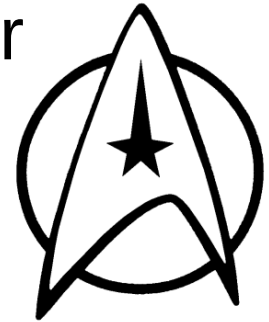
Federating HTCondor pools

Greg Thain

Agenda

Ways to send jobs from one pool to another

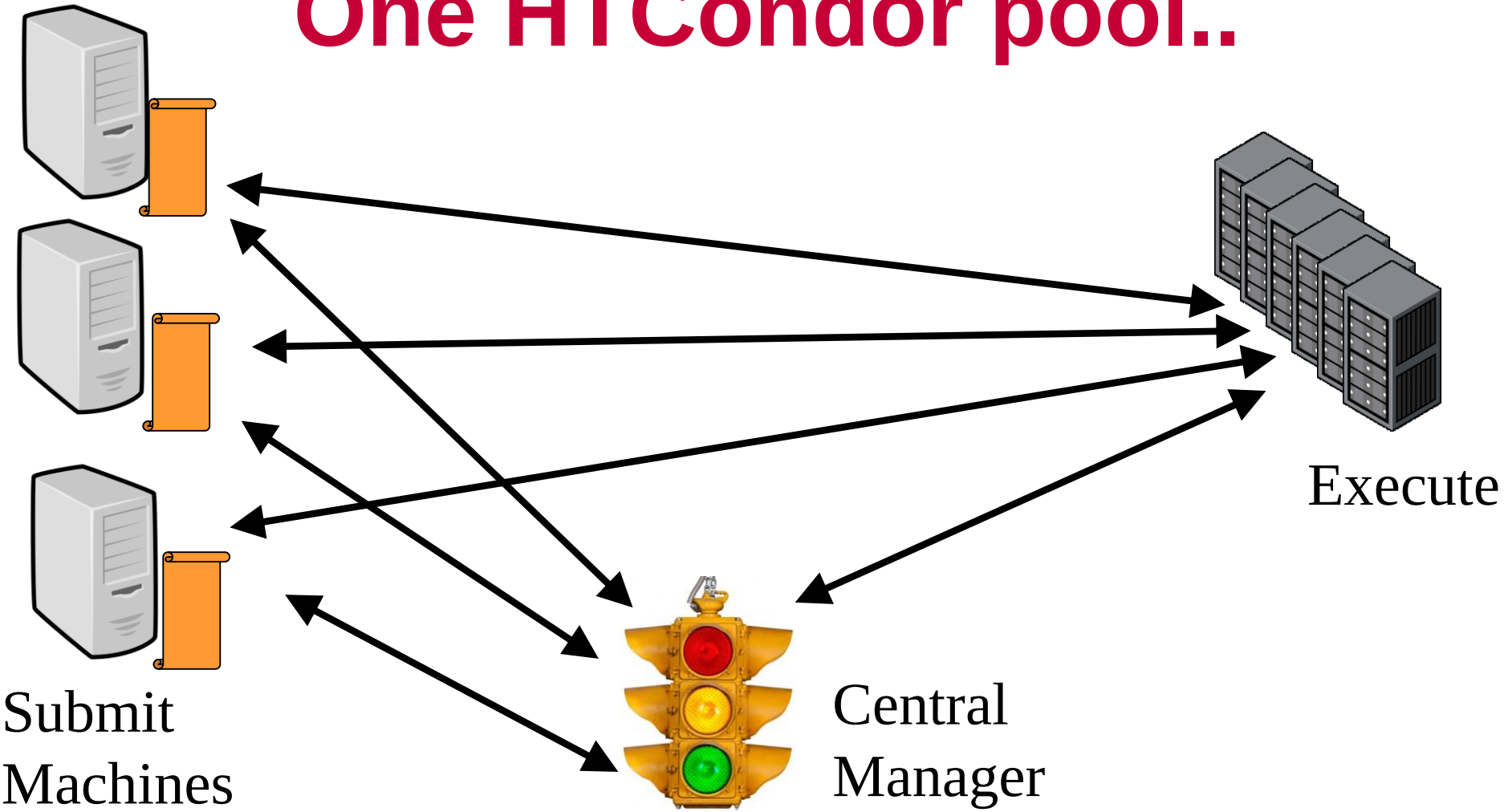
or… machines from one pool to another



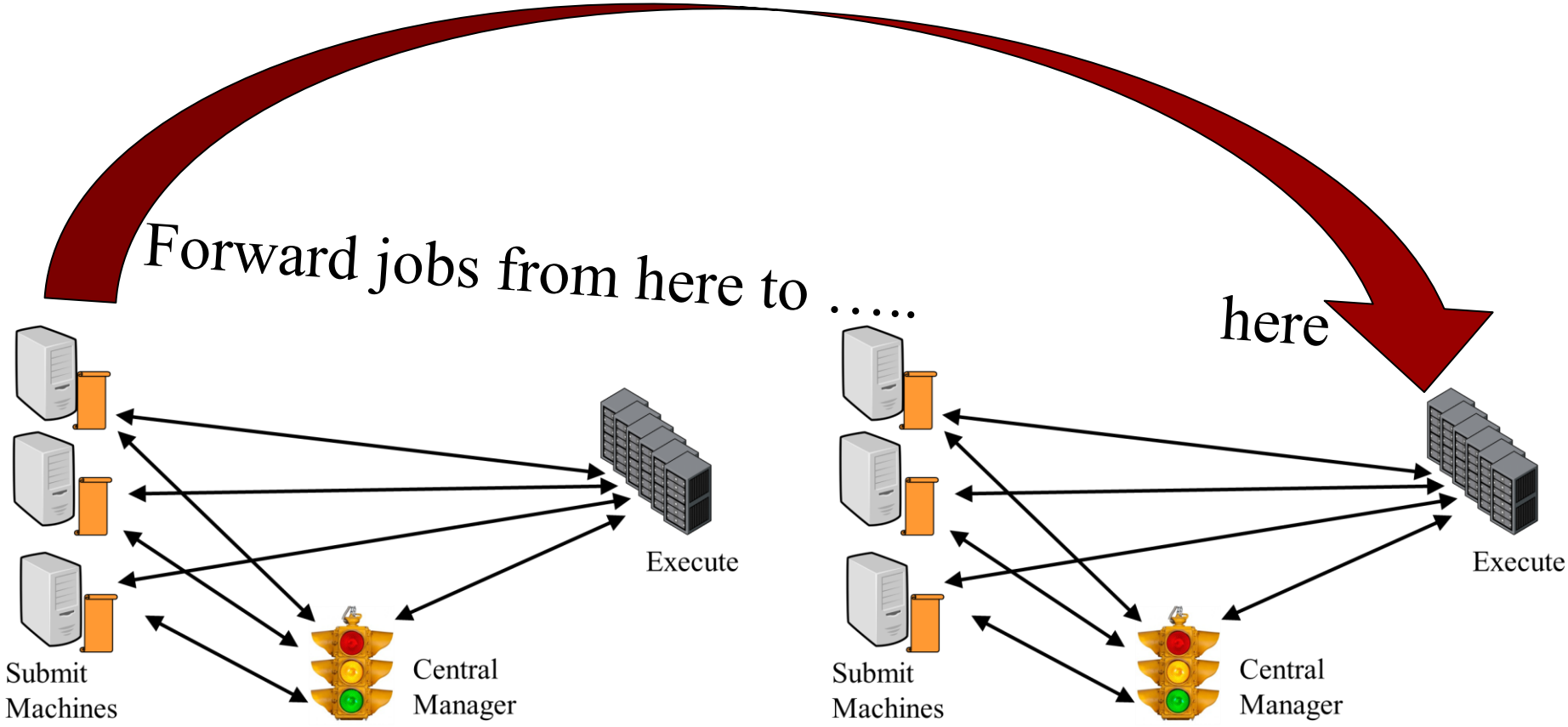
Advantages and Disadvantages to every way

- Merging
- Flocking
- Startd flocking
- Condor-C
- Job Router
- Glidein, in general
- GlideinWMS
- Condor CE

One HTCondor pool..



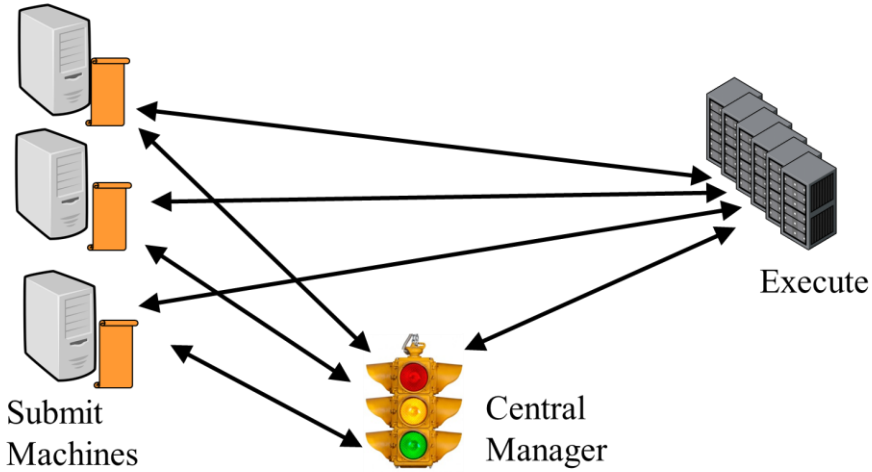
Two pools

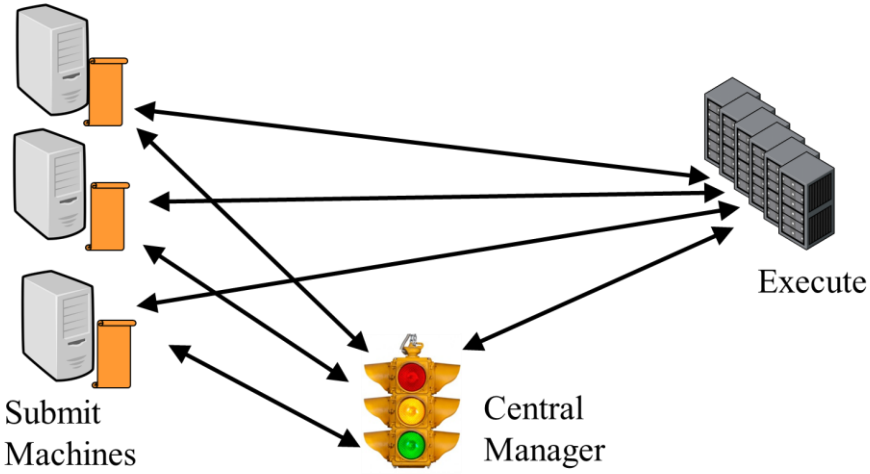

Forward jobs from here to .....
here
Execute
Submit Machines
Central Manager
Execute
Submit Machines
Central Manager

Many Policy Questions

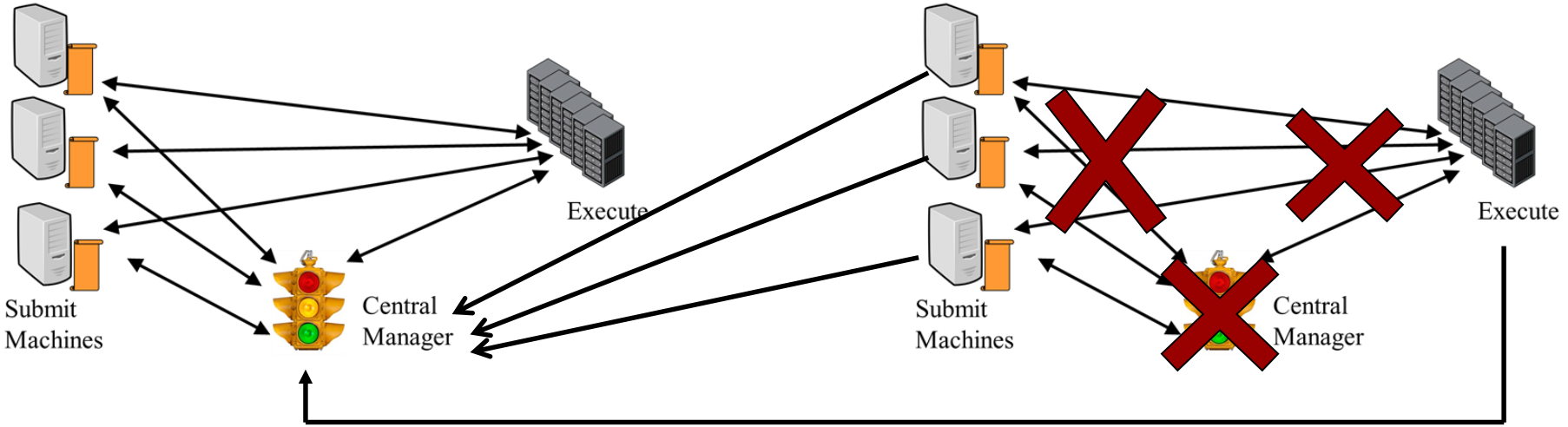
From just one schedd?
For all jobs?
To all startds?
Who decides to send jobs?

When to decide?
What about firewalls?
Who is the Administrator?
Accounting and fair share





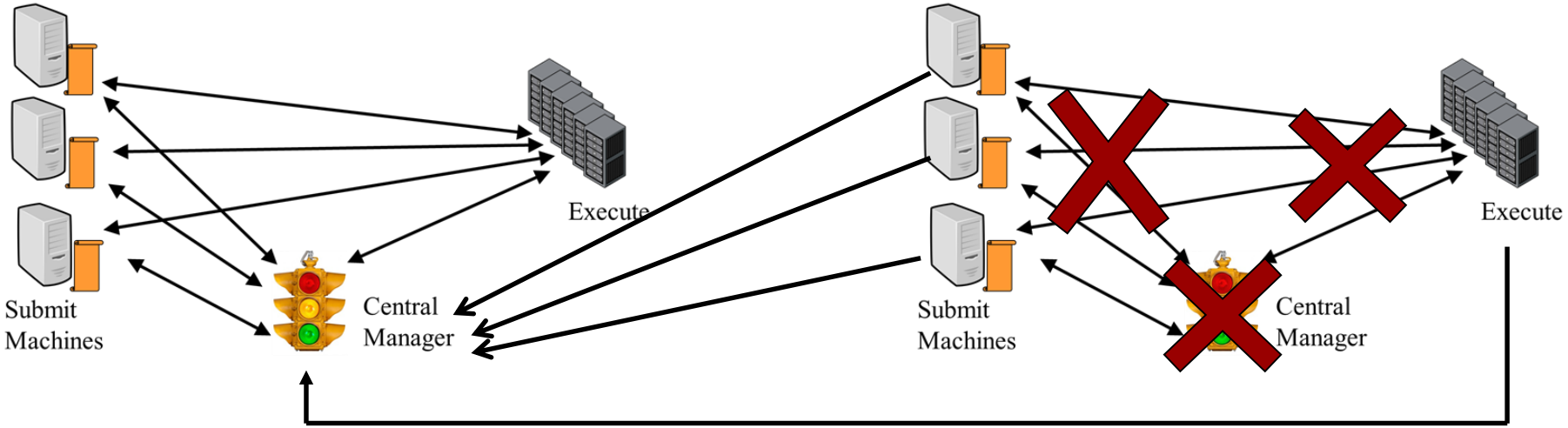
Merging: Just one 1 big pool




```
CONDOR_HOST = OTHER_CM_MACHINE
```

Change right hand condor pool's config file

Merging: Pros

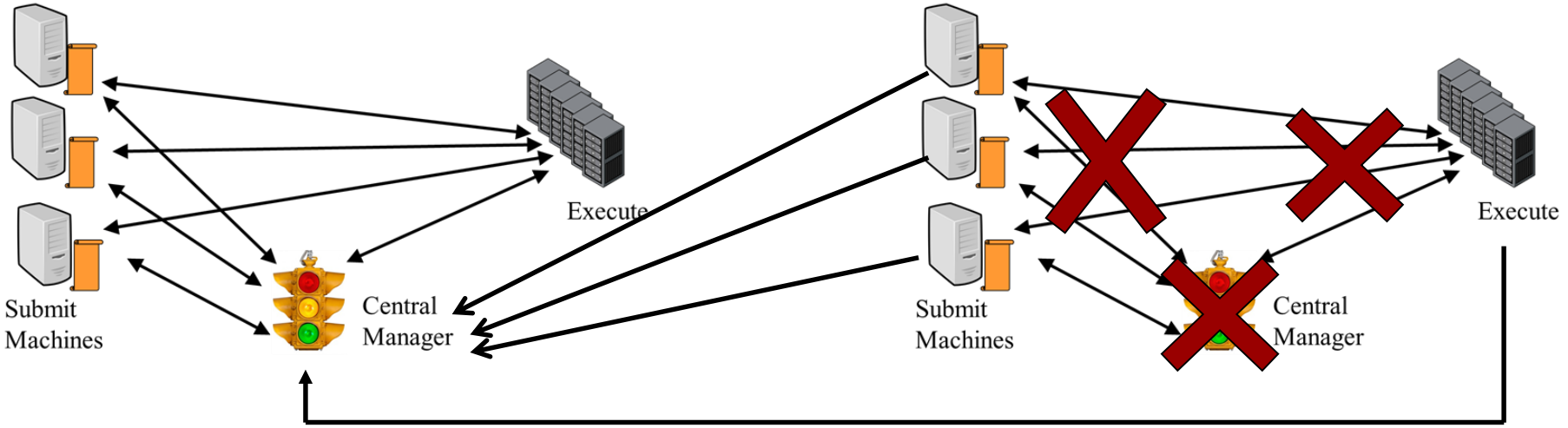


Easy to implement

All jobs go to all machines

Single fair share and accounting records

Merging: Cons



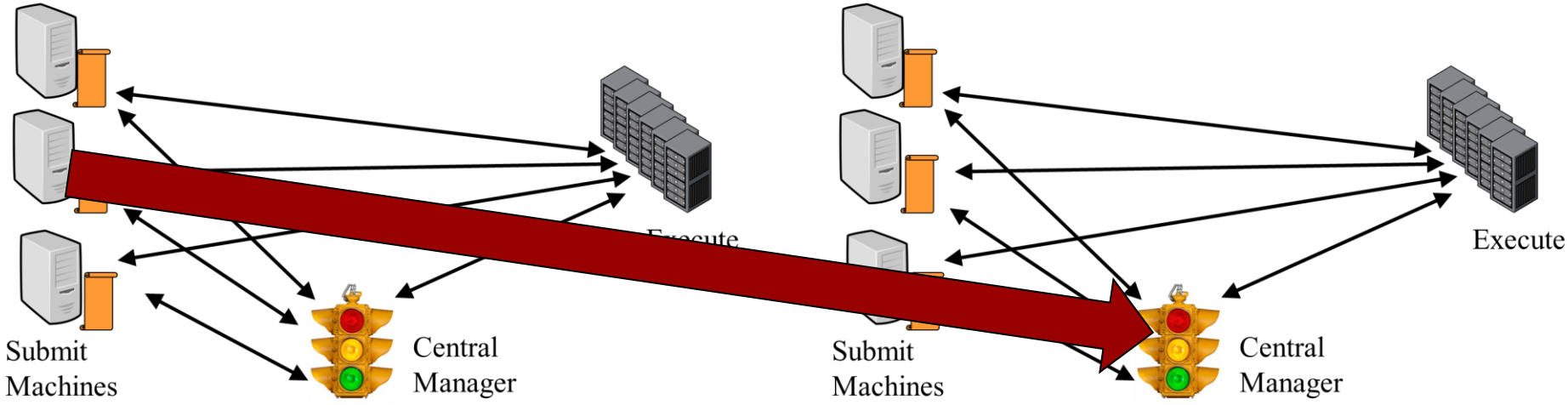
Requires one central manager – one accountant

May have firewall and networking problems

Can’t keep pools separate

Flocking

Flocking is a relationship from ONE SCHEDD to another CM



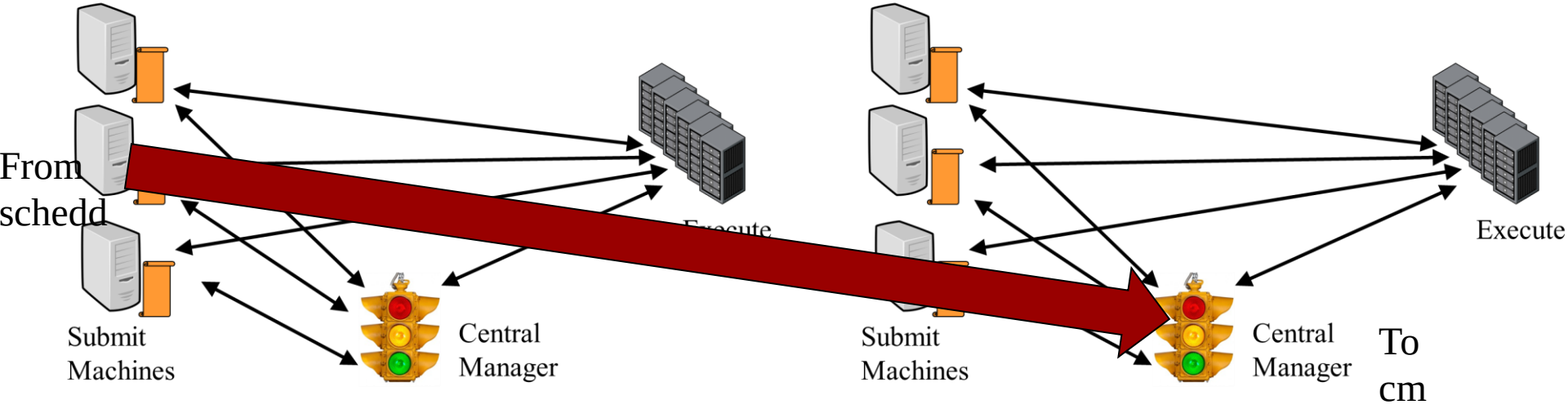
Flocking

```
FLOCK_TO = ip.addr.to.cm
```

From schedd config

```
FLOCK_FROM = \
  ip.addr.from.sched
```

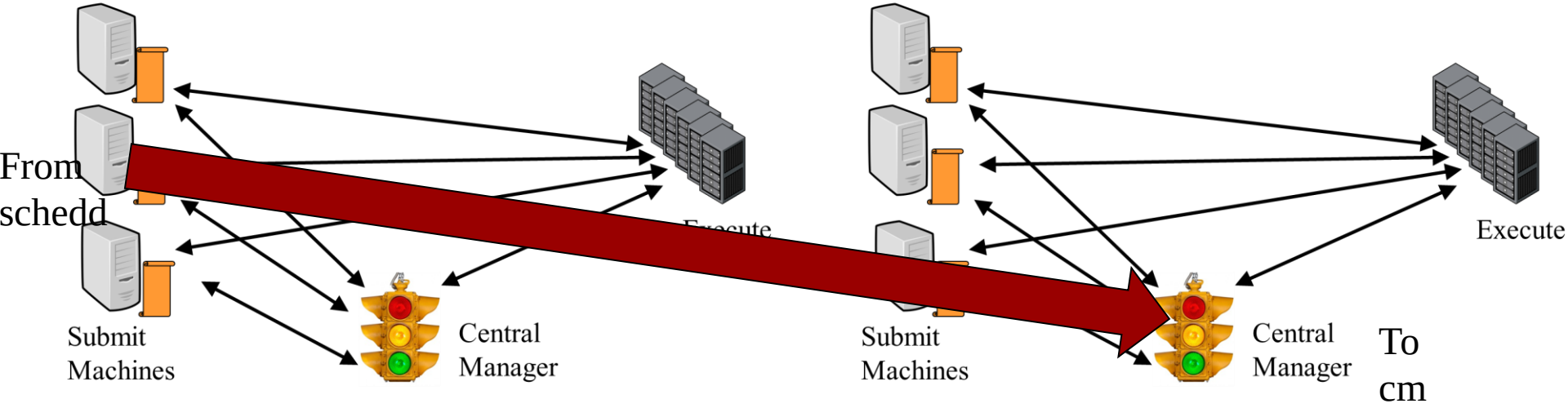
To cm config



Flocking: Pros

Easy to set up
Policy is fixed

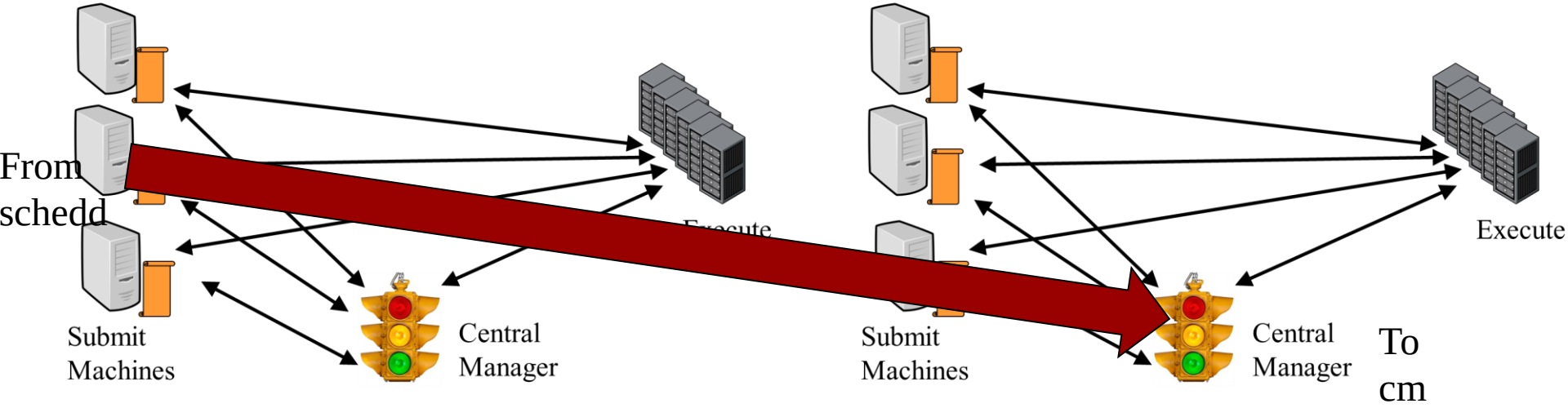
Works for many uses



Flocking: Cons

Difficult when many scheds
Or many cms
Policy is fixed

Requires trust between pools
Requires good networks



Selective Flocking

- By default, ALL jobs eligible to flock
- May want users to opt in via job submission

```
JOB_TRANSFORM_NAMES = REQUIREMENTS

JOB_TRANSFORM_REQUIREMENTS @= end
REQUIREMENTS JobUniverse == 5 && !(MY.WantGlidein?:0 )
SET requirements ( TARGET.PoolName == “MyHomePool" ) &&\
  $(MY.requirements)
@end
```

New schedd config

Selective Flocking

```
STARTD_ATTRS = PoolName, $(STARTD_ATTRS)
PoolName = “MyHomePool”
```

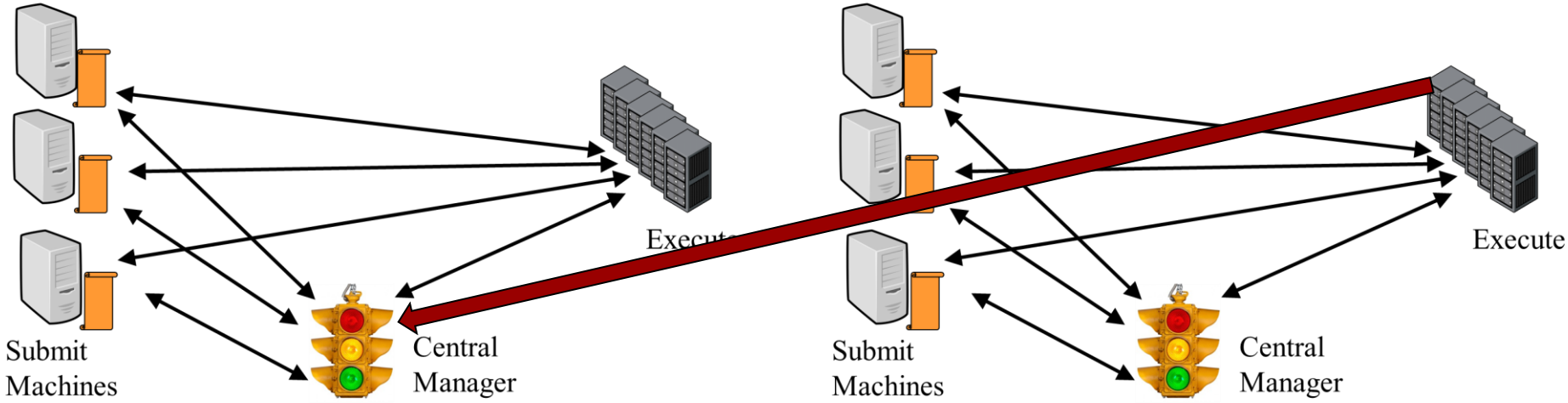
New startd config

```
Executable = foo
Arguments = 1 2 3
Log = log
+WantGlidein = true
queue
```

New submit file

Startd (reverse) Flocking

Startd flocking allows one startd to appear in > 1 pool



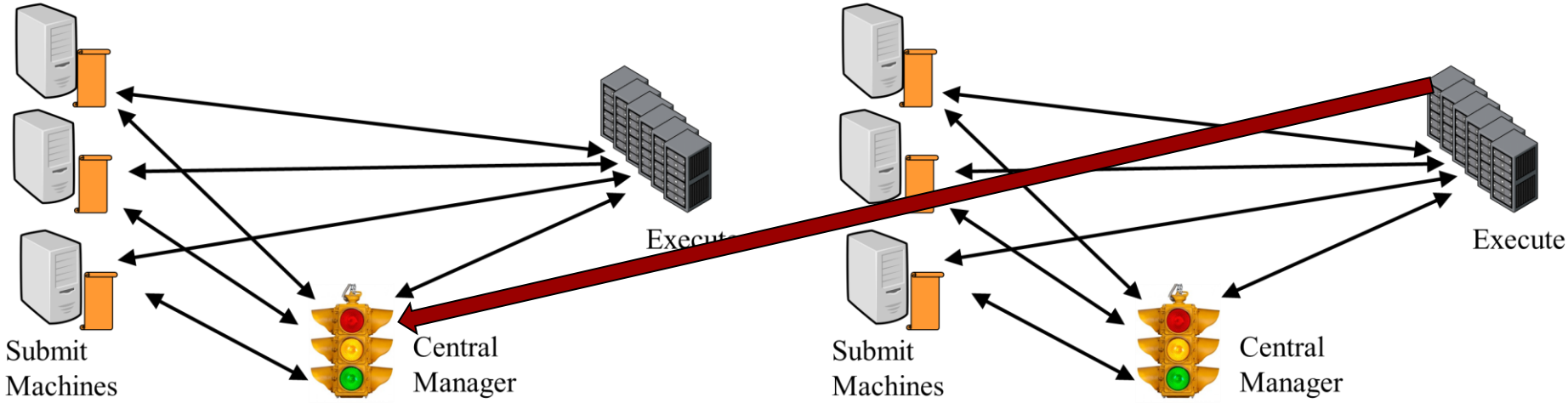
Startd Flocking Config

```
ALLOW_ADVERTISE_STARTD = \
   from.startd.addr
```

To cm config

```
COLLECTOR_HOST = \
   my.cm, your.cm
```

From startd config



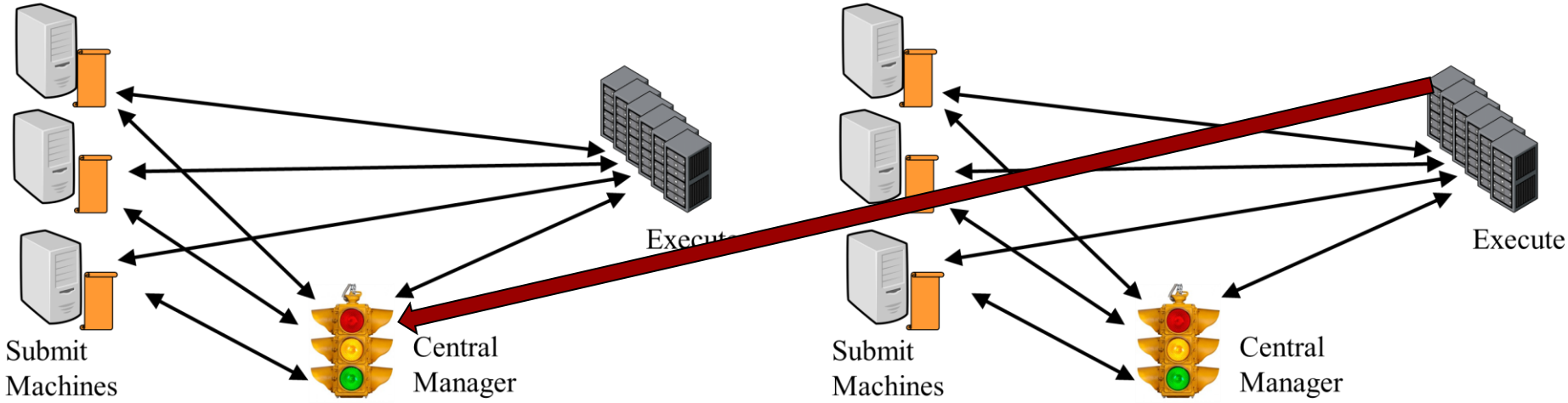
Startd Flocking: Pros

Per startd control

Easy to set up

Policy is fixed

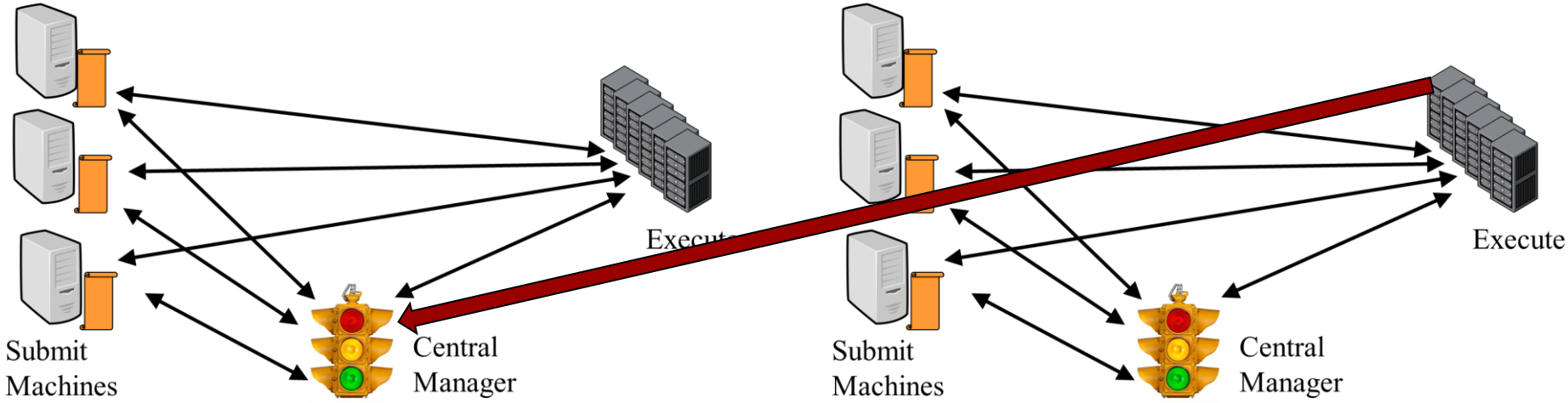
Good for friendly pools



Startd Flocking: Cons

Difficult when many pools
Accounting may be tricky
Policy is mostly fixed

Requires trust between pools
Requires good networks
No user mapping



Condor-C

Condor-c is a **job** that runs on foreign schedd

```
grid_resource = condor joe@remotesched.example.com\
       remotecm.example.com
+remote_jobuniverse = 5
+remote_requirements = True
+remote_ShouldTransferFiles = "YES"
+remote_WhenToTransferOutput = "ON_EXIT"

Executable = foo
Arguments = 1 2 3
Log = log
queue
```

Condor-C: Pros

Per job forwarding
No policy
Useful as a base for other systems
After job sent, network can be broken
Good scalability
User is in charge
Good for submitting pilots

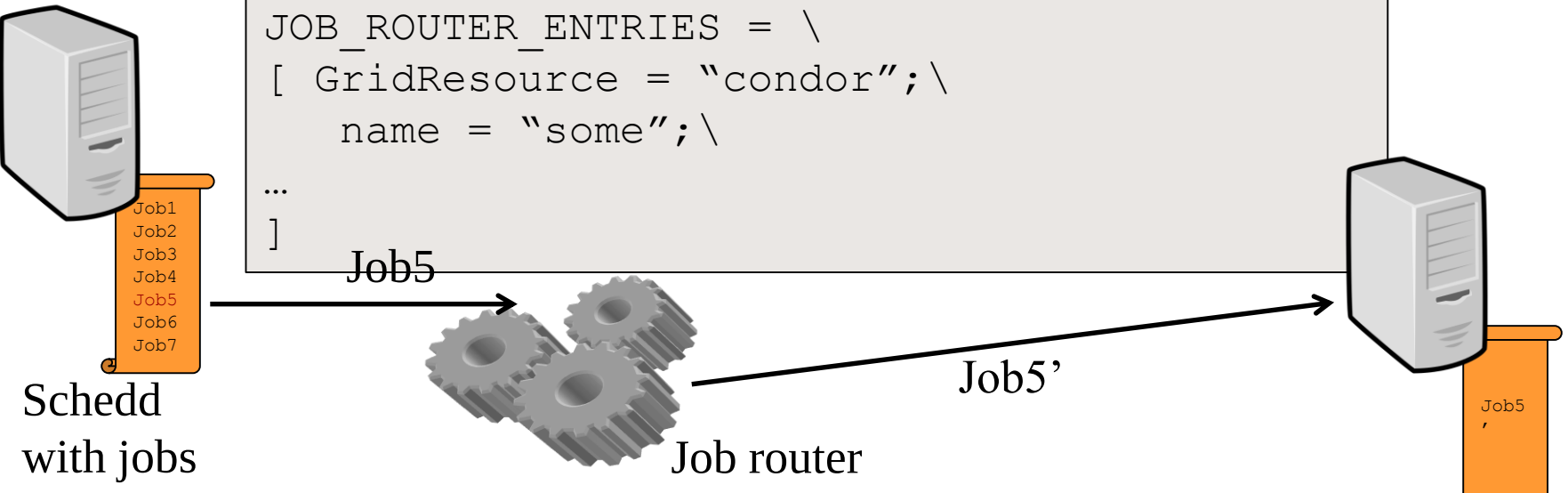
Condor-C: Cons

Requires GSI or SSL authentication – tough to set up

Job policy is fixed at submit time

Job Router: config

```
JOB_ROUTER_DEFAULTS = \
[ \
        requirements = WantJobRouter;\
        MaxJobs = 10;\
        delete_requirements = true;\
]
JOB_ROUTER_ENTRIES = \
[ GridResource = “condor”;\
   name = “some”;\
…
]
```



Job Router

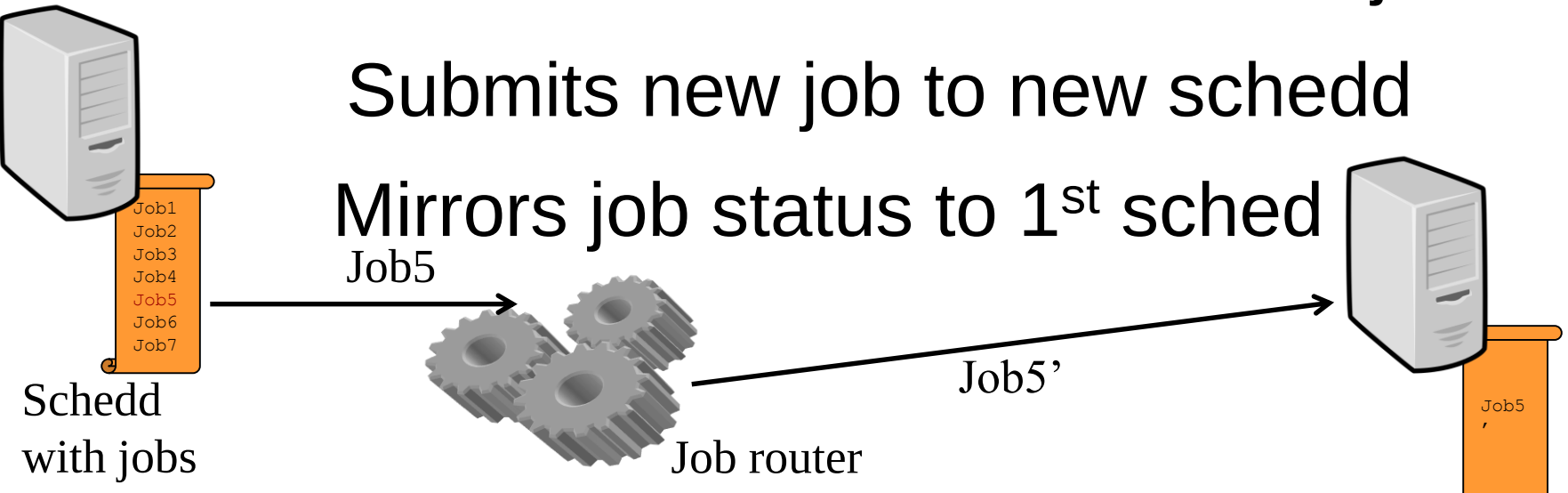
JobRouter is a condor daemon…

Grabs jobs from schedd, “I’ve got this one”

Uses rules to transform into new job

Submits new job to new schedd

Mirrors job status to 1st sched



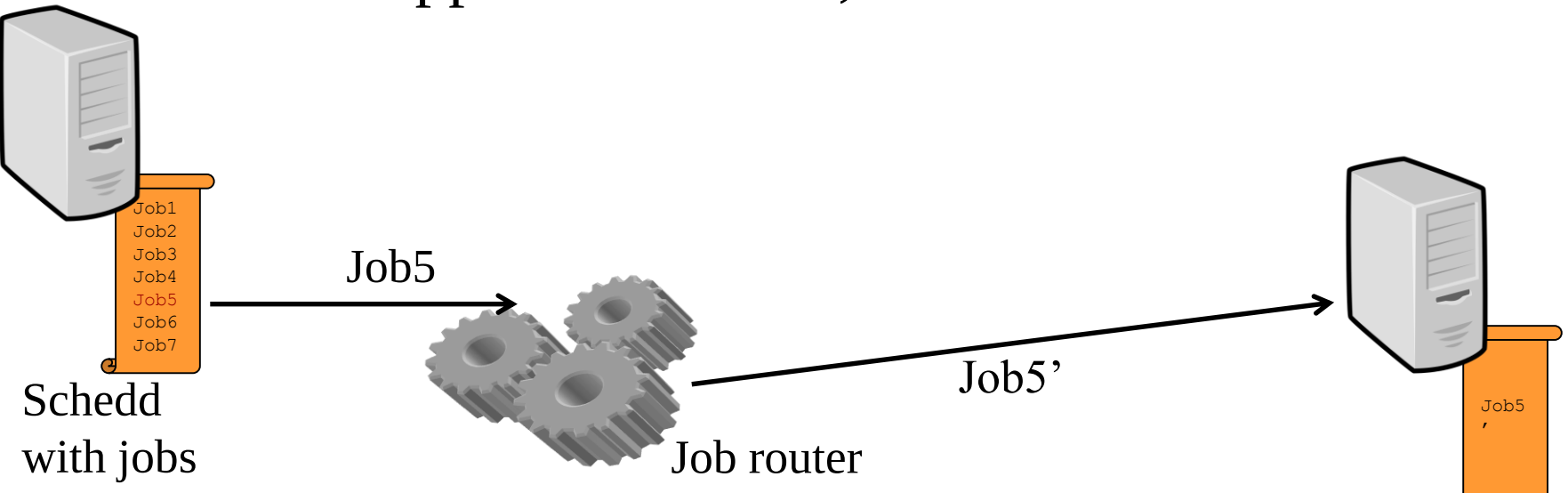
Job Router: pros

Works over slow WAN

Submitters don't need to know their jobs are moved

Easy for admin to mutate previously submitted jobs

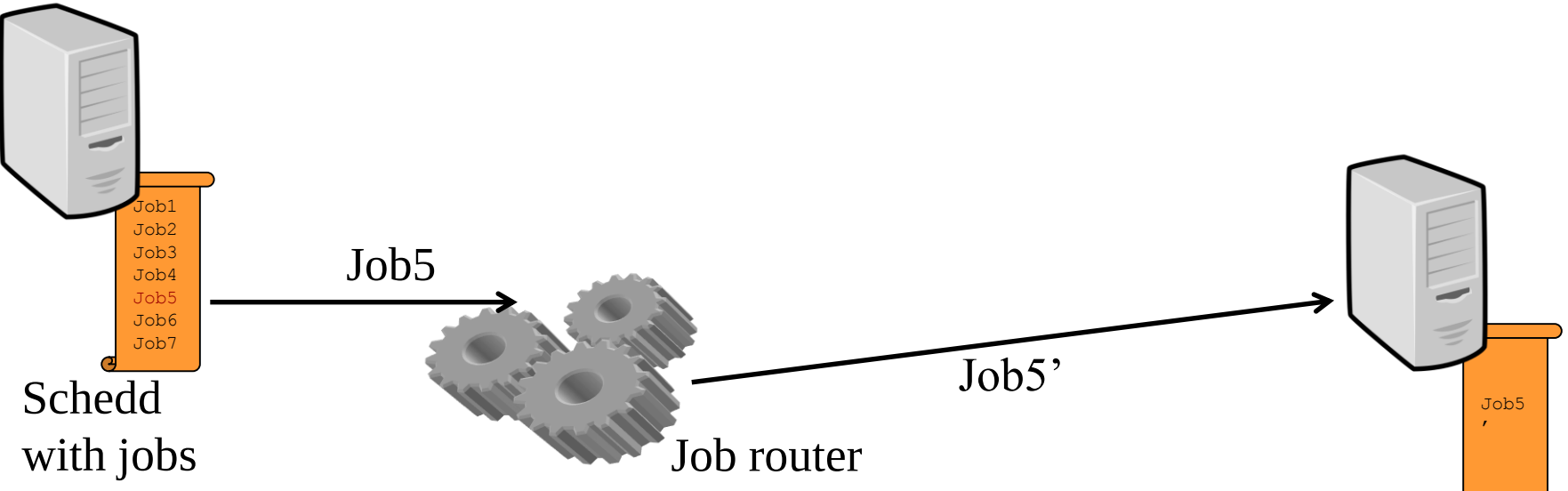
Job router supports > 1 route, can timeout and resubmit



Job Router: cons

Requires GSI, SSL, for remote auth

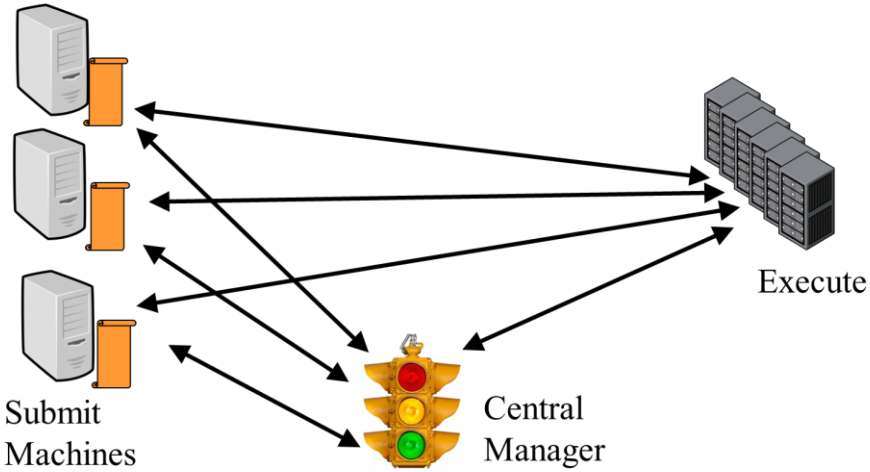
Early binding – Jobs can wait ‘in line’ when startds idle

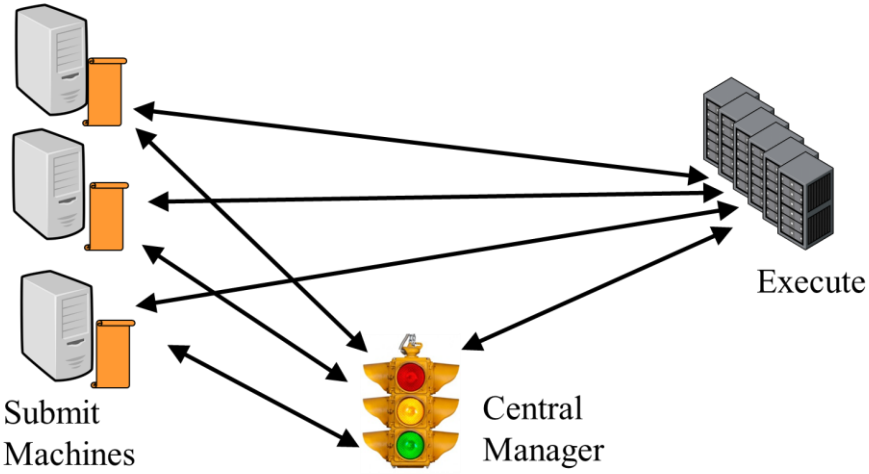


Glidein, HobbleIn, the idea

Like merging, but dynamic

Create Overlay pool

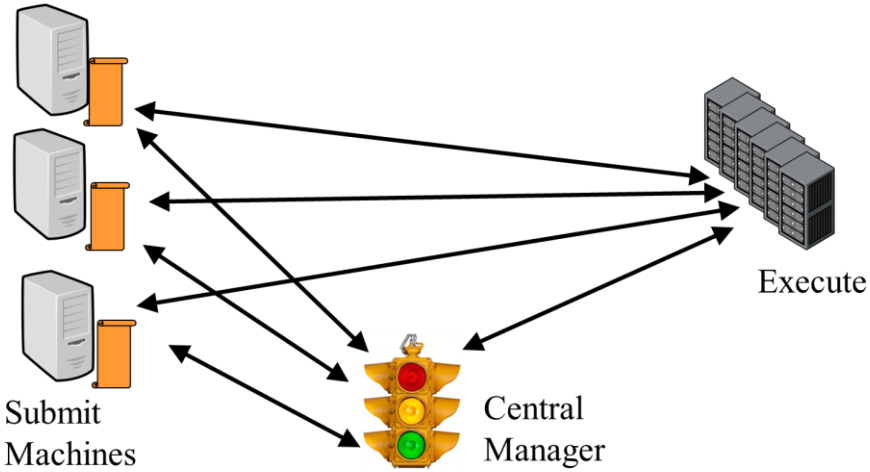


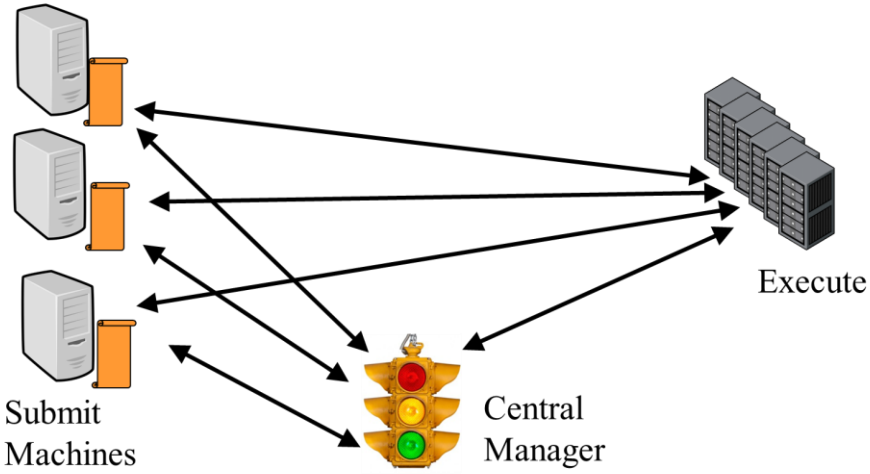


Glidein, HobbleIn, the idea

Like merging, but dynamic

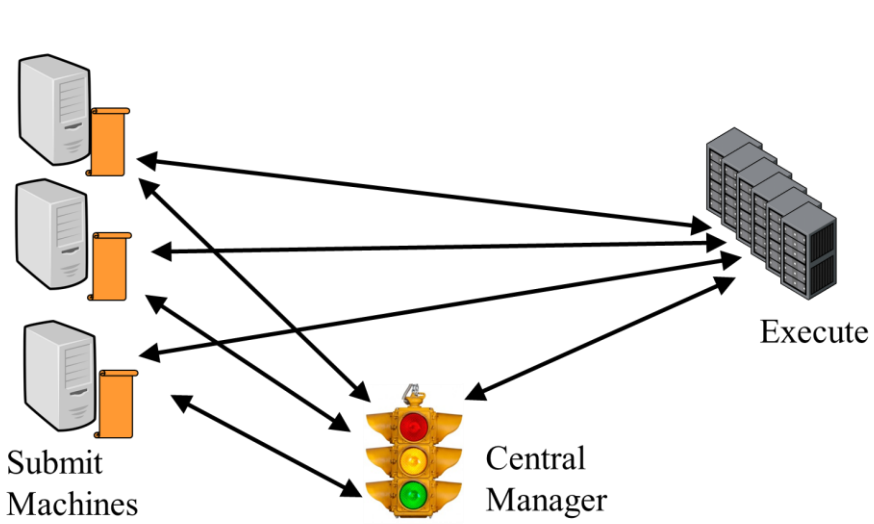
Submit jobs, startds reporting home

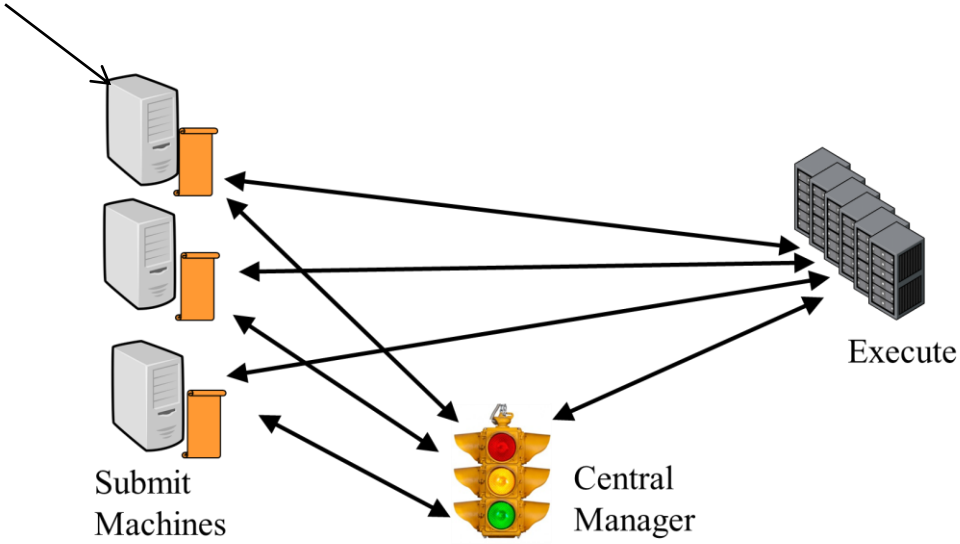




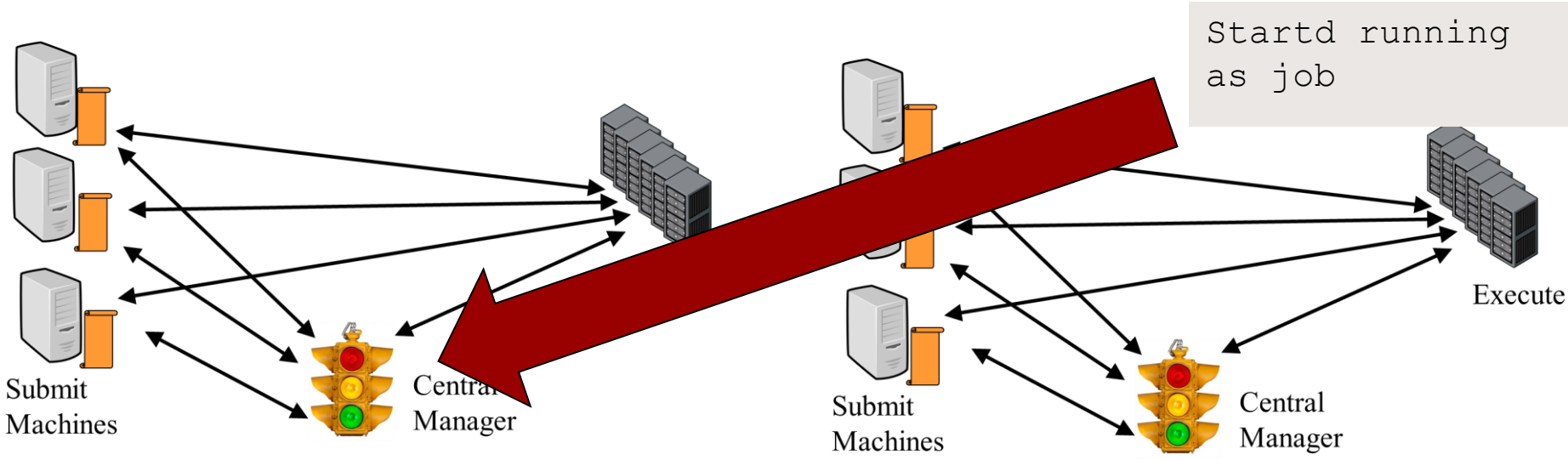
Glidein, HobbleIn

```
Executable = condor_master
Arguments = -f -t
Output = out
Queue 100
```





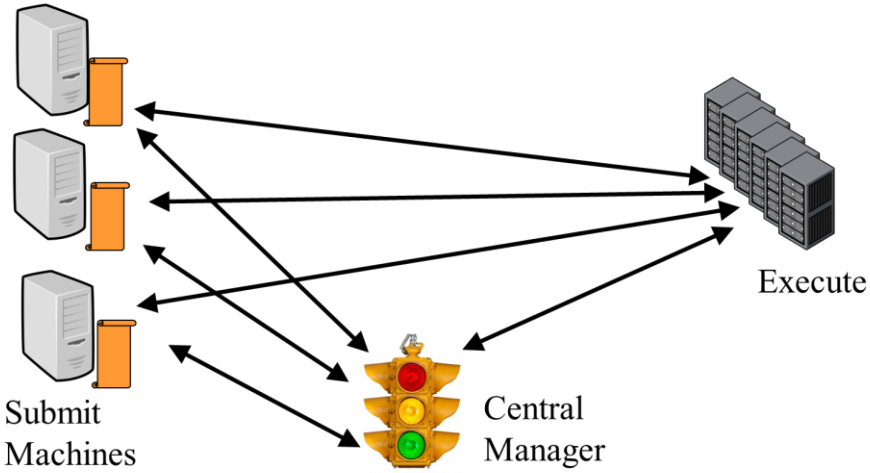
Glidein, HobbleIn

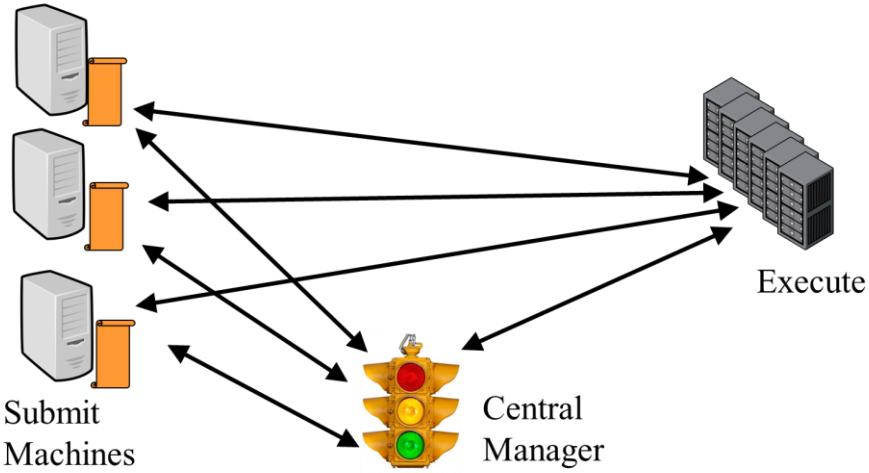


Glidein, HobbleIn, pros:

Late binding

Easy to merge lots of pools



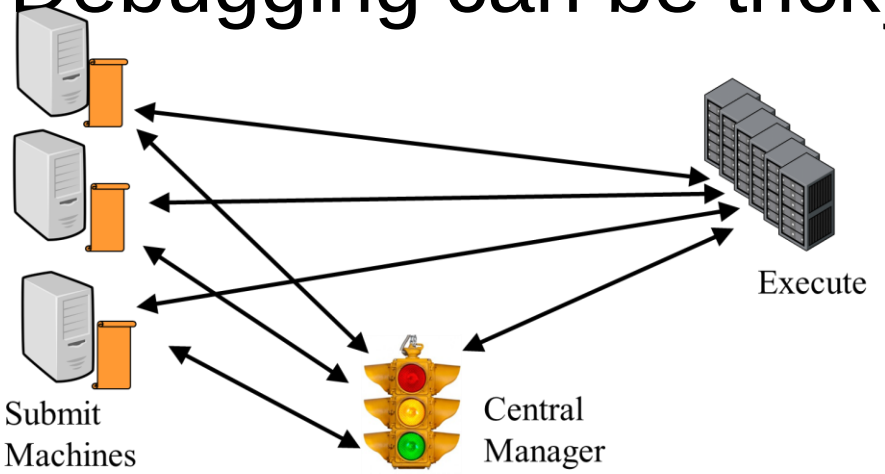


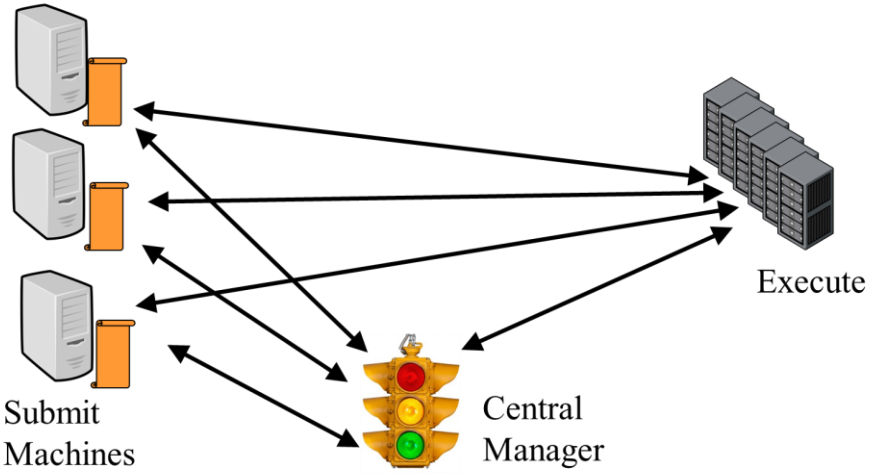
Glidein, HobbleIn, cons:

Startd runs as non-root, some feature gone

Need good networking

Debugging can be tricky





Annex

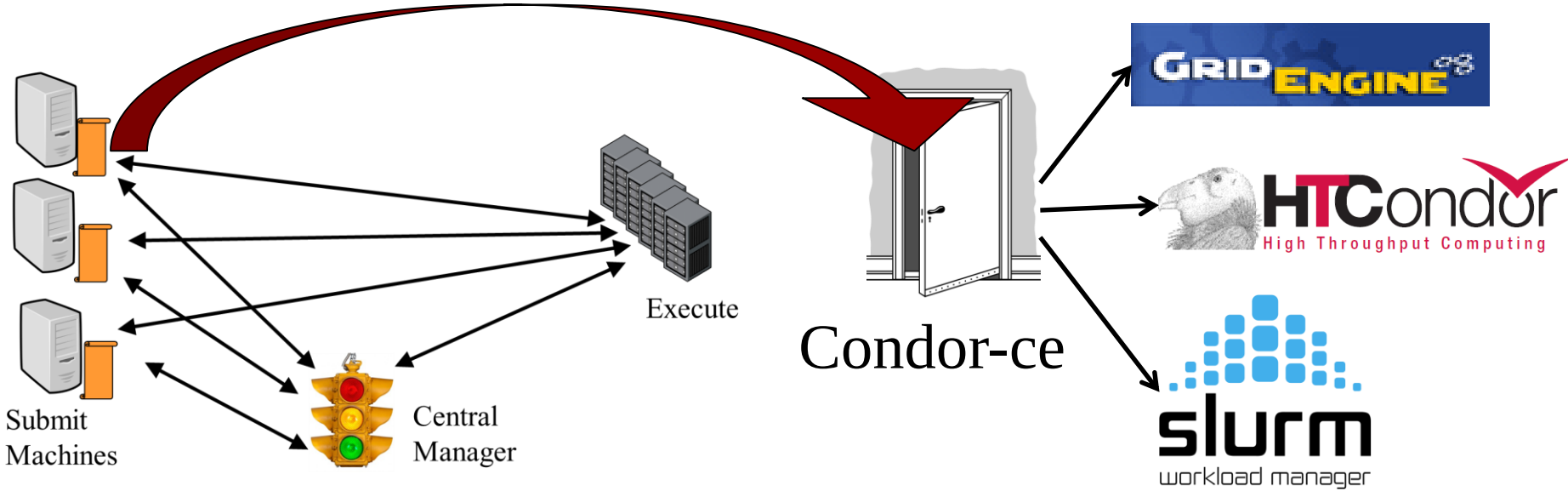
- What if we could:
  - Pay for a new standalone pool in AWS
  - Flock to that pool
- condor_annex makes this easy

GlideinWMS

- Implementation of Glidein idea for OSG
- Very sophisticated
- Needs GSI security
- Requires lot of work to setup, run

Condor-CE

- Combines condor-c, job router
- “Door” to non-condor remote pools



Conclusion