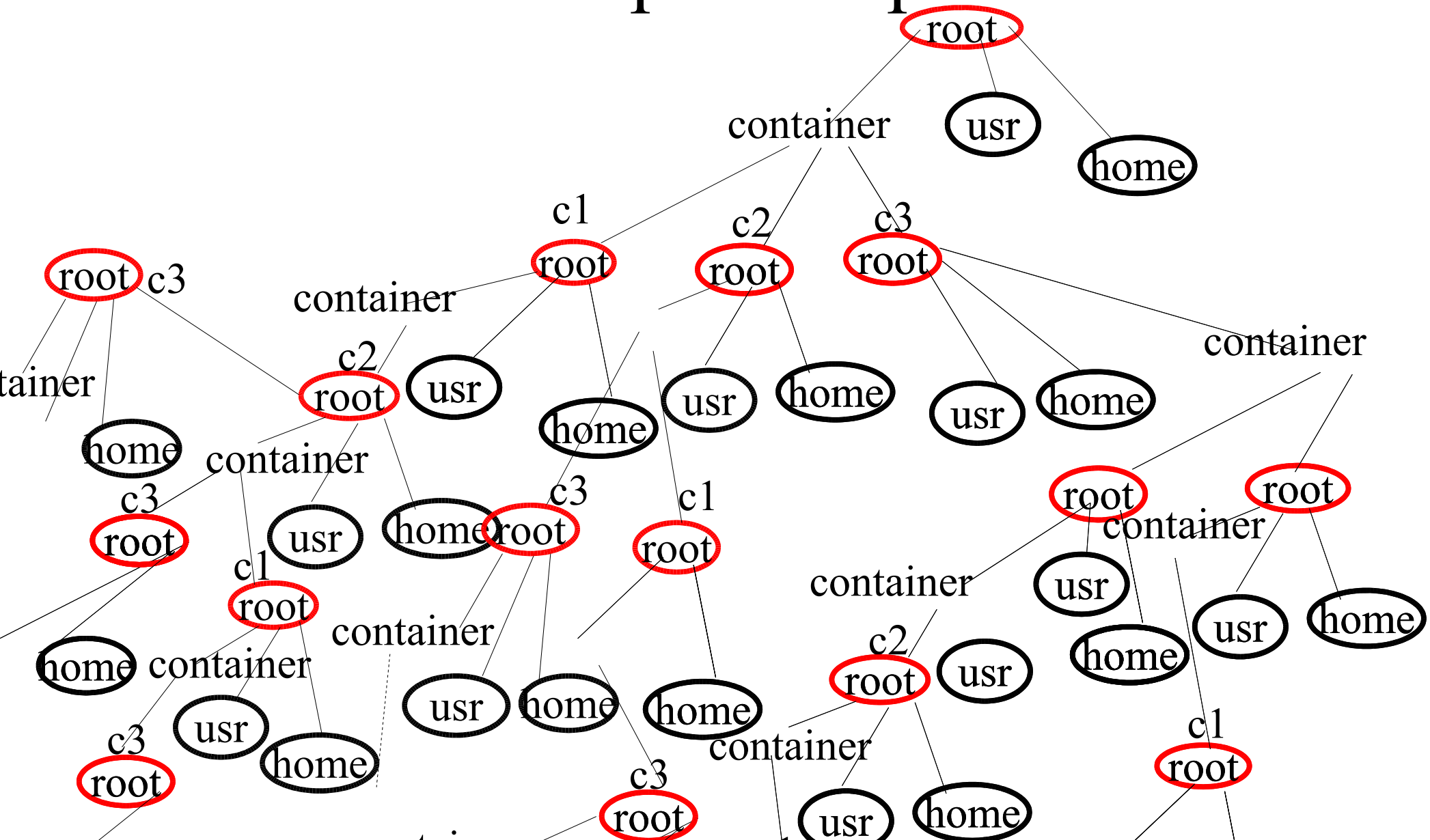
Mount explosion problem



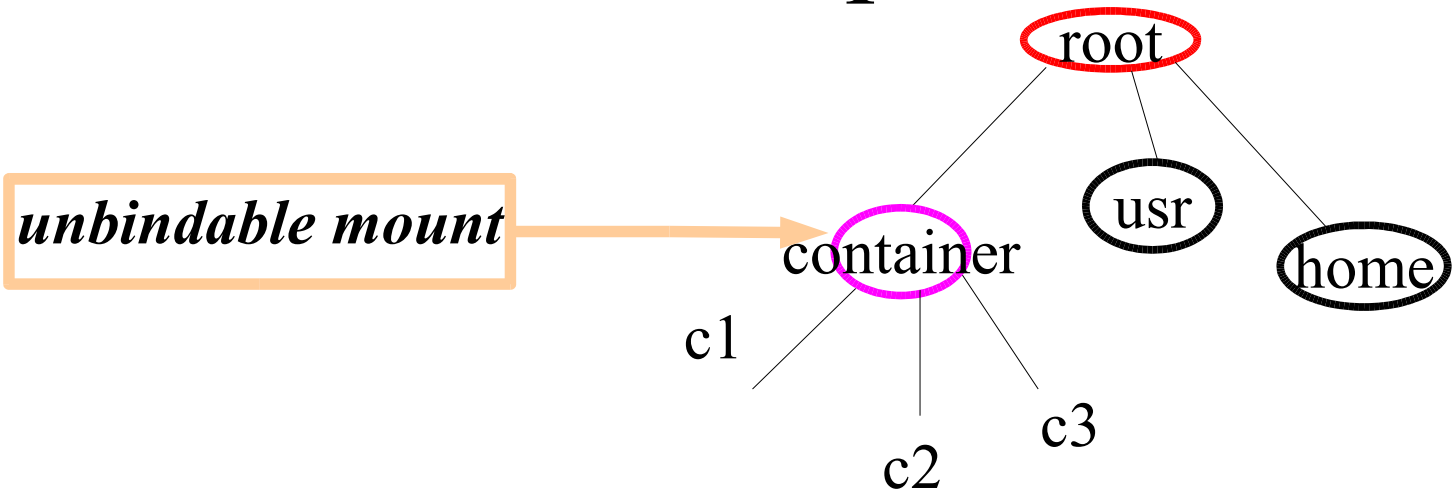
Mount explosion!!



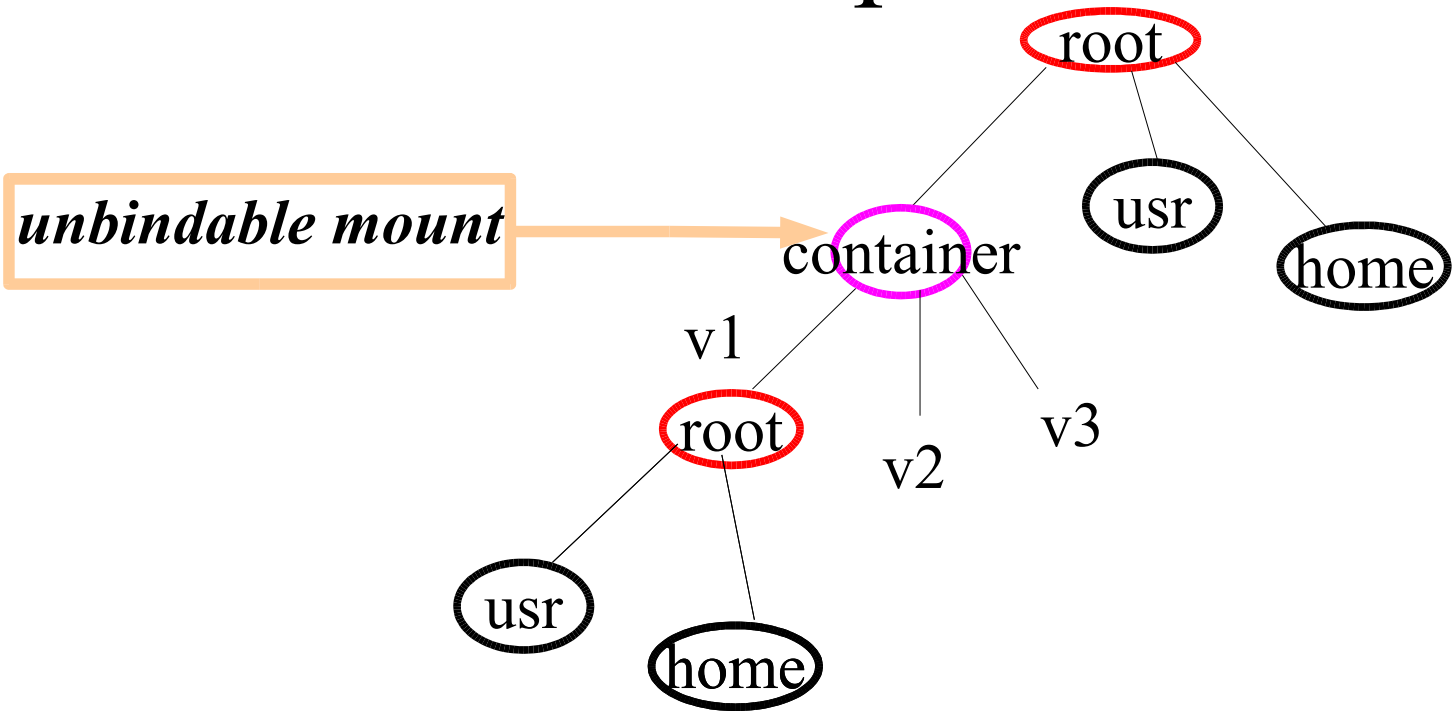
Mount explosion problem



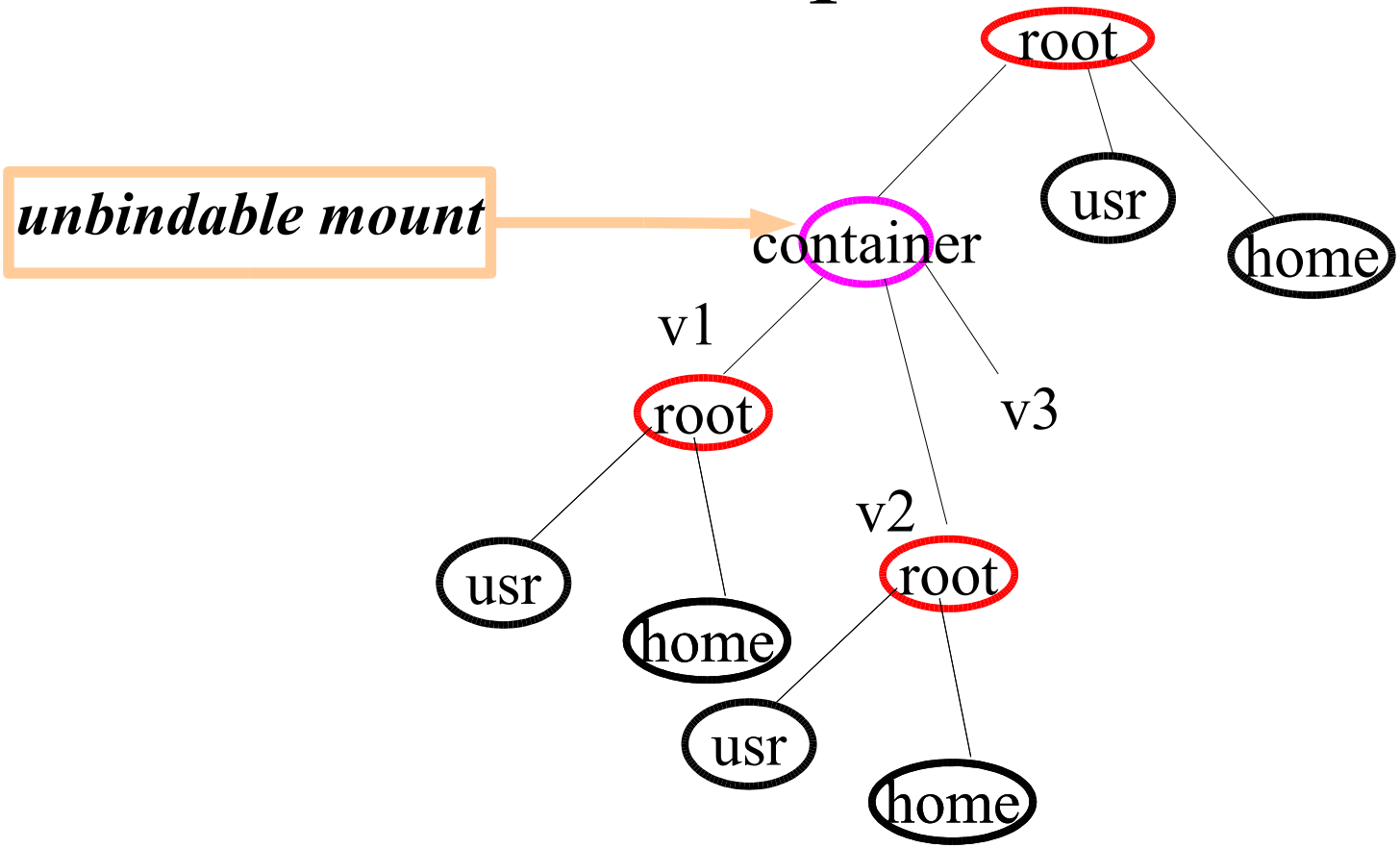
Mount explosion solution!!



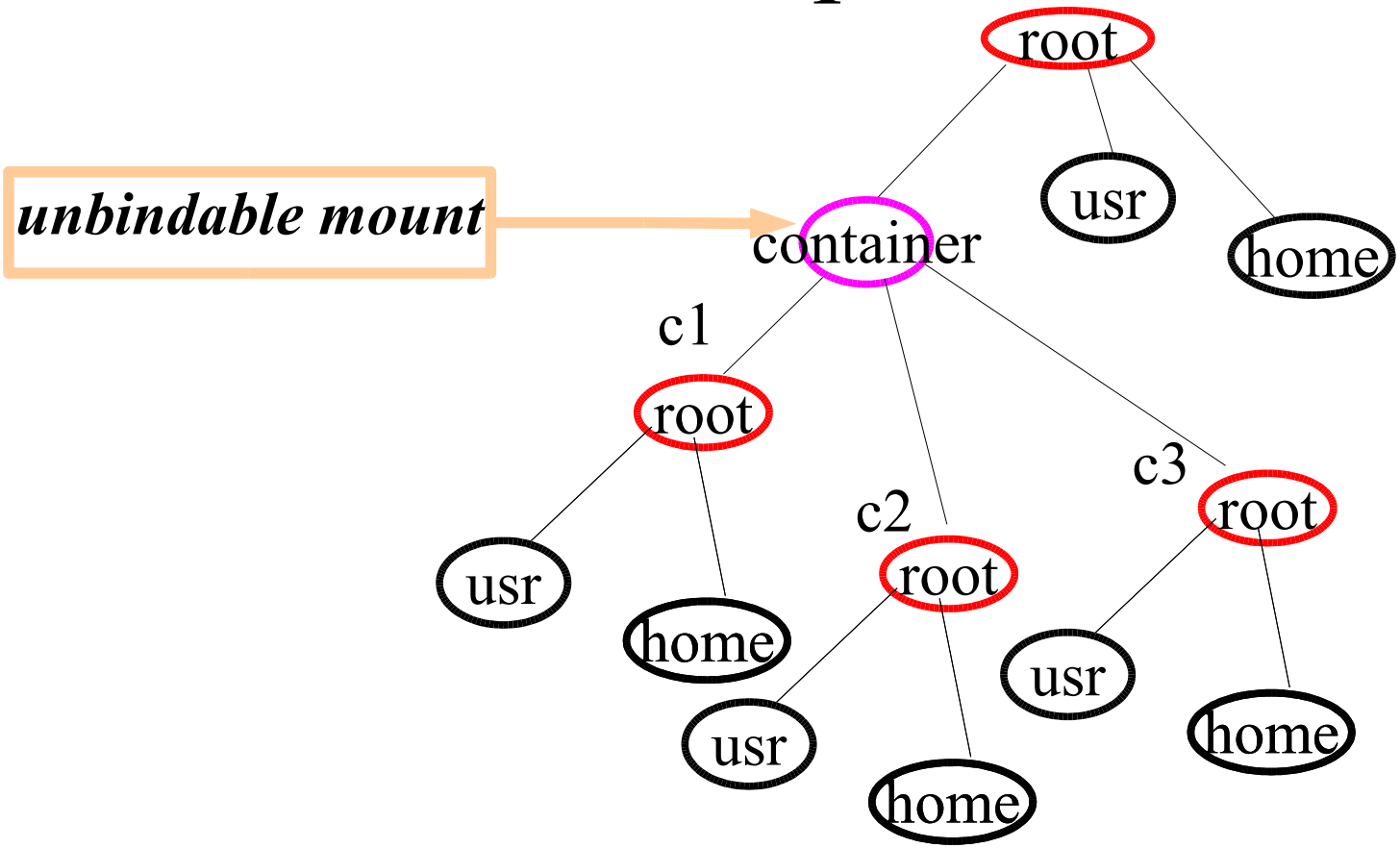
Mount explosion solution!!



Mount explosion solution!!

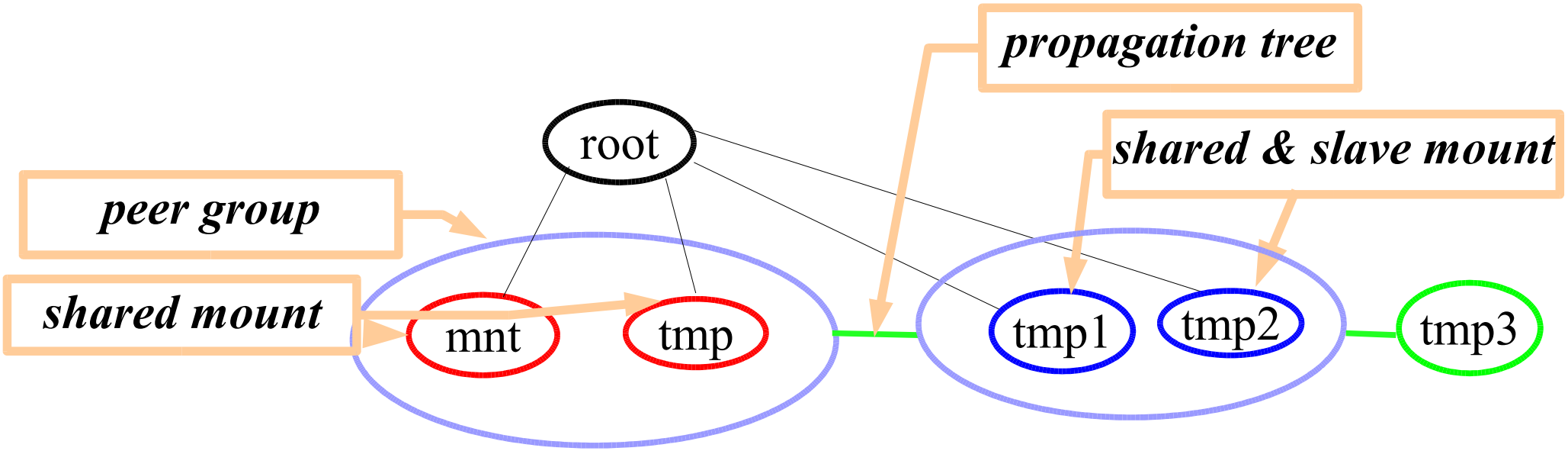


Mount explosion solution!!



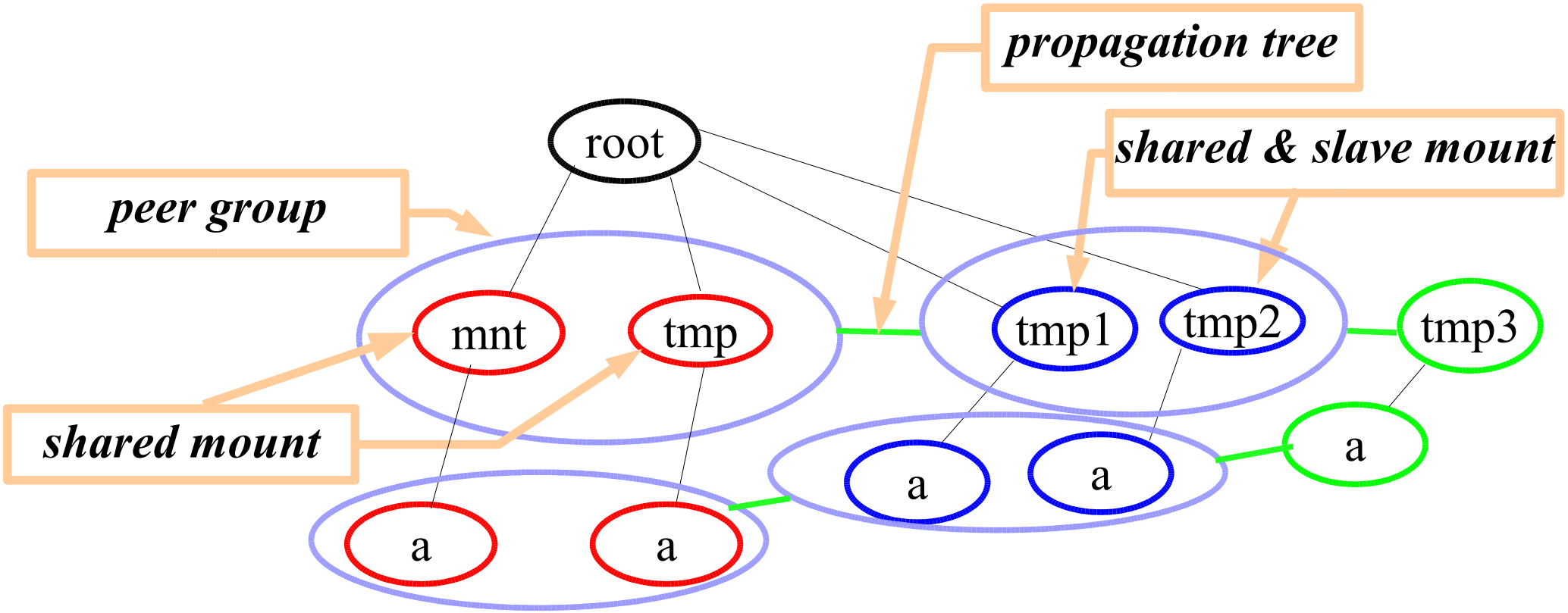
Shared and Slave mount

- shared and slave mount
  - mount,unmount events propagate towards it from master and it propagates them to its slaves



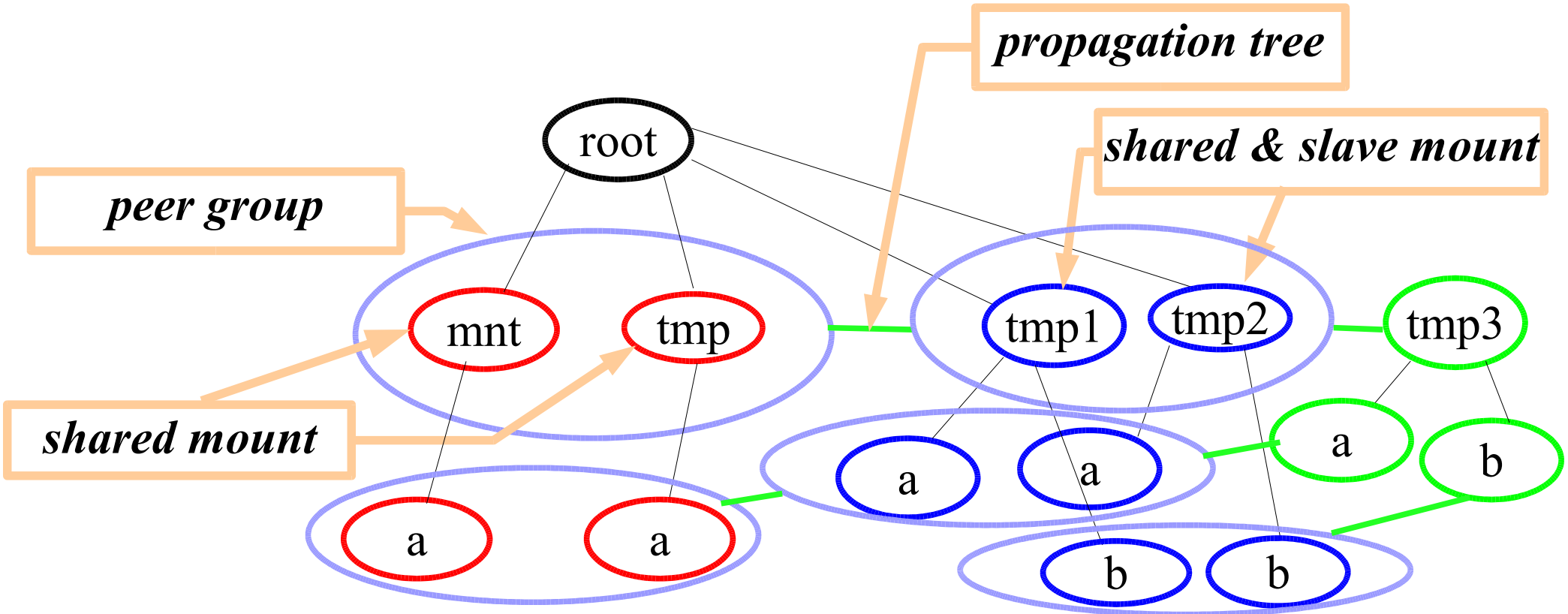
Shared and Slave mount

- shared and slave mount
  - mount,unmount events propagate towards it from master and it propagates them to its slaves

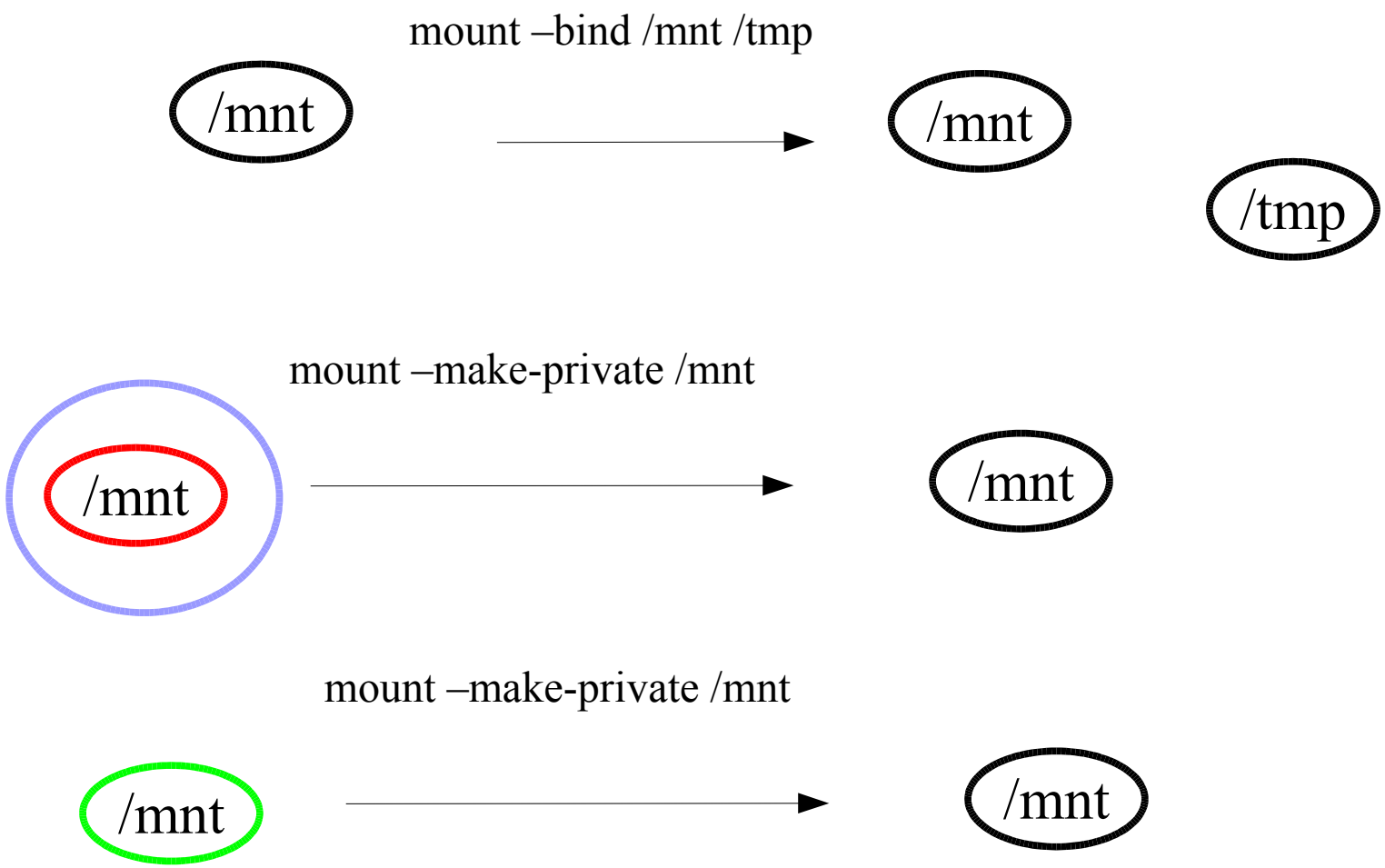


Shared and Slave mount

- shared and slave mount
  - mount,unmount events propagate towards it from master and it propagates them to its slaves

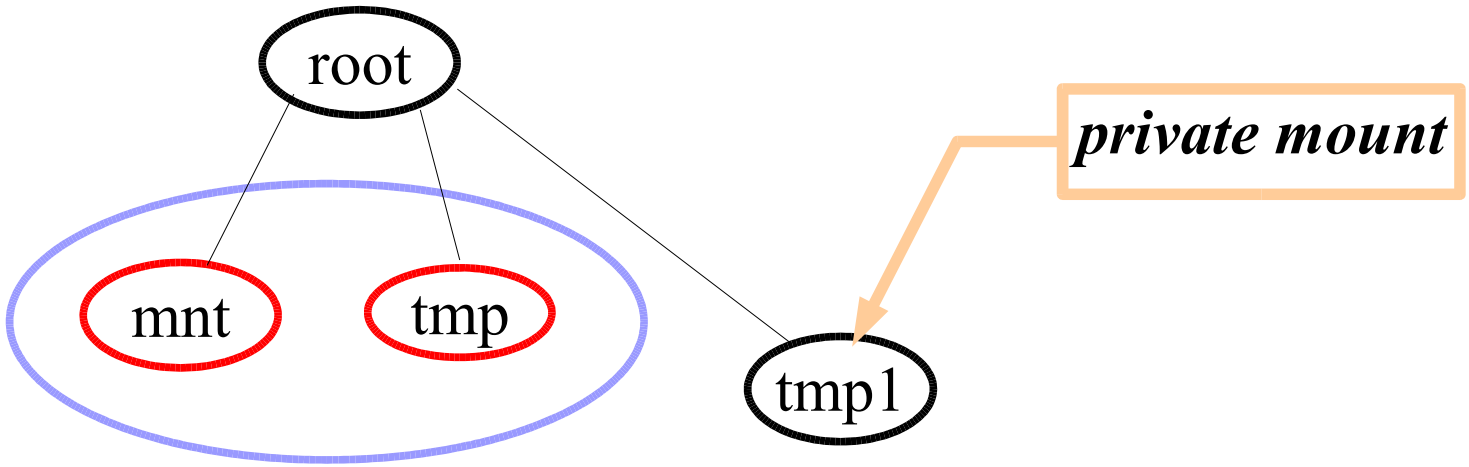


Private mount



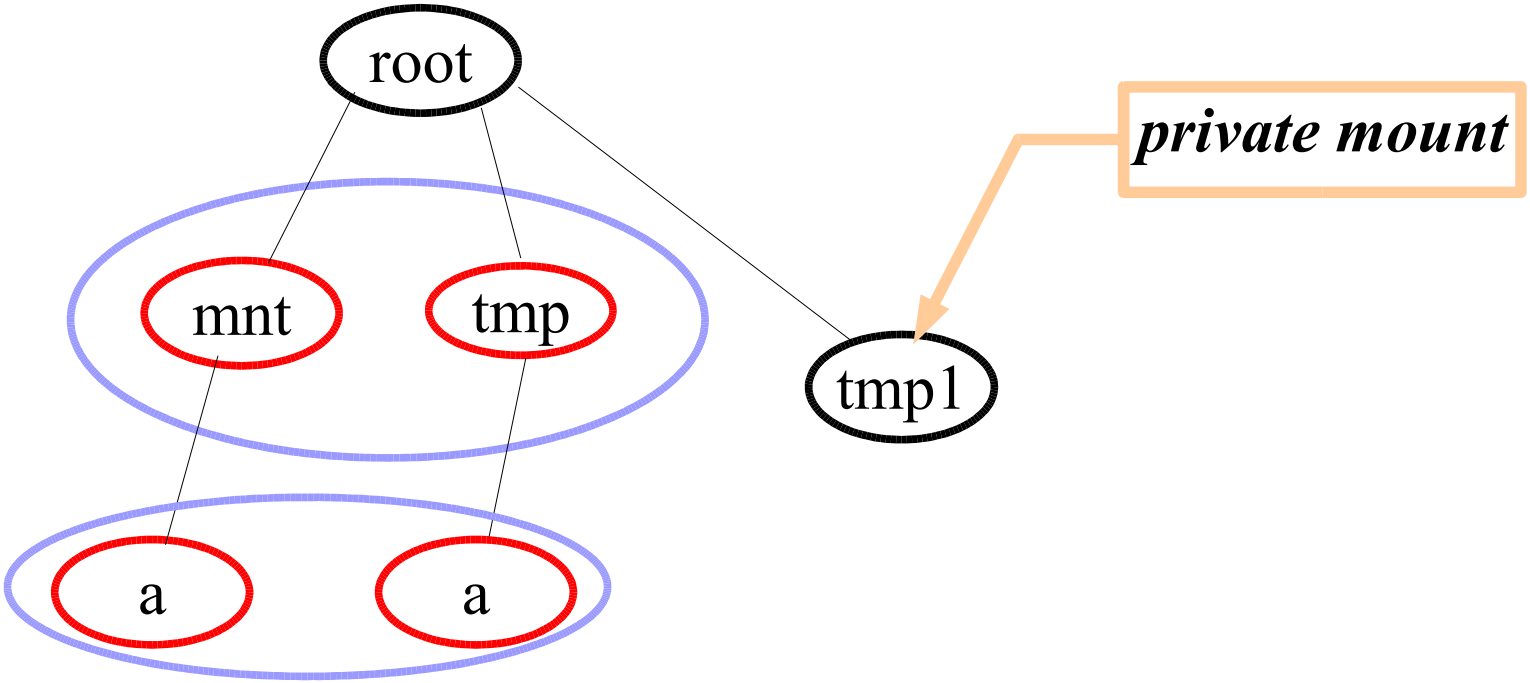
Private mount

- private mounts.
  - no propagation.
  - mounts by default are private.



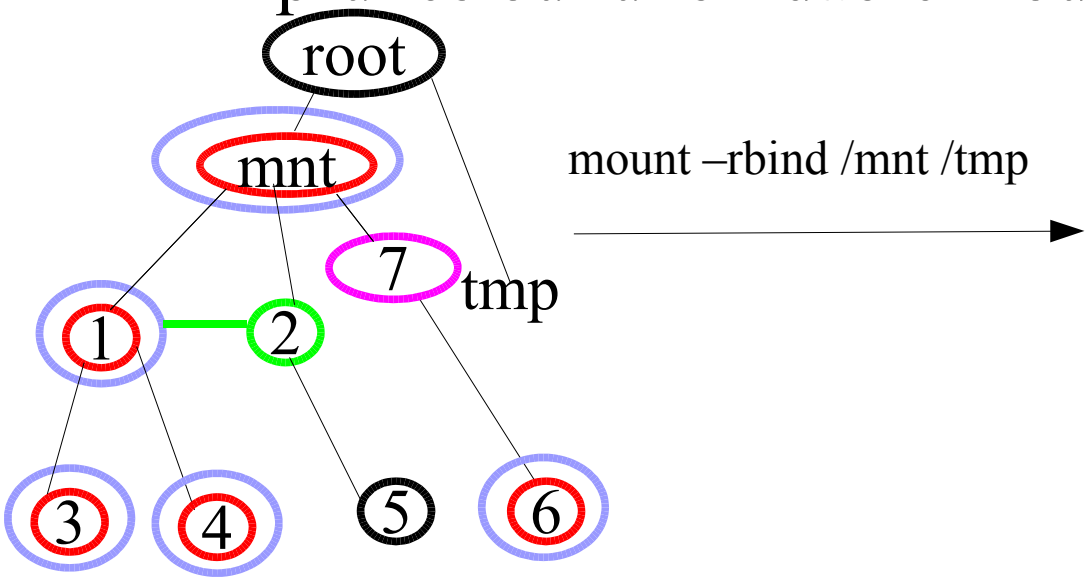
Private mount

- private mounts.
  - no propagation.
  - mounts by default are private.



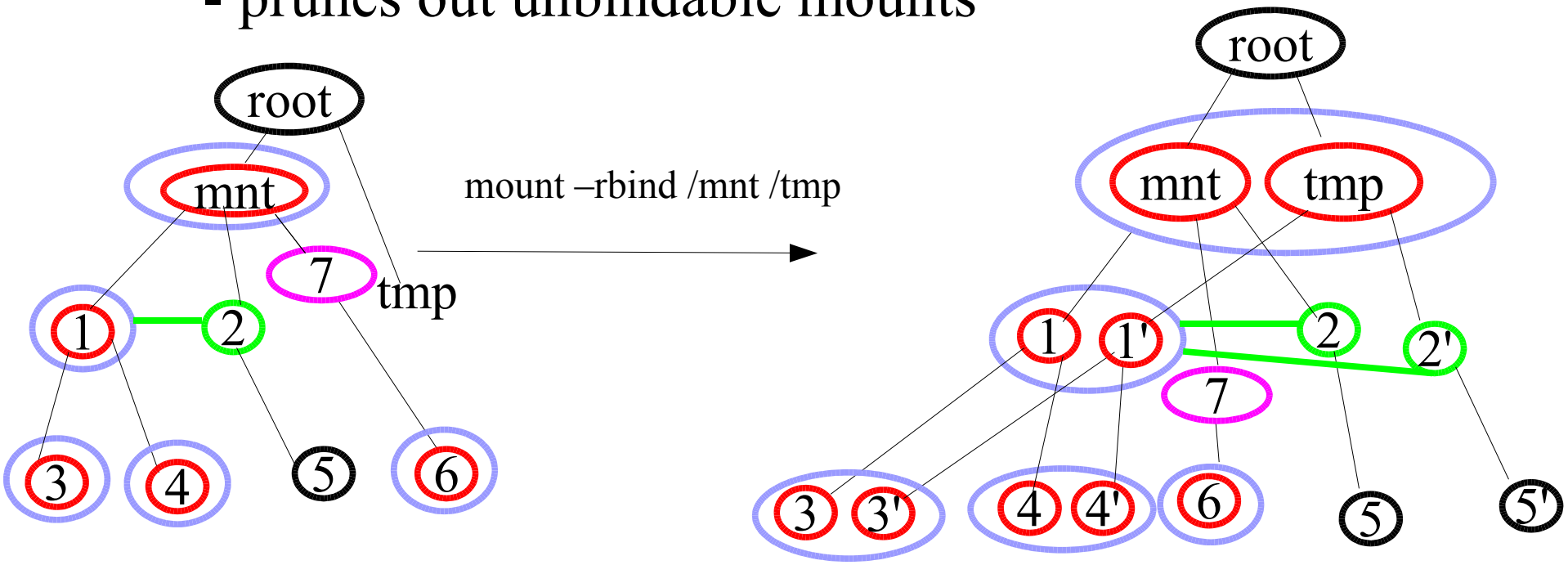
Rbind

- rbind
  - applies the bind-mount rules for each mount in the mount-tree
  - prunes out unbindable mounts



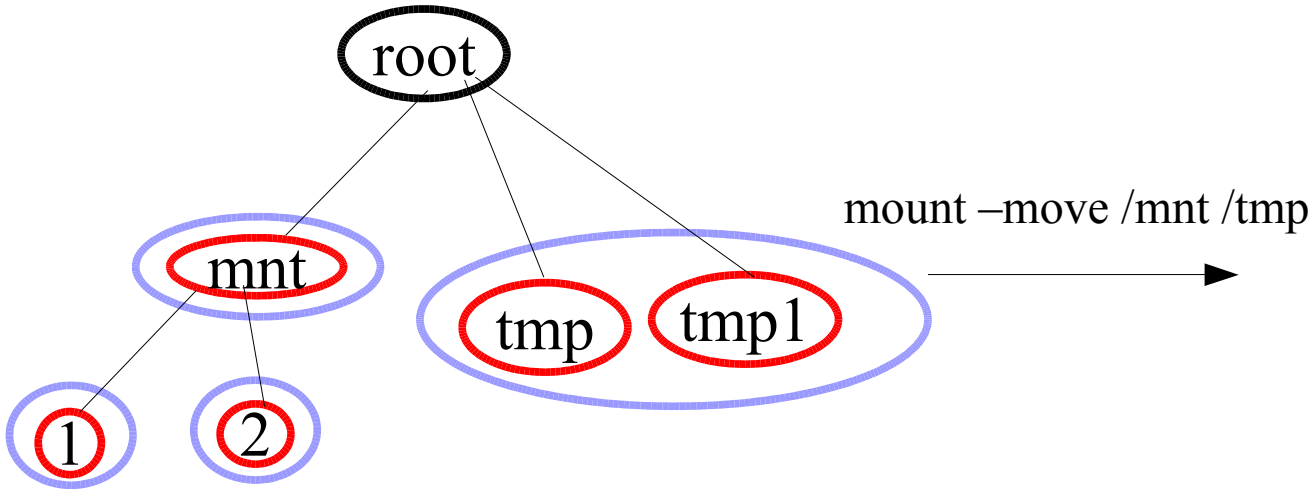
Rbind

- rbind
  - applies the bind-mount rules for each mount in the mount-tree
  - prunes out unbindable mounts



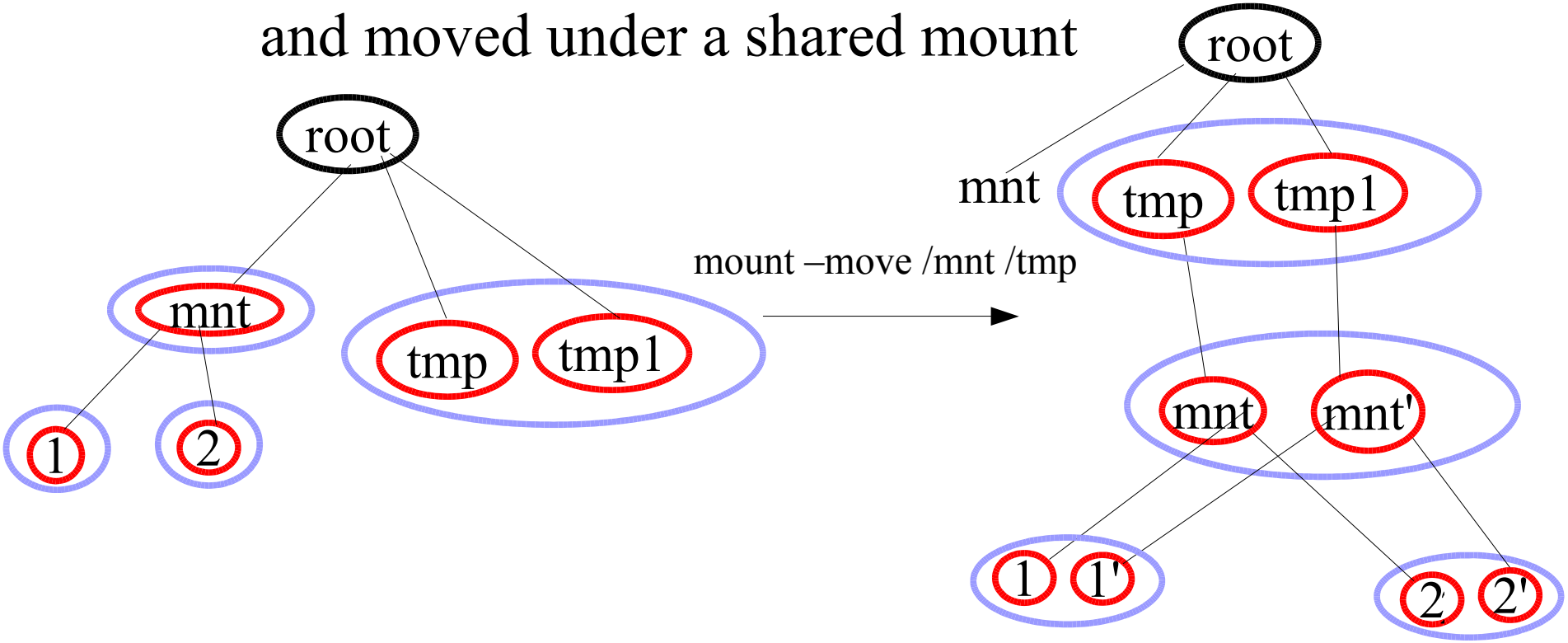
Move

- move
  - invalid if parent is a shared mount.
  - invalid if the tree contains unbindable mount and moved under a shared mount*



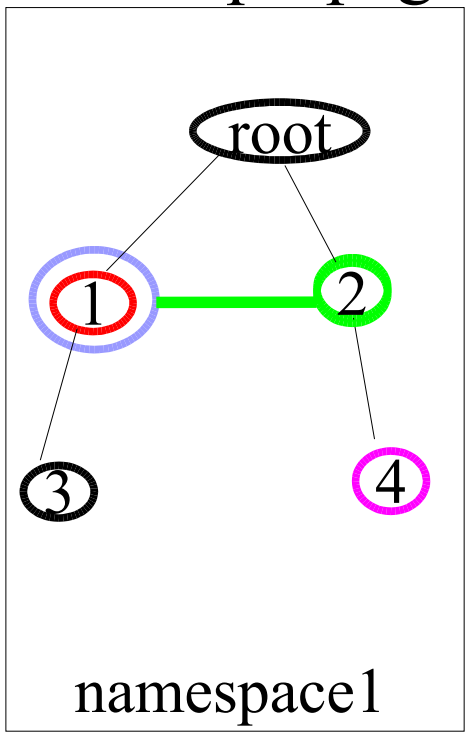
Move

- move
  - invalid if parent is a shared mount
  - invalid if the tree contains unbindable mount and moved under a shared mount



Clone namespace

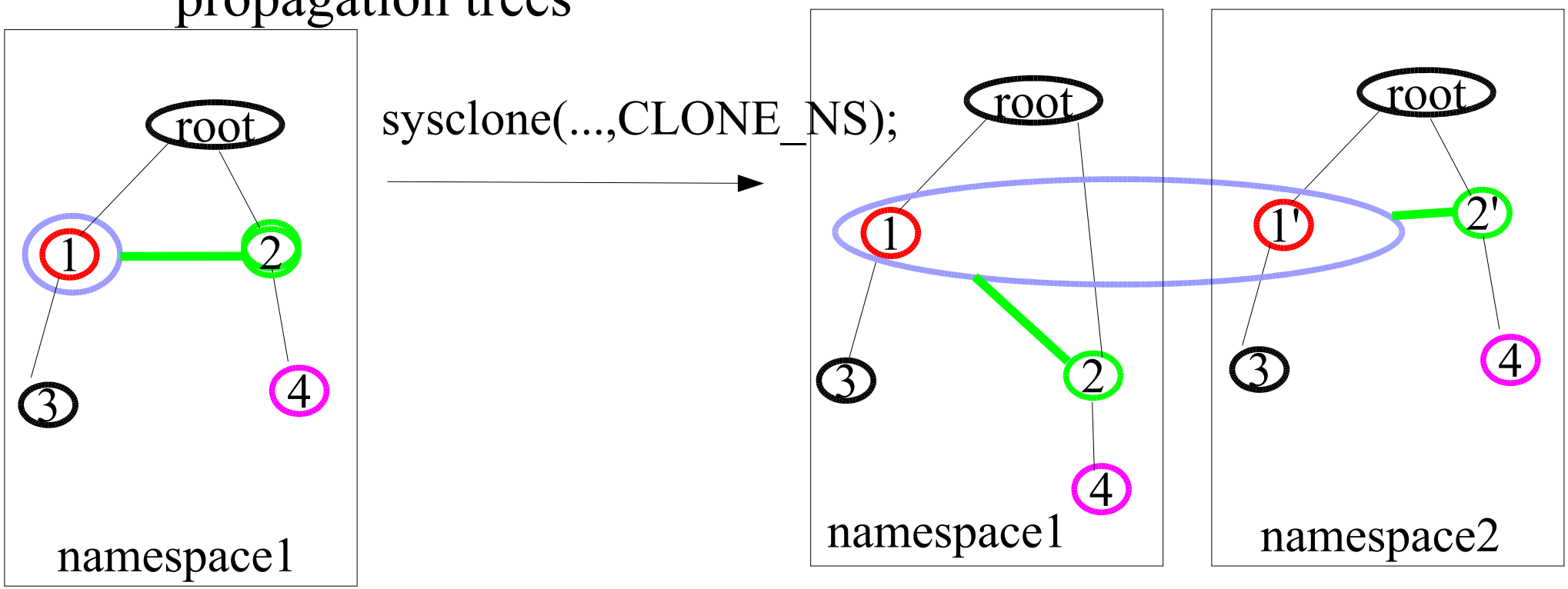
- clone namespace
  - Clones all the mounts including unbindable mounts
  - adds the new shared and slave mounts in their respective propagation trees



sysclone(...,CLONE_NS);

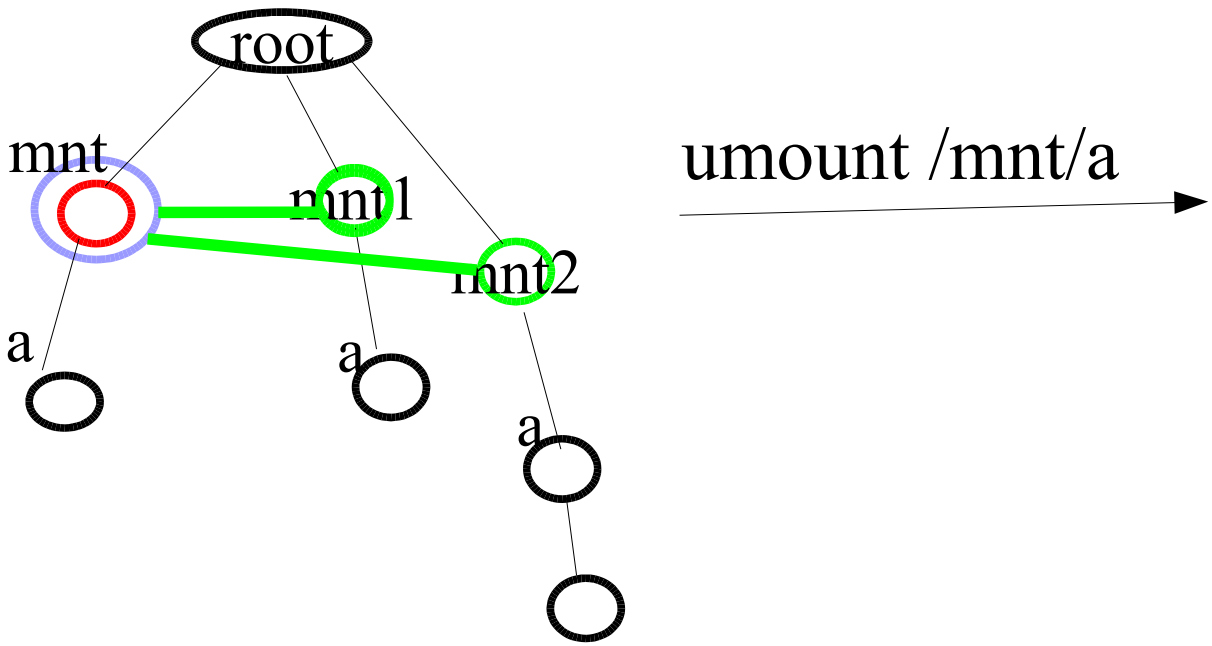
Clone namespace

- clone namespace
  - Clones all the mounts including unbindable mounts
  - adds the new shared and slave mounts in their respective propagation trees



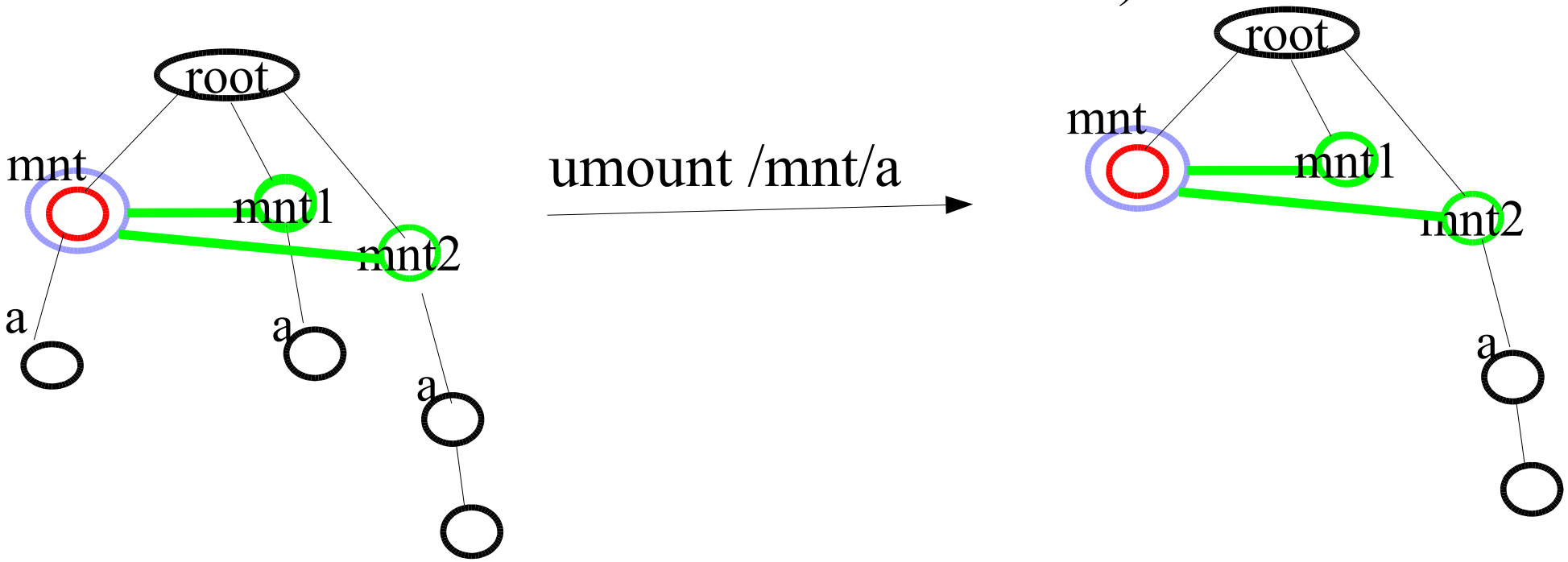
Umount

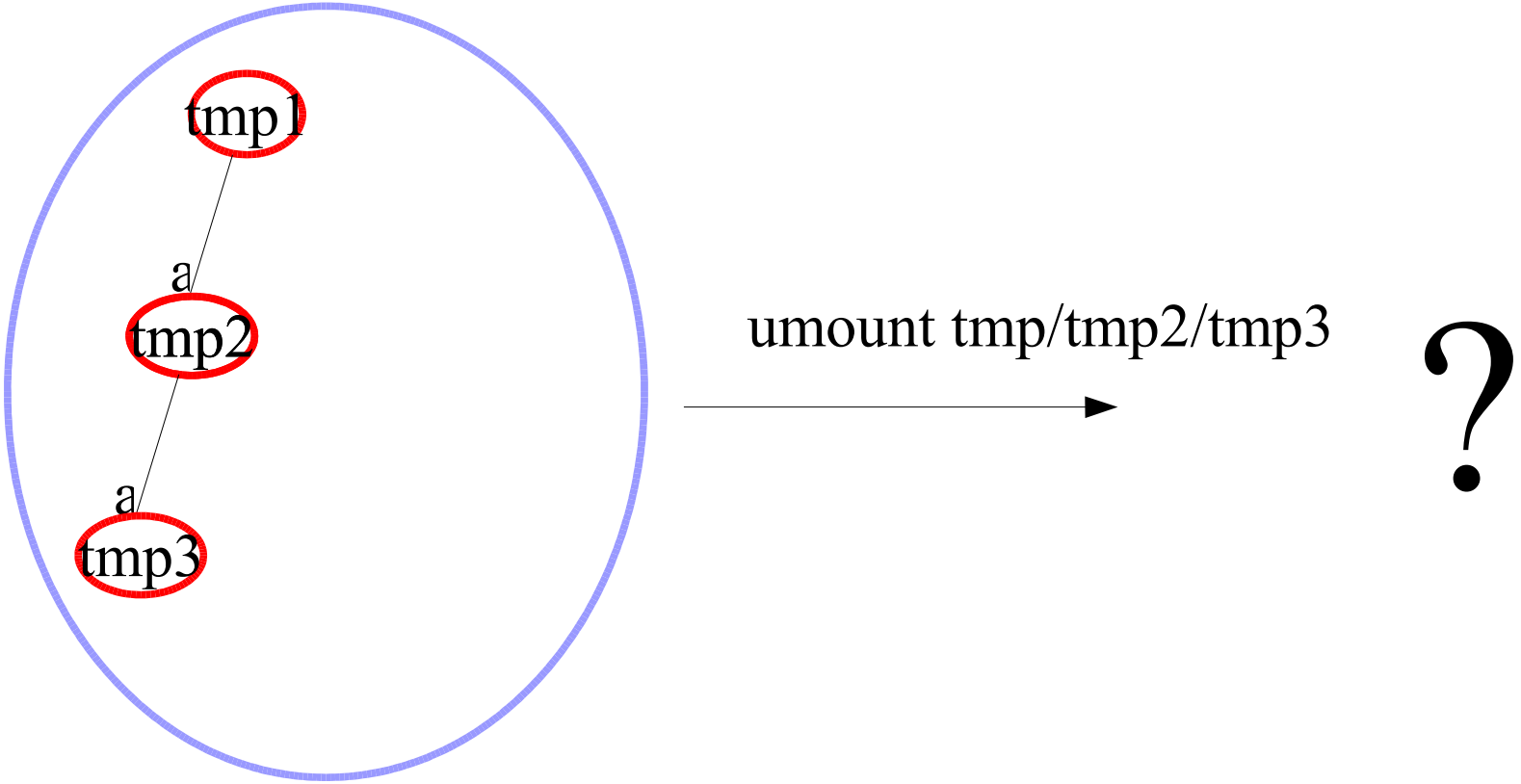
- umount
  - fail if the mount has submounts
  - unmount all child mounts on the mounts belonging to the parent's propagation tree (only if the child mounts do not have children mounts)



Umount

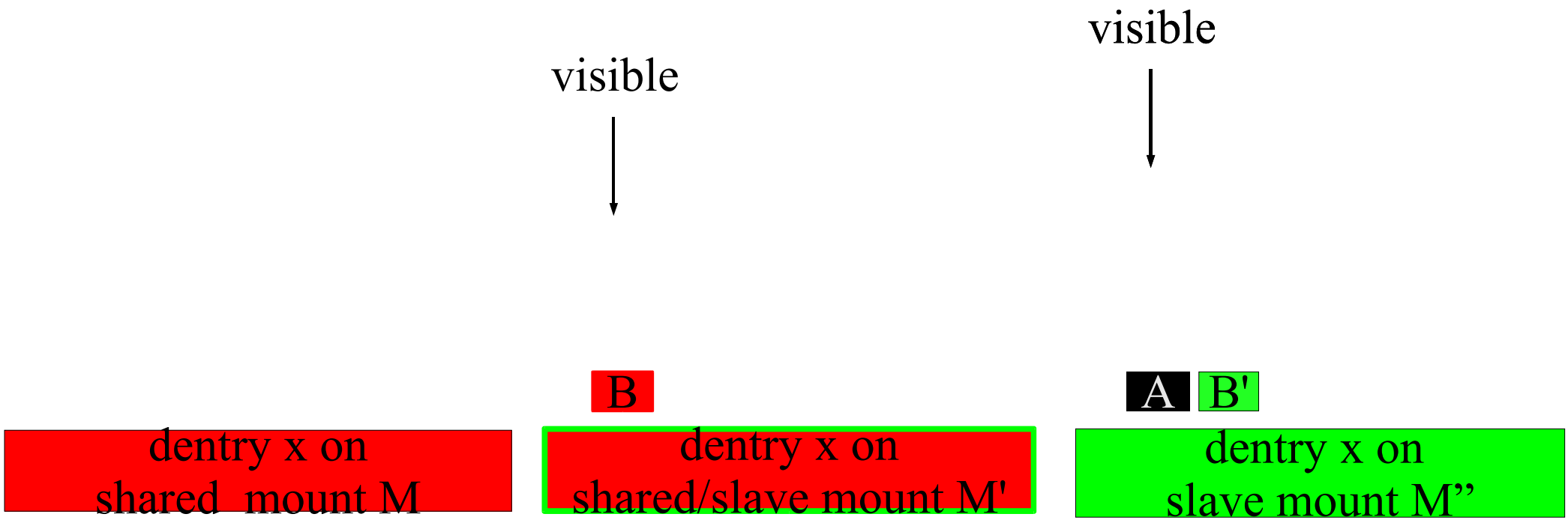
- umount
  - fail if the mount has submounts
  - unmount all child mounts on the mounts belonging to the parent's propagation tree (only if the child mounts do not have children mounts)




tmp1
a
tmp2
a
tmp3
umount tmp/tmp2/tmp3
?

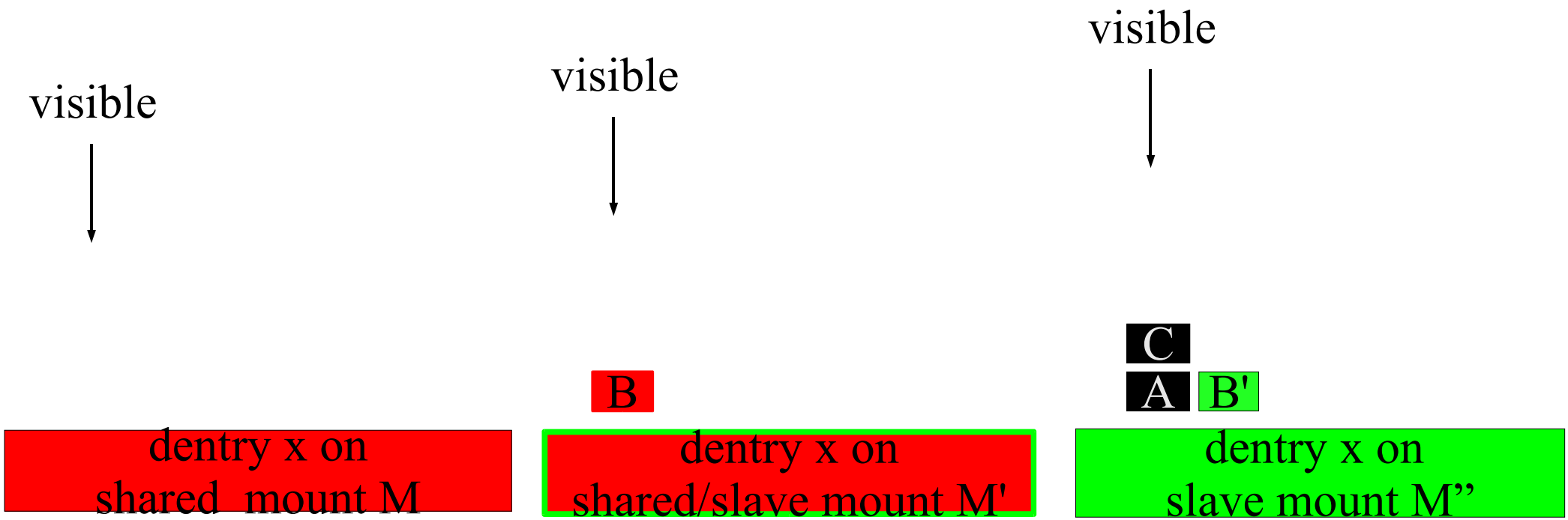
Side-mounts/Over-mounts

Side-mounts/Over-mounts



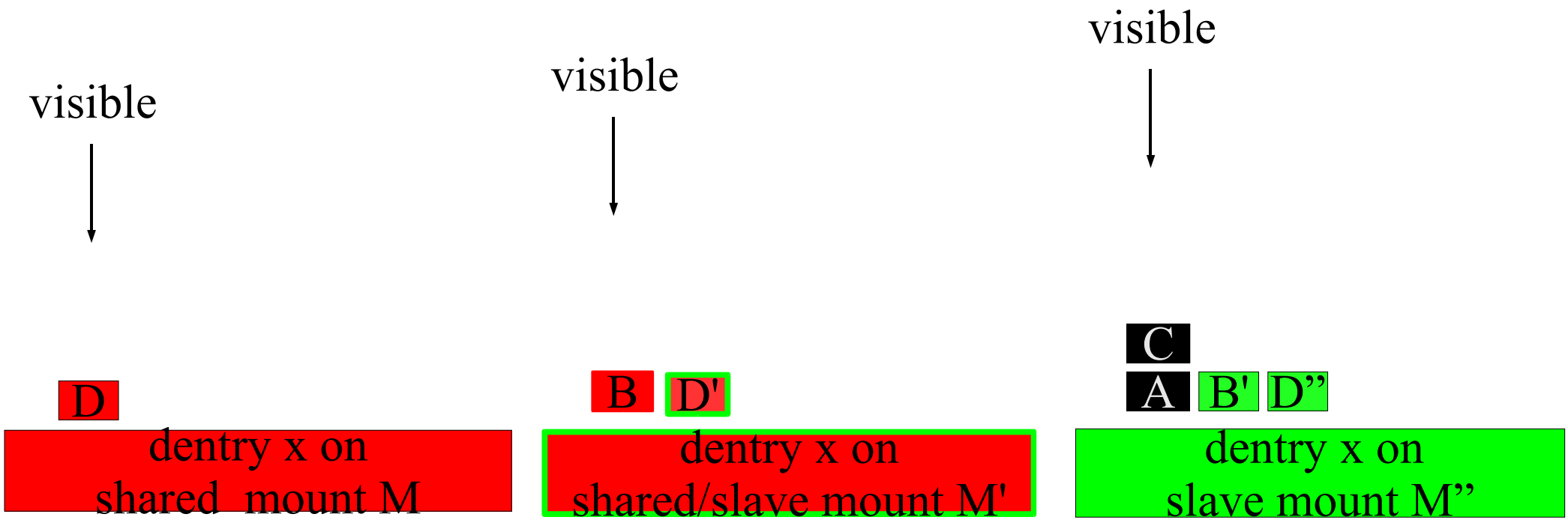
- Side-mount: vfsmounts on the same dentry of a vfsmount. (eg. A and B')
- Least recent side-mount always visible. (A is visible)

Side-mounts/Over-mounts



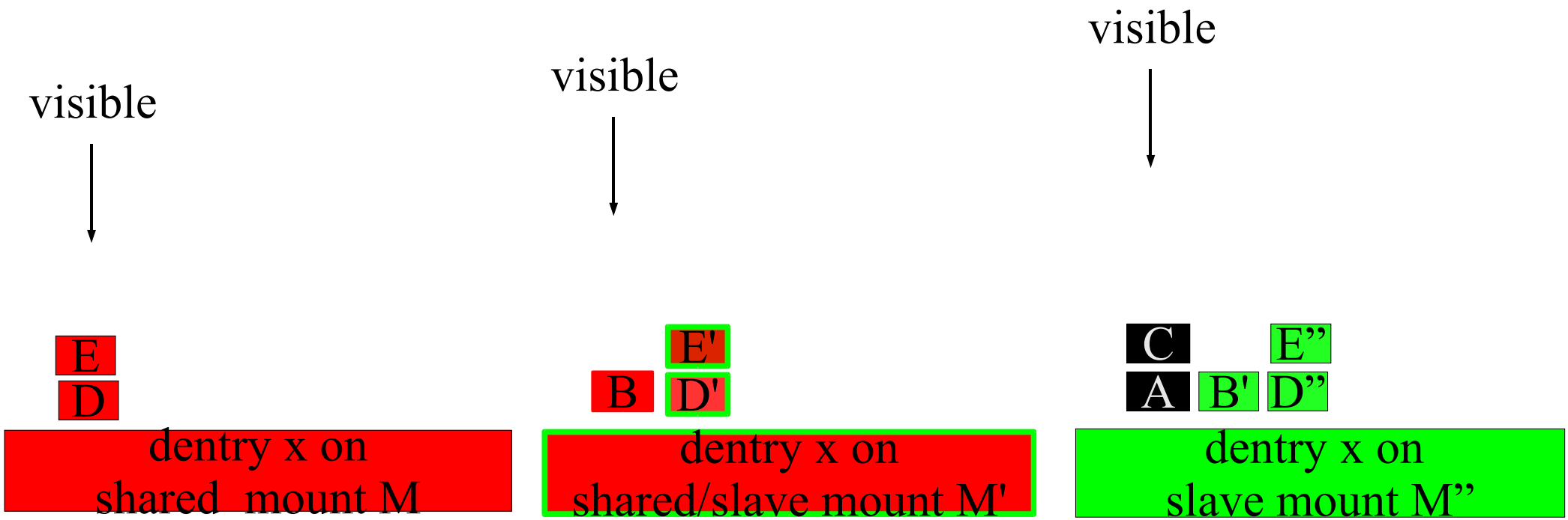
- Side-mount: vfsmount mounted on the same dentry. (eg A and B')
- Over-mount: vfsmount on top of another vfsmount. (eg C over-mount on A)
- Least recent side-mount always visible.(provided there is no over-mount).

Side-mounts/Over-mounts



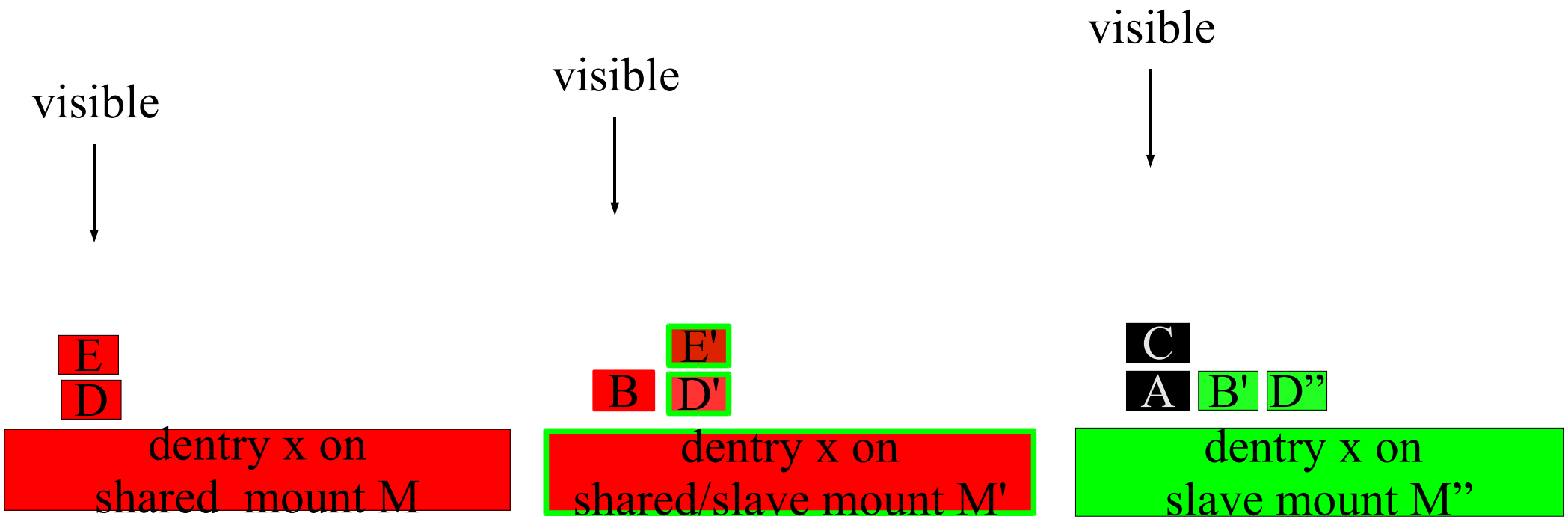
- Side-mount: vfsmount mounted on the same dentry. (eg A, B' and D”)
- Over-mount: vfsmount on top of another vfsmount. (eg C over-mount on A)
- Least recent side-mount always visible.(provided there is no over-mount).

Side-mounts/Over-mounts



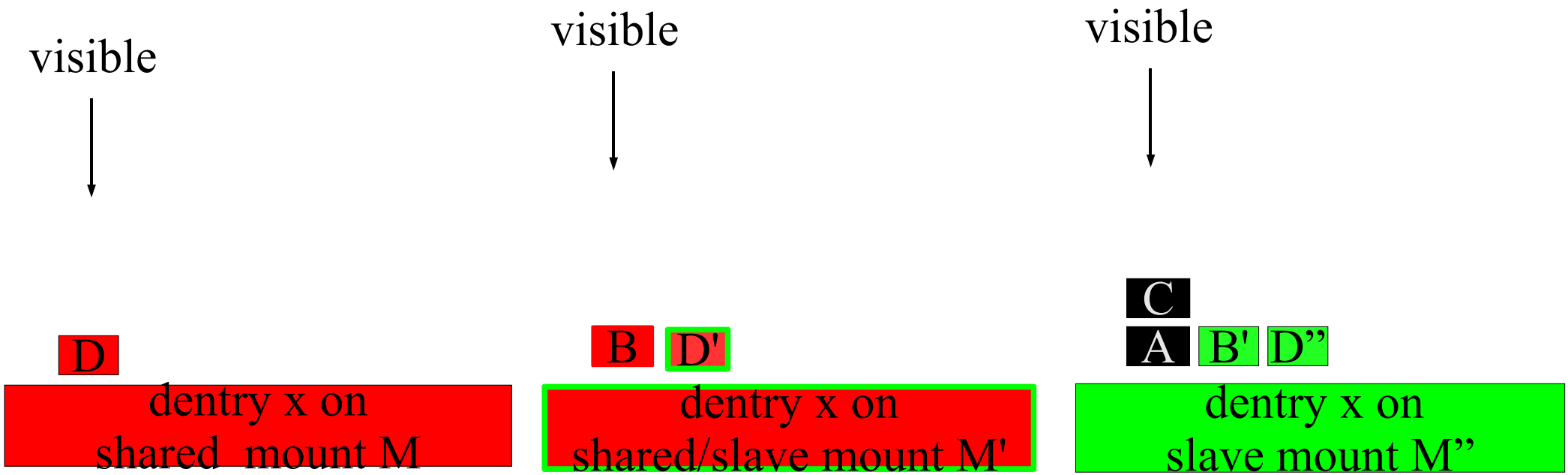
- Side-mount: vfsmounts on the same dentry of a vfsmount.(eg A, B' and D”)
- Over-mount: vfsmounts on top of another vfsmount.(eg C over-mount on A)
- Least recent side-mount always visible.(provided there is no over-mount).
- Over-mount on a side-mount is obscured too. (E' is obscured by B)

Side-mounts/Over-mounts



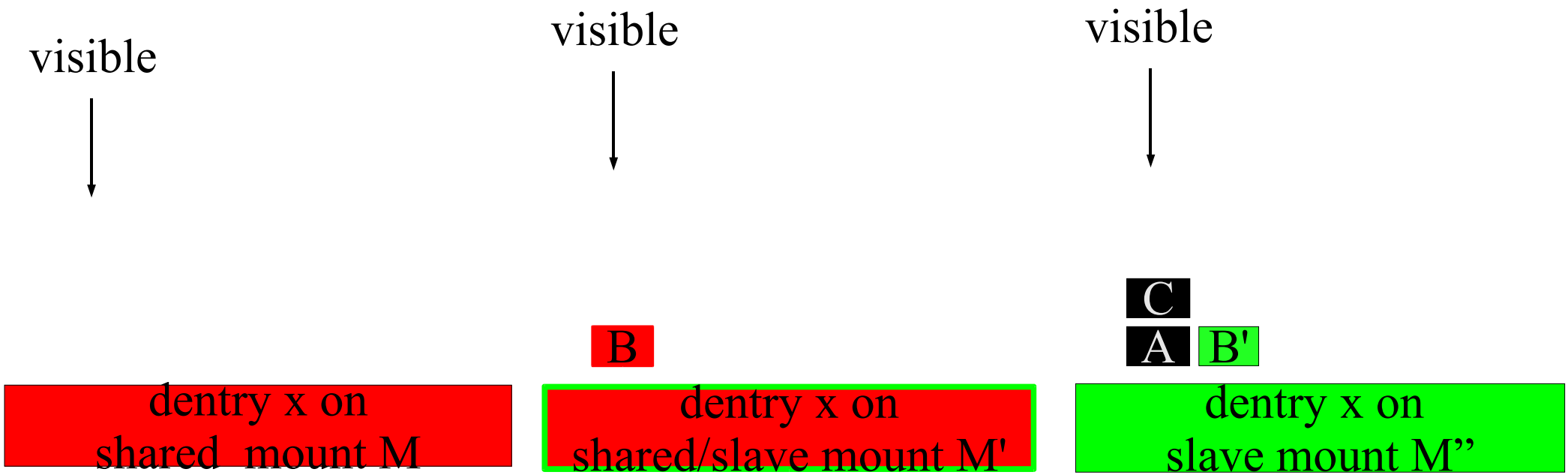
- Side-mount: vfsmounts on the same dentry of a vfsmount.(eg A, B' and D”)
- Over-mount: vfsmounts on top of another vfsmount.(eg C over-mount on A)
- Least recent side-mount always visible.(provided there is no over-mount).
- Over-mount on a side-mount is obscured too. (E' is obscured by B)
- Explicit-unmount unmounts the specified mount and propagates it. (eg E”).

Side-mounts/Over-mounts



- Side-mount: vfsmounts on the same dentry of a vfsmount.(eg A, B' and D”)
- Over-mount: vfsmounts on top of another vfsmount.(eg C over-mount on A)
- Least recent side-mount always visible.(provided there is no over-mount).
- Over-mount on a side-mount is obscured too. (E' is obscured by B).
- Explicit-unmount unmounts the specified mount and propagates it. (eg E”).
- Propagated unmount always unmounts the most recent mount on the dentry.

Side-mounts/Over-mounts



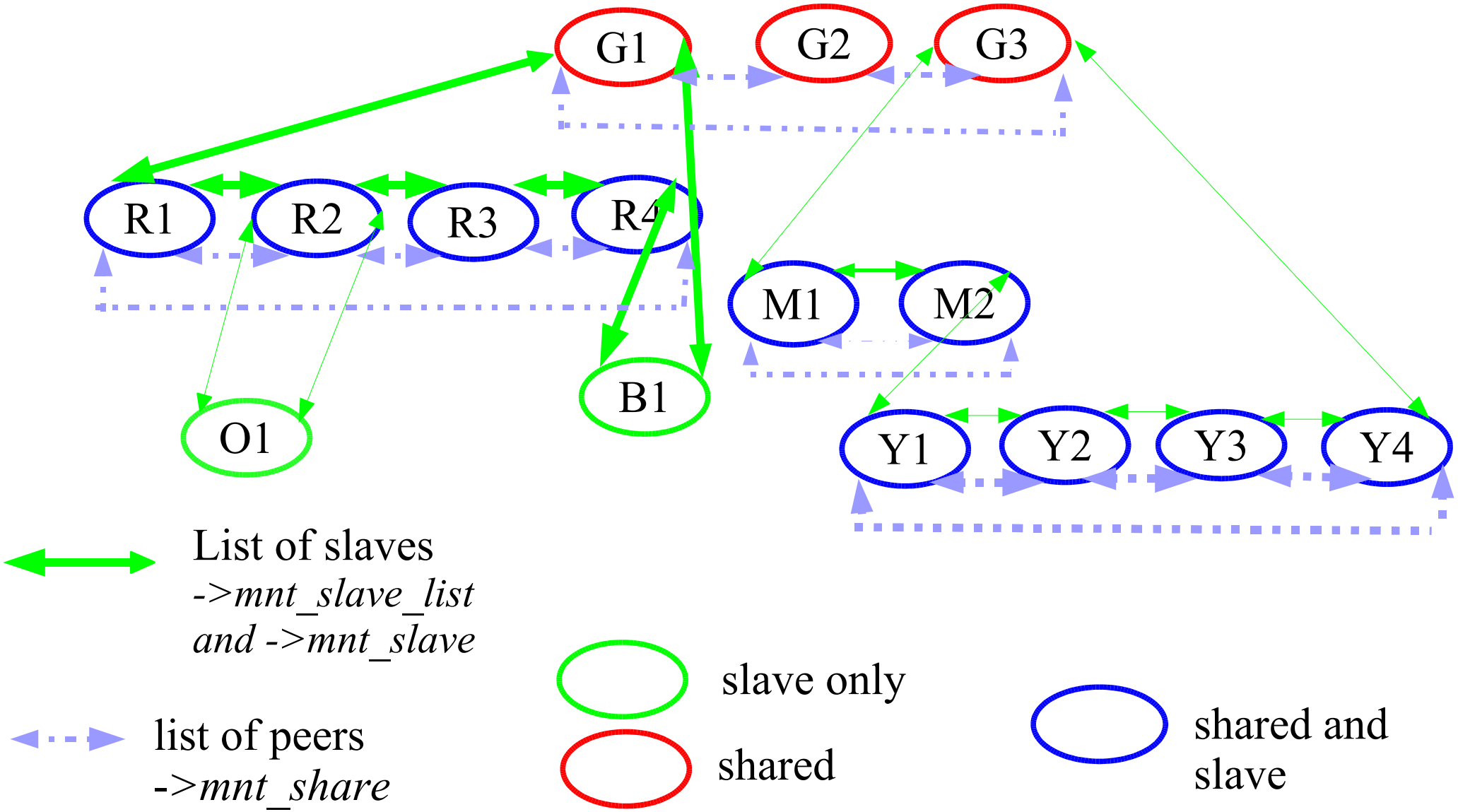
- Side-mount: vfsmounts on the same dentry of a vfsmount.(eg A, B' and D”)
- Over-mount: vfsmounts on top of another vfsmount.(eg C over-mount on A)
- Least recent side-mount always visible.(provided there is no over-mount).
- Over-mount on a side-mount is obscured too. (E' is obscured by B).
- Explicit-unmount unmounts the specified mount and propagates it. (eg E”).
- Propagated unmount always unmounts the most recent side-mount.

Implementation detail

additions to vfsmount structure

- *->mnt_share* circular list of peer mounts
- *->mnt_master* if slave, points to master mount
- *->mnt_slave_list* list of slave mounts
- *->mnt_slave* slave list entry
- additional flags in *->mnt_flags*
  - *MNT_SHARED*
  - *MNT_UNBINDABLE*

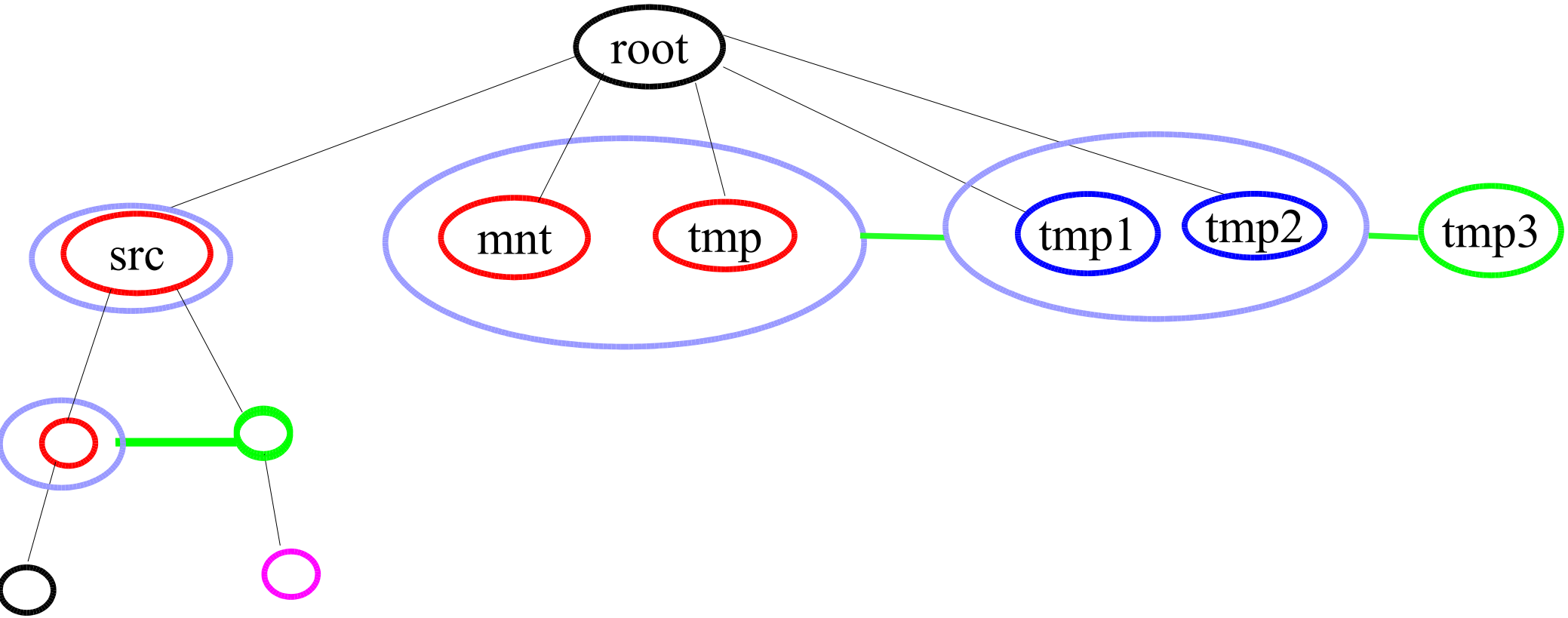
Propagation tree representation

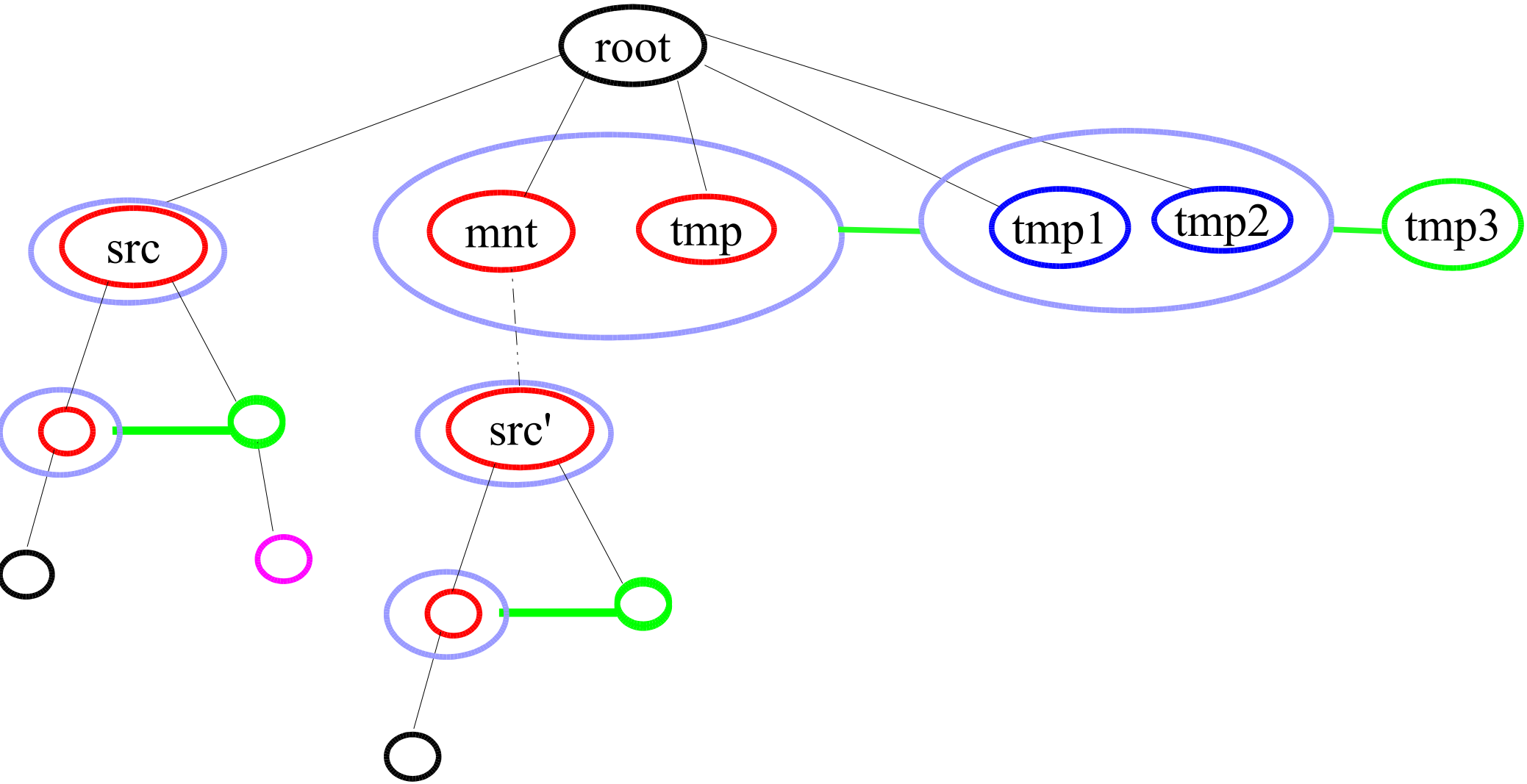


Implementation (continued)

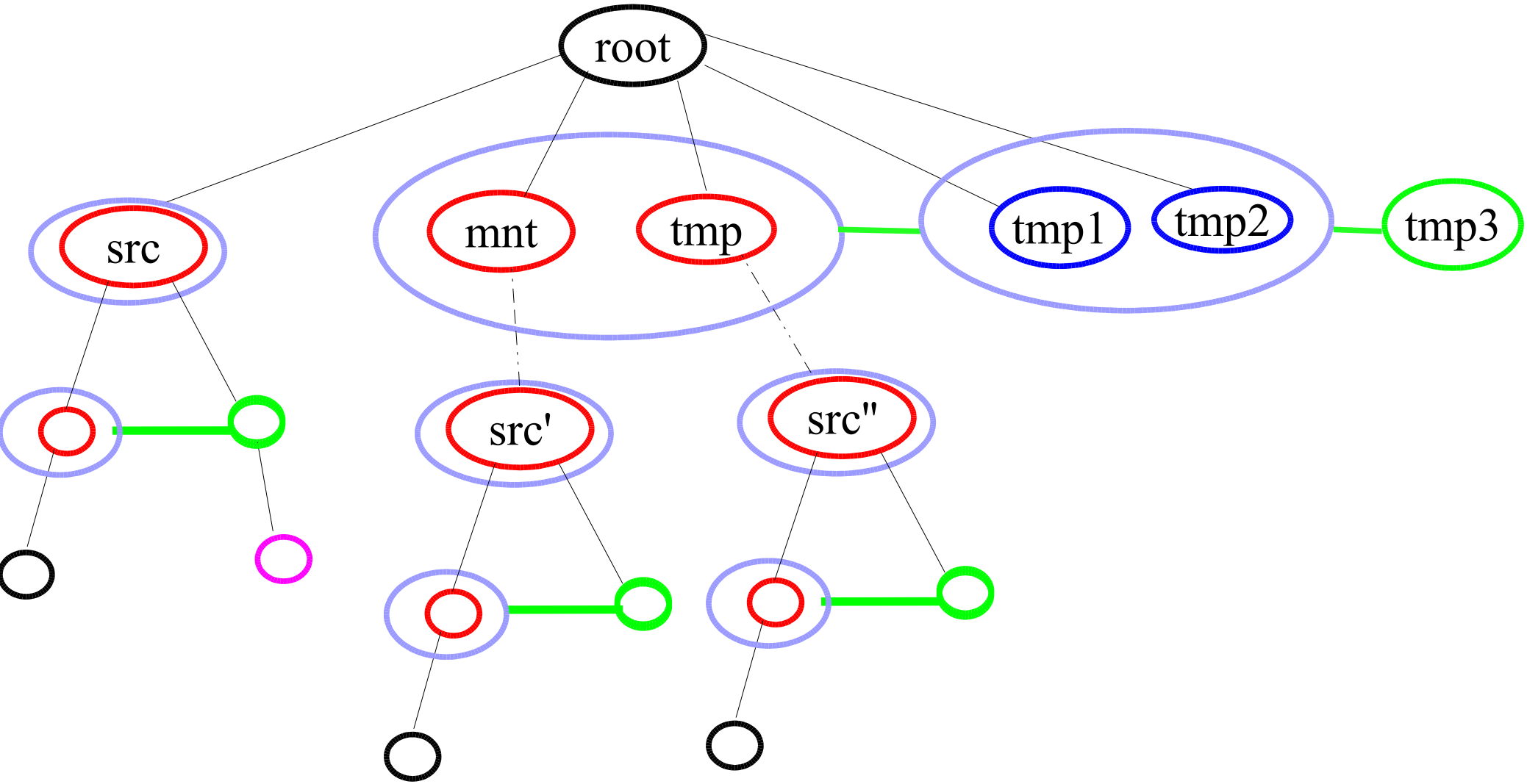
crux of the bind/move operation in

- *attach_recursive_mnt()*
  - clone a copy of the source mount tree for each mount that receives propagation from the destination (*propagate_mnt()/copy_tree()*)
  - build-up the propagation tree for each of the child mount (*clone_mnt()*)
  - if successful, attach the trees to their parents and place them in the hash list. (*commit_tree()*)
  - if allocation fails, release all the newly allocated mount trees (*propagate_mnt()*/*umount_tree()*)

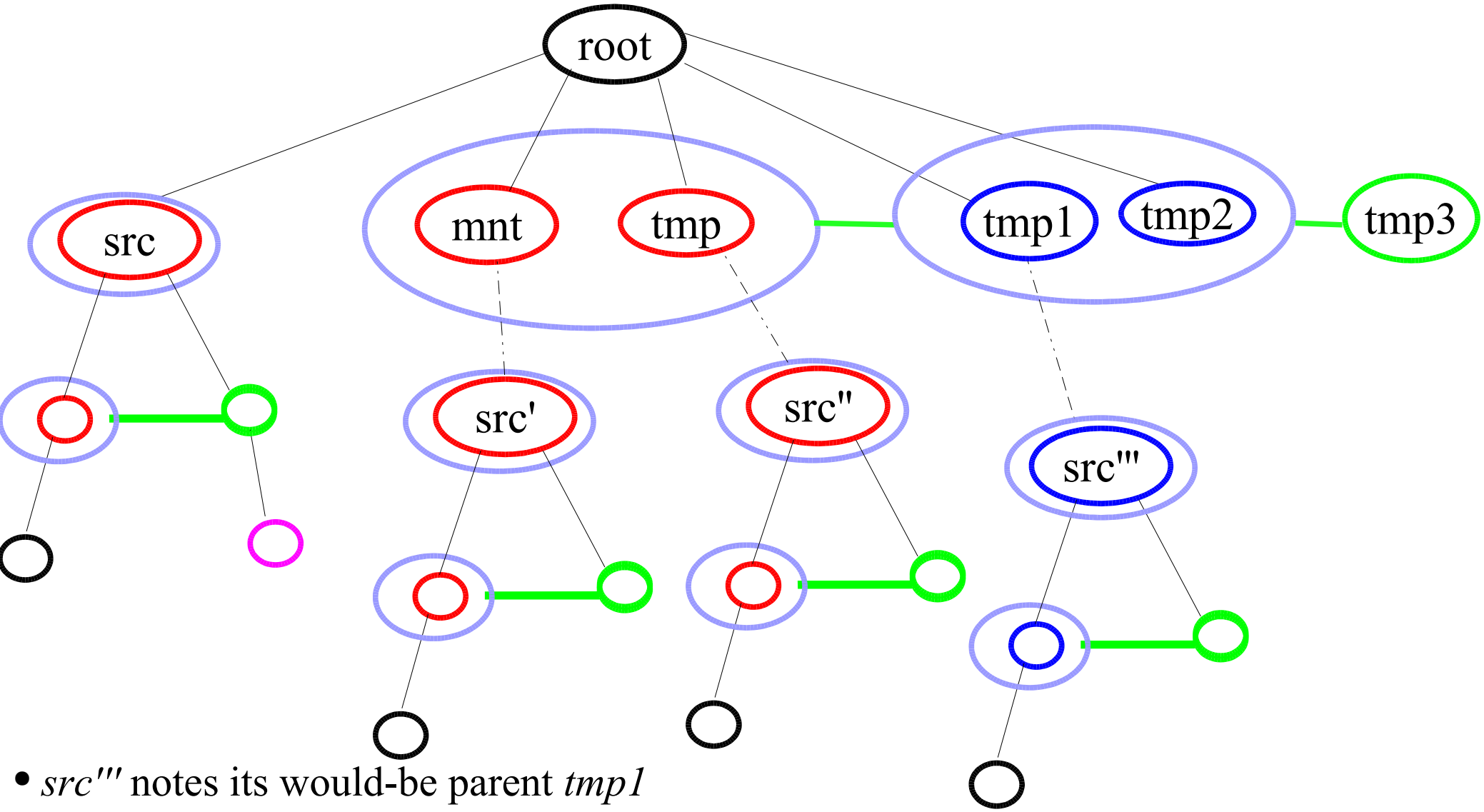

root
src
mnt
tmp
tmp1
tmp2
tmp3



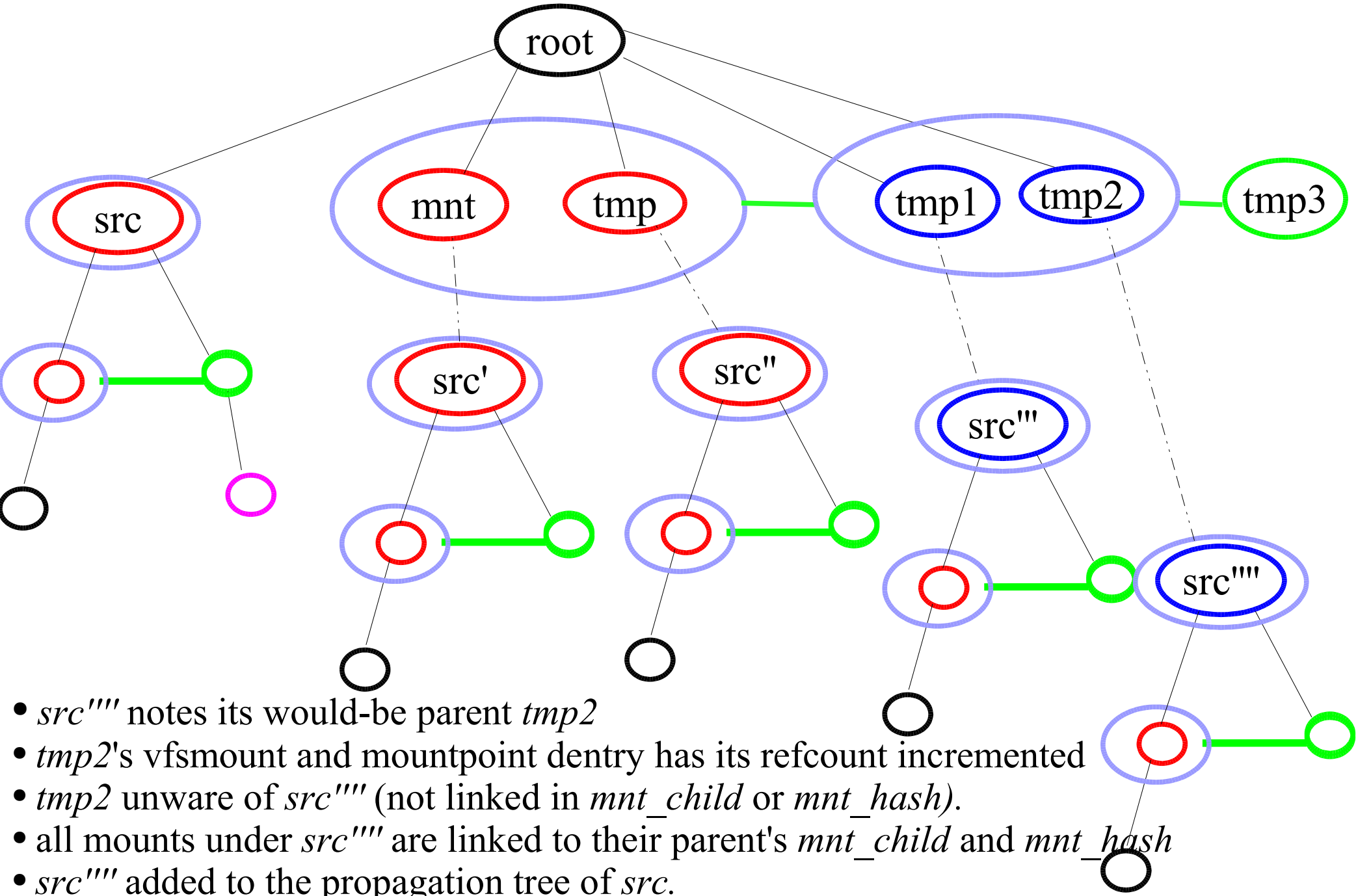
- *src'* notes its would-be parent *mnt*
- *mnt*'s vfsmount and mountpoint dentry has its refcount incremented
- *mnt* unware of *src'* (not linked in *mnt_child* or *mnt_hash).*
- all mounts under *src'* are linked to their parent's *mnt_child* and *mnt_hash*
- *src'* added to the propagation tree of *src.*



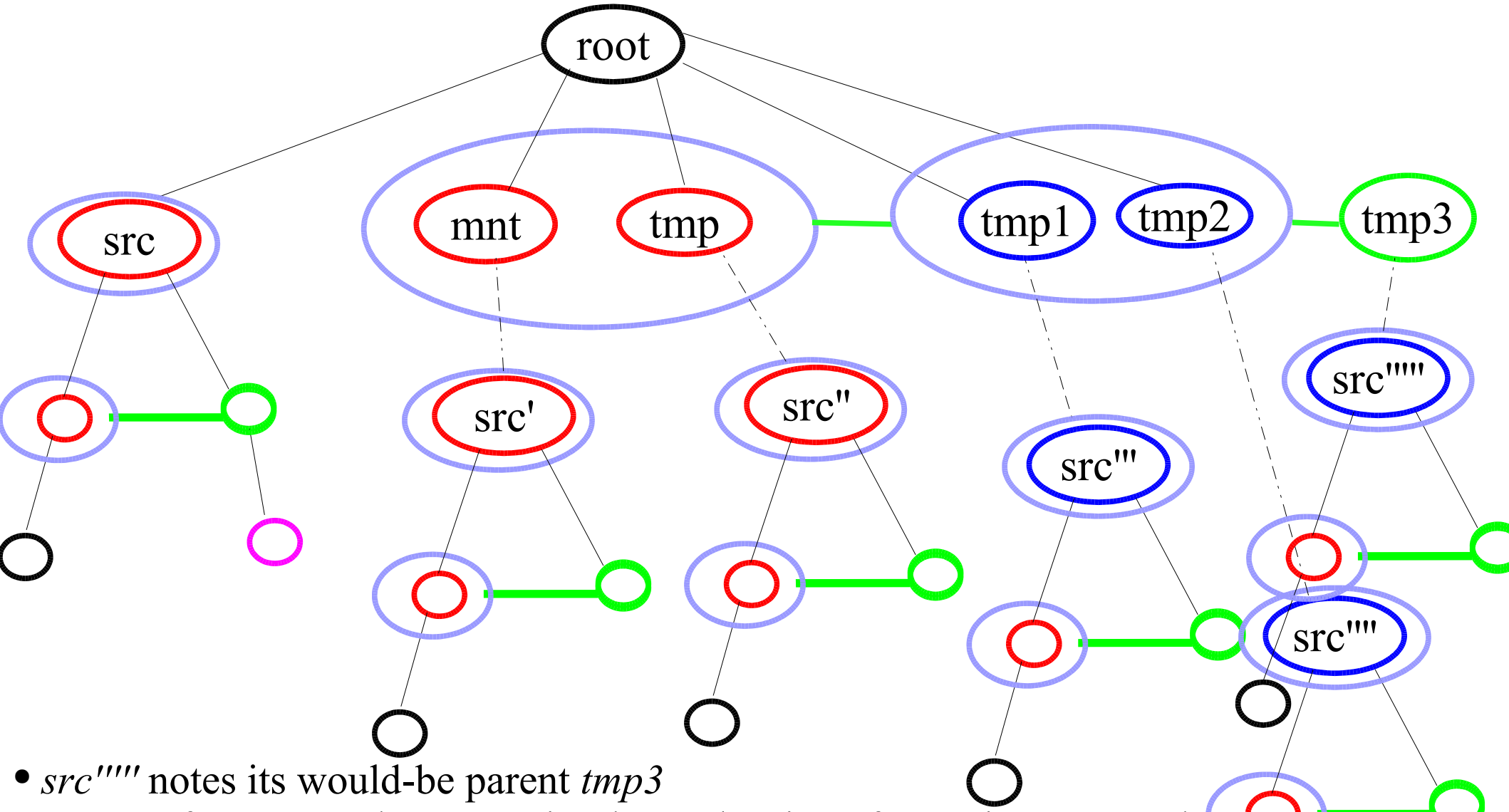
- *src"* notes its would-be parent *tmp*
- *tmp*'s vfsmount and mountpoint dentry has its refcount incremented
- *tmp* unware of *src"* (not linked in *mnt_child* or *mnt_hash).*
- all mounts under *src"* are linked to their parent's *mnt_child* and *mnt_hash*
- *src"* added to the propagation tree of *src*.



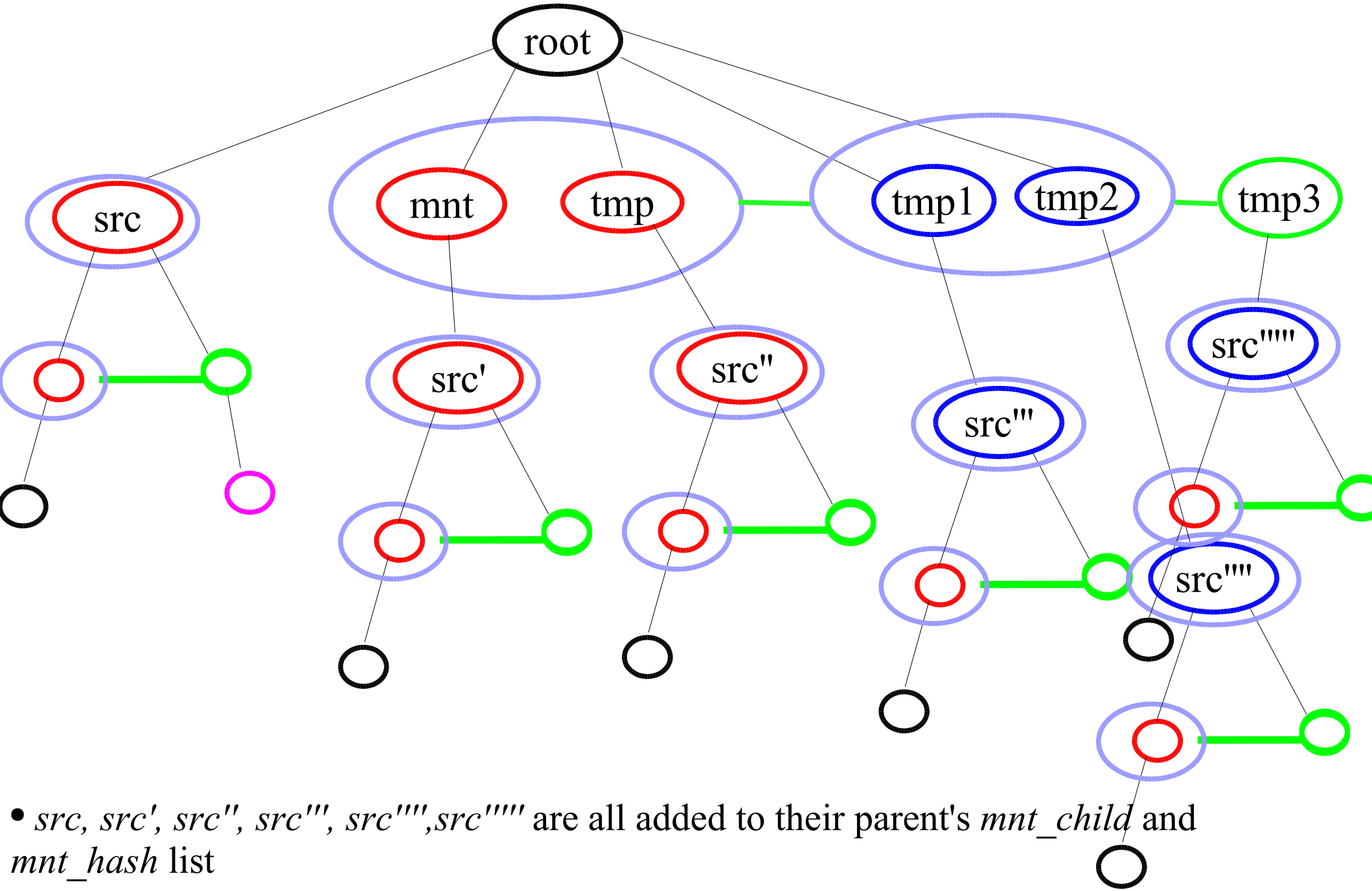
- *src'''* notes its would-be parent *tmp1*
- *tmp1*'s vfsmount and mountpoint dentry has its refcount incremented
- *tmp1* unware of *src'''* (not linked in *mnt_child* or *mnt_hash).*
- all mounts under *src'''* are linked to their parent's *mnt_child* and *mnt_hash*
- *src'''* added to the propagation tree of *src*.



- *src''''* notes its would-be parent *tmp2*
- *tmp2*'s vfsmount and mountpoint dentry has its refcount incremented
- *tmp2* unware of *src''''* (not linked in *mnt_child* or *mnt_hash).*
- all mounts under *src''''* are linked to their parent's *mnt_child* and *mnt_hash*
- *src''''* added to the propagation tree of *src*.



- *src'''''* notes its would-be parent *tmp3*
- *tmp3*'s vfsmount and mountpoint dentry has its refcount incremented
- *tmp3* unware of *src'''''* (not linked in *mnt_child* or *mnt_hash).*
- all mounts under *src'''''* are linked to their parent's *mnt_child* and *mnt_hash*
- *src'''''* added to the propagation tree of *src*.



- *src, src', src'', src''', src'''',src'''''* are all added to their parent's *mnt_child* and *mnt_hash* list

Implementation (continued)

crux of the umount operation in

- umount_tree*()*
  - collect all the mount trees that can be unmounted (*propagate_umount()*)
  - unhash the mounts. No longer available through *lookup_mnt().*
  - detach them from their propagation trees. (*change_mnt_propagation()*)
  - detach each mount from their mount trees. (*release_mounts()*)

Pointers to documentation

Al Viro's RFC:
http://lwn.net/Articles/119232/

Ram Pai's implementation history:
https://www.sudhaa.com/~ram/sharedsubtree

Extensive Documentation:
http://lwn.net/Articles/159077/

This paper (updated):
https://www.sudhaa.com/~ram/sharedsubtree/paper/sharedsubtree.pdf

per-user namespace

A user-space solution

- maintain /share as a shared mount in original namespace.
- maintain a mount for each user in /share.
- when user logs in, sshd clones-off a new namespace.
- rbind /share/$USER /home/$USER
- mount –make-private /share
- umount -l /share

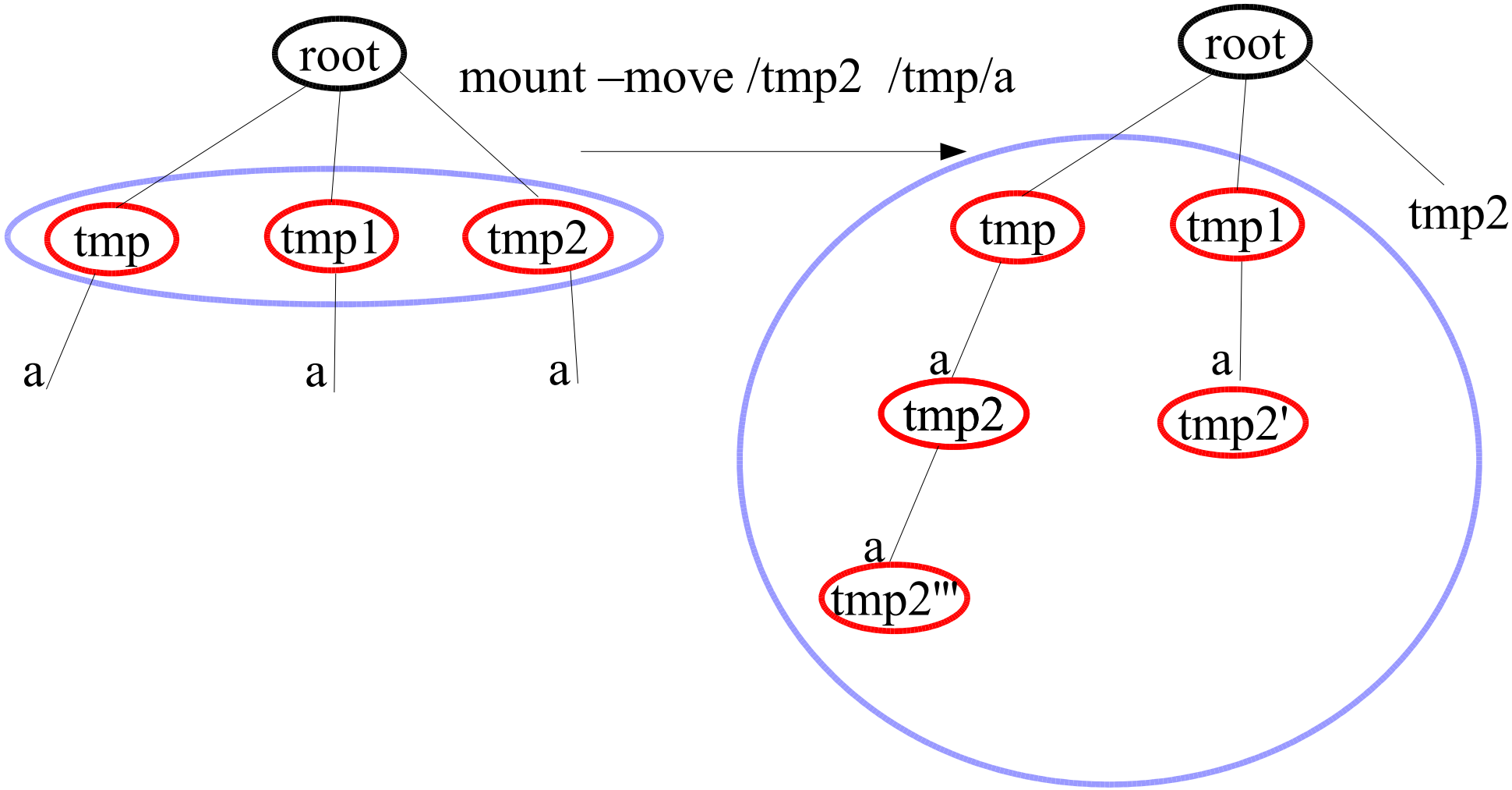
Future work

- /proc interface to view the propagation tree.
- /proc/mount confusion fix.
- mount command using the new interfaces.
  - may need revamp of /proc/mount interface or something similar
- user-mount accounting.
- Ability to lazy unmount a tree without dismantling it.
- union-mount semantics definition and implementation.

Summary

- Ability to share mount tree.
  - fixes namespace isolation
  - makes bind/rbind dynamic
- provides building blocks for
  - per-user-namespace.
  - versioned filesystem.
  - containers.

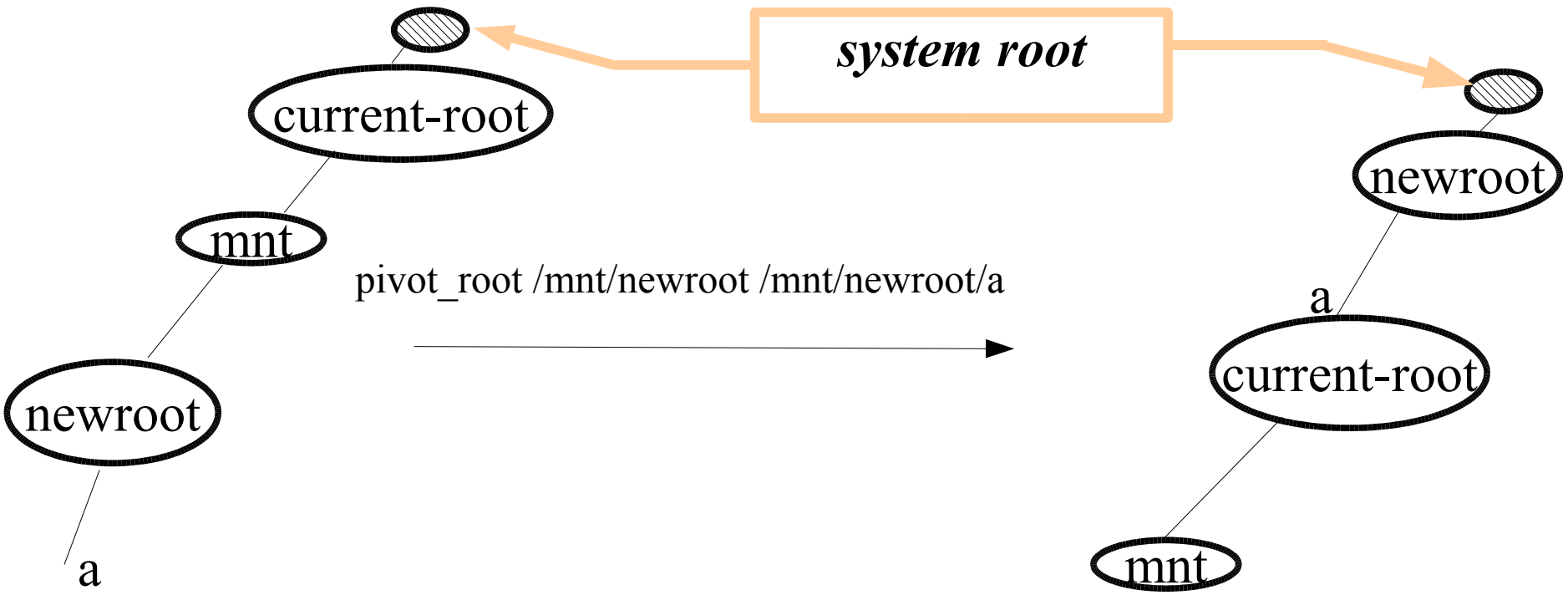
Legal Statement


root
mount –move /tmp2 /tmp/a
tmp
tmp1
tmp2
a
a
a
root
tmp
tmp1
tmp2
a
a
tmp2
tmp2'
a
tmp2'''

Pivot root

- pivot_root

- parent of current-root cannot be shared
- parent of new-root cannot be shared
- new-root cannot be shared



Requirement

- Ability to maintain multiple identical mount trees, each tree associated with some entity.
  - containers (associate a system mount tree per container)
  - MVFS (associate a mount tree per view)
- Ability to make private changes to part of a tree.
  - FUSE (private mount that is not visible to anybody else).
  - SeLinux LSPP (private mounts not visible to anybody else).
- How?
  - Filesystem Namespace (per-process namespace)?
  - Rbind?