Nelson

用函數方式設計系統

Adelbert Chang
@adelbertchang

Anil Madhavapeddy @ Haskell 2014

We build applications in a **safe**, **compositional style** using functional programming...

Anil Madhavapeddy @ Haskell 2014

We build applications in a **safe**, **compositional style** using functional programming...

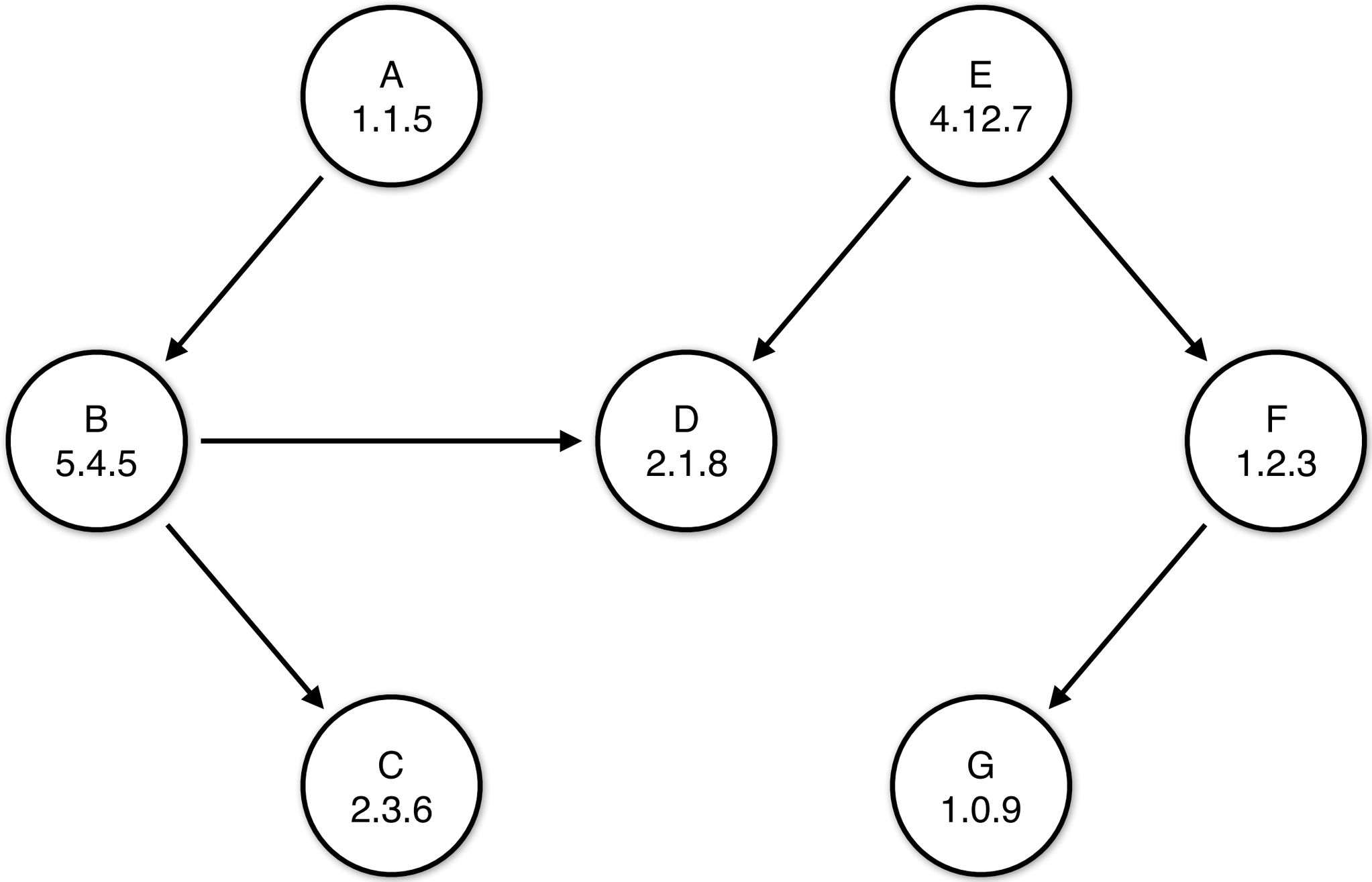
...and then surround it in **15 million lines of unsafe code** to interact with the outside world.

Anil Madhavapeddy @ Haskell 2014

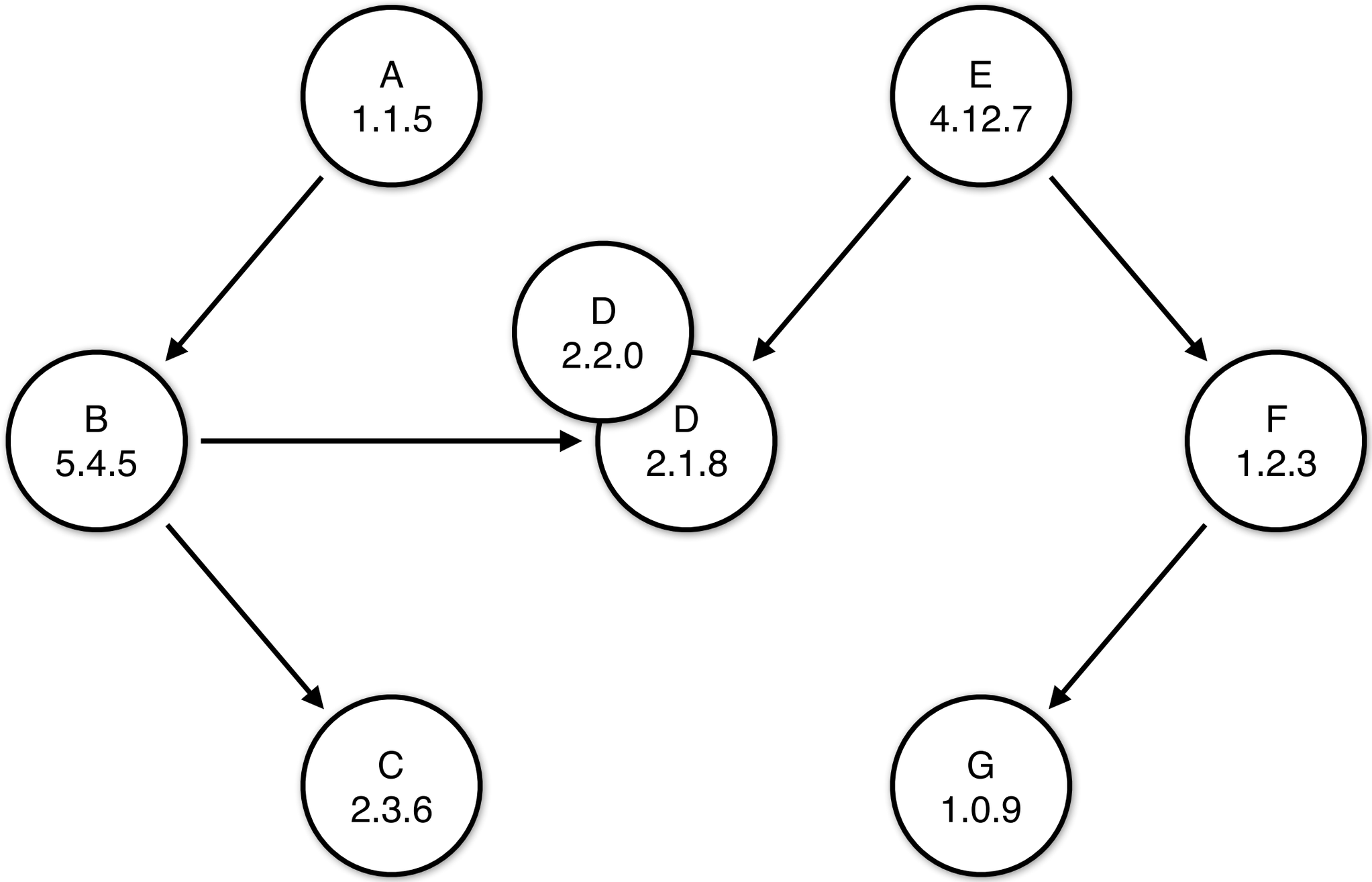
We build applications in a **safe**, **compositional style** using functional programming...

...and then surround it in **mutable, complex infrastructure** to deploy it.

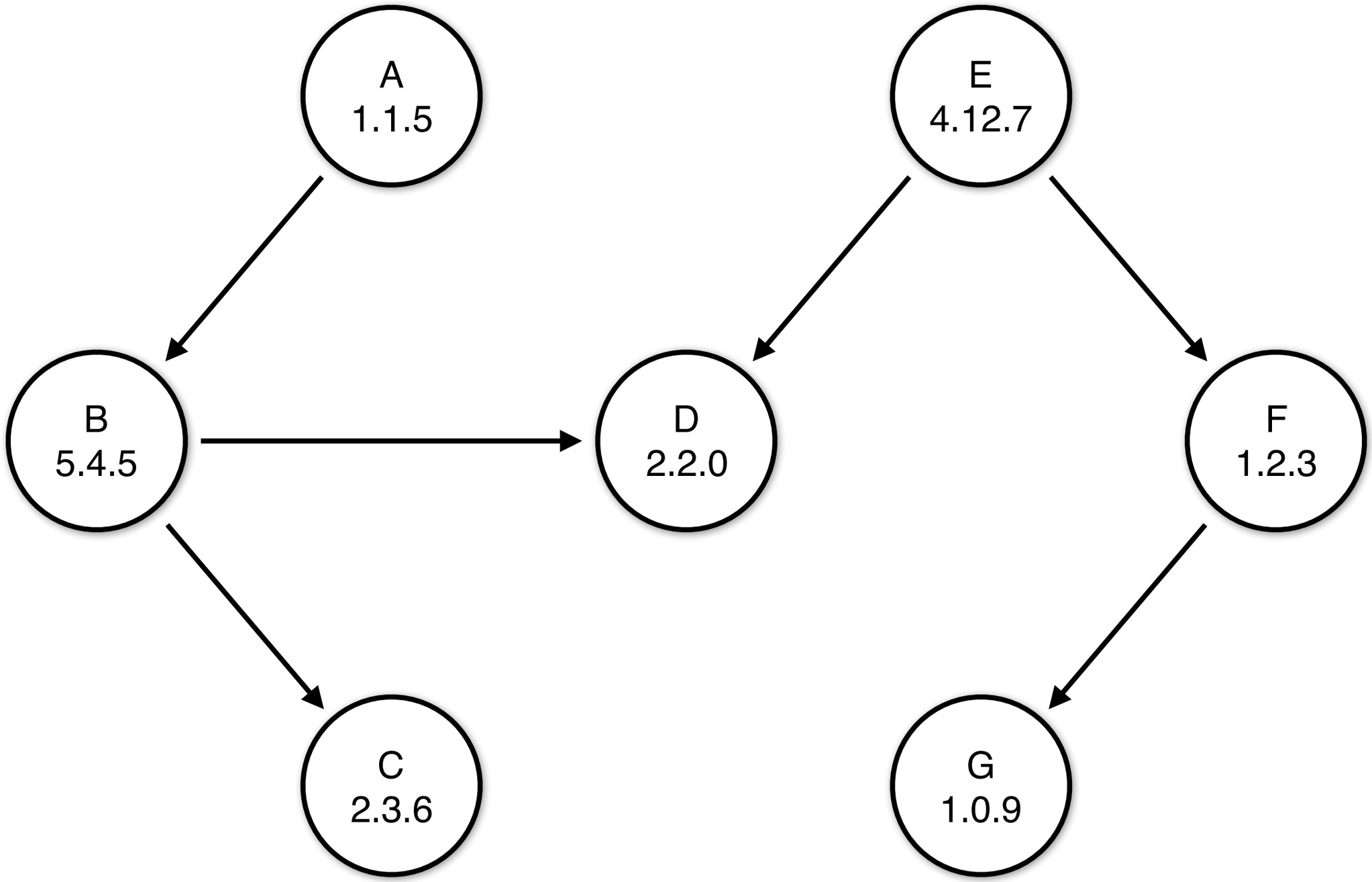
Mutable deployments



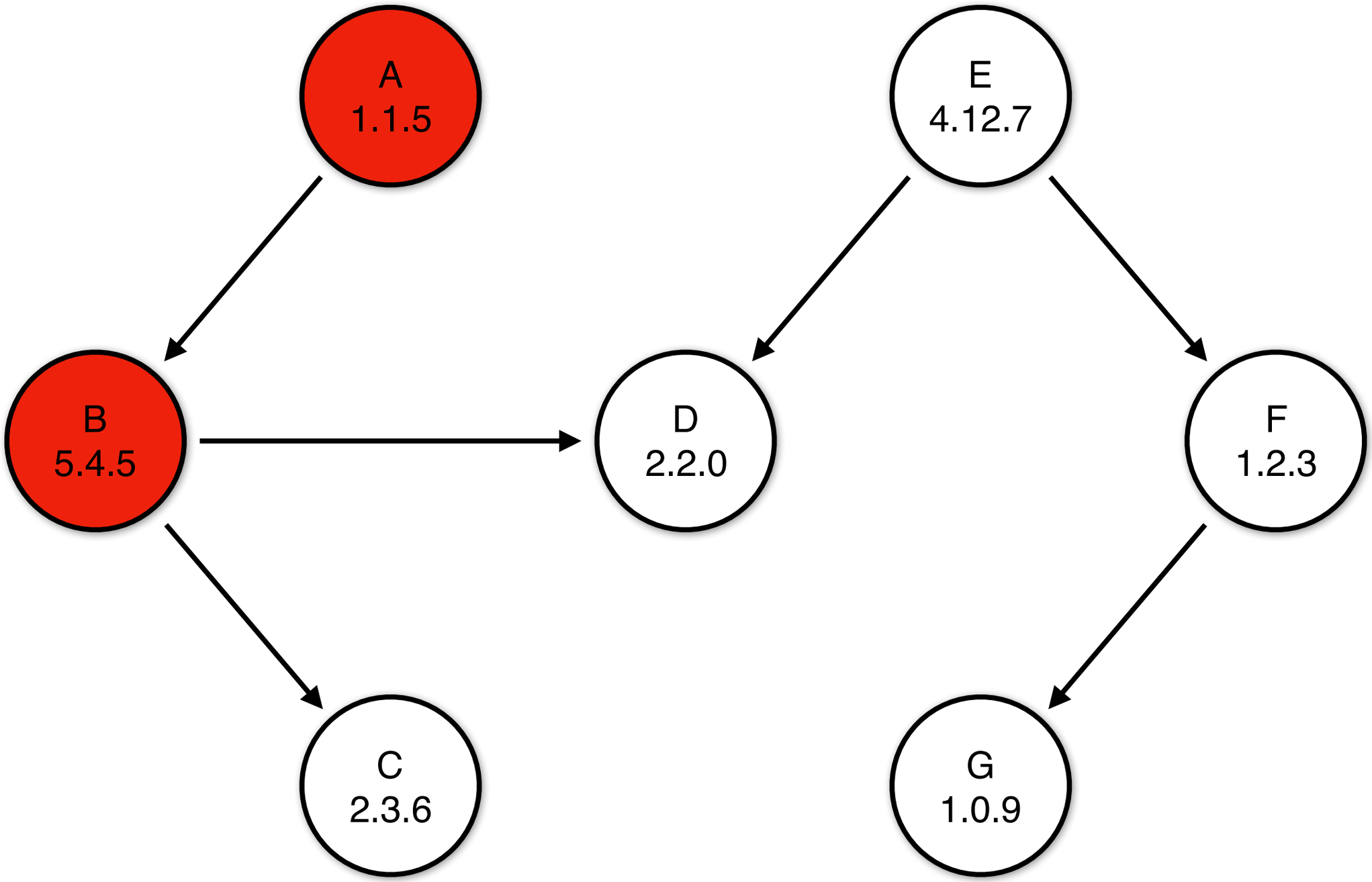
Mutable deployments



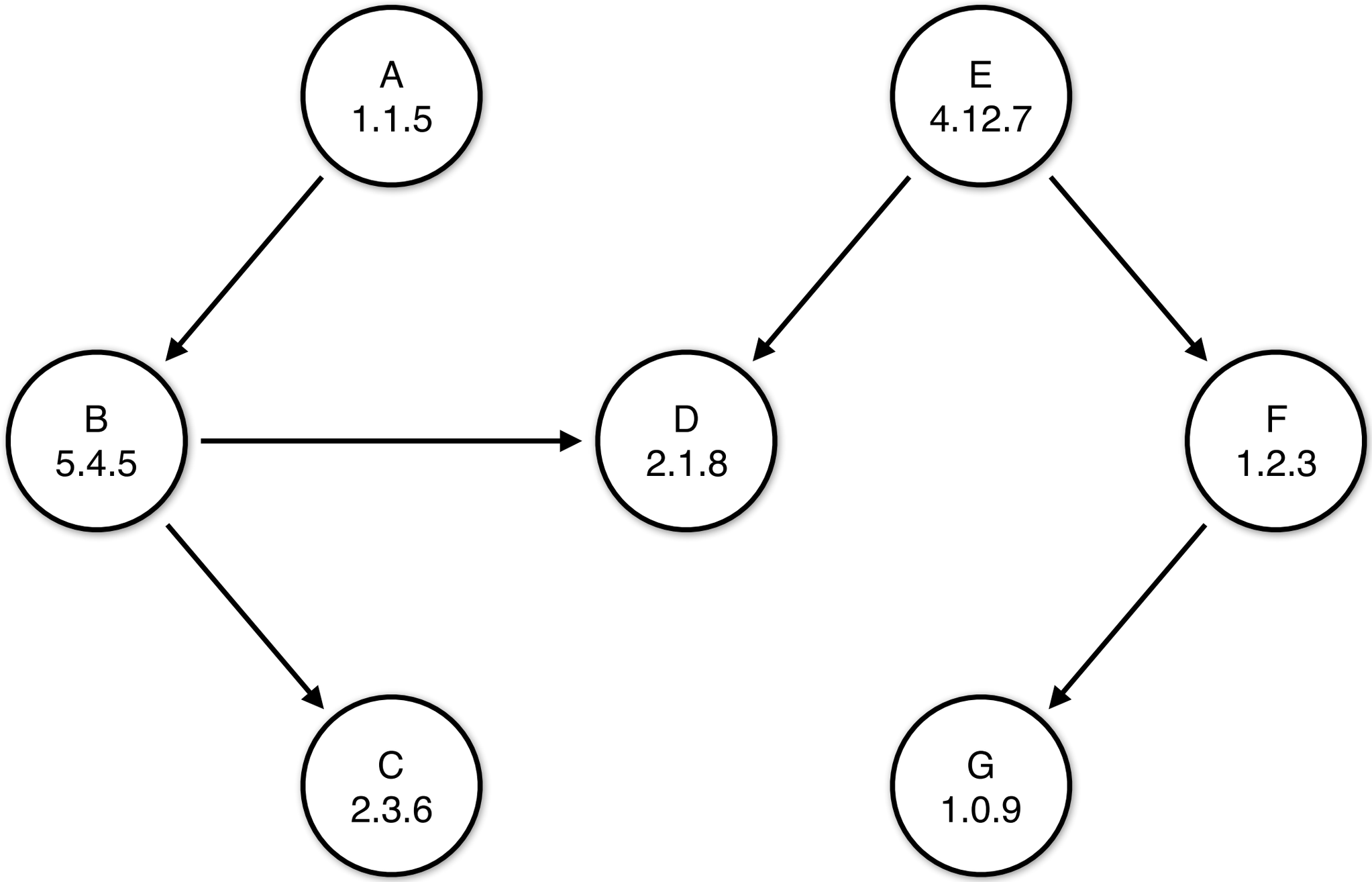
Mutable deployments



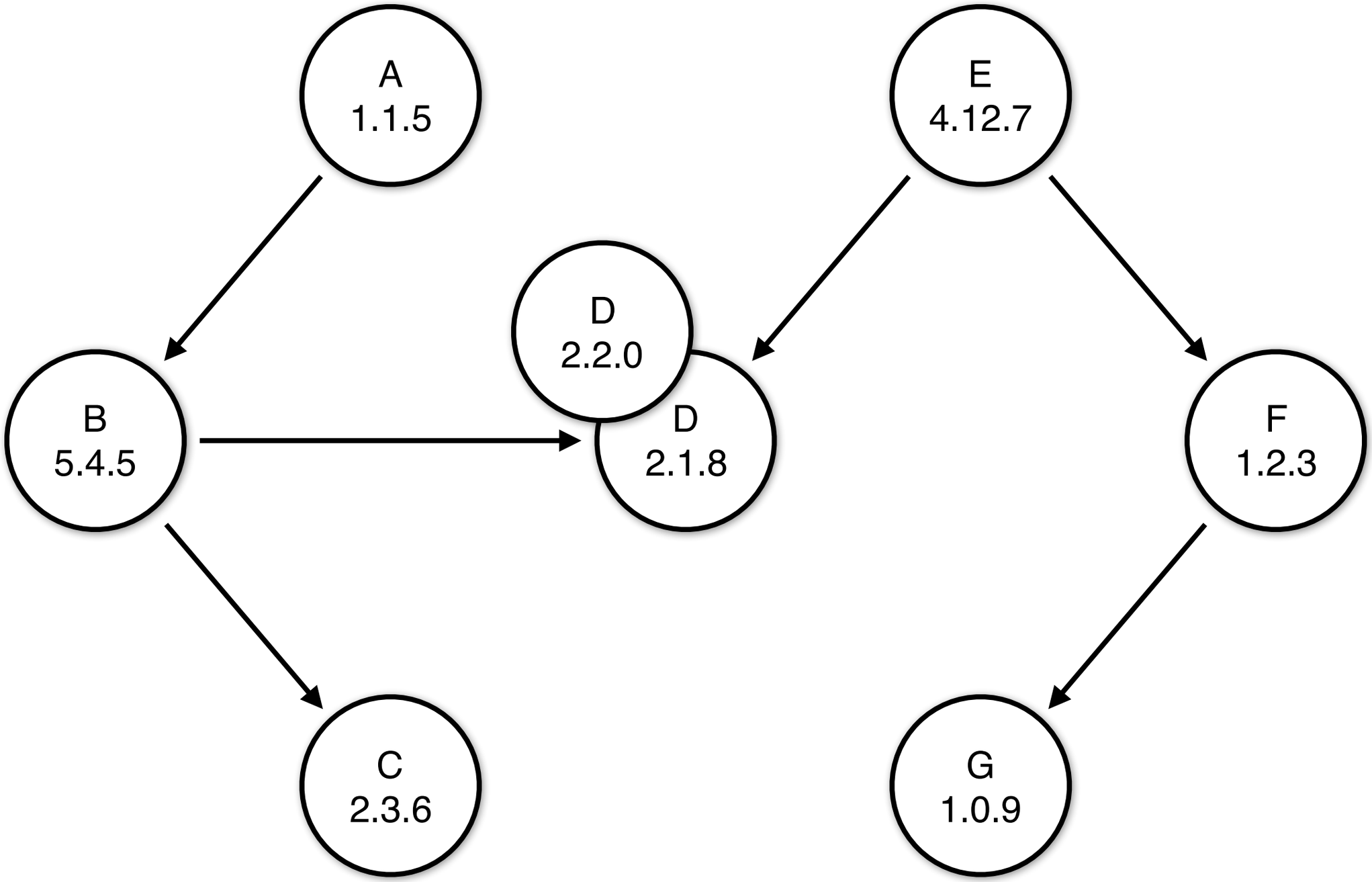
Mutable deployments



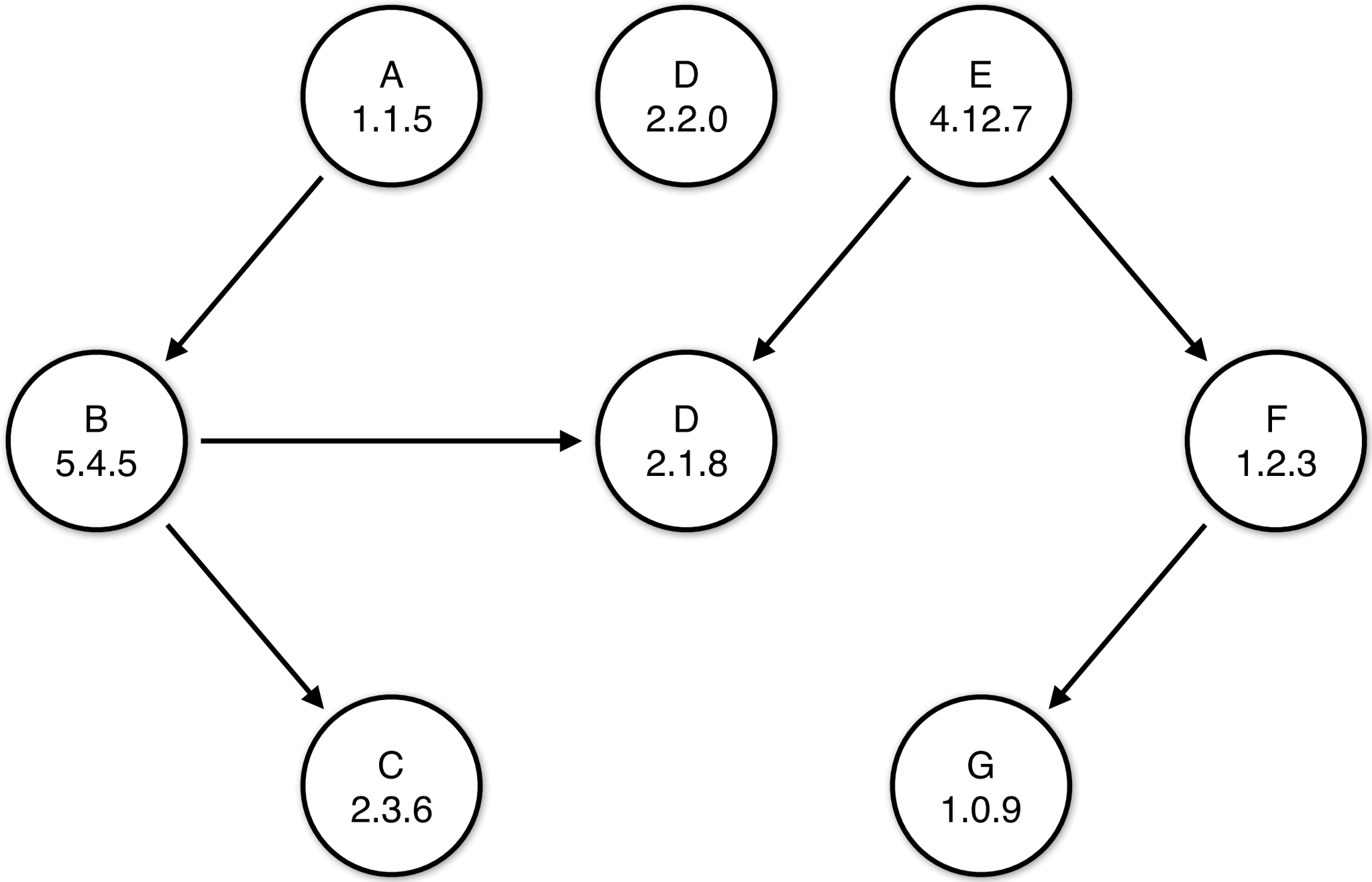
Immutable deployments



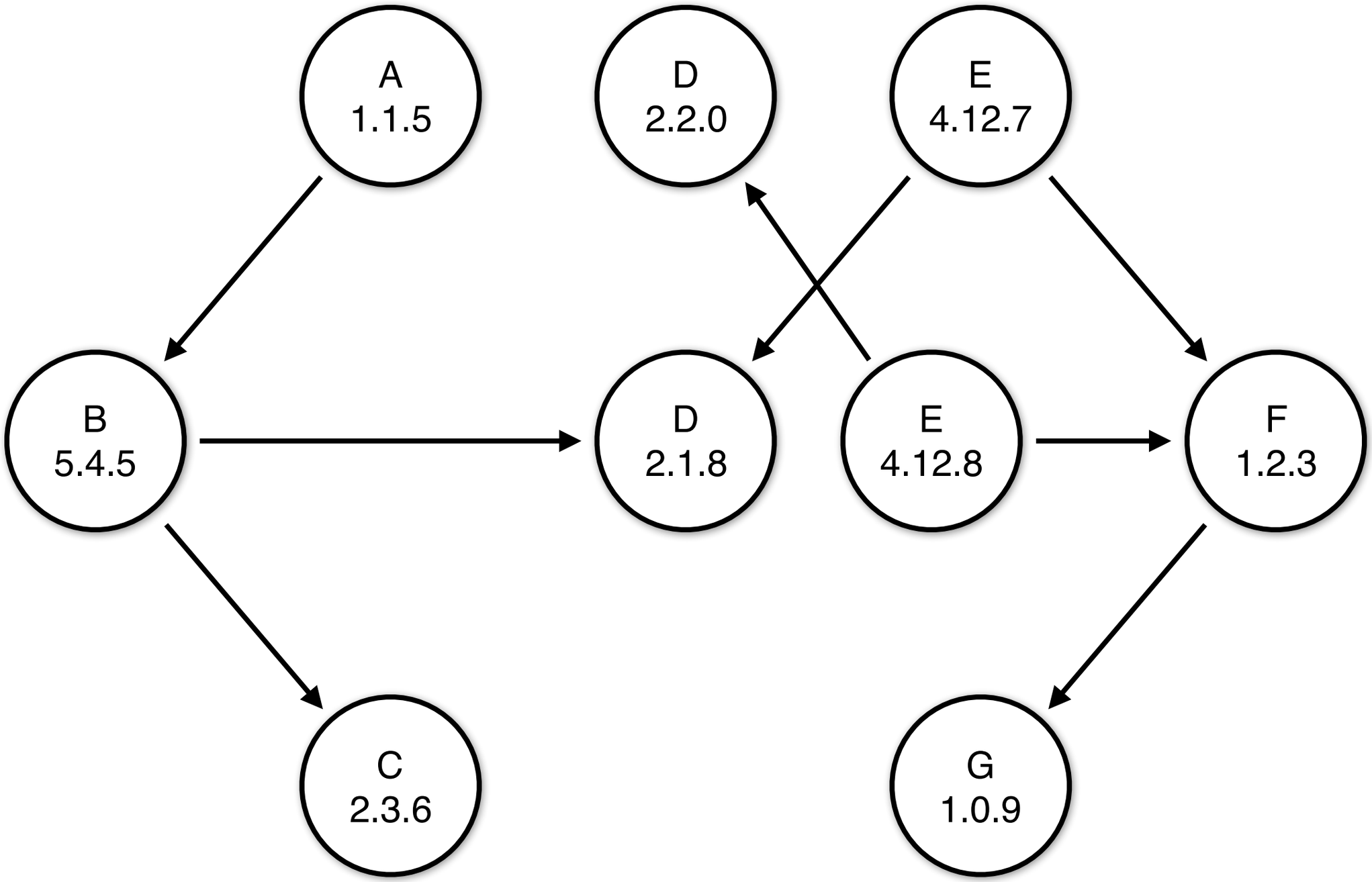
Immutable deployments



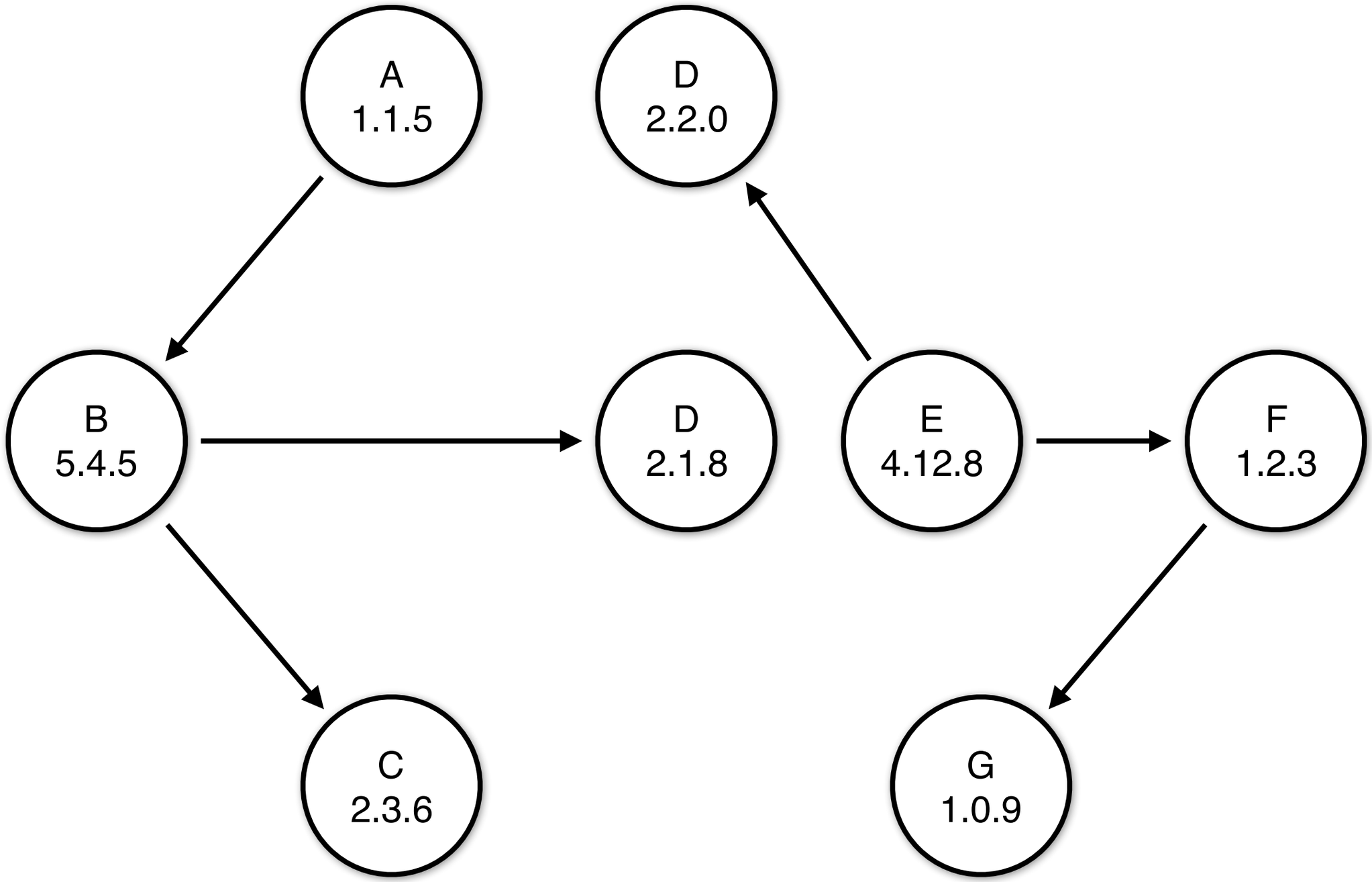
Immutable deployments



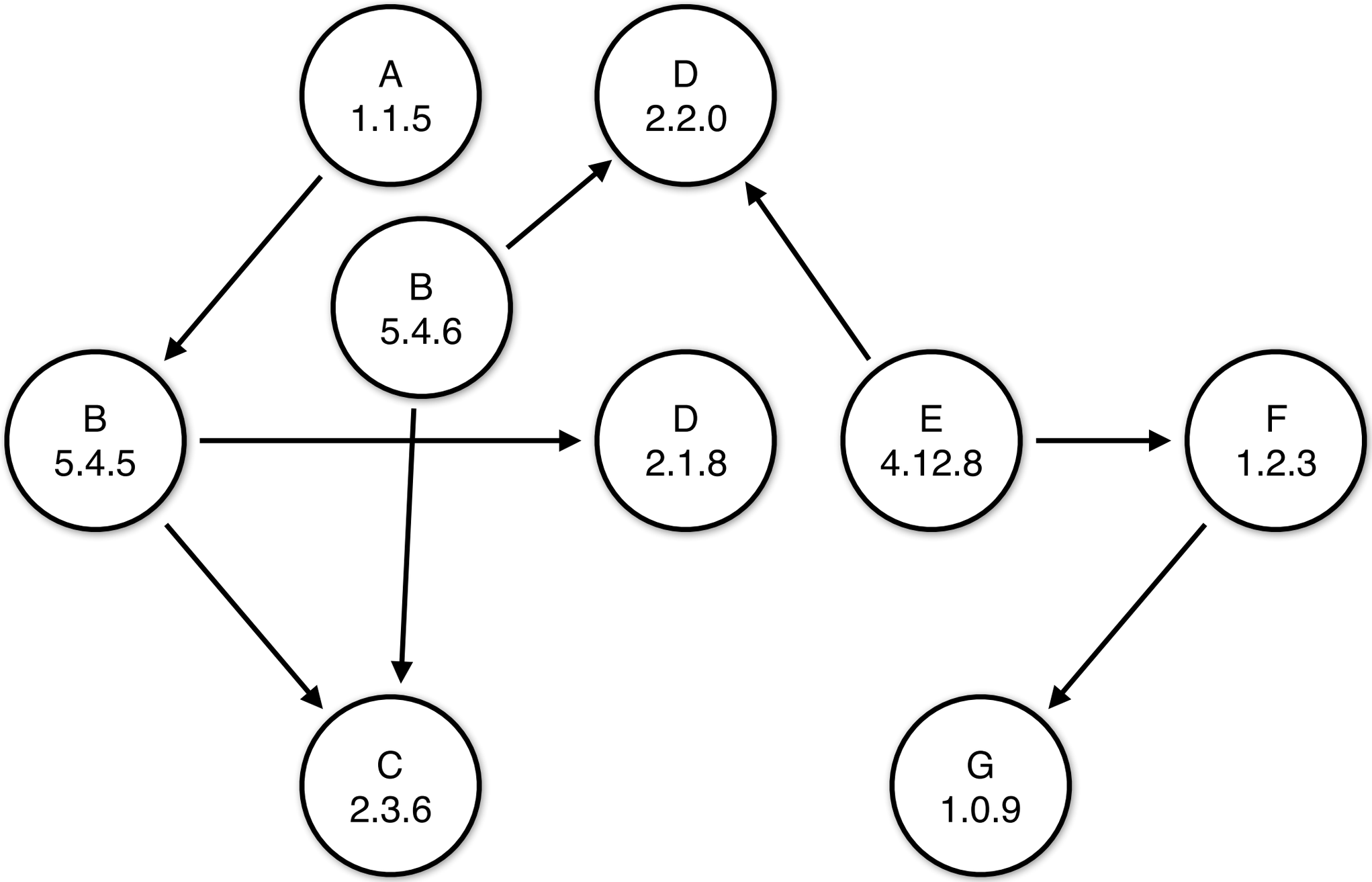
Immutable deployments



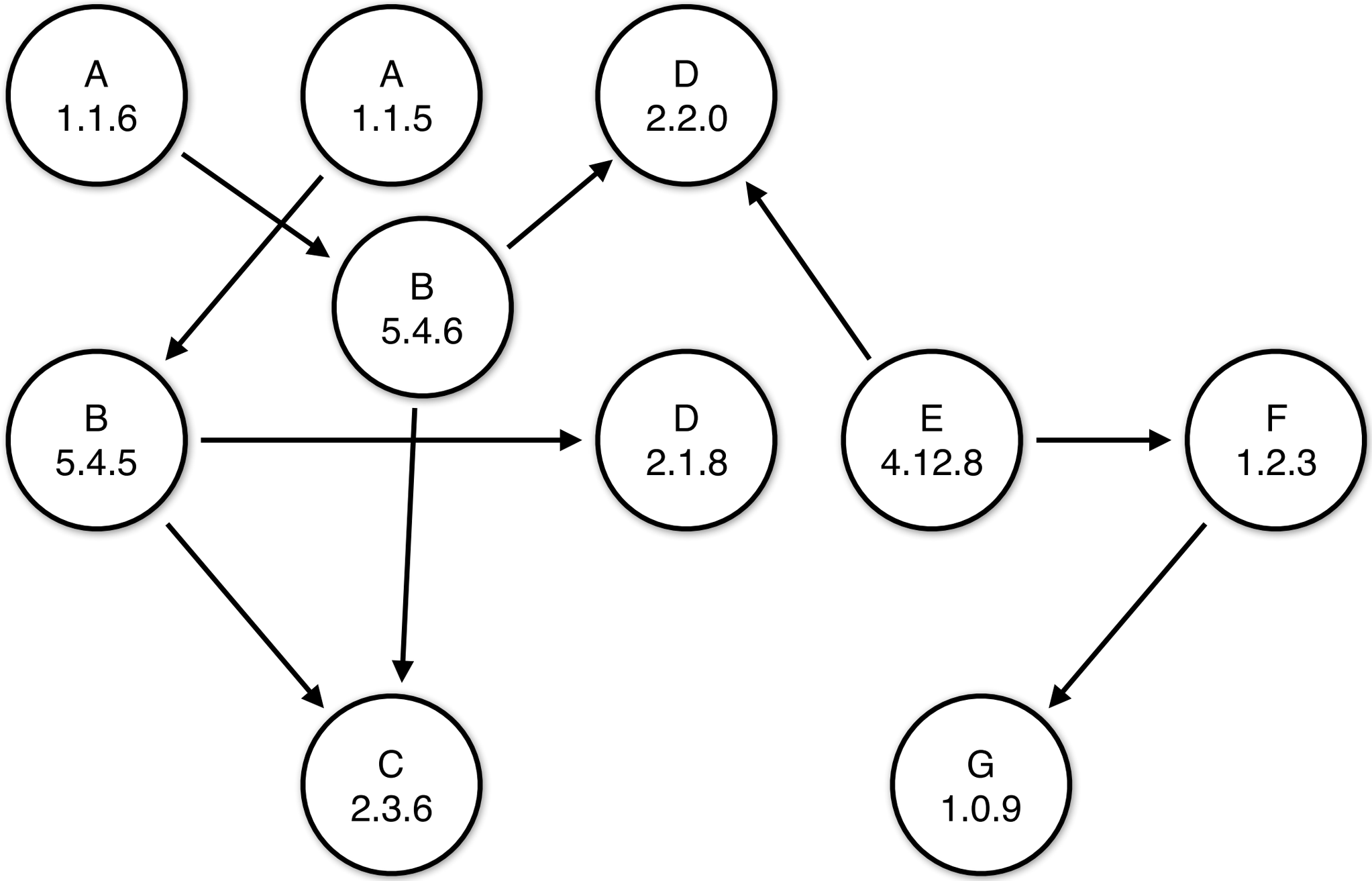
Immutable deployments



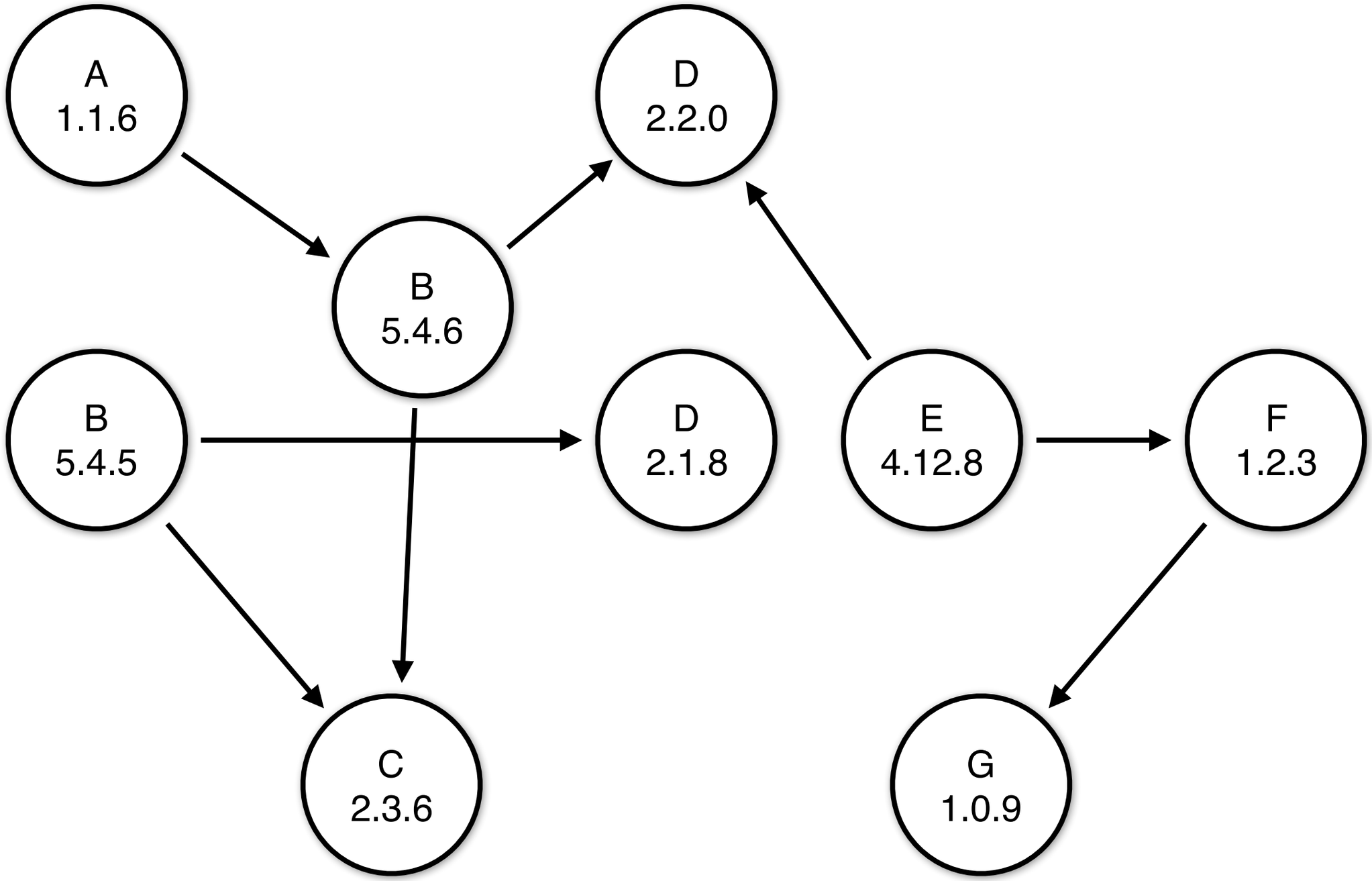
Immutable deployments



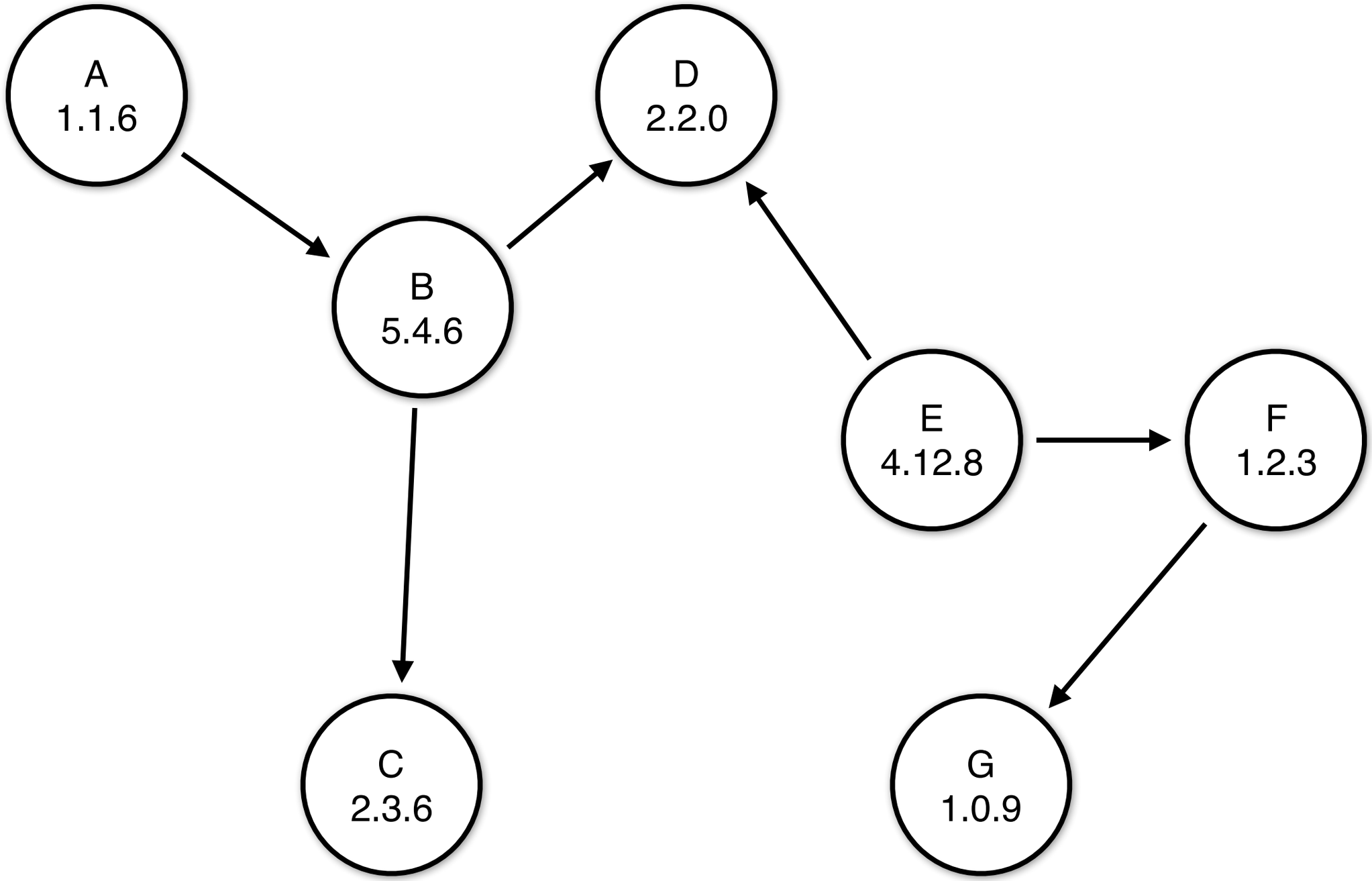
Immutable deployments



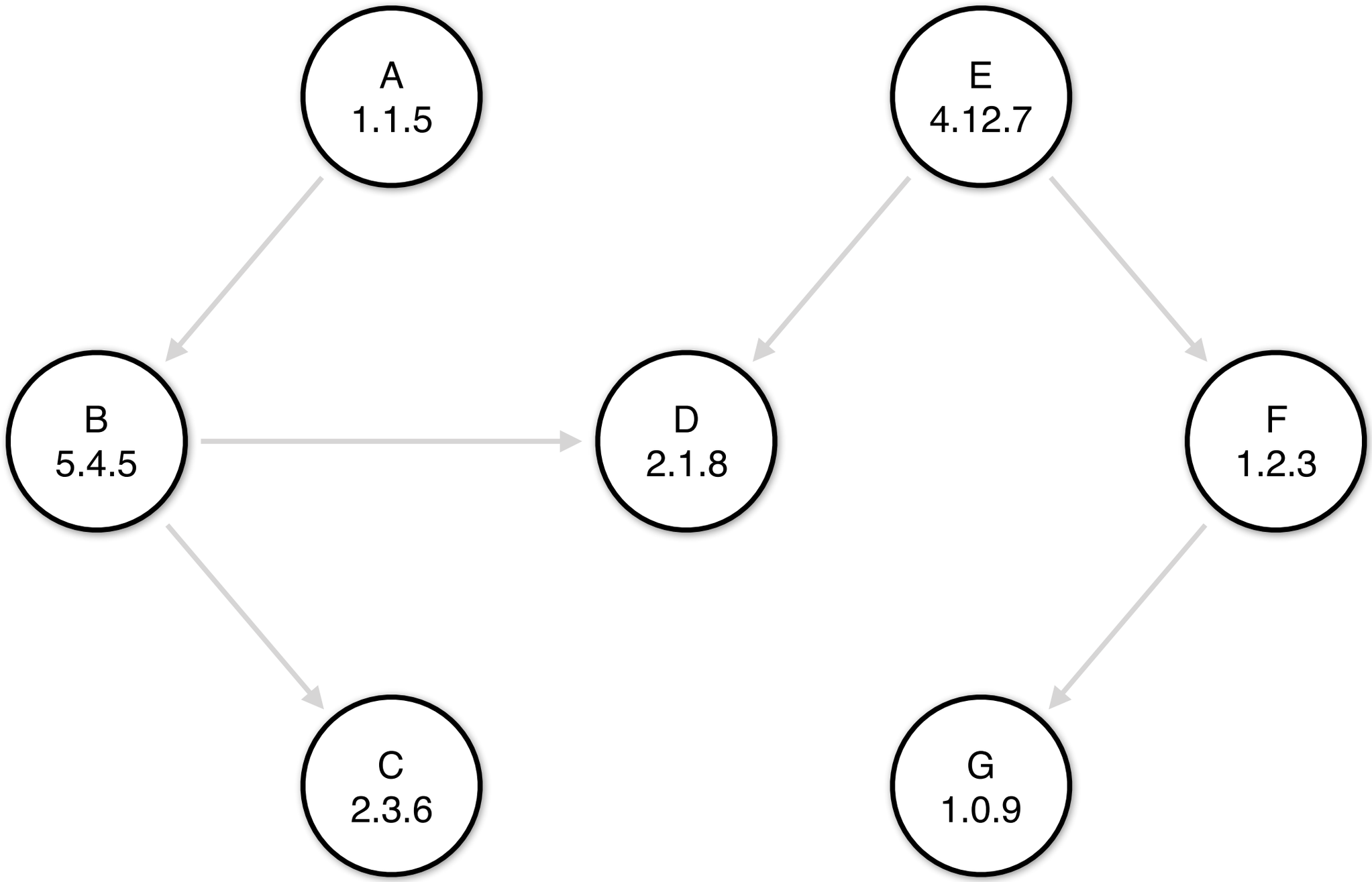
Immutable deployments



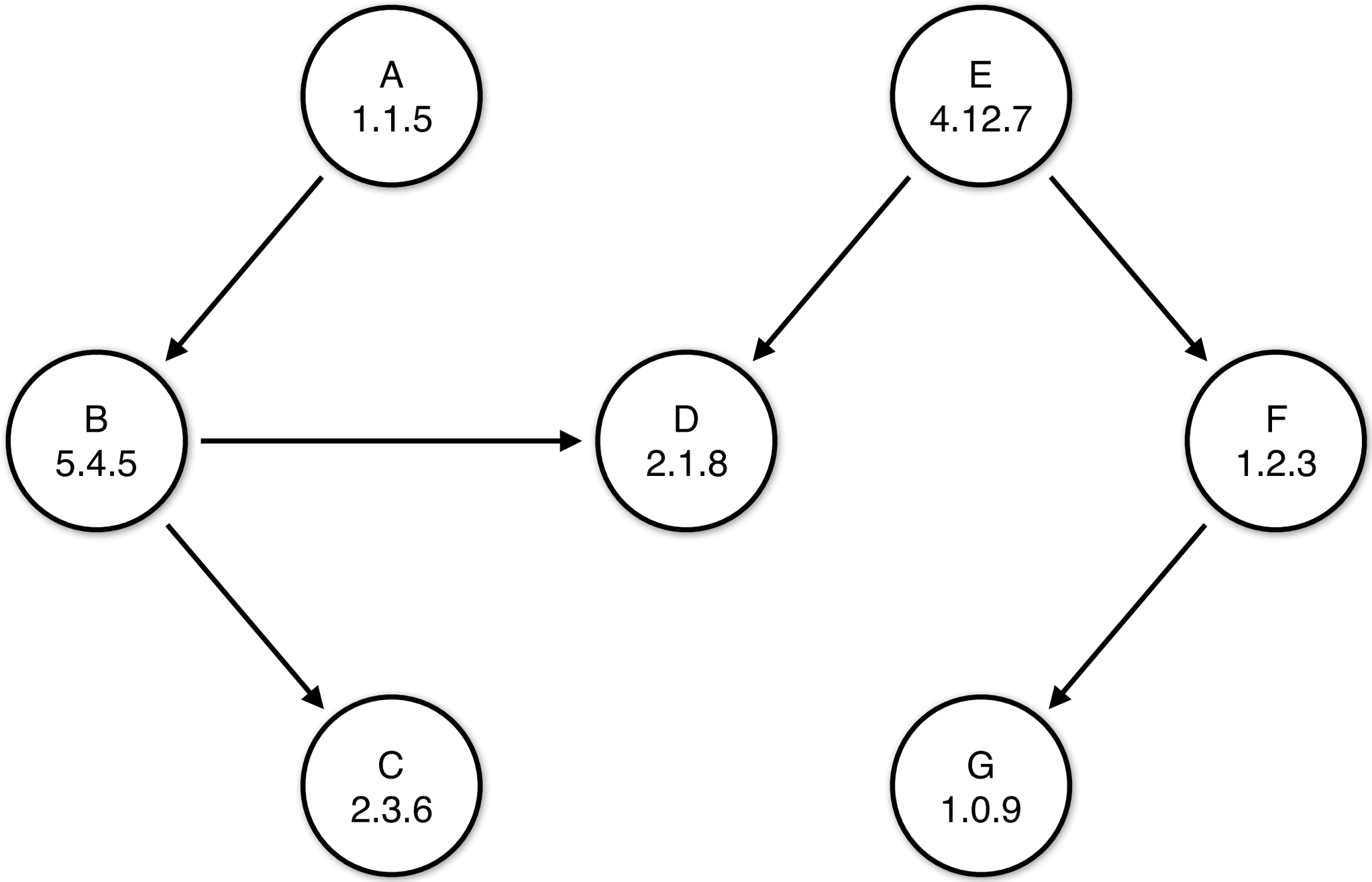
Immutable deployments



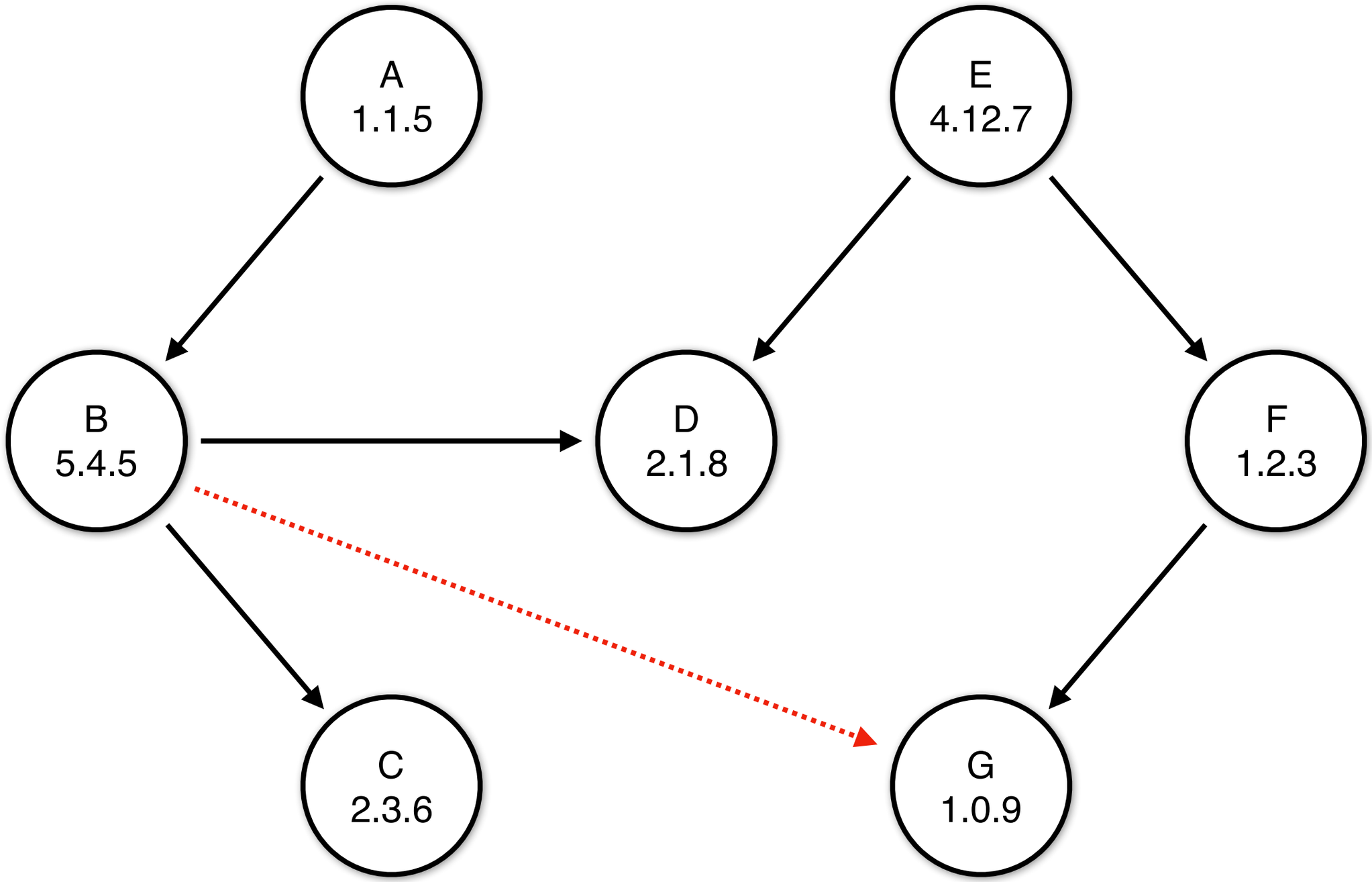
Reify the call graph



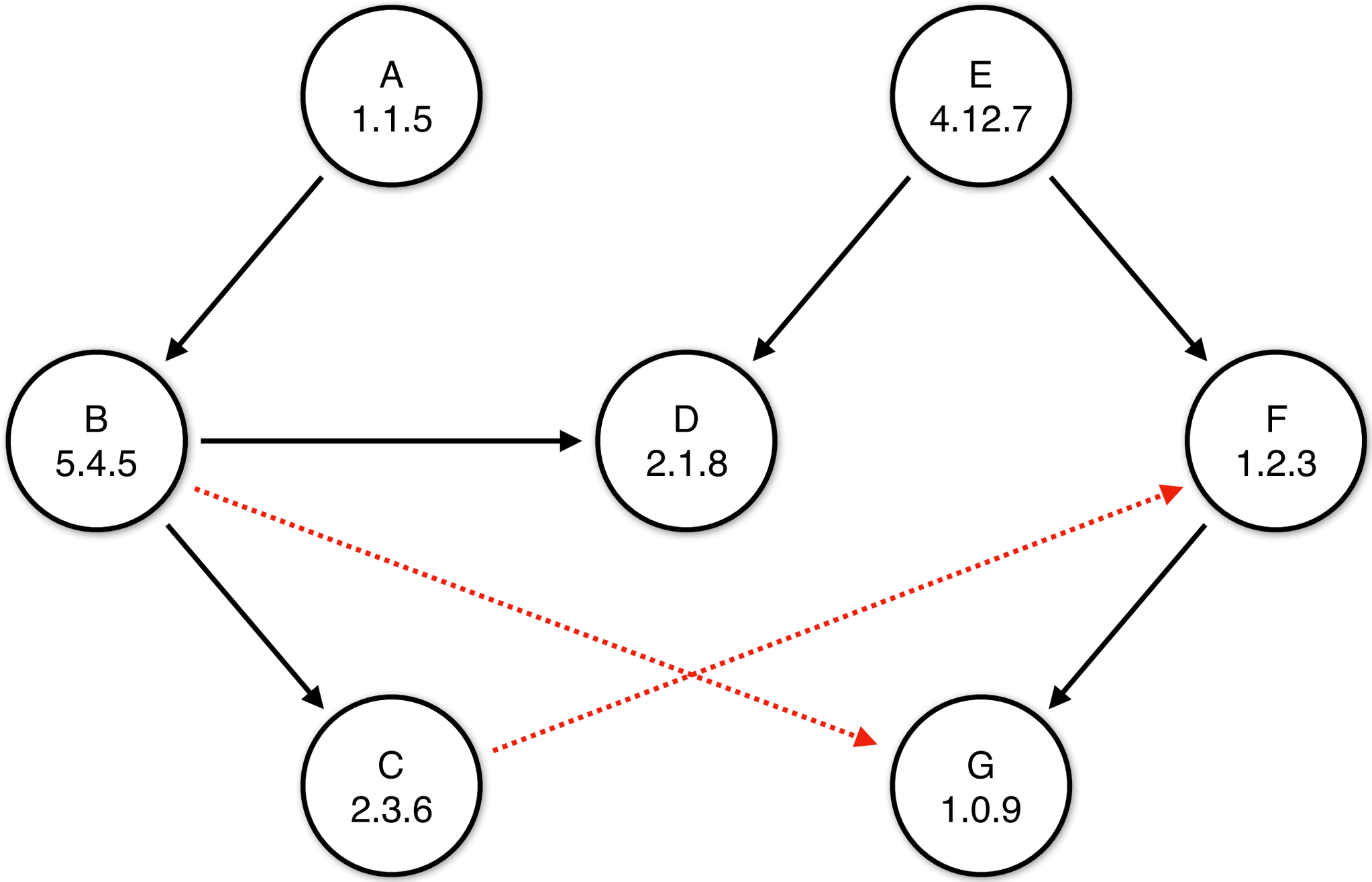
Reify the call graph



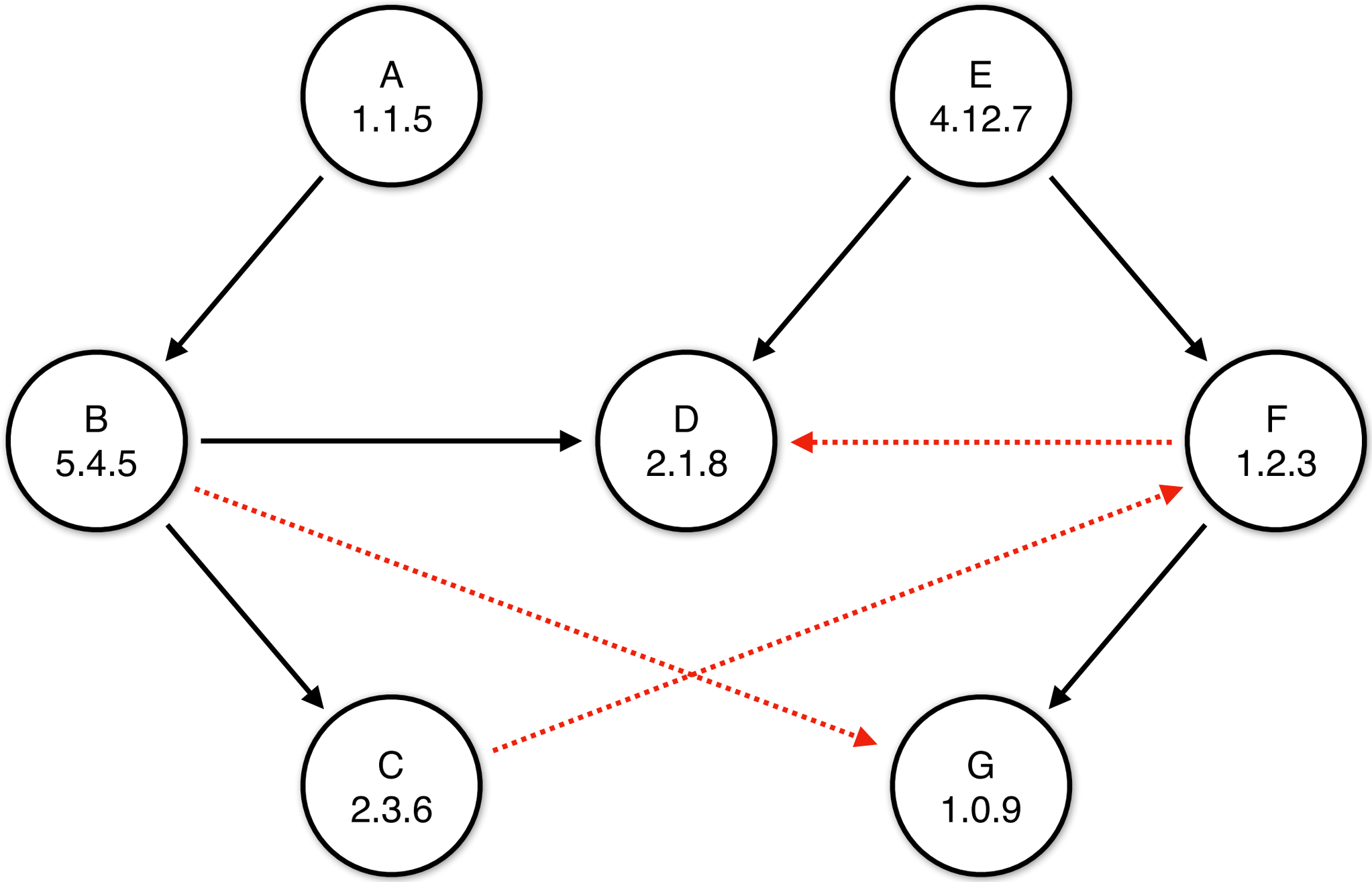
Reify the call graph



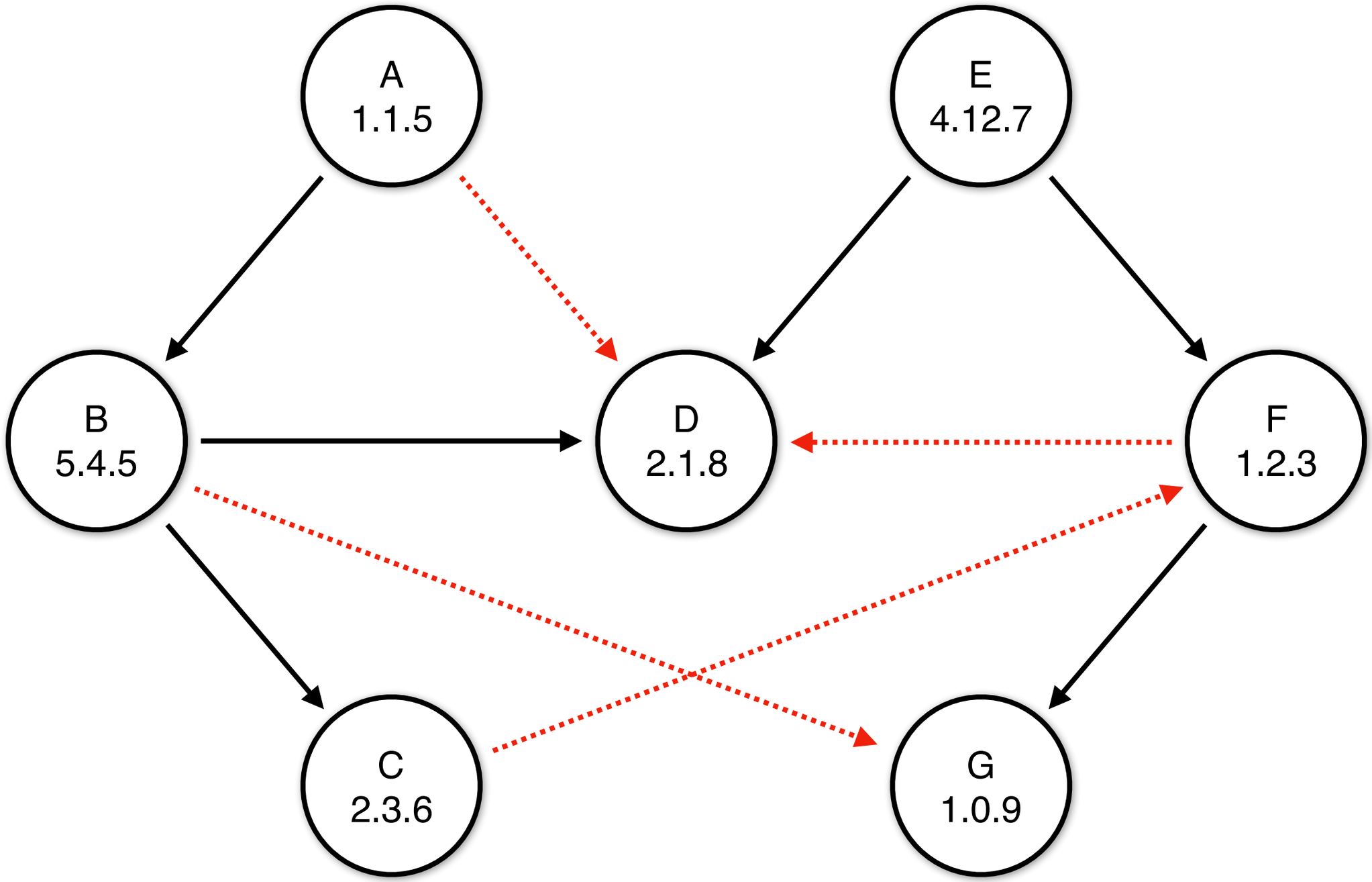
Reify the call graph



Reify the call graph



Reify the call graph

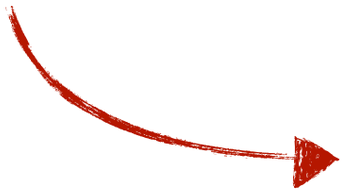



N
W
E
S

```yaml
name: foo
version: 0.3.19
output:
  kind: container
  image: units/foo:0.3.19
```

```
                                    ption of foo
        - default->9000/http
      dependencies:
        - ref: bar@2.1
        - ref: baz@2.3

plans:
  - name: default

namespaces:
  - name: production
    units:
      - ref: foo
        plans:
          - default
```

Name and version



```
name: foo
version: 0.3.19
output:
  kind: container
  image: units/foo:0.3.19
```

```
                                    ption of foo
        - default->9000/http
      dependencies:
        - ref: bar@2.1
        - ref: baz@2.3

plans:
  - name: default

namespaces:
  - name: production
    units:
      - ref: foo
        plans:
          - default
```

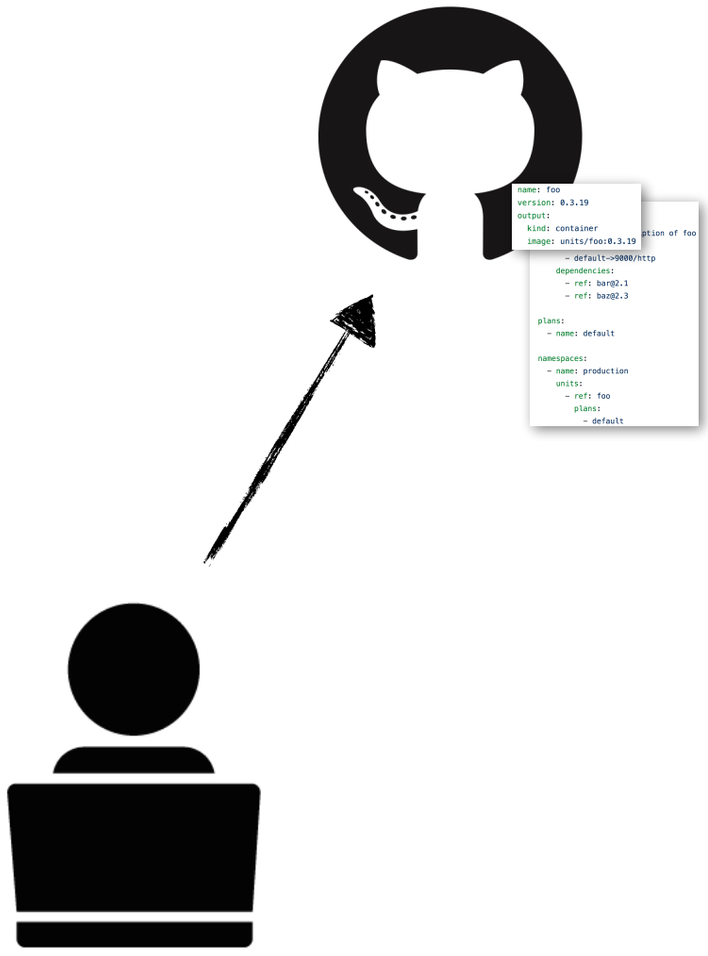
Name and version

```
name: foo
version: 0.3.19
output:
  kind: container
  image: units/foo:0.3.19
```

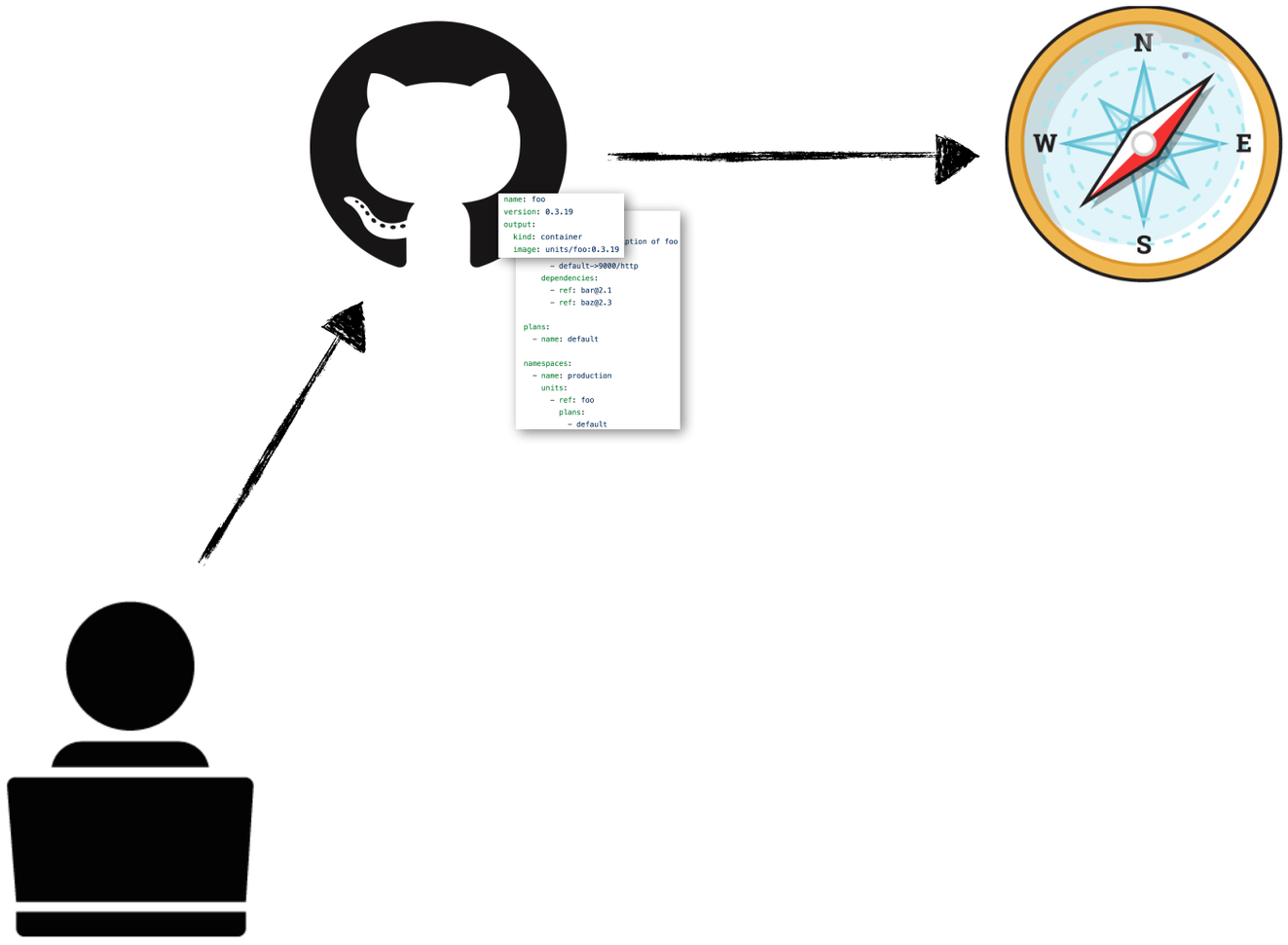
```
                                    ption of foo
        - default->9000/http
      dependencies:
        - ref: bar@2.1
        - ref: baz@2.3

plans:
  - name: default

namespaces:
  - name: production
    units:
      - ref: foo
        plans:
          - default
```

Dependencies

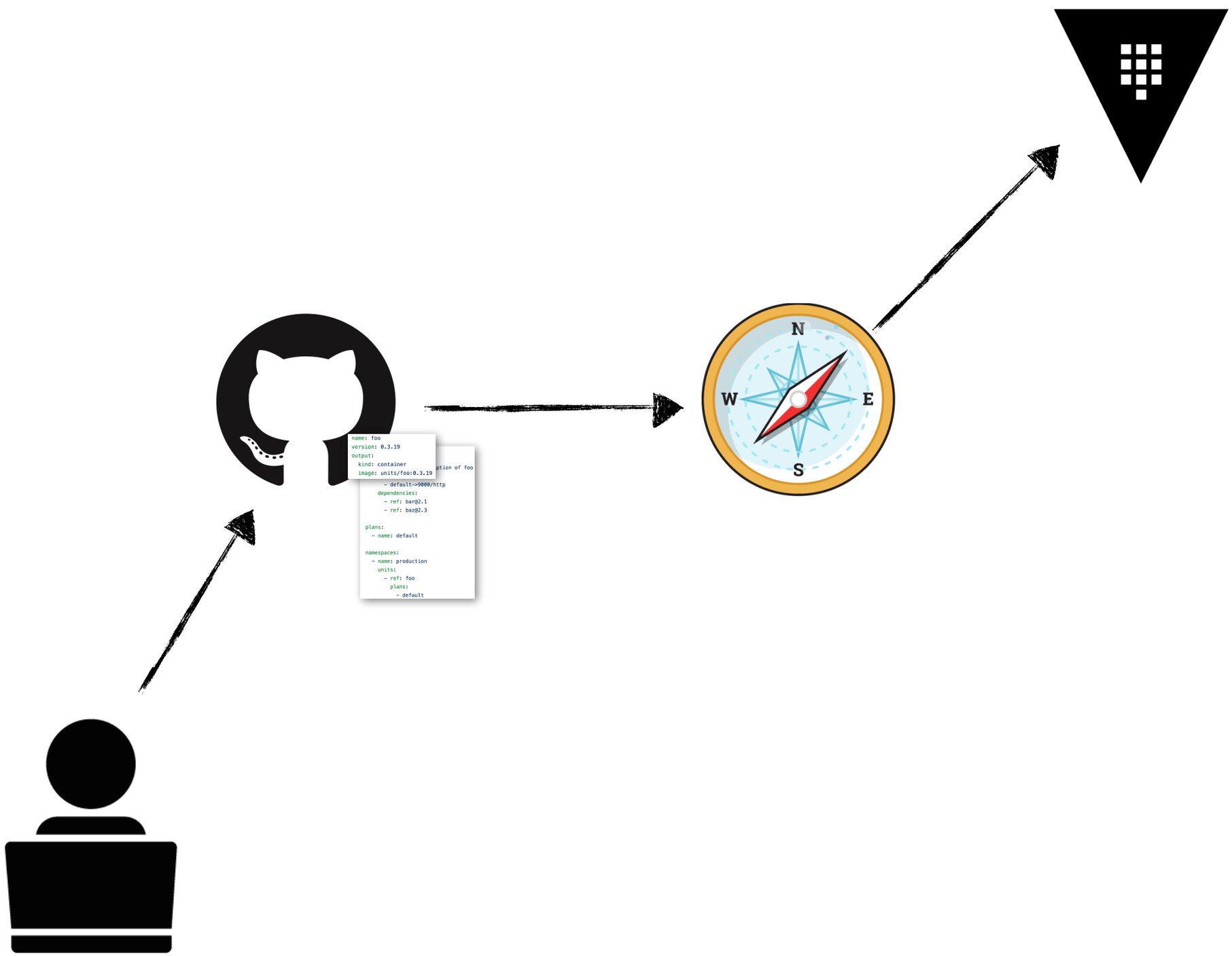
Nelson workflow



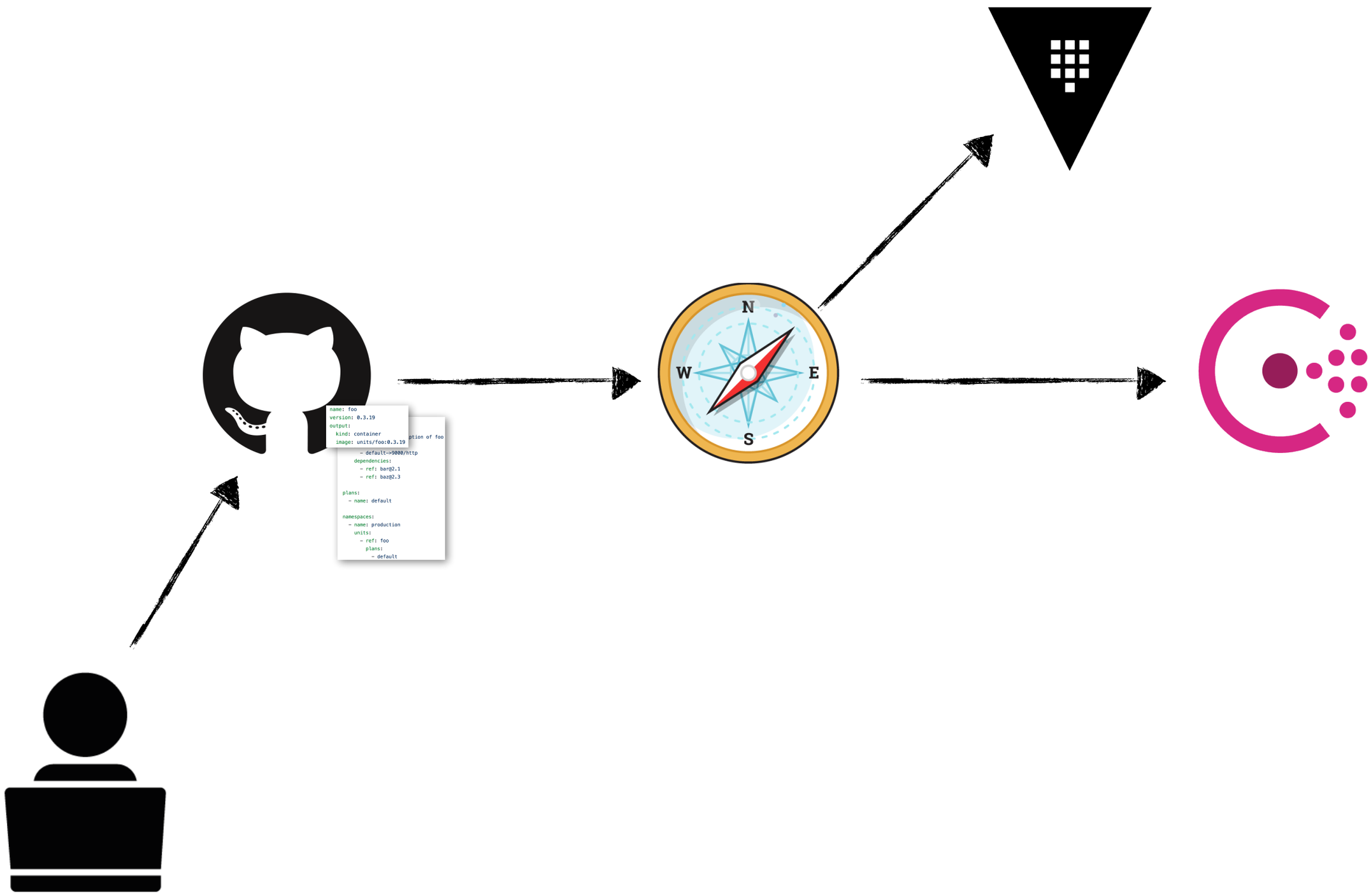
Nelson workflow



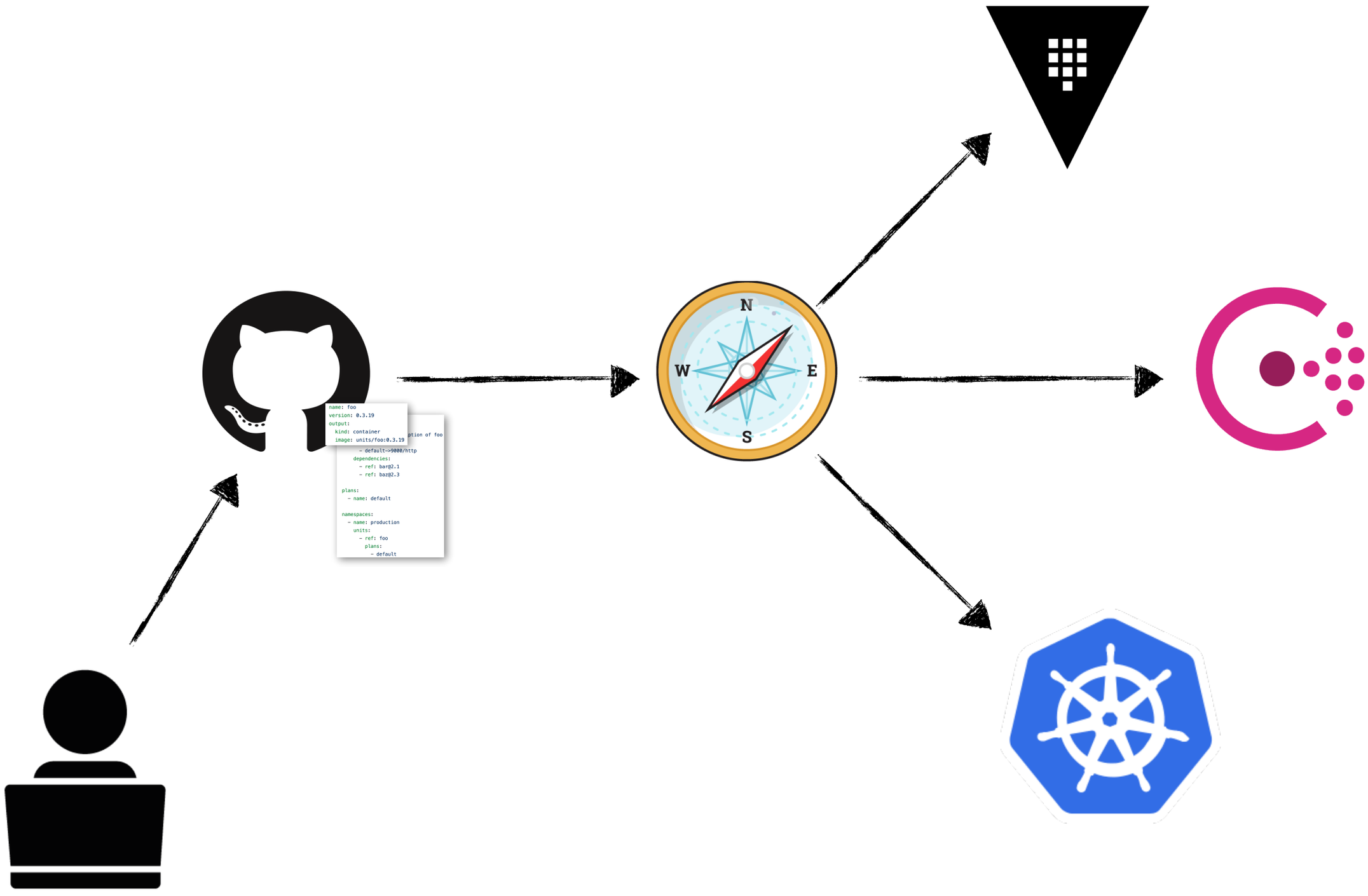
Nelson workflow



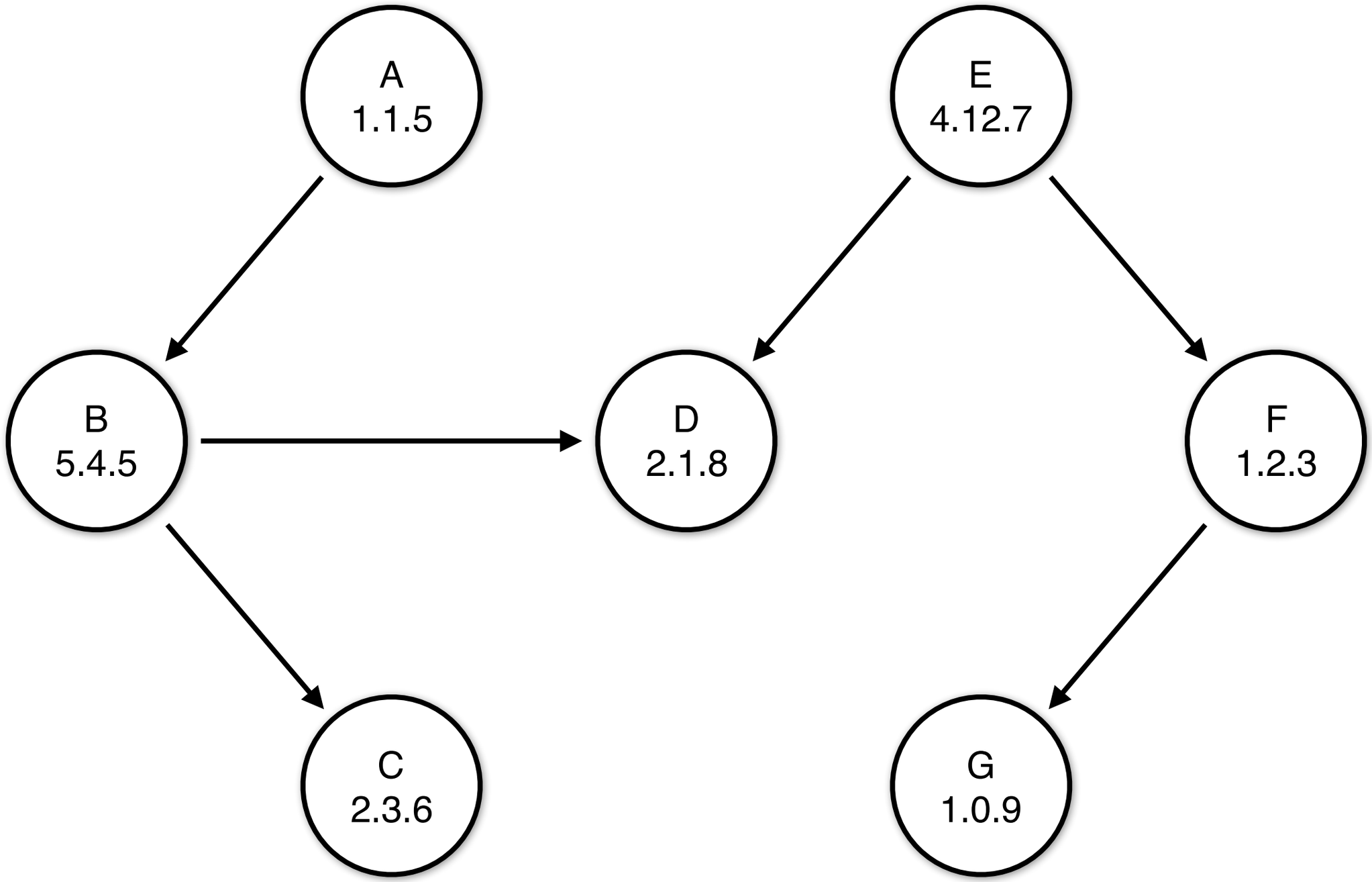
Nelson workflow



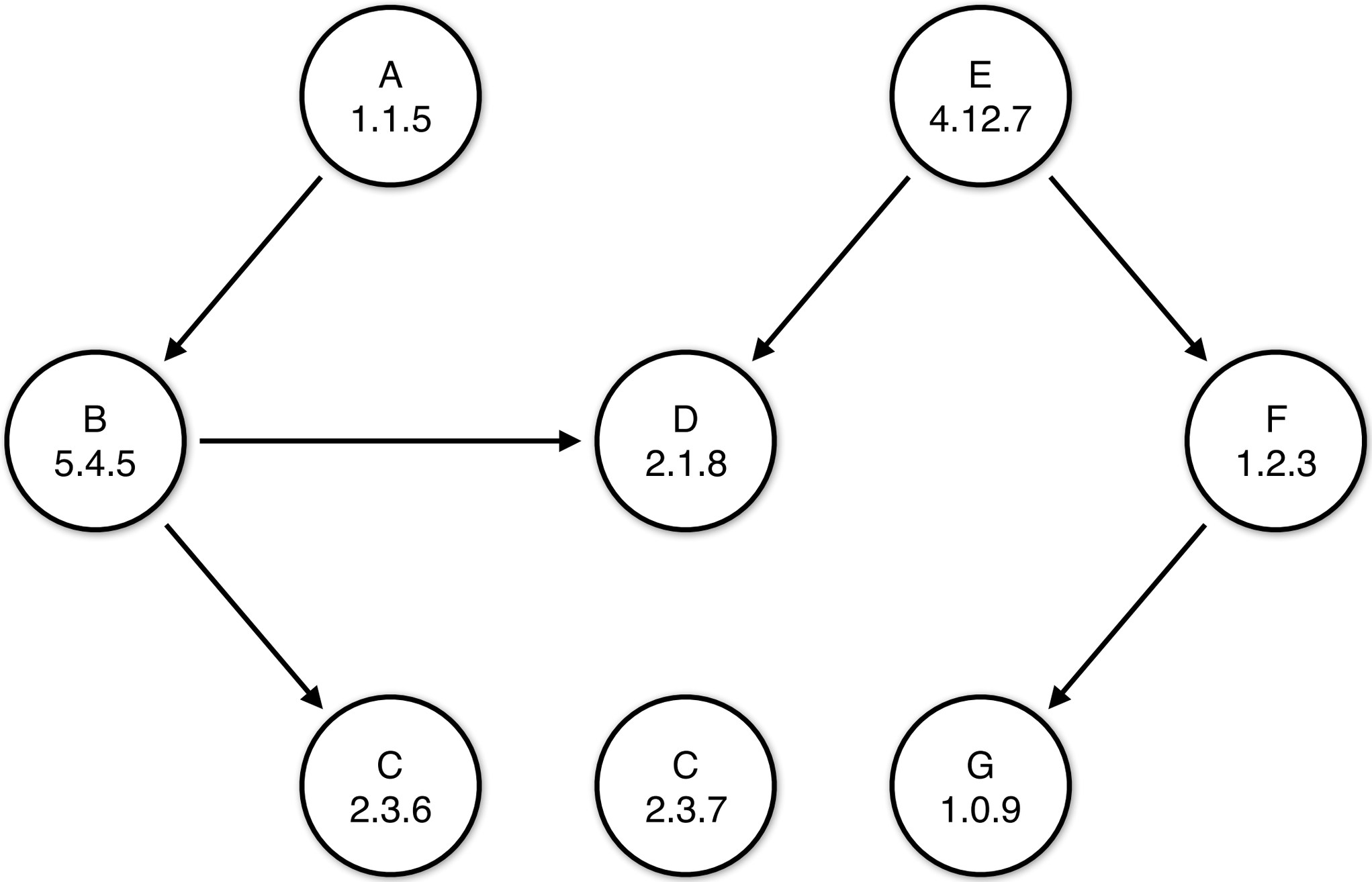
Nelson workflow



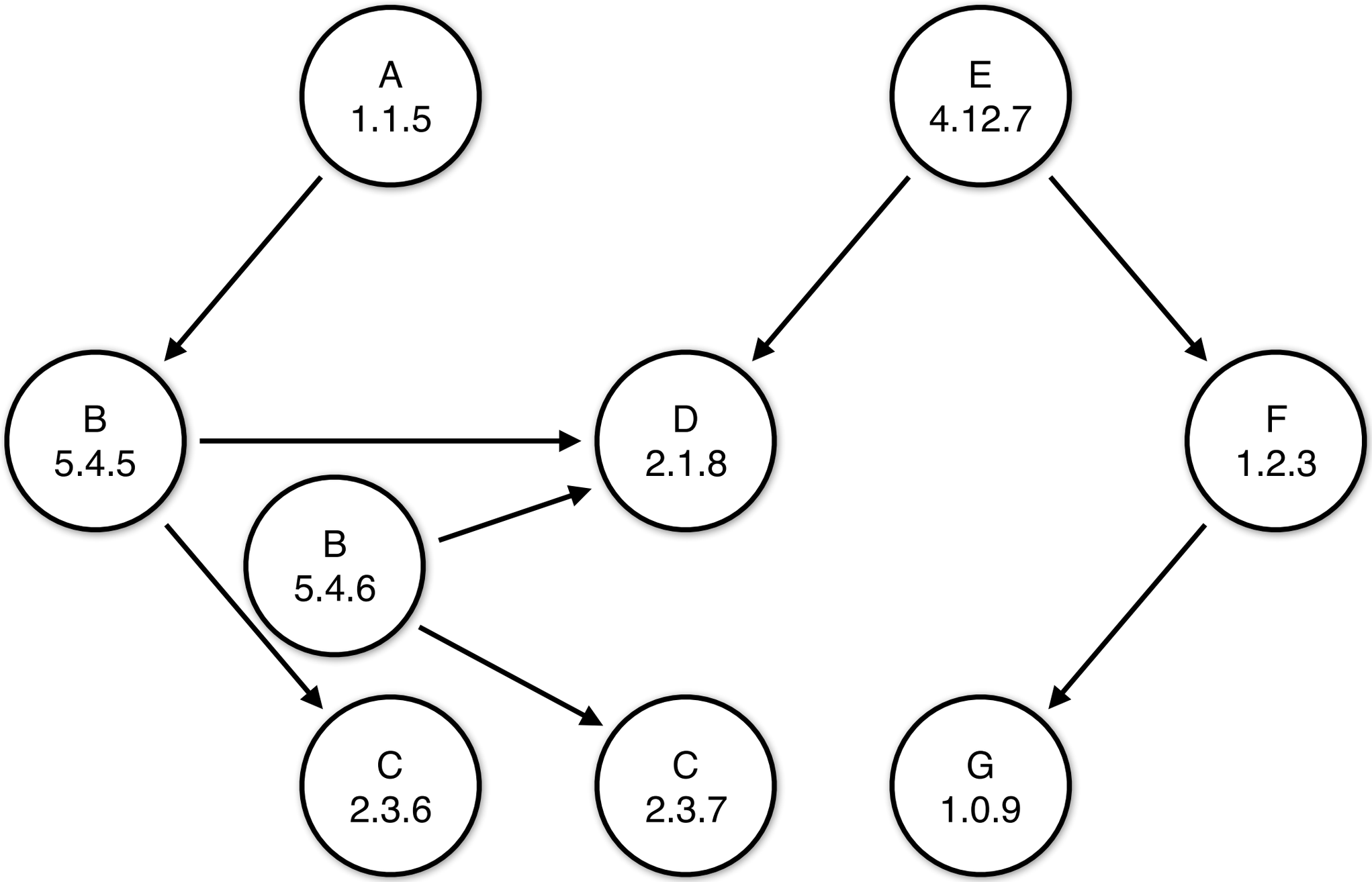
Graph management: traffic shifting



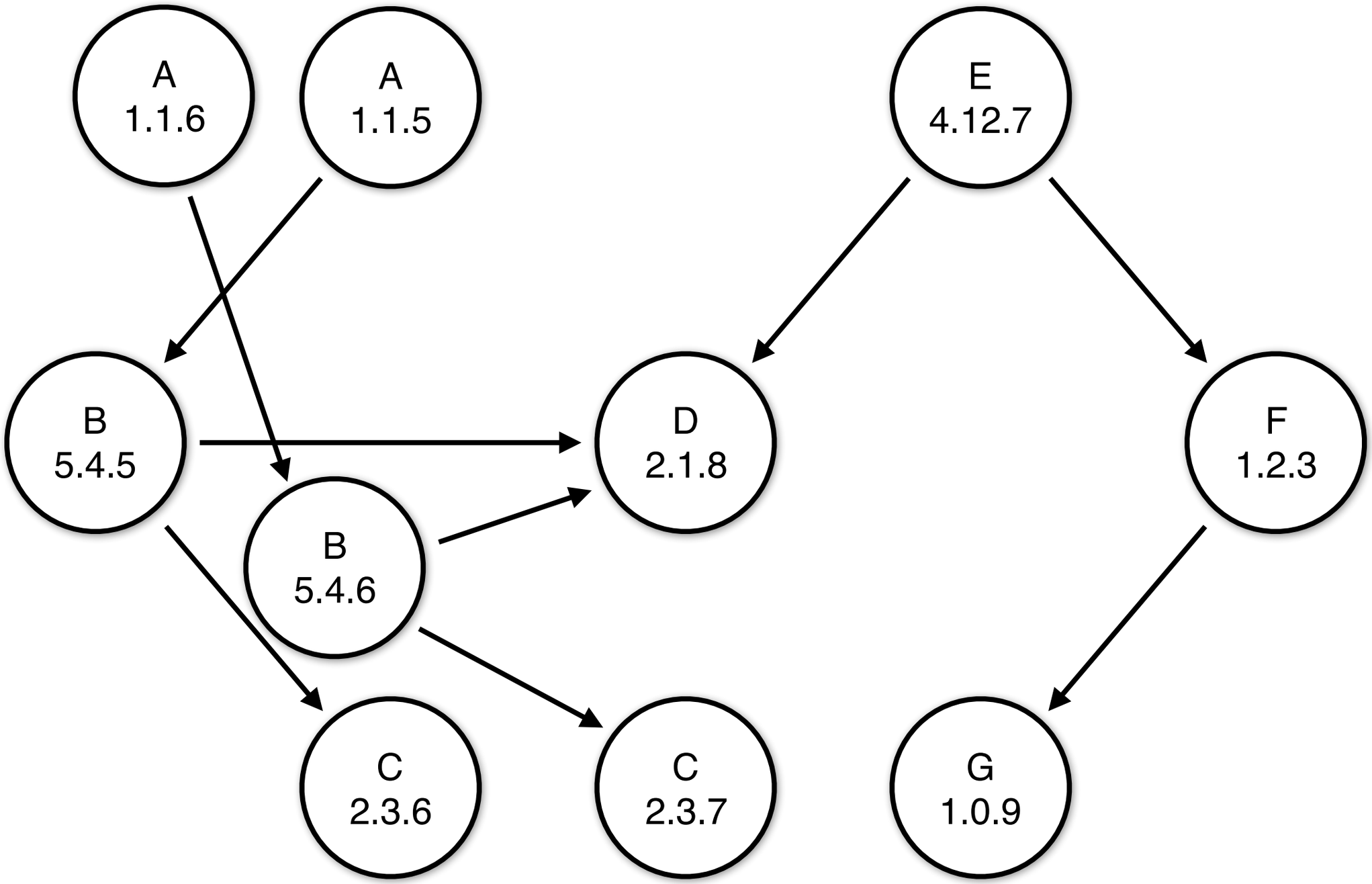
Graph management: traffic shifting



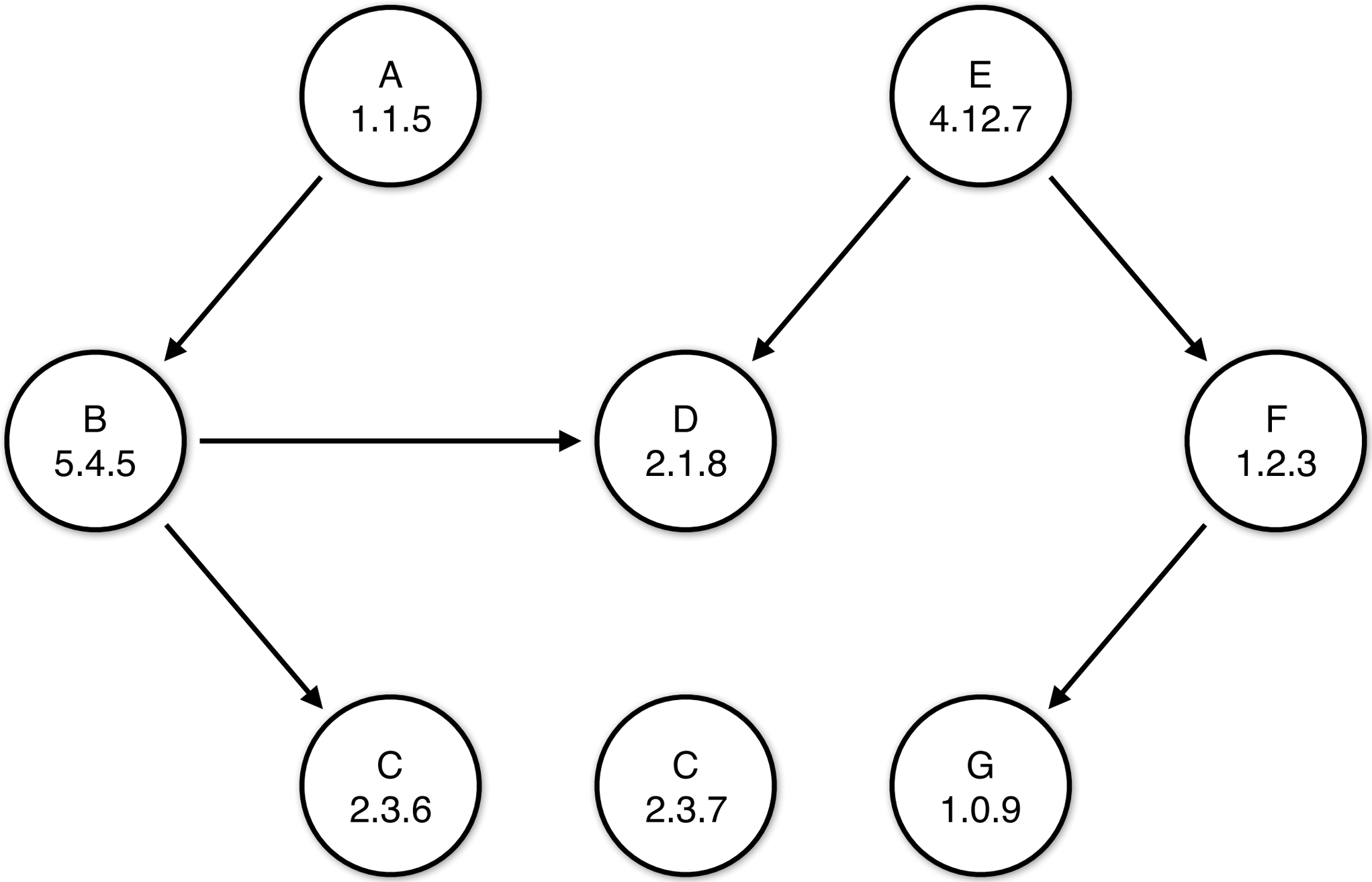
Graph management: traffic shifting



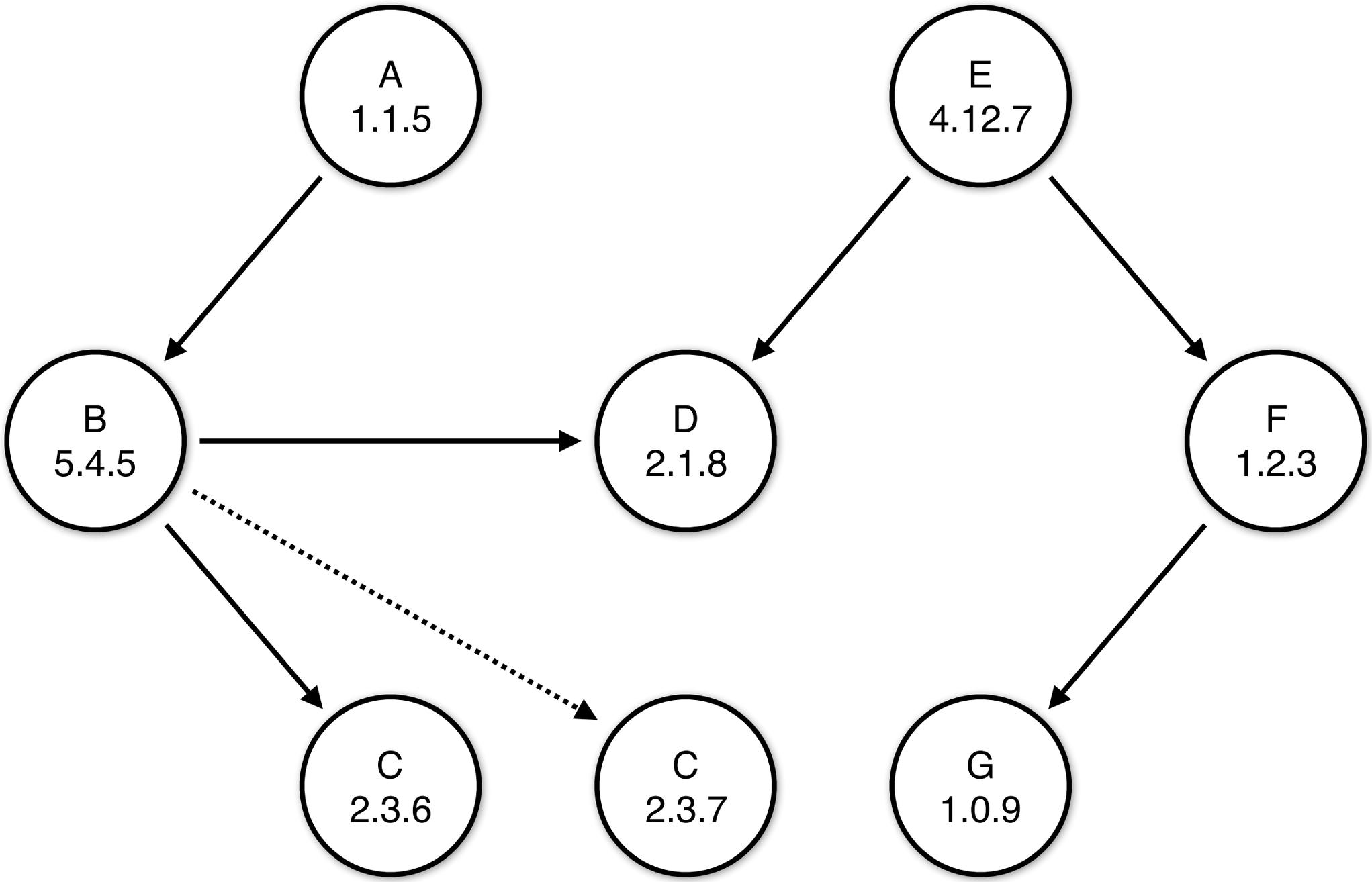
Graph management: traffic shifting



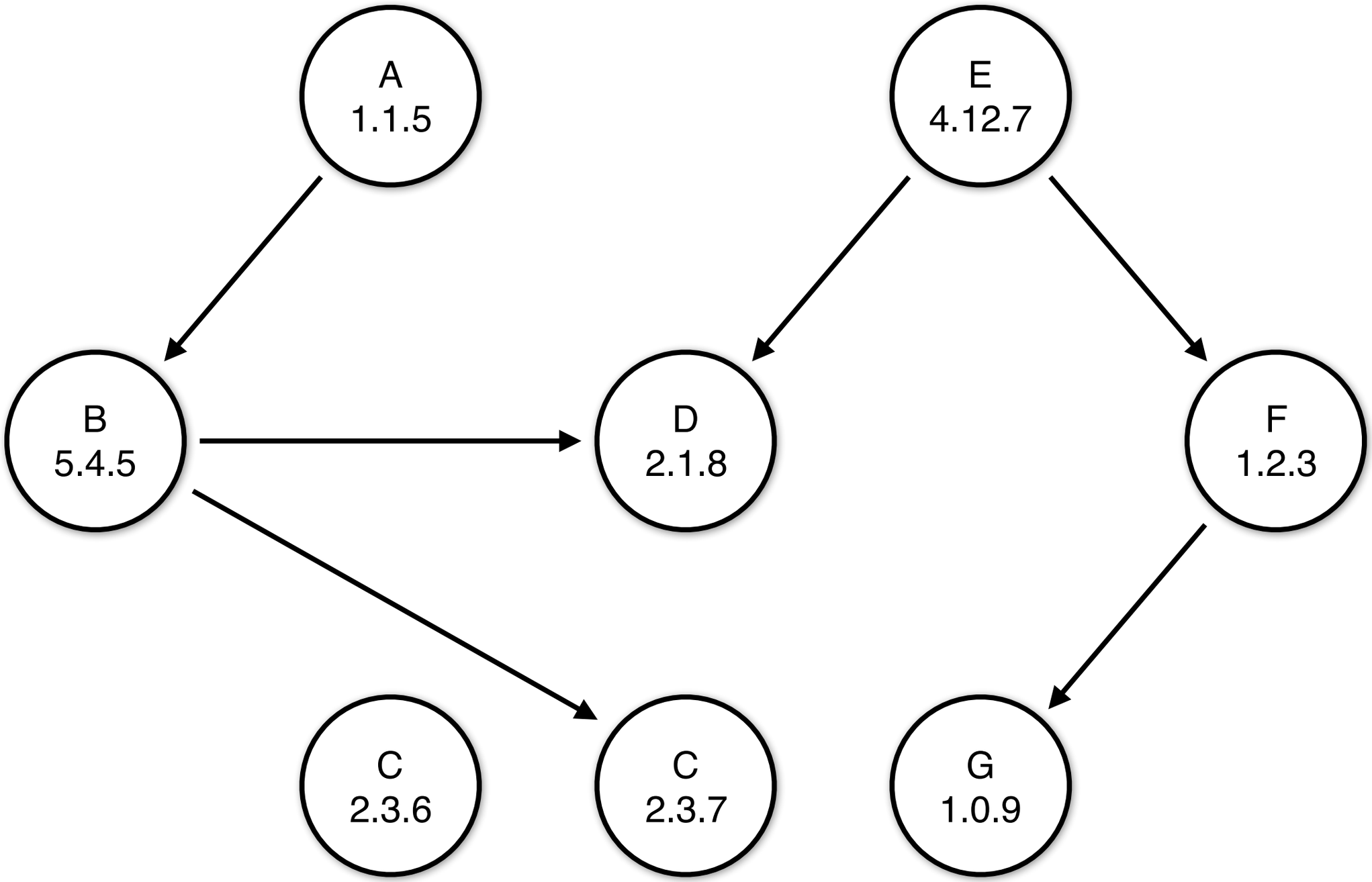
Graph management: traffic shifting



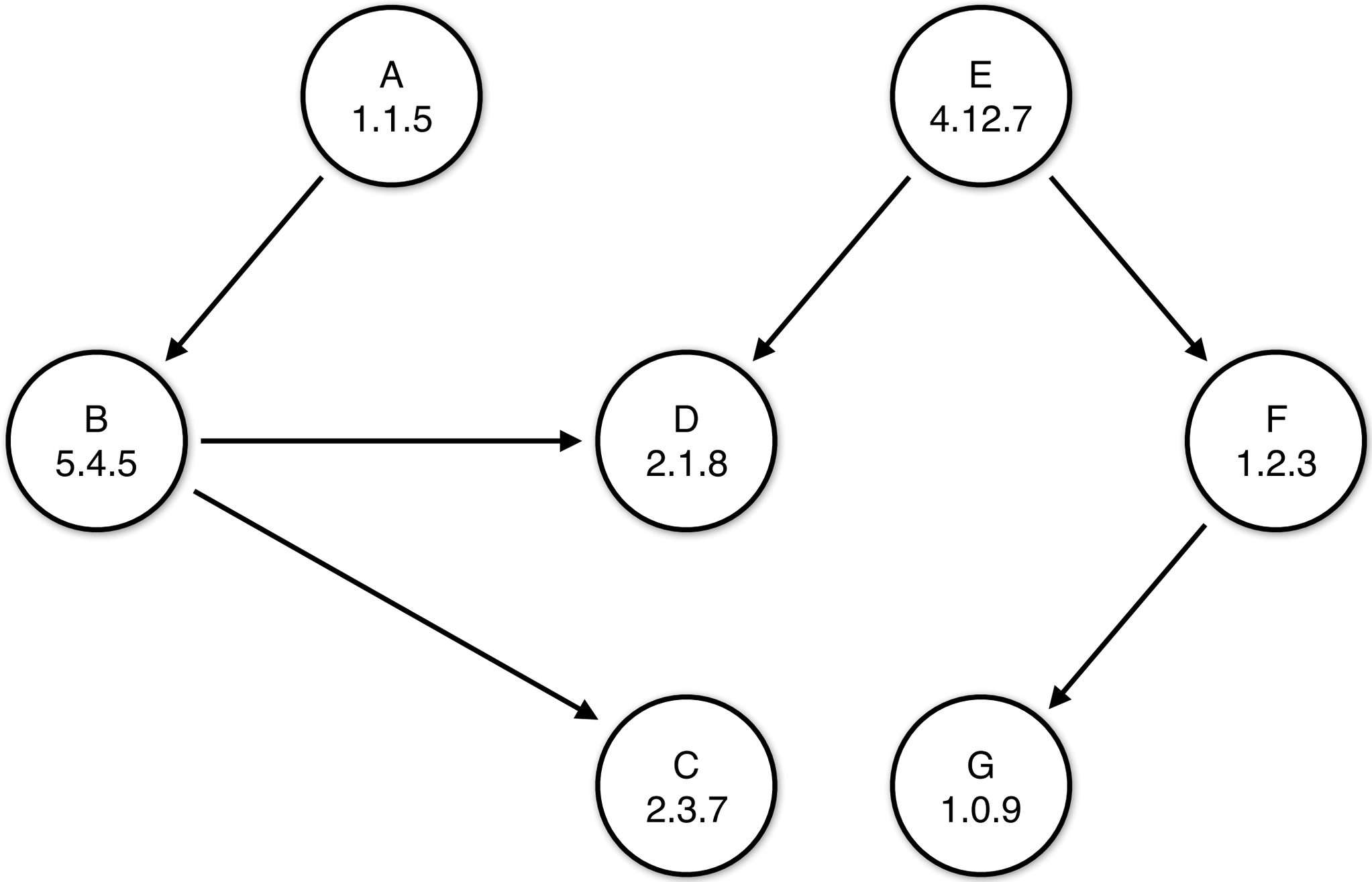
Graph management: traffic shifting



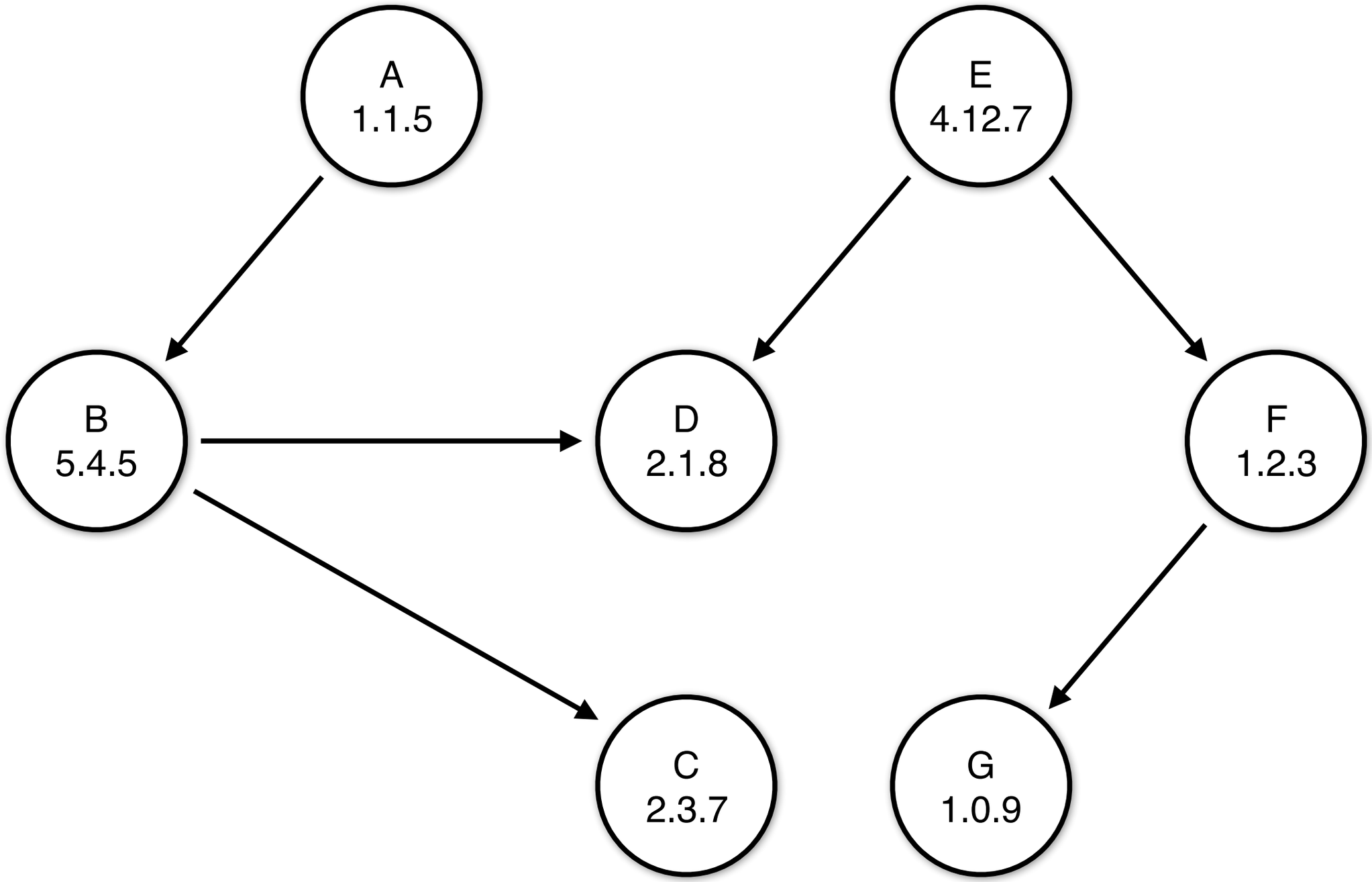
Graph management: traffic shifting



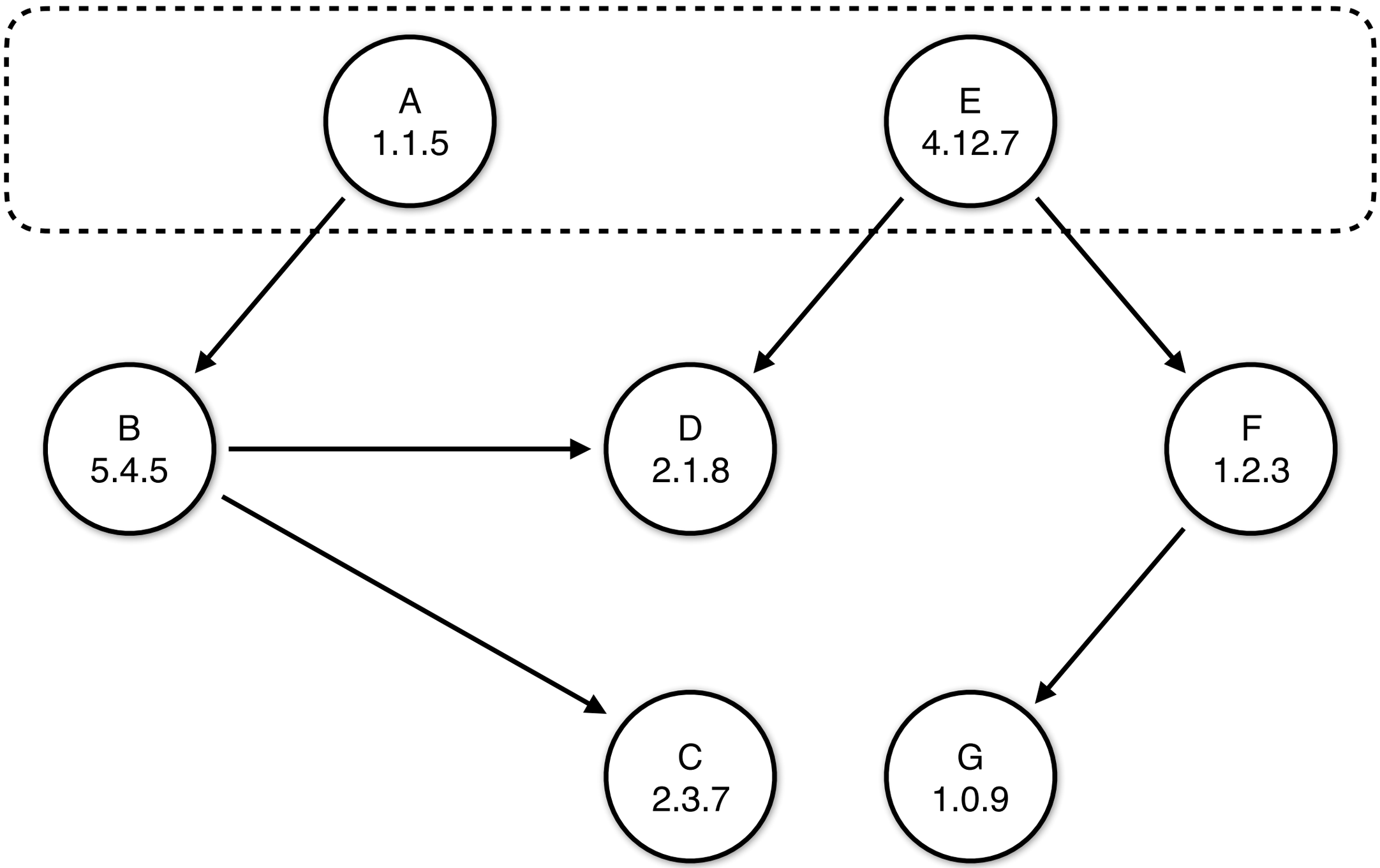
Graph management: traffic shifting



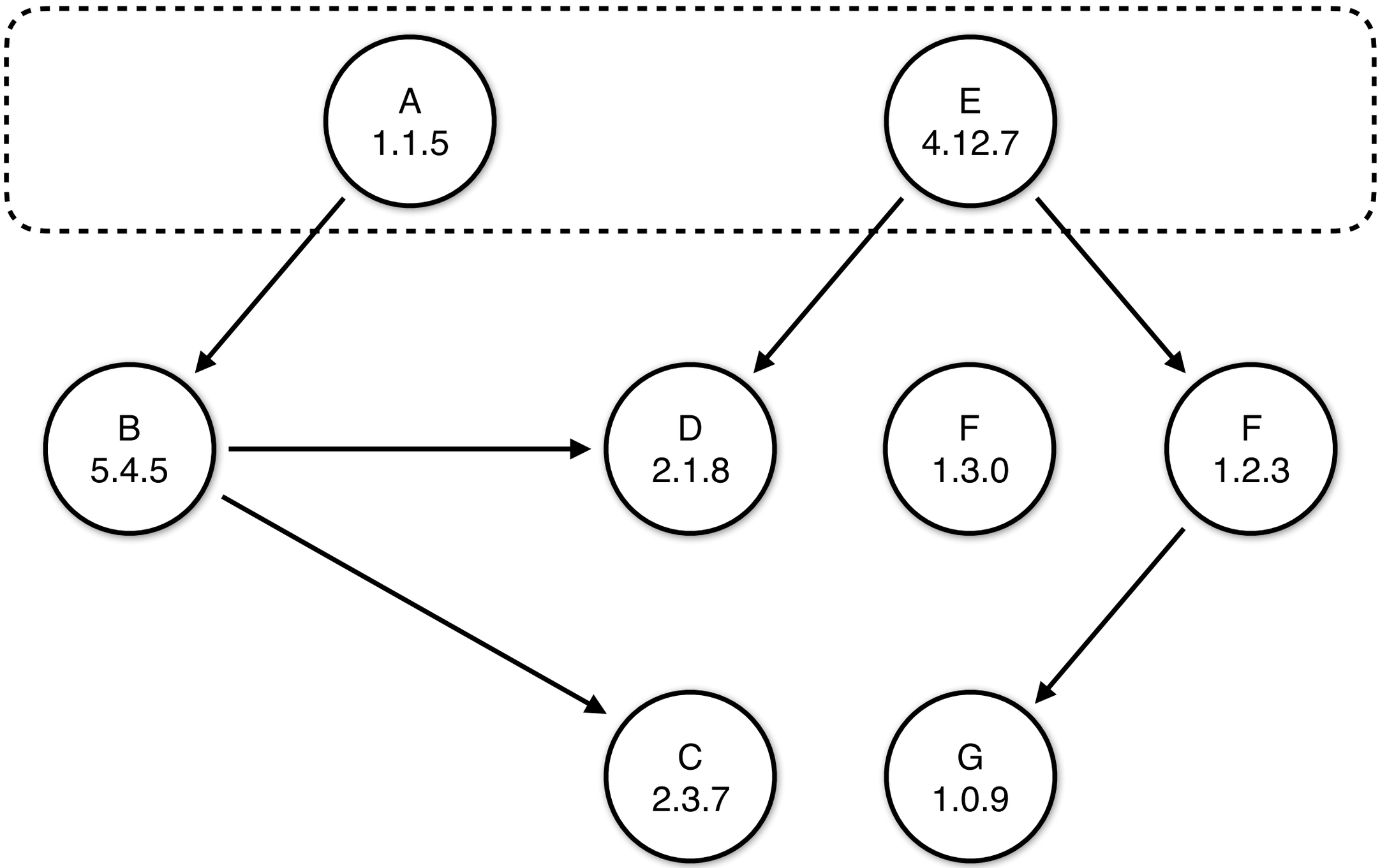
Graph management: pruning



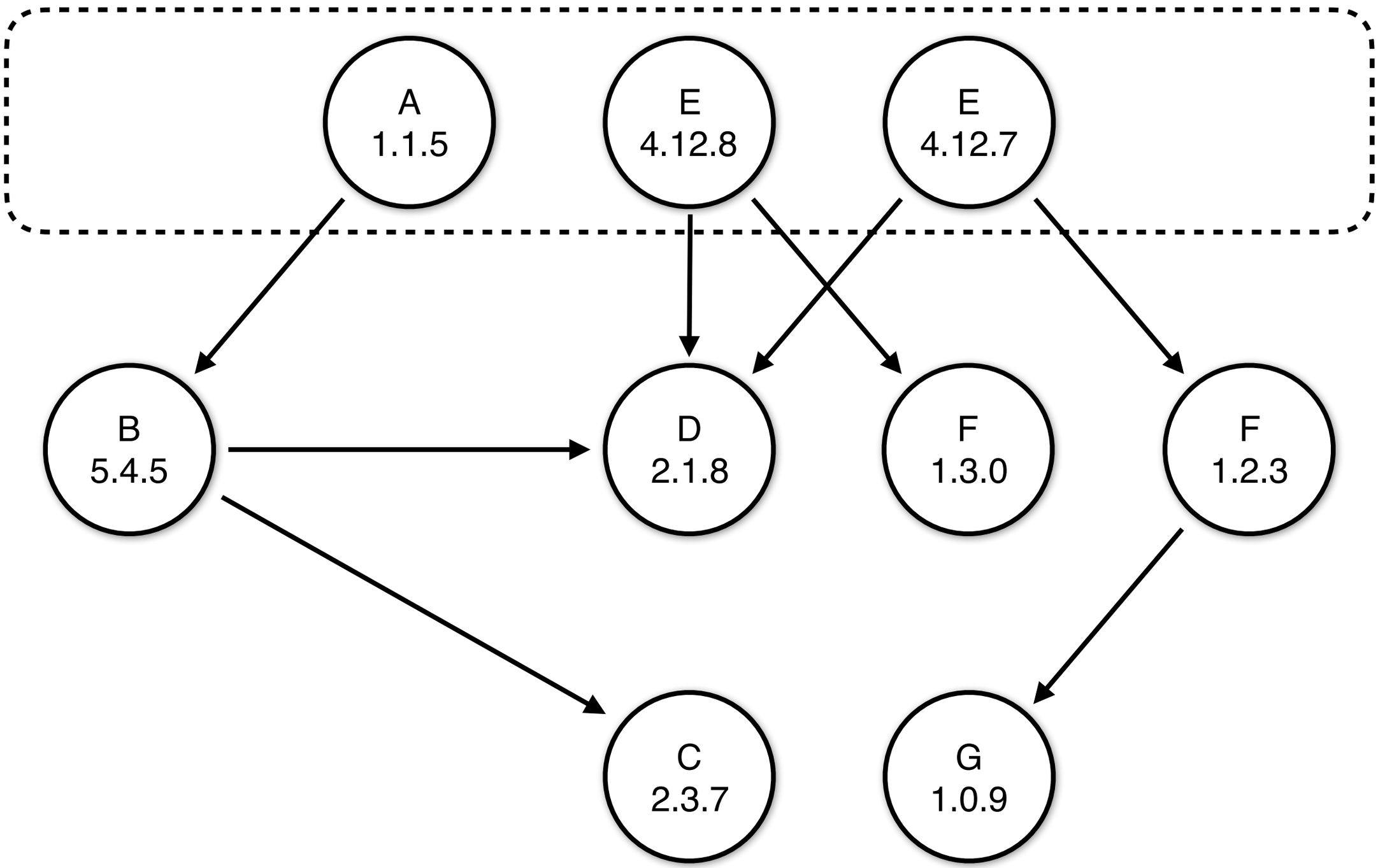
Graph management: pruning



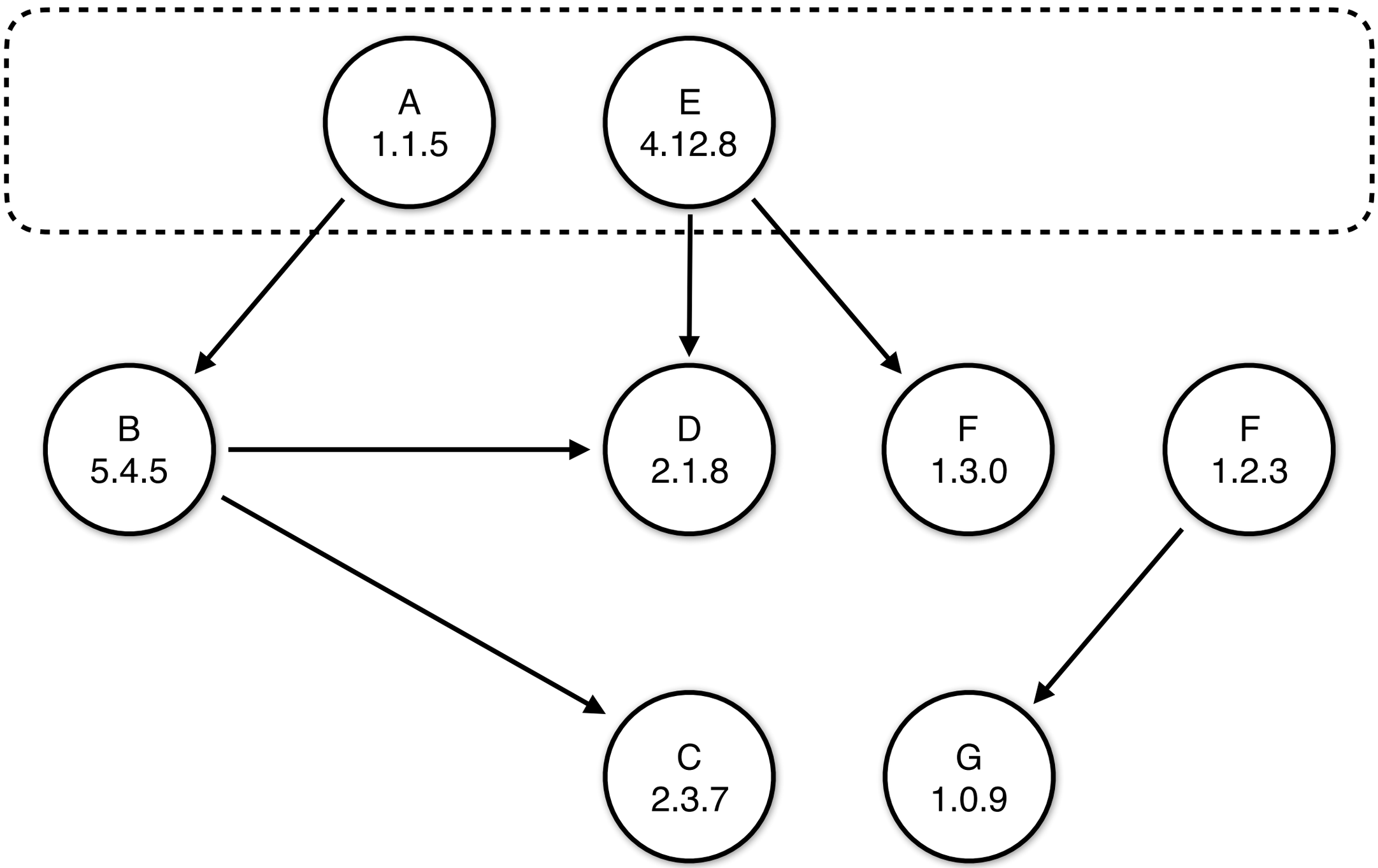
Graph management: pruning



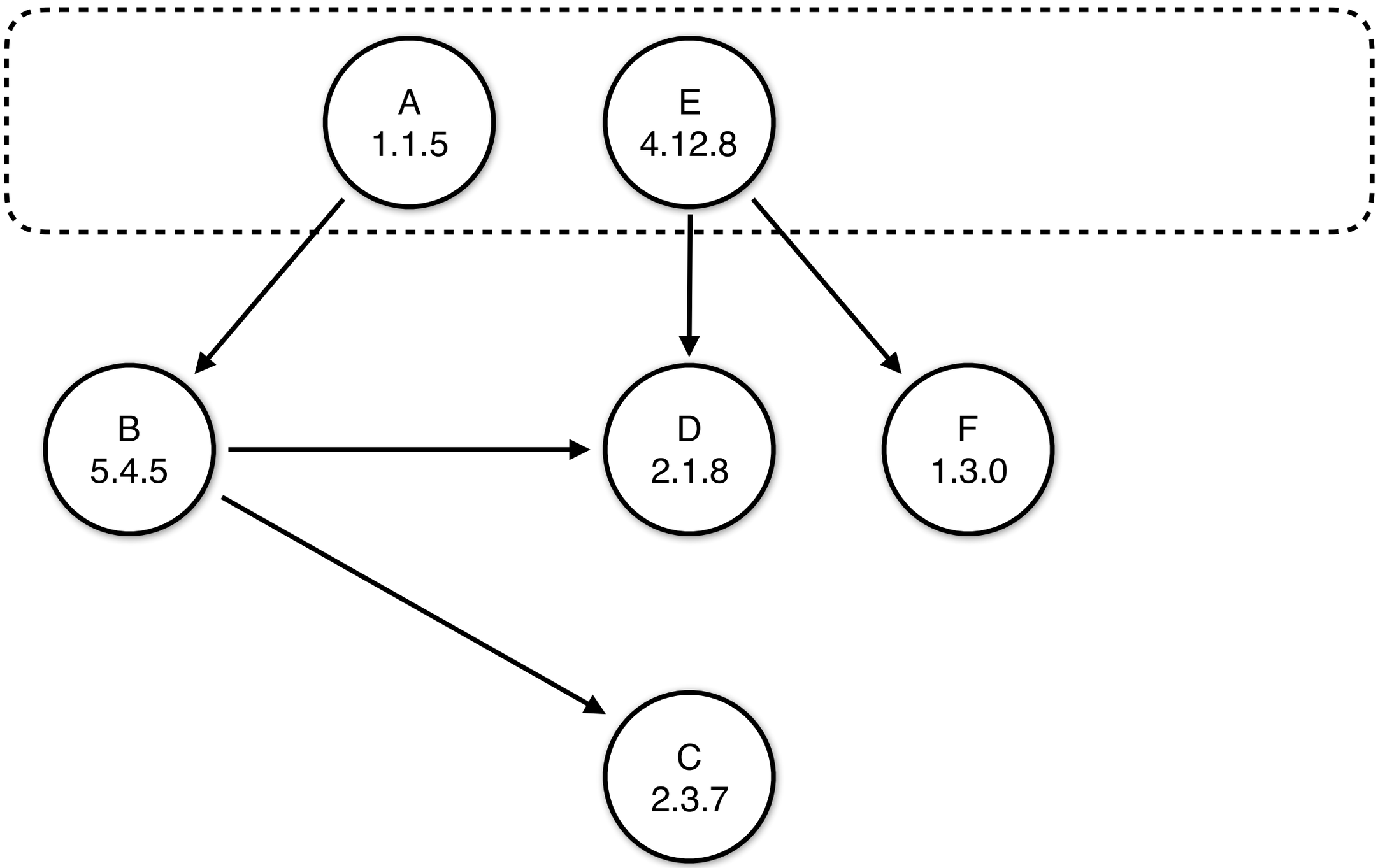
Graph management: pruning



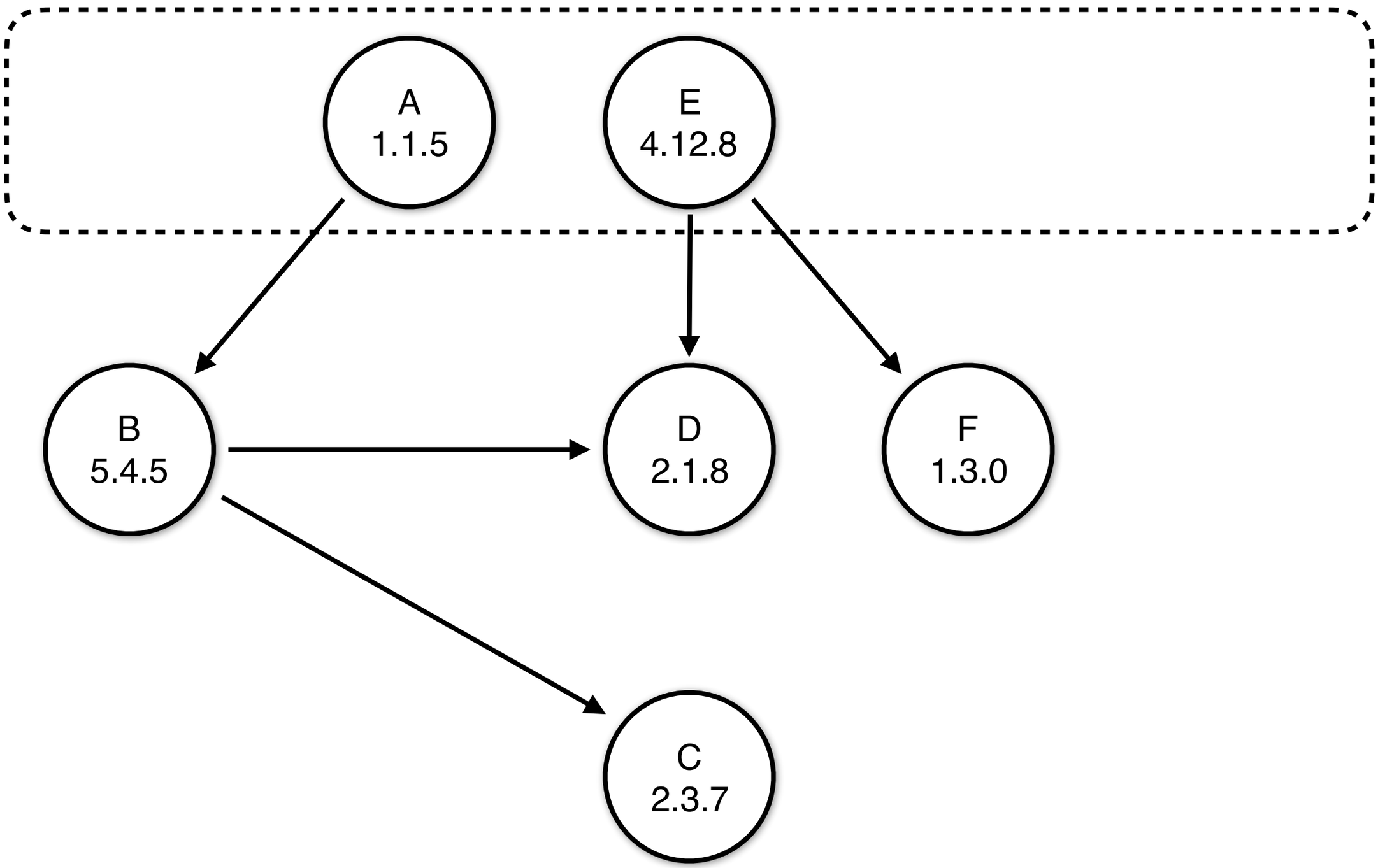
Graph management: pruning



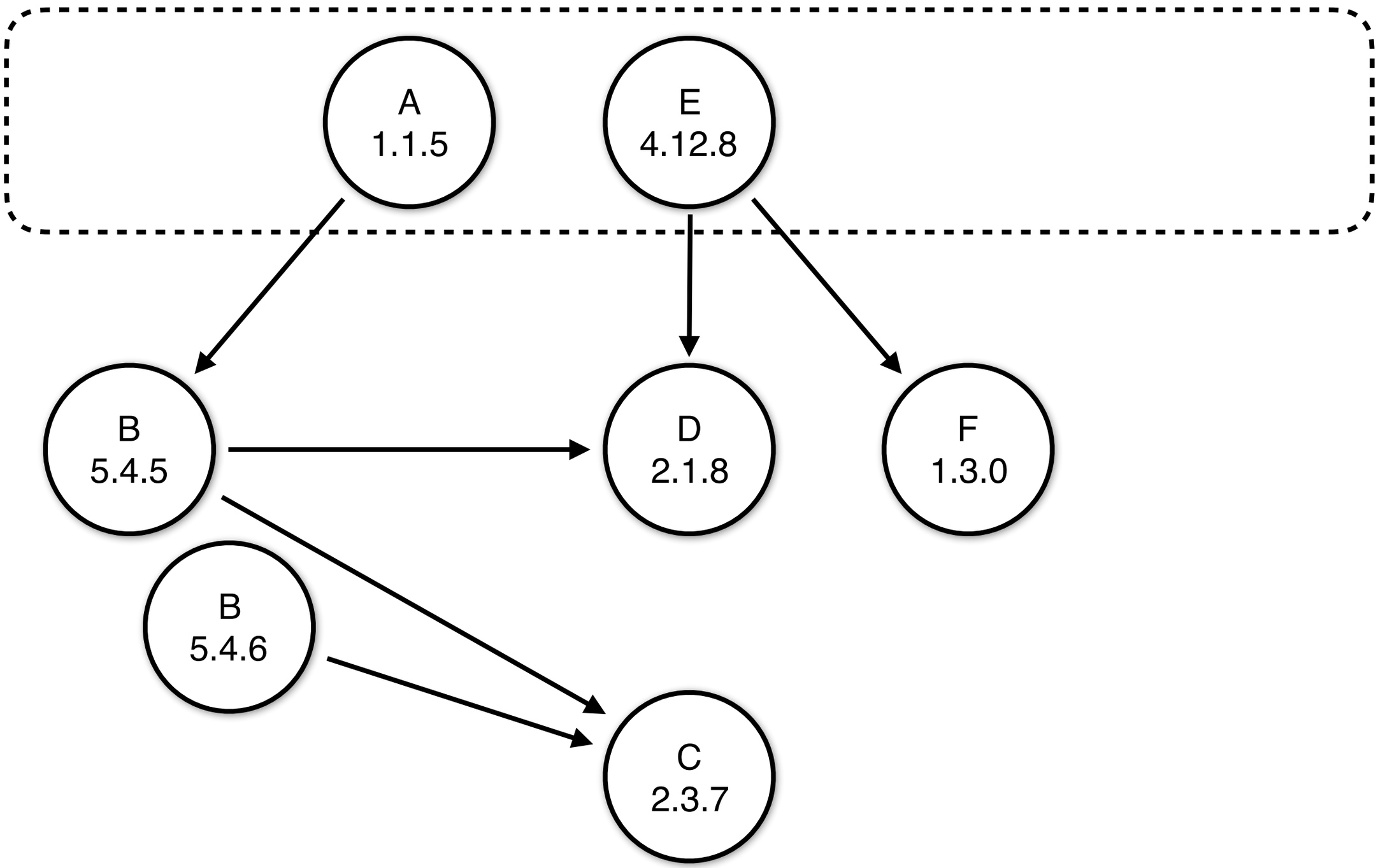
Graph management: pruning



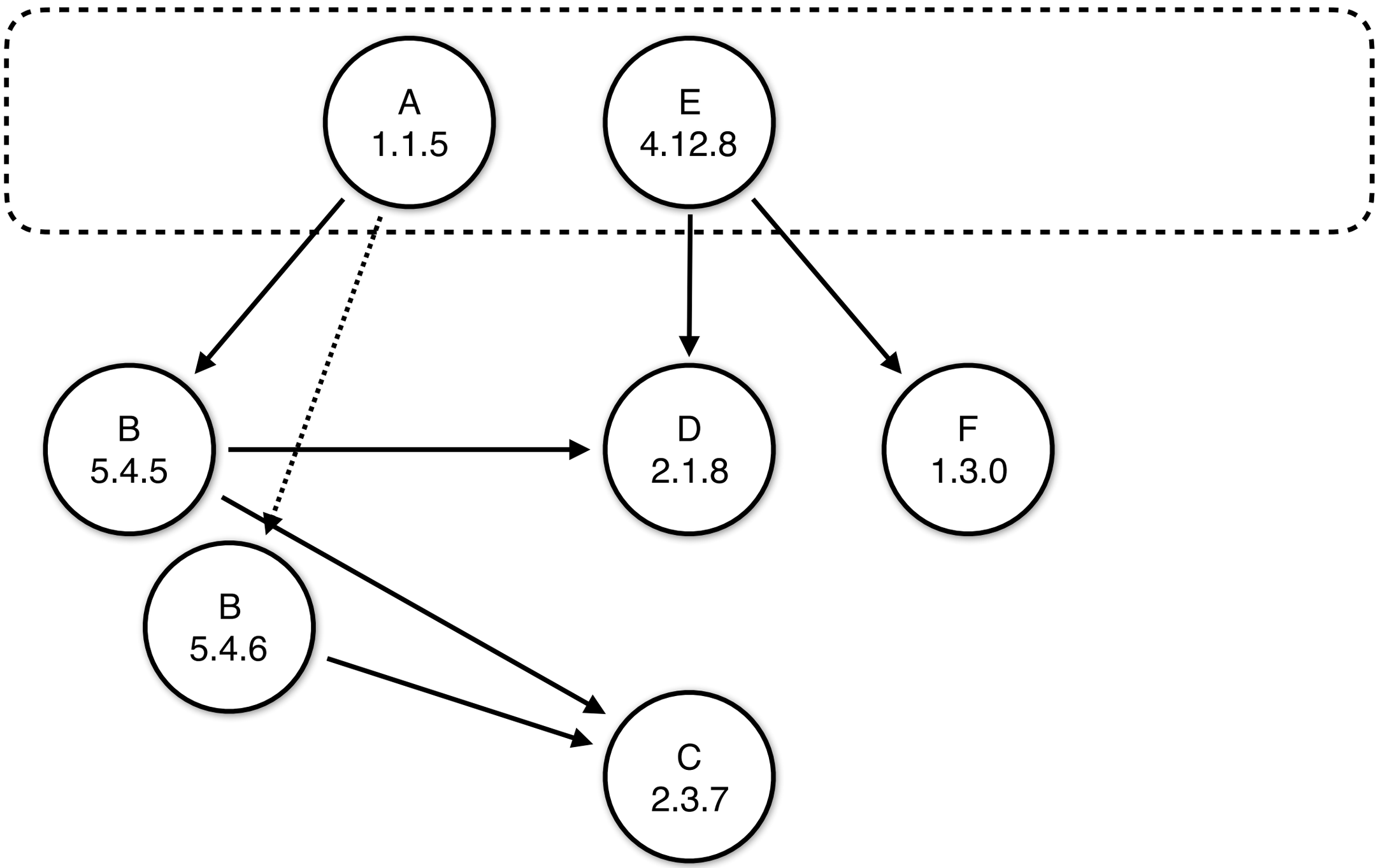
Graph management: pruning



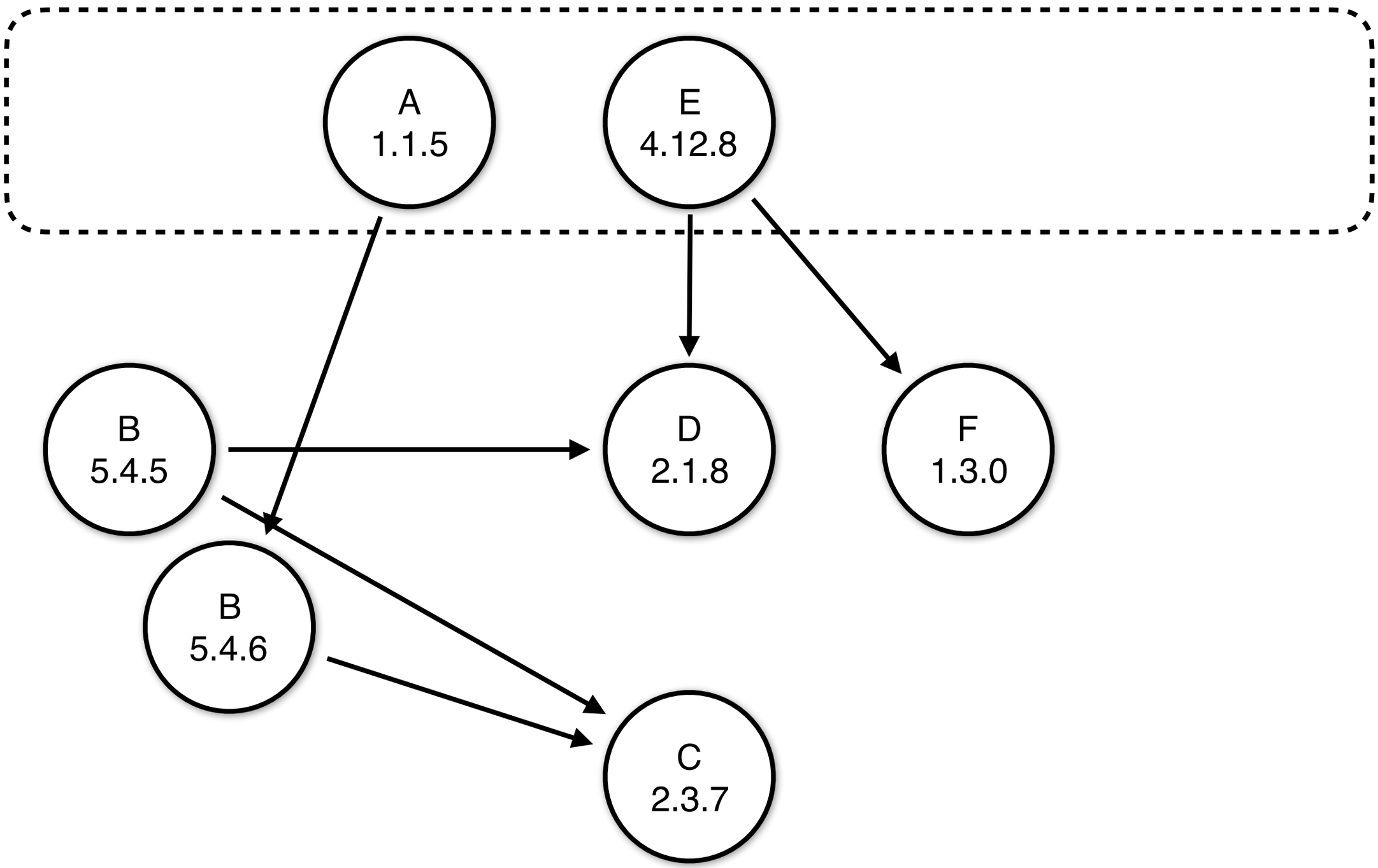
Graph management: pruning



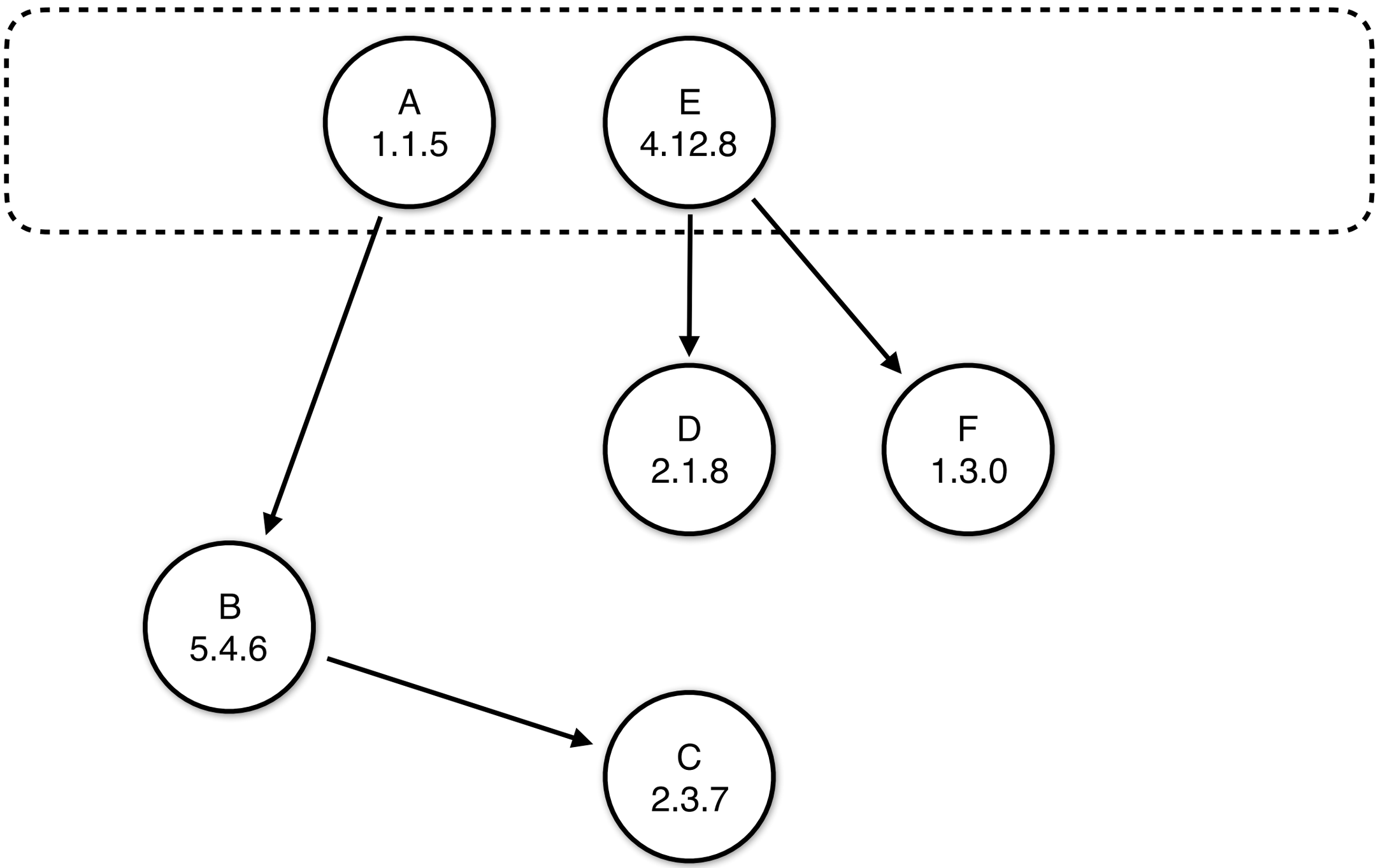
Graph management: pruning



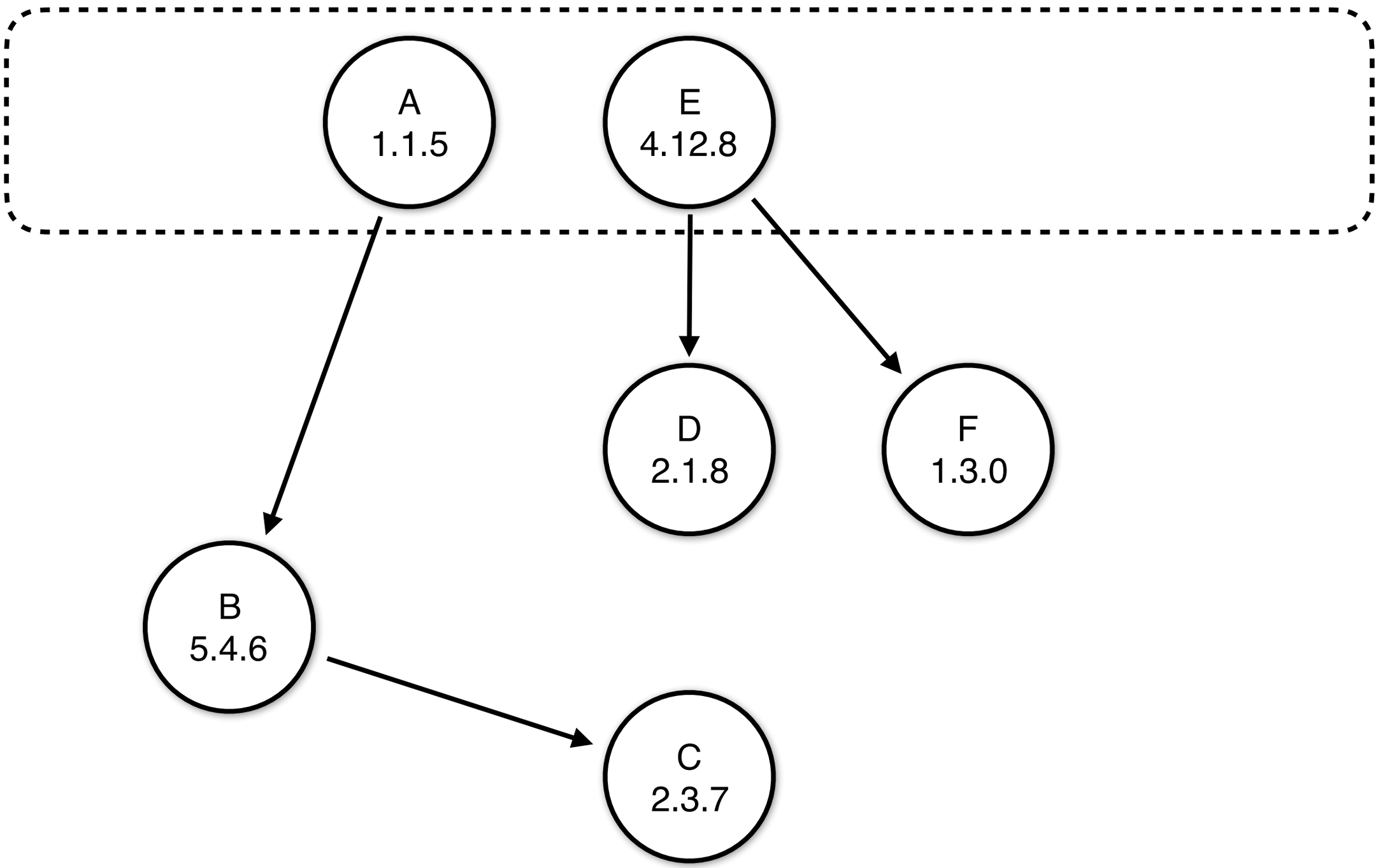
Graph management: pruning



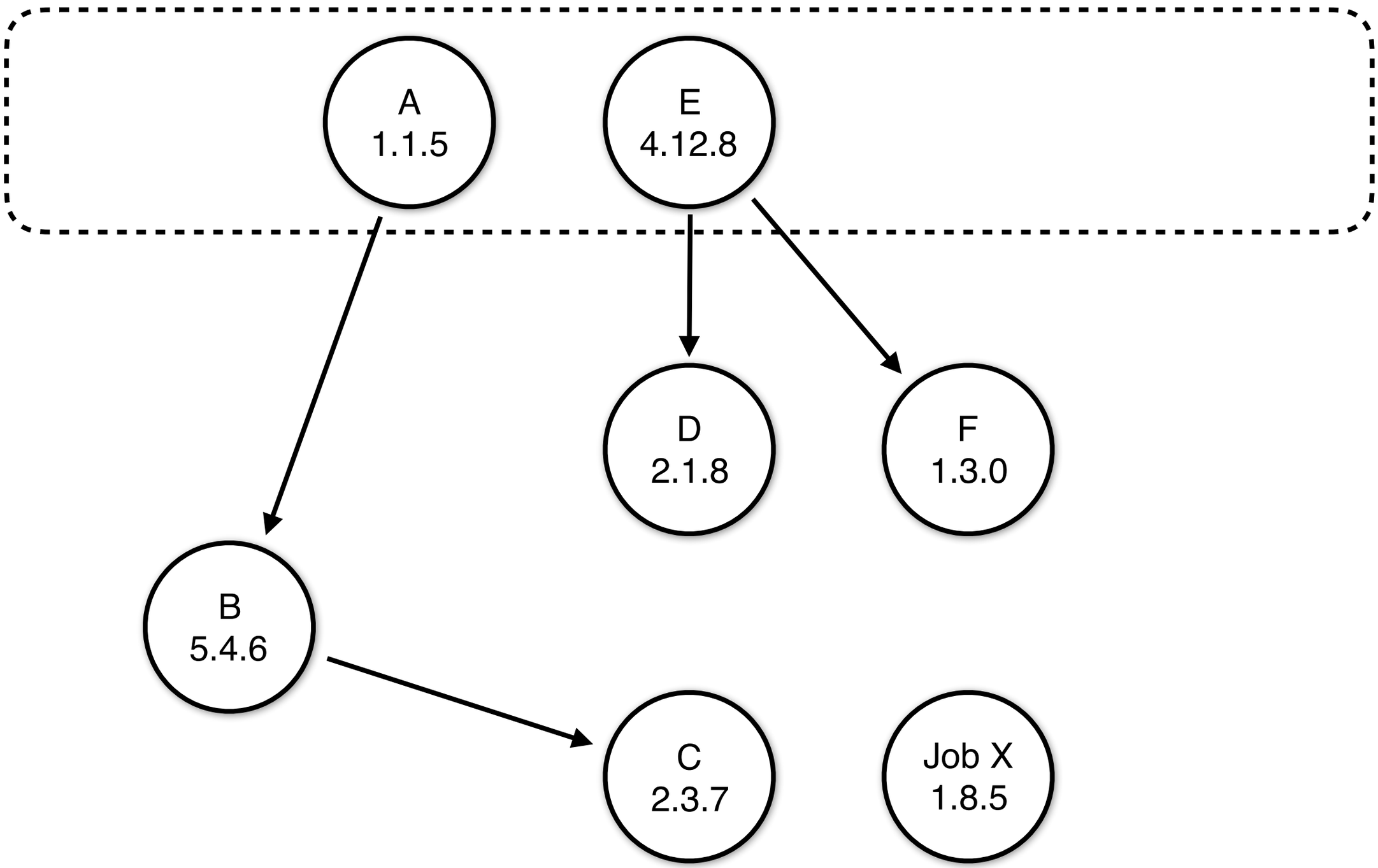
Graph management: pruning



Graph management: pruning



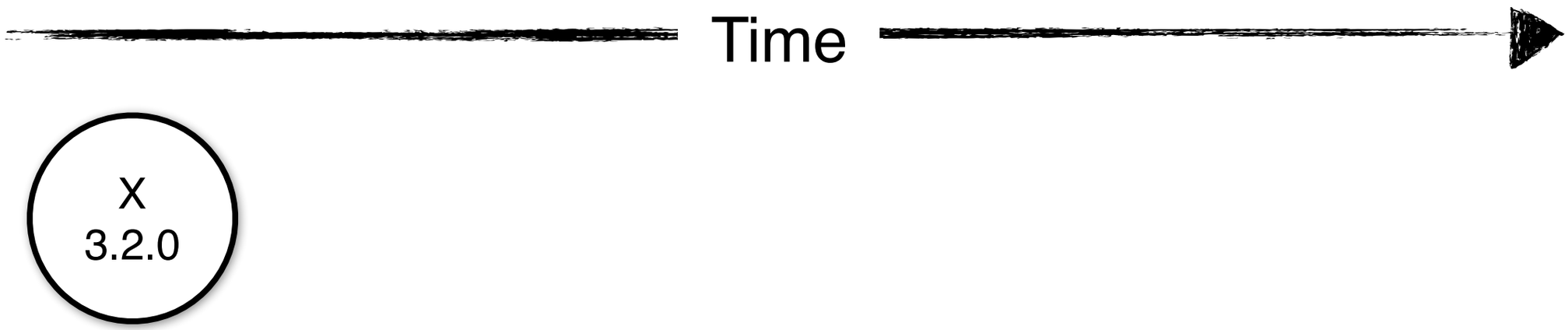
Graph management: pruning



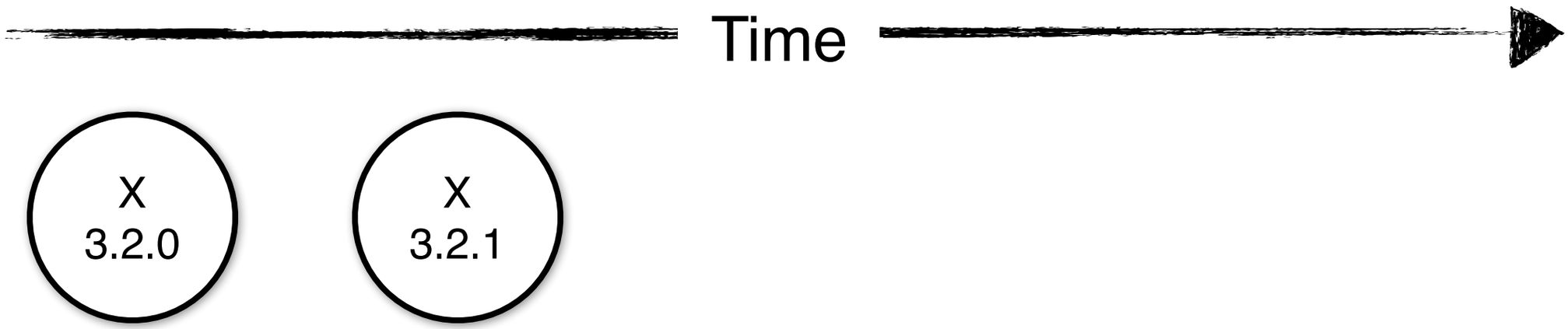

Time

Time

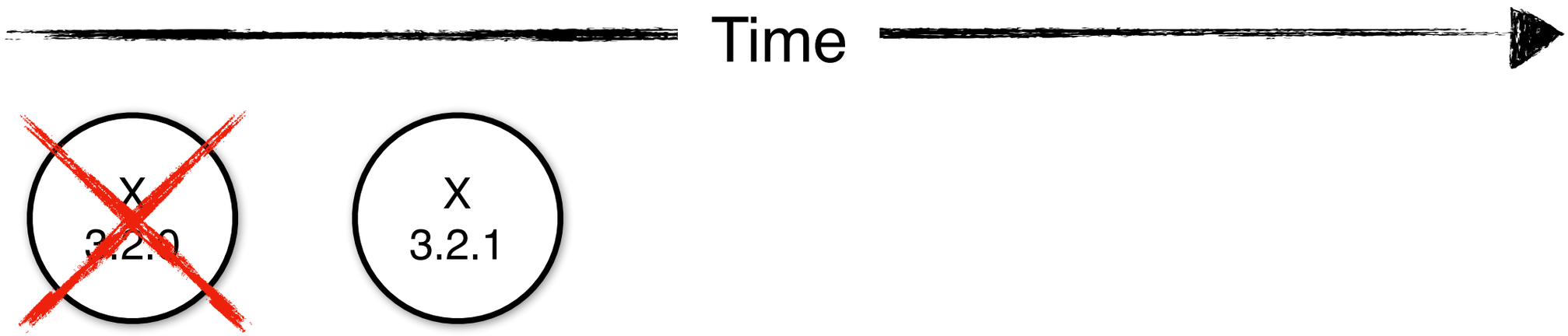
`retain-latest`



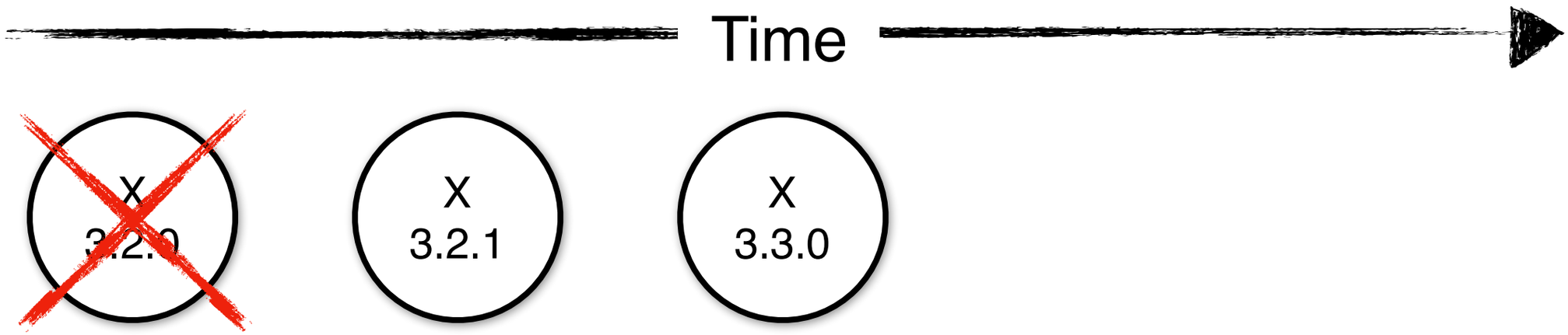
`retain-latest`

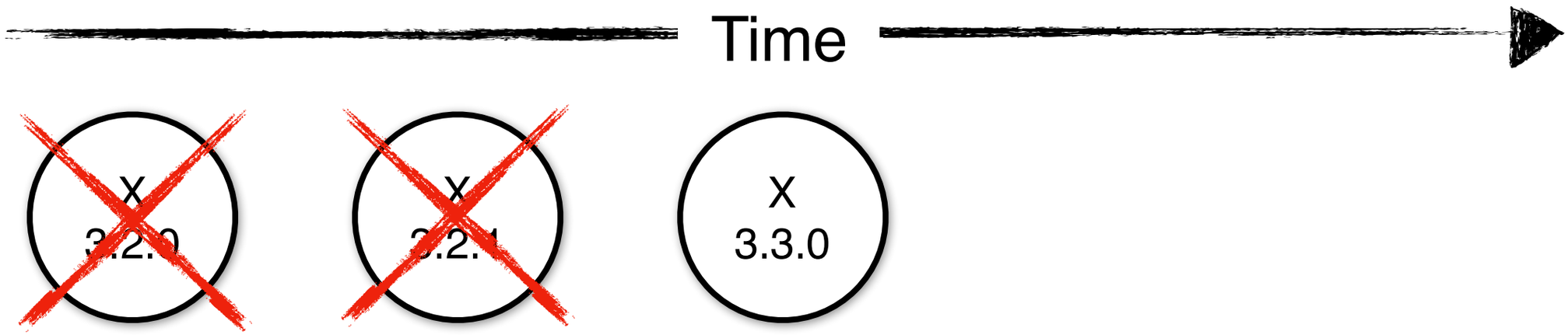


`retain-latest`

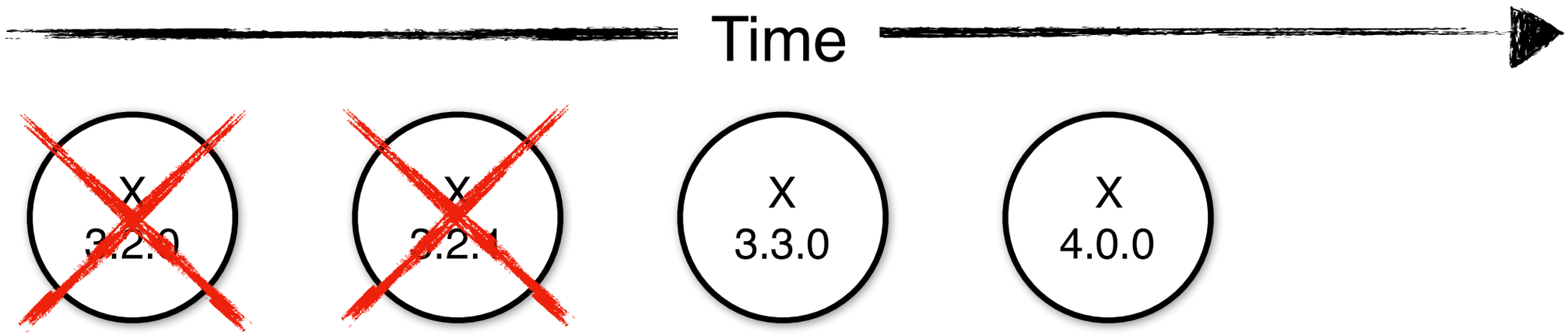


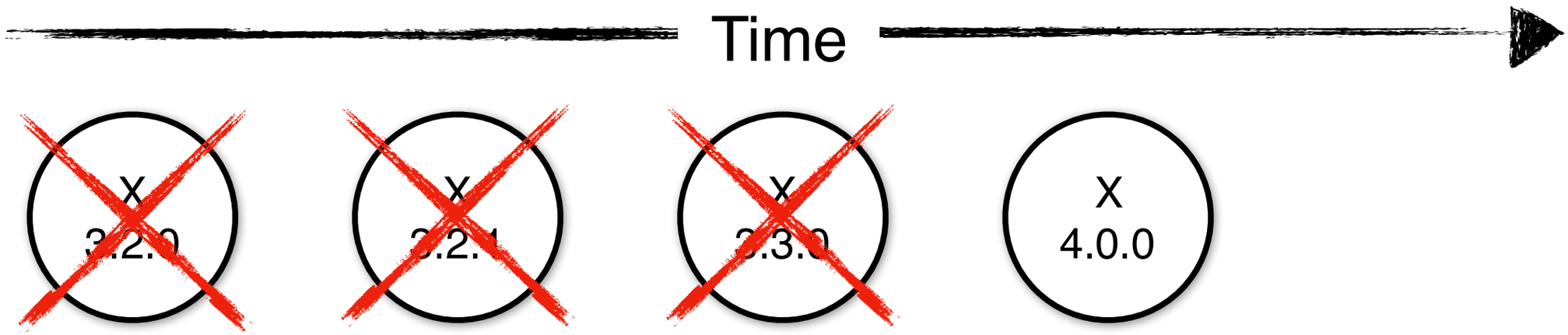
`retain-latest`


Time
X
3.2.0
X
3.2.1
X
3.3.0
retain-latest

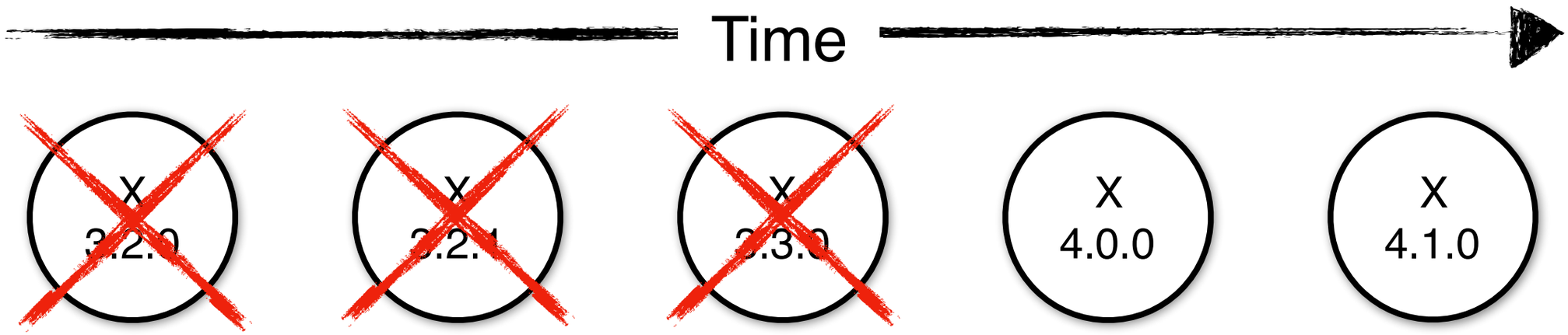


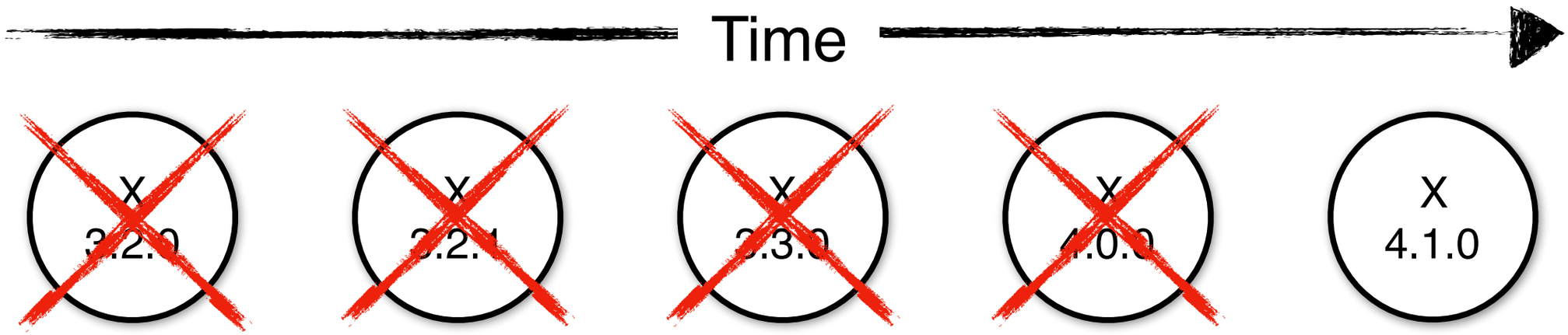
`retain-latest`


Time
X
3.2.0
X
3.2.1
X
3.3.0
X
4.0.0
retain-latest

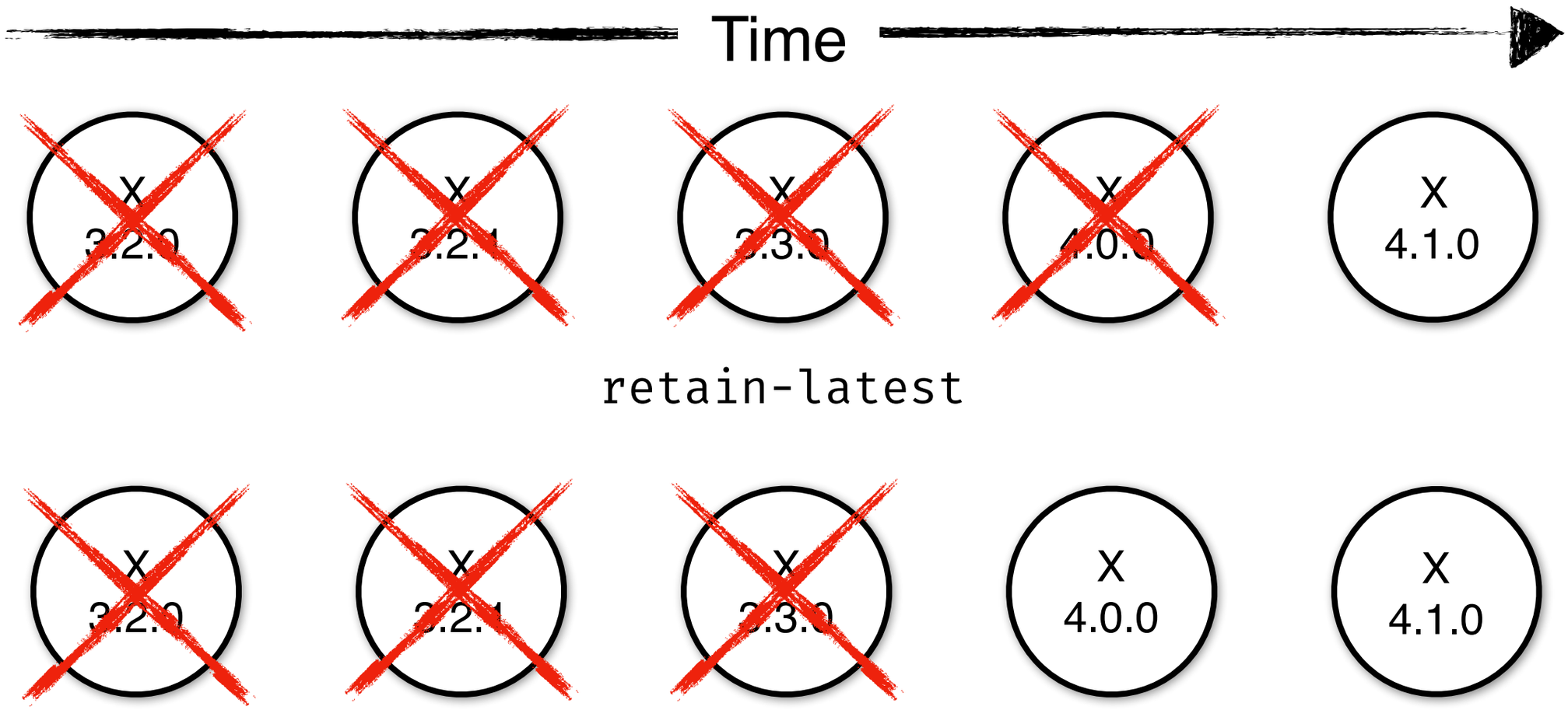


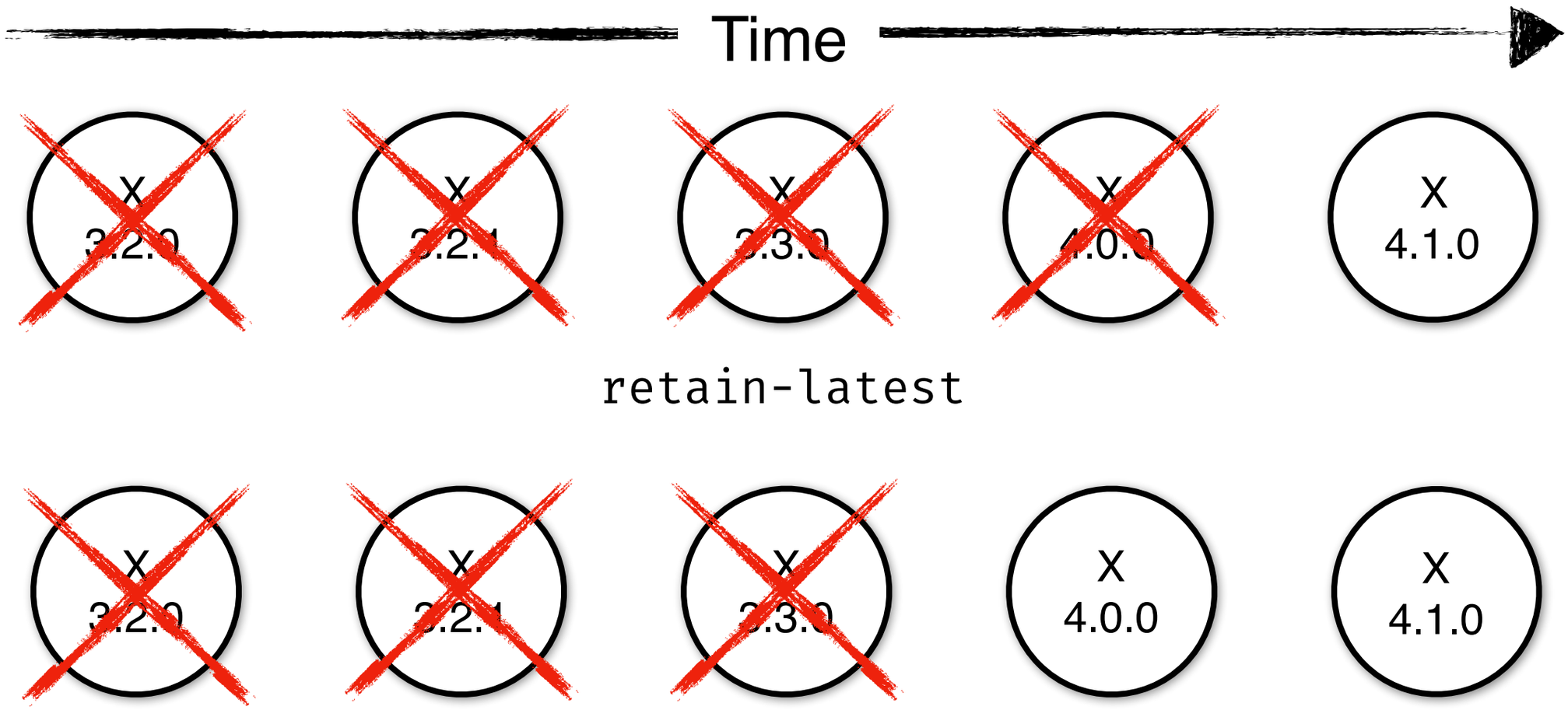
`retain-latest`


Time
X
3.2.0
X
3.2.1
X
3.3.0
X
4.0.0
X
4.1.0
retain-latest

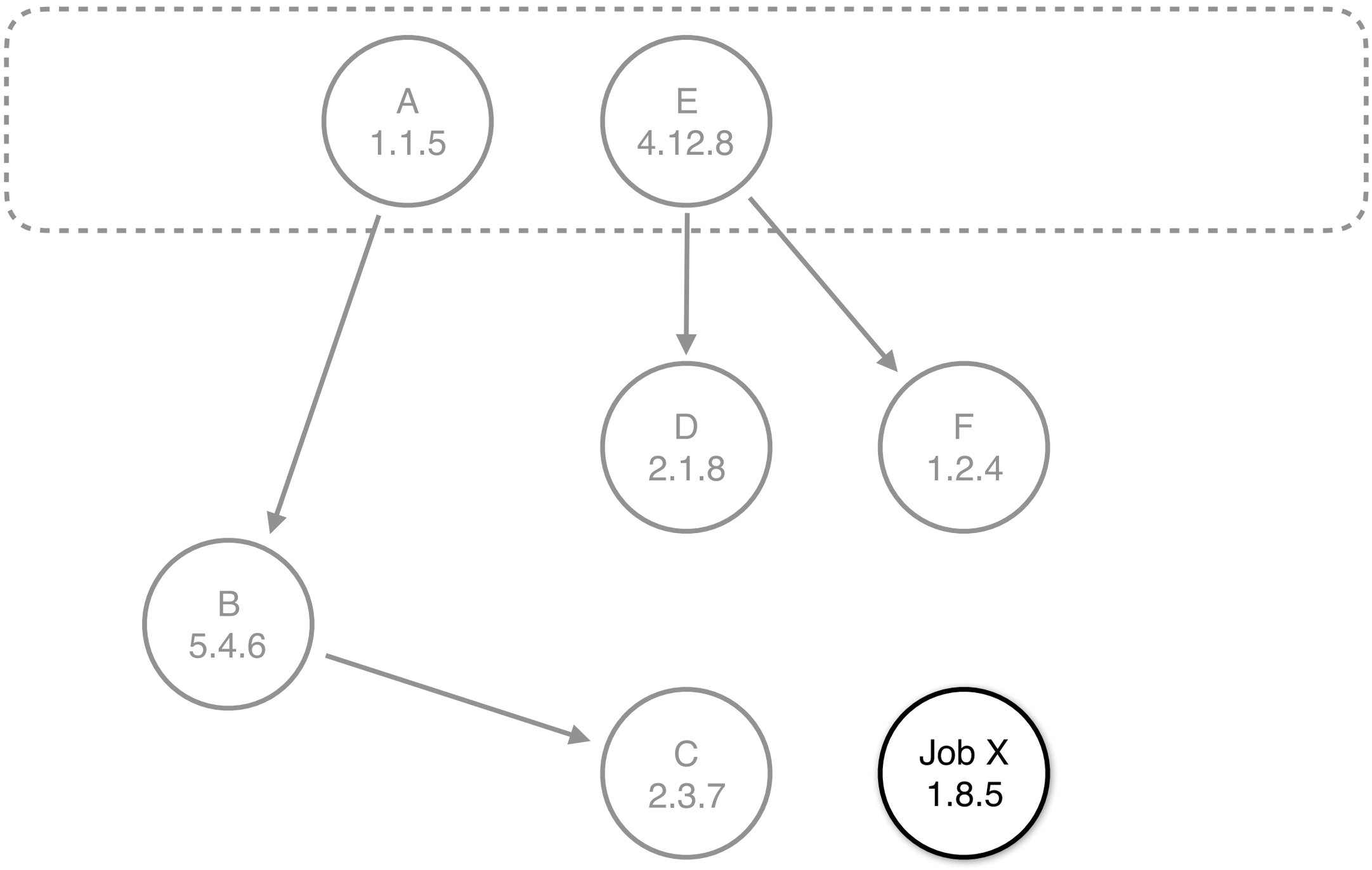


`retain-latest`

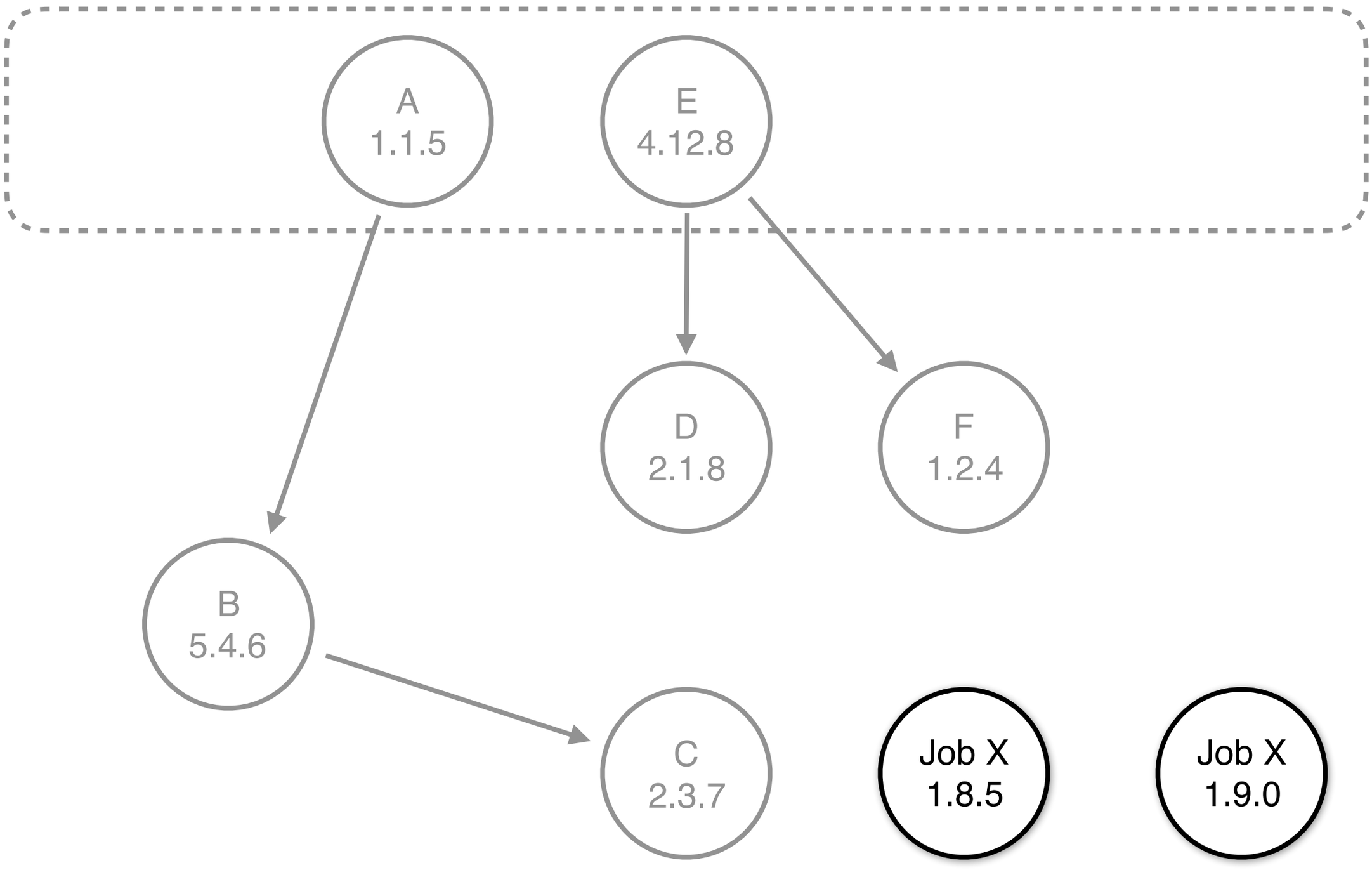

Time
X
3.2.0
X
3.2.1
X
3.3.0
X
4.0.0
X
4.1.0
retain-latest
X
3.2.0
X
3.2.1
X
3.3.0
X
4.0.0
X
4.1.0
retain-latest-two-major


Time
X
3.2.0
X
3.2.1
X
3.3.0
X
4.0.0
X
4.1.0
retain-latest
X
3.2.0
X
3.2.1
X
3.3.0
X
4.0.0
X
4.1.0
retain-latest-two-major
X
3.2.0
X
3.2.1
X
3.3.0
X
4.0.0
X
4.1.0
retain-latest-two-feature

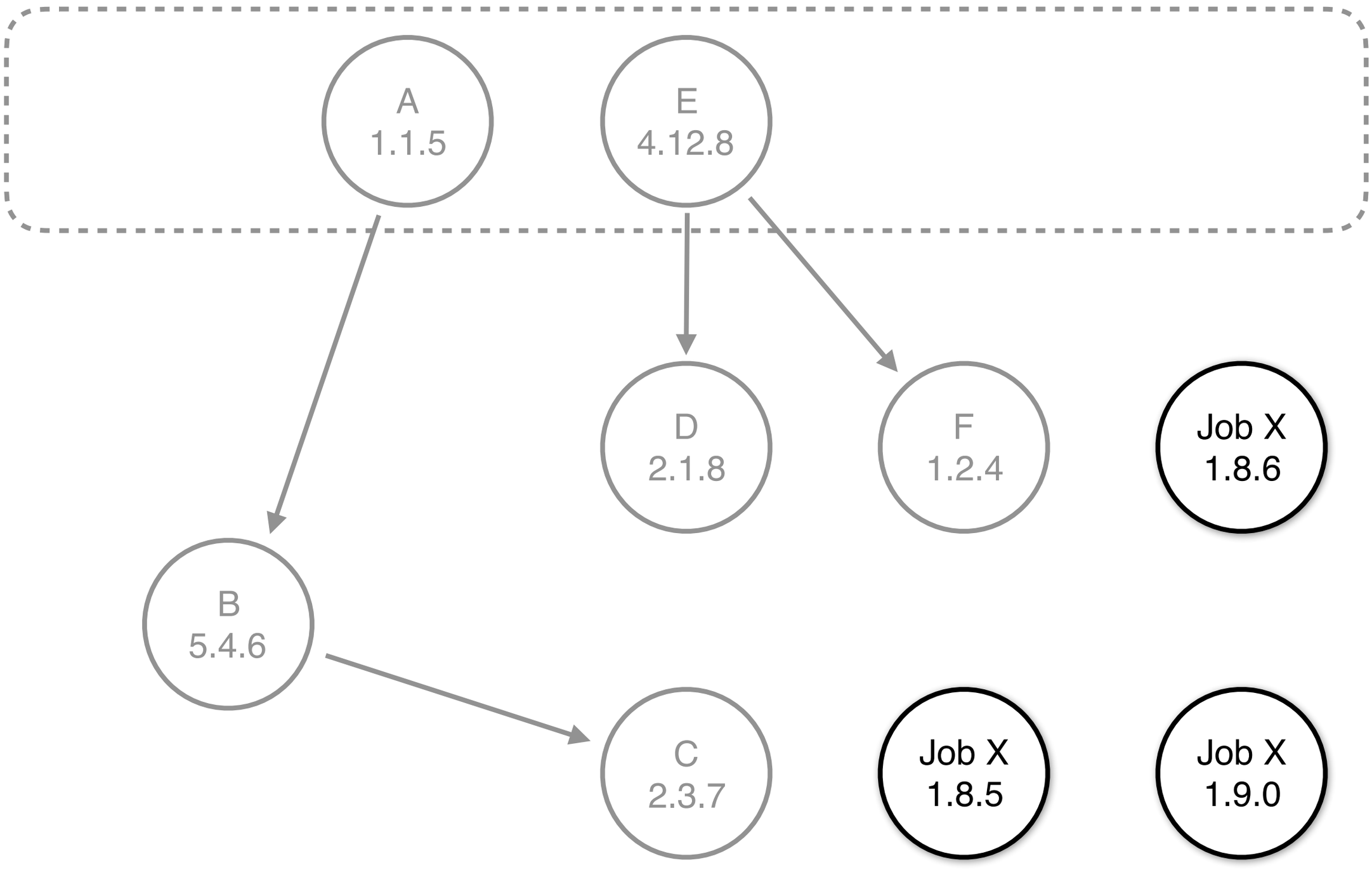
Graph management: pruning



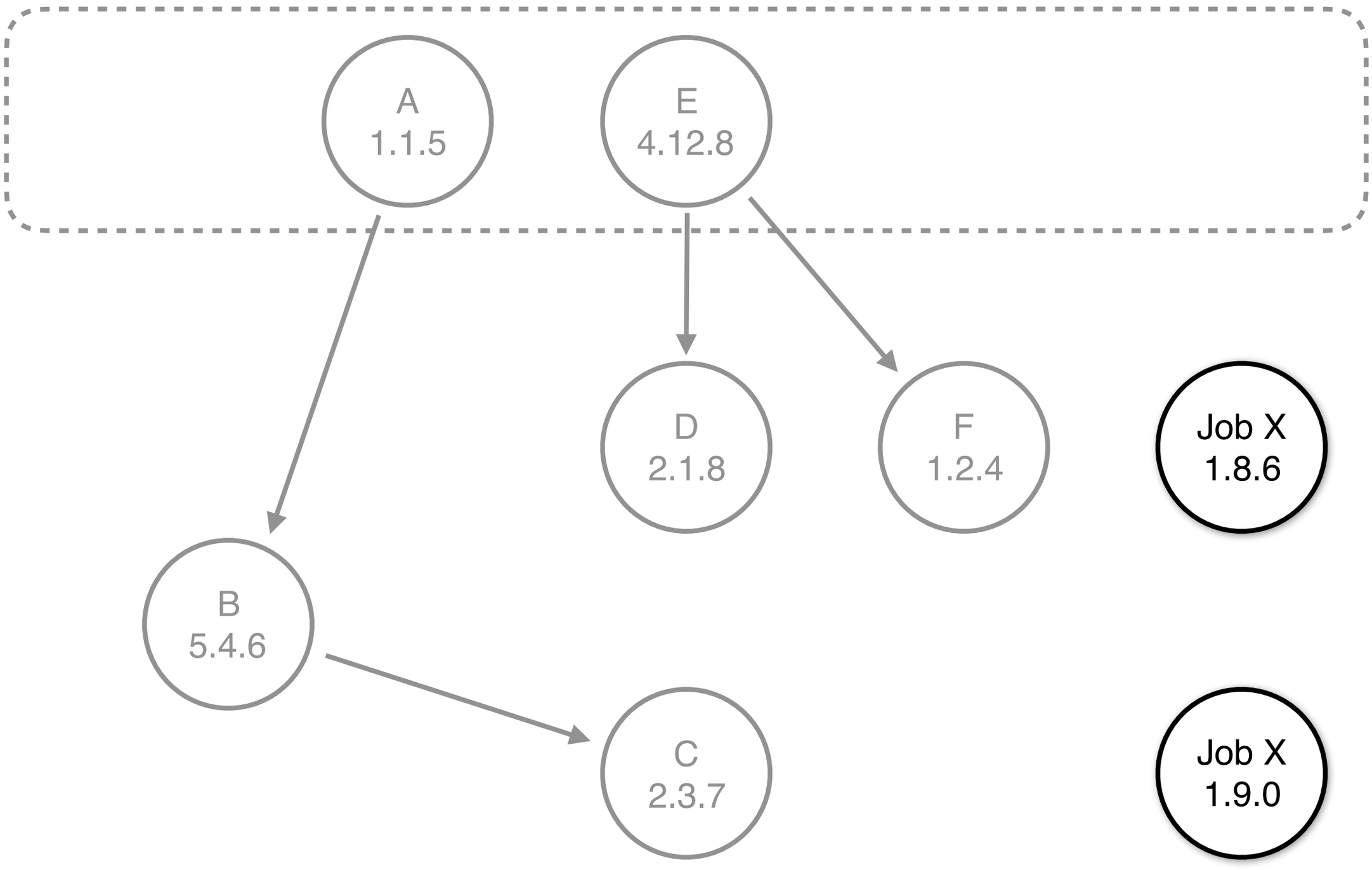
Graph management: pruning



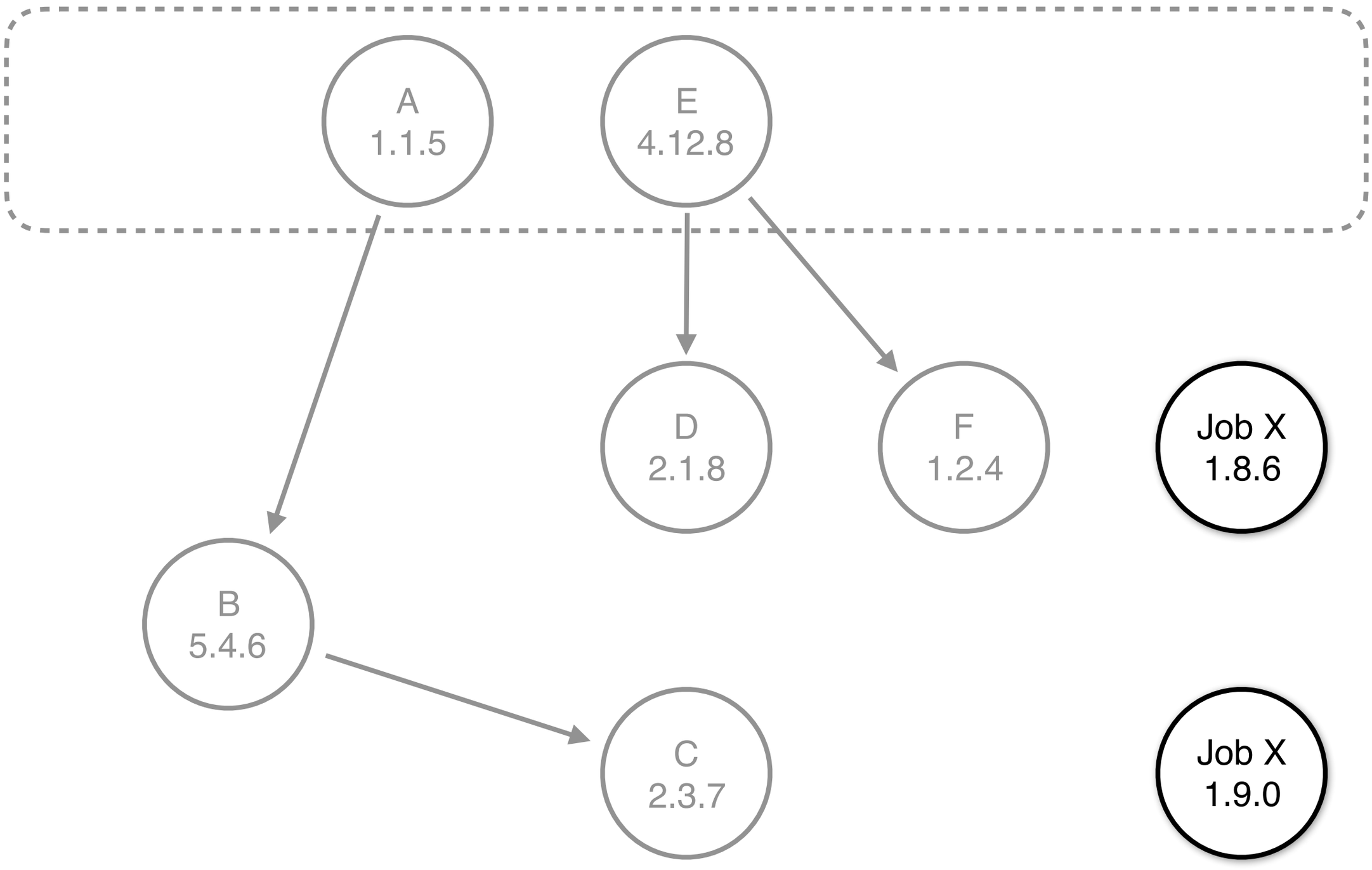
Graph management: pruning



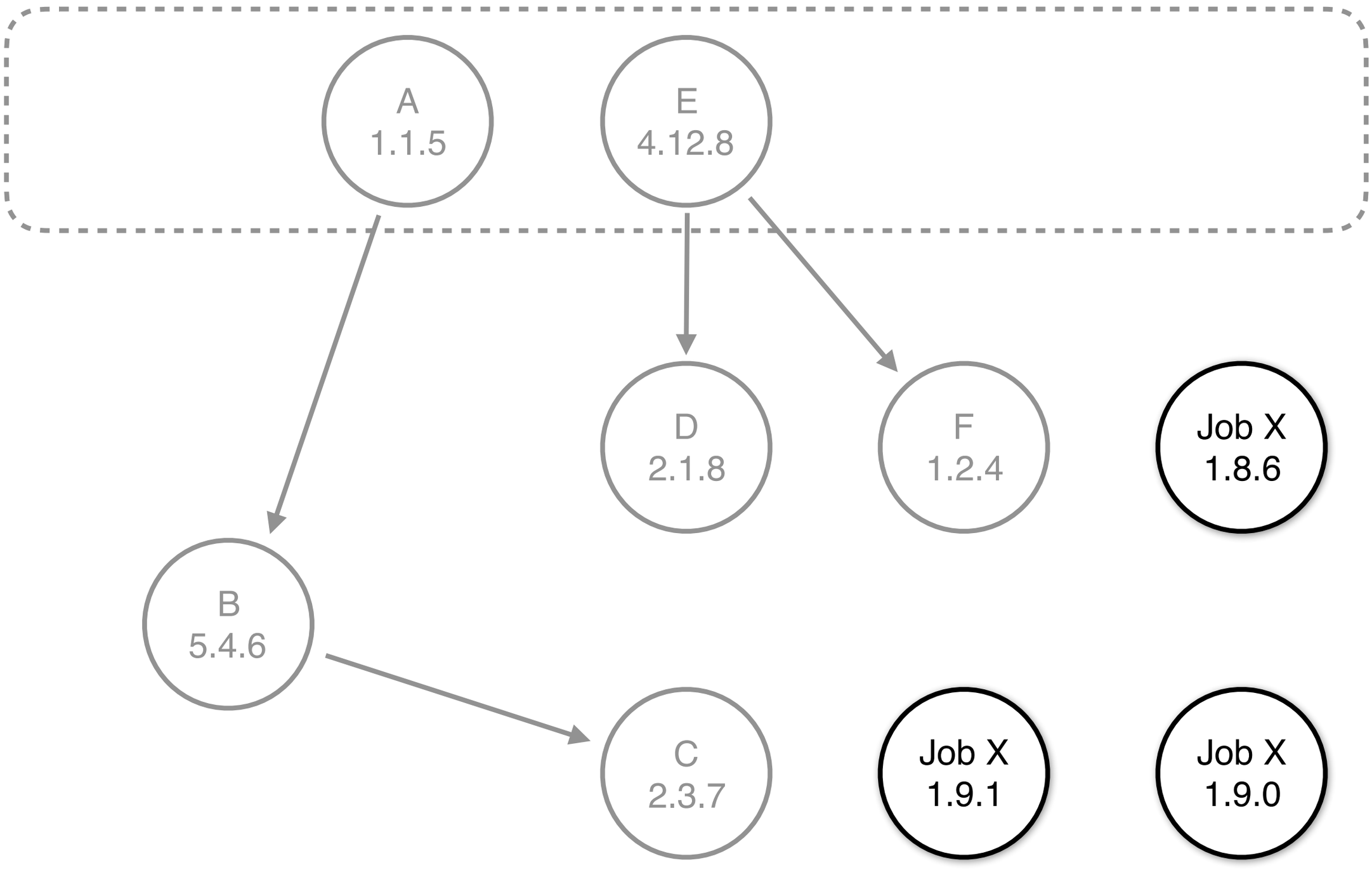
Graph management: pruning



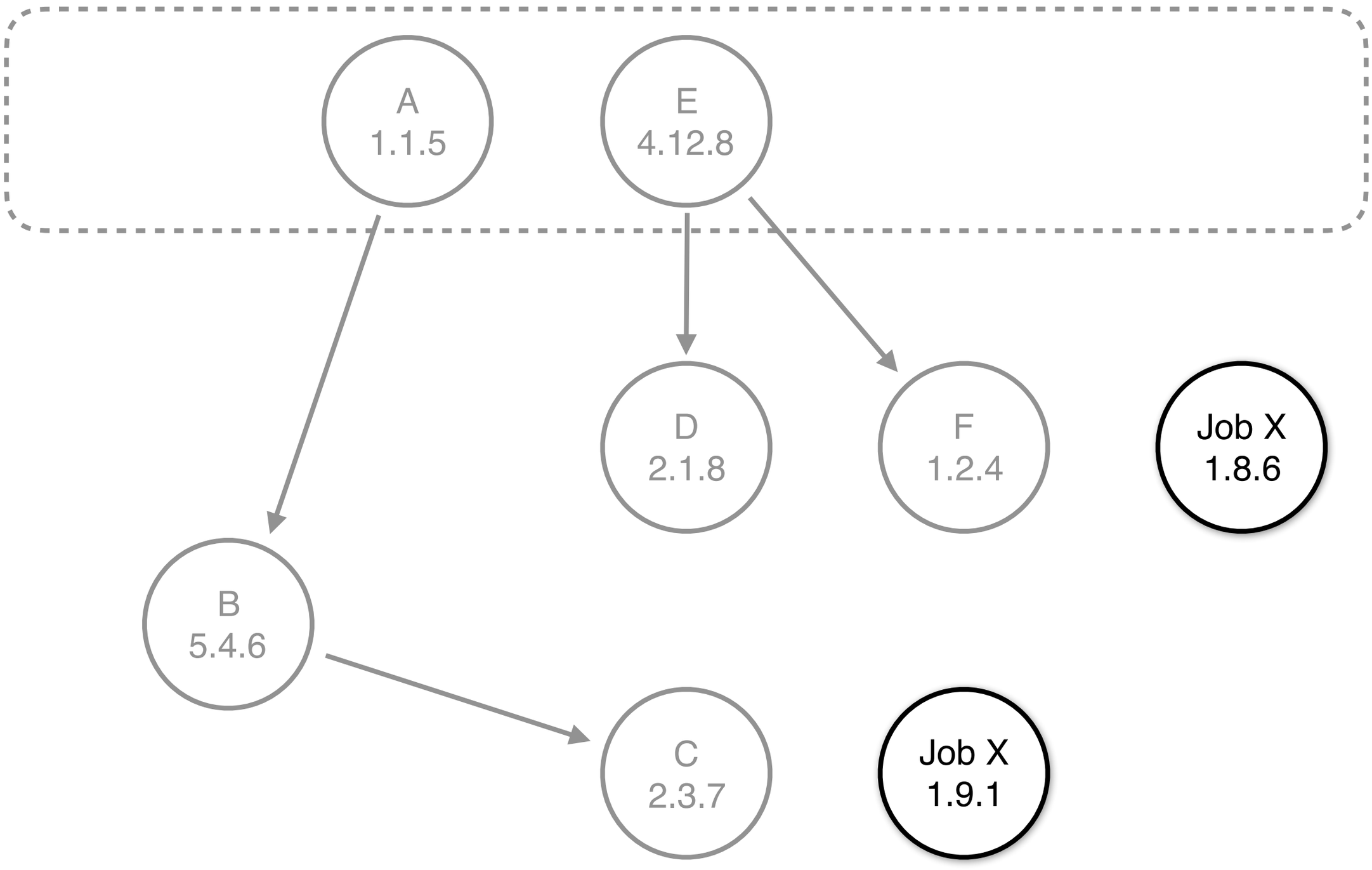
Graph management: pruning

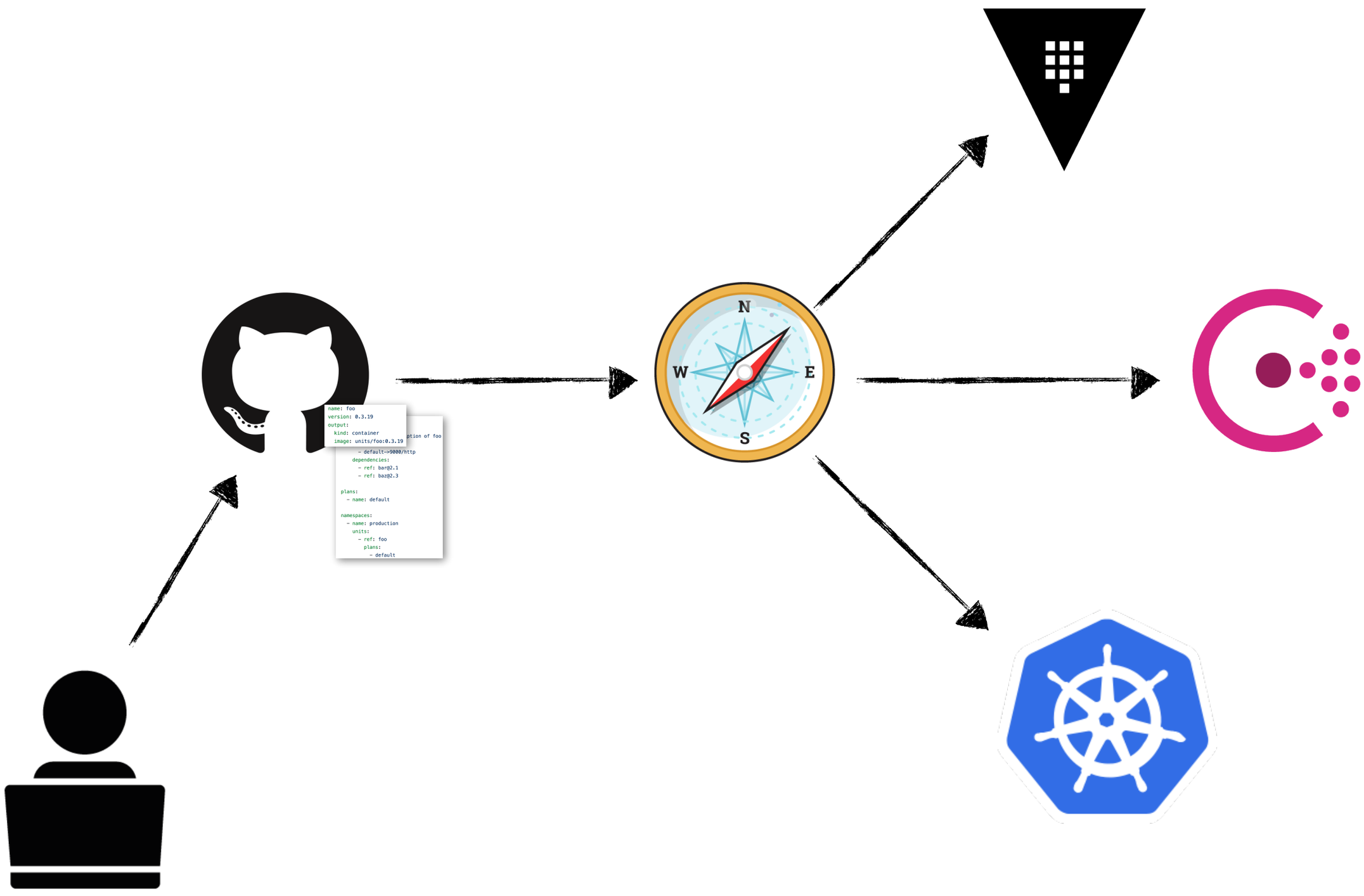


Graph management: pruning



Graph management: pruning




name: foo
version: 0.3.19
output:
kind: container
image: units/foo:0.3.19
- default->9000/http
dependencies:
- ref: bar@2.1
- ref: baz@2.3
plans:
- name: default
namespaces:
- name: production
units:
- ref: foo
plans:
- default
N
W
E
S

Launch workflow

```
for {
    _  <- status(id, Pending, "workflow about to start")
    i  <- dockerOps(id, unit, dc.docker.registry)
```

Launch workflow

```
for {
    _  <- status(id, Pending, "workflow about to start")
    i  <- dockerOps(id, unit, dc.docker.registry)

    _  <- status(id, Deploying, s"writing alert definitions to ${dc.name}'s consul")
    _  <- writeAlertsToConsul(sn, ns.name, p.name, unit, p.environment.alertOptOuts)
```

Launch workflow

```
for {
    _  <- status(id, Pending, "workflow about to start")
    i  <- dockerOps(id, unit, dc.docker.registry)

    _  <- status(id, Deploying, s"writing alert definitions to ${dc.name}'s consul")
    _  <- writeAlertsToConsul(sn, ns.name, p.name, unit, p.environment.alertOptOuts)

    _  <- logToFile(...)
    _  <- writePolicyToVault(cfg = dc.policy, sn = sn, ns = ns.name, rs = rs)
```

Launch workflow

```
for {
    _  <- status(id, Pending, "workflow about to start")
    i  <- dockerOps(id, unit, dc.docker.registry)

    _  <- status(id, Deploying, s"writing alert definitions to ${dc.name}'s consul")
    _  <- writeAlertsToConsul(sn, ns.name, p.name, unit, p.environment.alertOptOuts)

    _  <- logToFile(...)
    _  <- writePolicyToVault(cfg = dc.policy, sn = sn, ns = ns.name, rs = rs)

    _  <- logToFile(...)
    _  <- writeDiscoveryToConsul(id, sn, ns.name, dc)
```

Launch workflow

```
for {
    _  <- status(id, Pending, "workflow about to start")
    i  <- dockerOps(id, unit, dc.docker.registry)

    _  <- status(id, Deploying, s"writing alert definitions to ${dc.name}'s consul")
    _  <- writeAlertsToConsul(sn, ns.name, p.name, unit, p.environment.alertOptOuts)

    _  <- logToFile(...)
    _  <- writePolicyToVault(cfg = dc.policy, sn = sn, ns = ns.name, rs = rs)

    _  <- logToFile(...)
    _  <- writeDiscoveryToConsul(id, sn, ns.name, dc)

    _  <- getTrafficShift.cata(...)
    _  <- logToFile(...)
```

Launch workflow

```
for {
    _  <- status(id, Pending, "workflow about to start")
    i  <- dockerOps(id, unit, dc.docker.registry)

    _  <- status(id, Deploying, s"writing alert definitions to ${dc.name}'s consul")
    _  <- writeAlertsToConsul(sn, ns.name, p.name, unit, p.environment.alertOptOuts)

    _  <- logToFile(...)
    _  <- writePolicyToVault(cfg = dc.policy, sn = sn, ns = ns.name, rs = rs)

    _  <- logToFile(...)
    _  <- writeDiscoveryToConsul(id, sn, ns.name, dc)

    _  <- getTrafficShift.cata(...)
    _  <- logToFile(...)

    l  <- launch(i, dc, ns.name, vunit, p, hash)
    _  <- debug(s"response from scheduler $l")
```

Launch workflow

```
for {
    _  <- status(id, Pending, "workflow about to start")
    i  <- dockerOps(id, unit, dc.docker.registry)

    _  <- status(id, Deploying, s"writing alert definitions to ${dc.name}'s consul")
    _  <- writeAlertsToConsul(sn, ns.name, p.name, unit, p.environment.alertOptOuts)

    _  <- logToFile(...)
    _  <- writePolicyToVault(cfg = dc.policy, sn = sn, ns = ns.name, rs = rs)

    _  <- logToFile(...)
    _  <- writeDiscoveryToConsul(id, sn, ns.name, dc)

    _  <- getTrafficShift.cata(...)
    _  <- logToFile(...)

    l  <- launch(i, dc, ns.name, vunit, p, hash)
    _  <- debug(s"response from scheduler $l")

    _  <- status(id, getStatus(unit, p), "======> workflow completed <======")
} yield ()
```

Delete workflow

```
logToFile(d.id, s"removing policy from vault: ${vaultLoggingFields(...)}") >>
deletePolicyFromVault(d.stackName, ns.name) >>
```

Delete workflow

```
logToFile(d.id, s"removing policy from vault: ${vaultLoggingFields(...)}") >>
deletePolicyFromVault(d.stackName, ns.name) >>

logToFile(d.id, s"removing alerts from consul ${alerts.alertingKey(sn)}") >>
deleteAlertsFromConsul(d.stackName) >>
```

Delete workflow

```
logToFile(d.id, s"removing policy from vault: ${vaultLoggingFields(...)}") >>
deletePolicyFromVault(d.stackName, ns.name) >>

logToFile(d.id, s"removing alerts from consul ${alerts.alertingKey(sn)}") >>
deleteAlertsFromConsul(d.stackName) >>

logToFile(d.id, ...) >>
deleteDiscoveryInfoFromConsul(sn) >>
```

Delete workflow

```
logToFile(d.id, s"removing policy from vault: ${vaultLoggingFields(...)}") >>
deletePolicyFromVault(d.stackName, ns.name) >>

logToFile(d.id, s"removing alerts from consul ${alerts.alertingKey(sn)}") >>
deleteAlertsFromConsul(d.stackName) >>

logToFile(d.id, ...) >>
deleteDiscoveryInfoFromConsul(sn) >>

logToFile(d.id, s"instructing ${dc.name}'s scheduler to decommission ${sn}") >>
delete(dc,d) >>
```

Delete workflow

```
logToFile(d.id, s"removing policy from vault: ${vaultLoggingFields(...)}") >>
deletePolicyFromVault(d.stackName, ns.name) >>

logToFile(d.id, s"removing alerts from consul ${alerts.alertingKey(sn)}") >>
deleteAlertsFromConsul(d.stackName) >>

logToFile(d.id, ...) >>
deleteDiscoveryInfoFromConsul(sn) >>

logToFile(d.id, s"instructing ${dc.name}'s scheduler to decommission ${sn}") >>
delete(dc,d) >>

status(d.id, Terminated, s"Decommissioning deployment ${sn} in ${dc.name}")
```

Background processes

```
runBackgroundJob("auditor", cfg.auditor.process(cfg.storage))
```

Background processes

```
runBackgroundJob("auditor", cfg.auditor.process(cfg.storage))

runBackgroundJob("pipeline_processor",
                 Process.eval(Pipeline.task(cfg)(Pipeline.sinks.runAction(cfg))))
```

Background processes

```
runBackgroundJob("auditor", cfg.auditor.process(cfg.storage))

runBackgroundJob("pipeline_processor",
                 Process.eval(Pipeline.task(cfg)(Pipeline.sinks.runAction(cfg))))

runBackgroundJob("workflow_logger", cfg.workflowLogger.process)
```

Background processes

```
runBackgroundJob("auditor", cfg.auditor.process(cfg.storage))

runBackgroundJob("pipeline_processor",
                 Process.eval(Pipeline.task(cfg)(Pipeline.sinks.runAction(cfg))))

runBackgroundJob("workflow_logger", cfg.workflowLogger.process)

runBackgroundJob("routing_cron",
                 routing.cron.consulRefresh(cfg) to Http4sConsul.consulSink)
```

Background processes

```
runBackgroundJob("auditor", cfg.auditor.process(cfg.storage))

runBackgroundJob("pipeline_processor",
                 Process.eval(Pipeline.task(cfg)(Pipeline.sinks.runAction(cfg))))

runBackgroundJob("workflow_logger", cfg.workflowLogger.process)

runBackgroundJob("routing_cron",
                 routing.cron.consulRefresh(cfg) to Http4sConsul.consulSink)

runBackgroundJob("cleanup_pipeline", cleanup.CleanupCron.pipeline(cfg))
```

Background processes

```
runBackgroundJob("auditor", cfg.auditor.process(cfg.storage))

runBackgroundJob("pipeline_processor",
                 Process.eval(Pipeline.task(cfg)(Pipeline.sinks.runAction(cfg))))

runBackgroundJob("workflow_logger", cfg.workflowLogger.process)

runBackgroundJob("routing_cron",
                 routing.cron.consulRefresh(cfg) to Http4sConsul.consulSink)

runBackgroundJob("cleanup_pipeline", cleanup.CleanupCron.pipeline(cfg))

runBackgroundJob("sweeper", cleanup.Sweeper.process(cfg))
```

Background processes

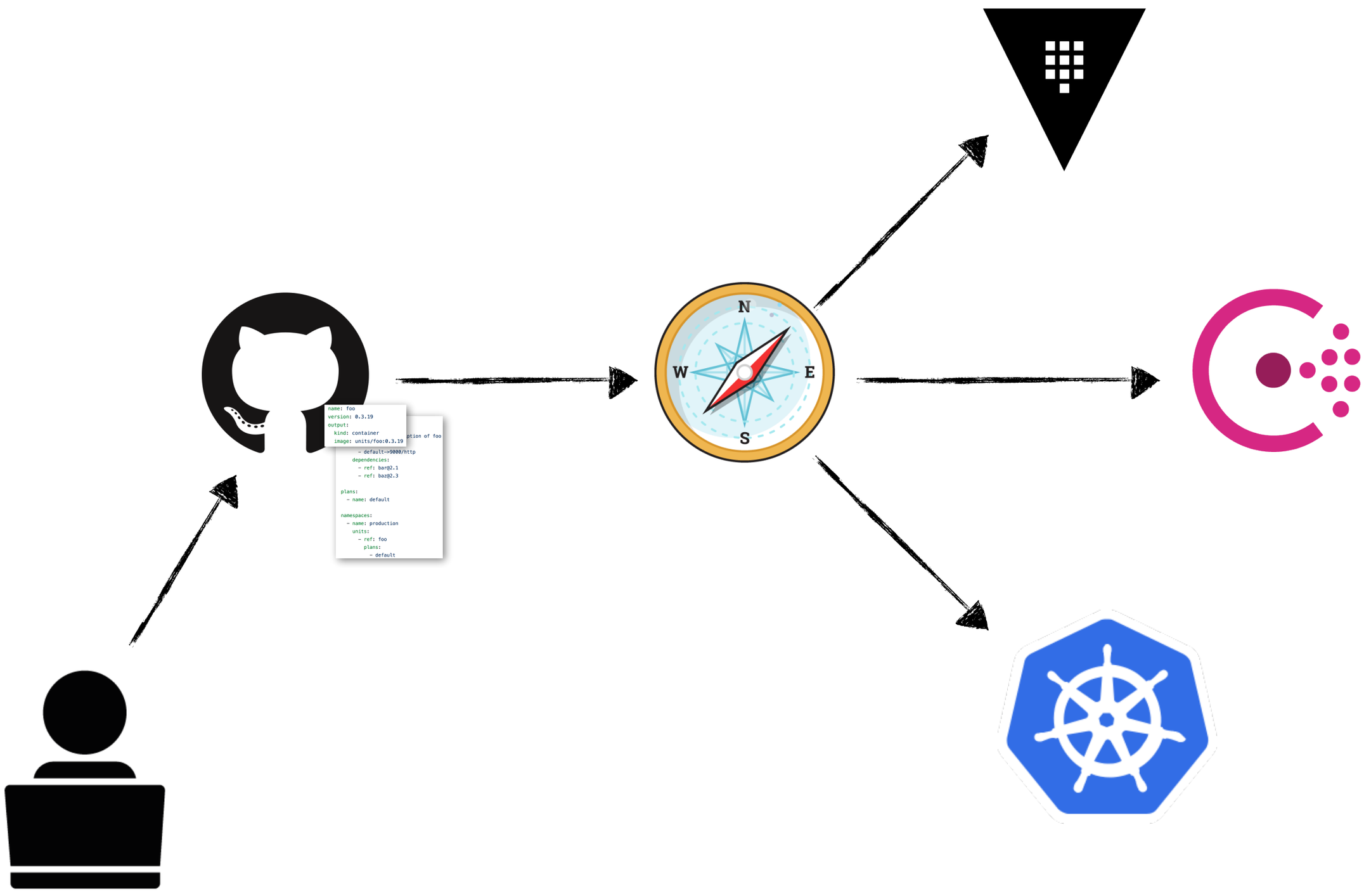
```
runBackgroundJob("auditor", cfg.auditor.process(cfg.storage))

runBackgroundJob("pipeline_processor",
                 Process.eval(Pipeline.task(cfg)(Pipeline.sinks.runAction(cfg))))

runBackgroundJob("workflow_logger", cfg.workflowLogger.process)

runBackgroundJob("routing_cron",
                 routing.cron.consulRefresh(cfg) to Http4sConsul.consulSink)

runBackgroundJob("cleanup_pipeline", cleanup.CleanupCron.pipeline(cfg))

runBackgroundJob("sweeper", cleanup.Sweeper.process(cfg))

runBackgroundJob("deployment_monitor", DeploymentMonitor.loop(cfg))
```

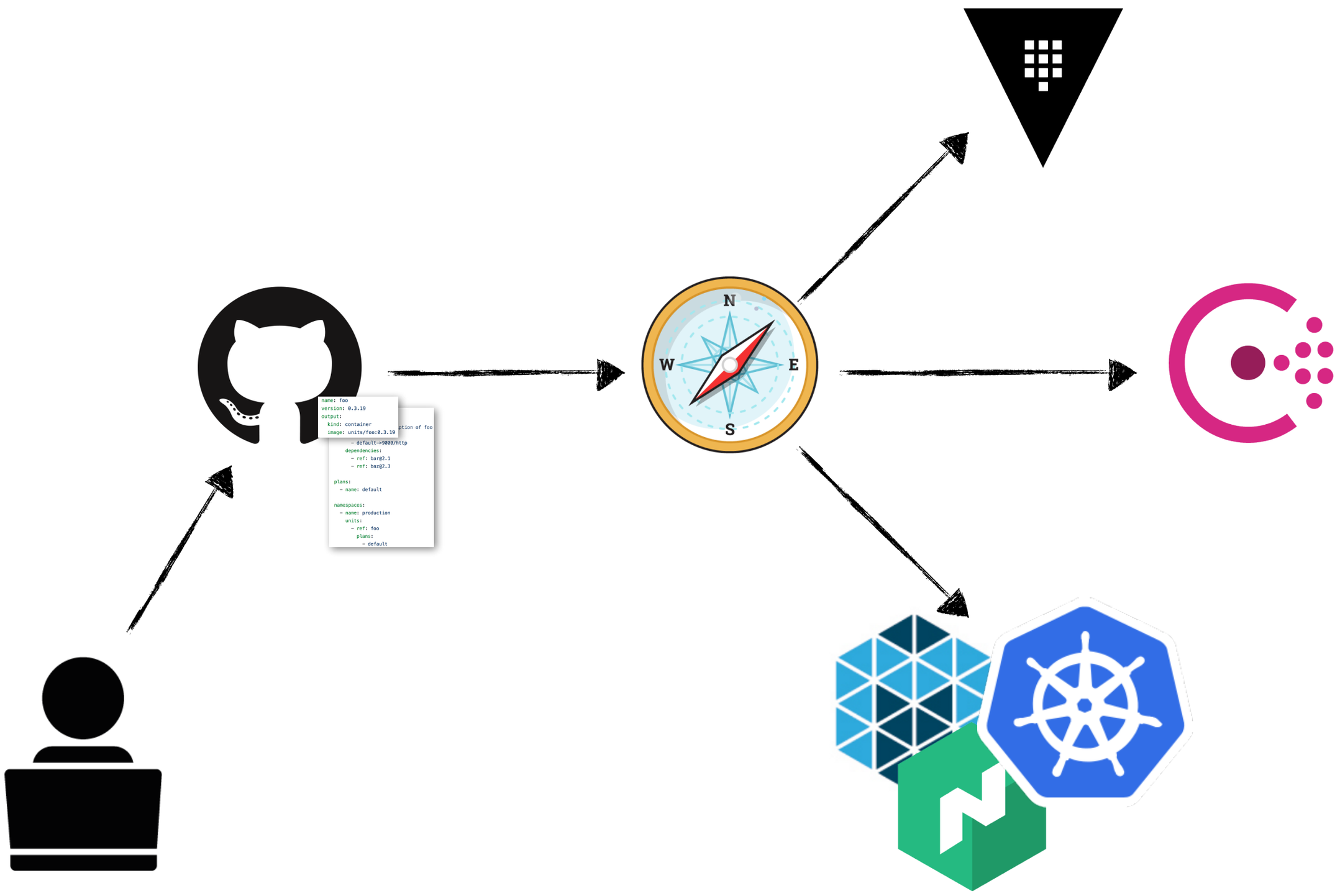
Workflow algebra

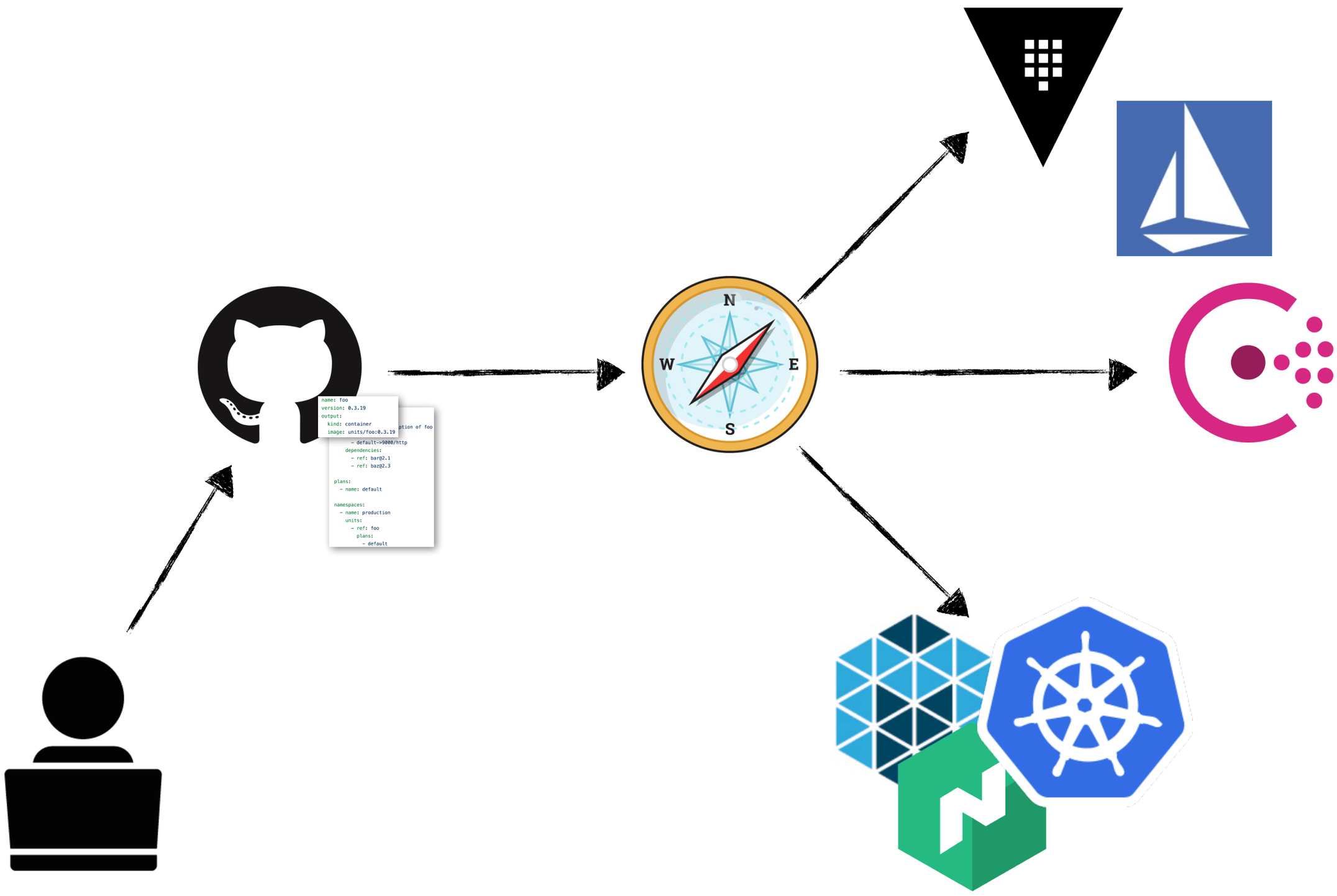
```
type Op0[A] = Coproduct[DockerOp, ConsulOp, A]

type Op1[A] = Coproduct[LoggingOp, Op0,A]

type Op2[A] = Coproduct[StoreOp, Op1, A]

type Op3[A] = Coproduct[WorkflowControlOp, Op2,A]

type Op4[A] = Coproduct[Vault, Op3, A]

type WorkflowOp[A] = Coproduct[SchedulerOp, Op4, A]

type WorkflowF[A] = Free.FreeC[WorkflowOp, A]
```

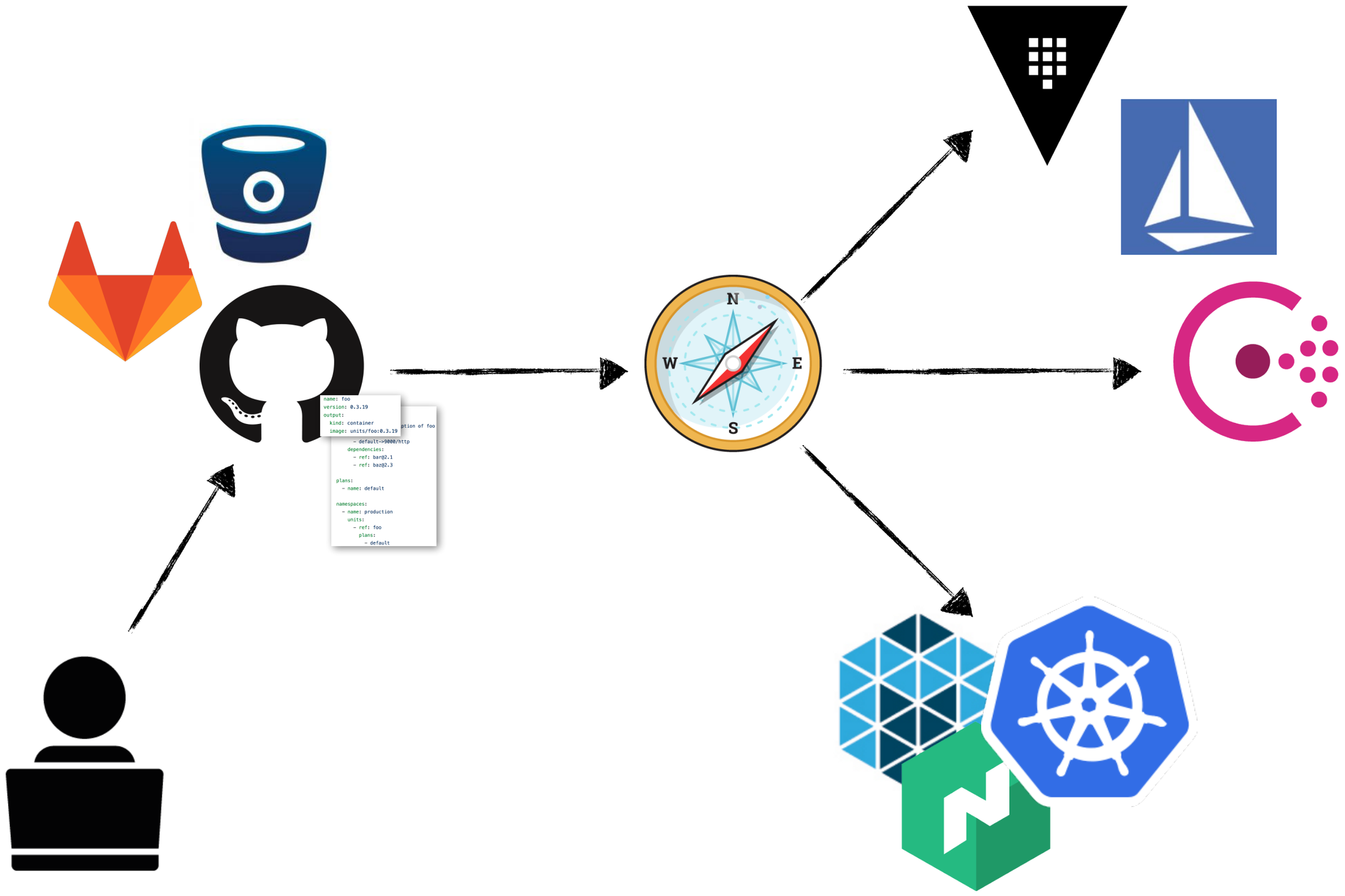
Workflow interpreter(s)

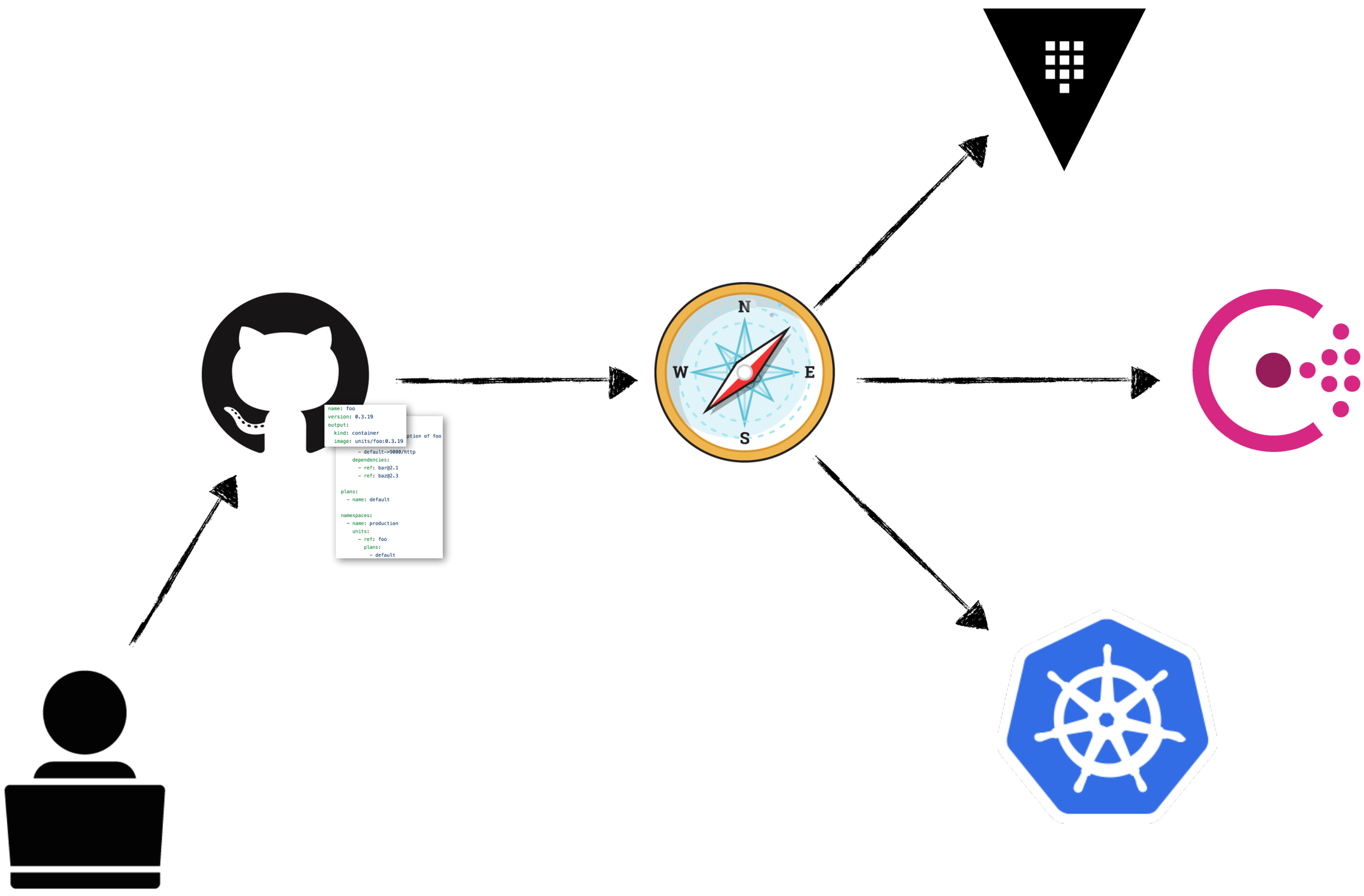
```
final case class Interpreters(
  scheduler: SchedulerOp        ~> Task,
  consul:    ConsulOp           ~> Task,
  vault:     Vault              ~> Task,
  storage:   StoreOp            ~> Task,
  logger:    LoggingOp          ~> Task,
  docker:    DockerOp           ~> Task,
  control:   WorkflowControlOp  ~> Task,
  health:    HealthCheckOp      ~> Task
)
```

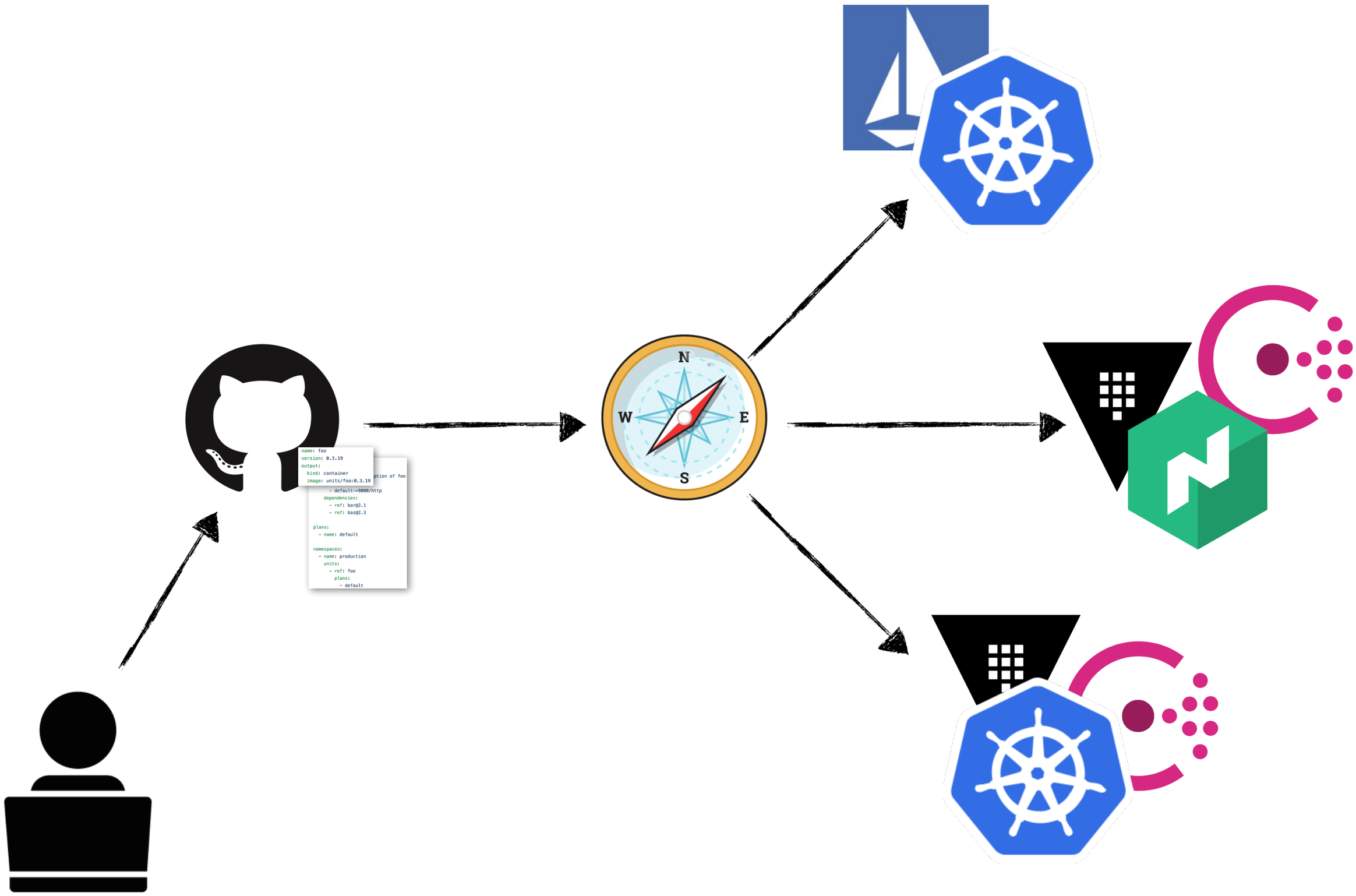

name: foo
version: 0.3.19
output:
kind: container
image: units/foo:0.3.19
- default->9000/http
dependencies:
- ref: bar@2.1
- ref: baz@2.3
plans:
- name: default
namespaces:
- name: production
units:
- ref: foo
plans:
- default
N
W
E
S

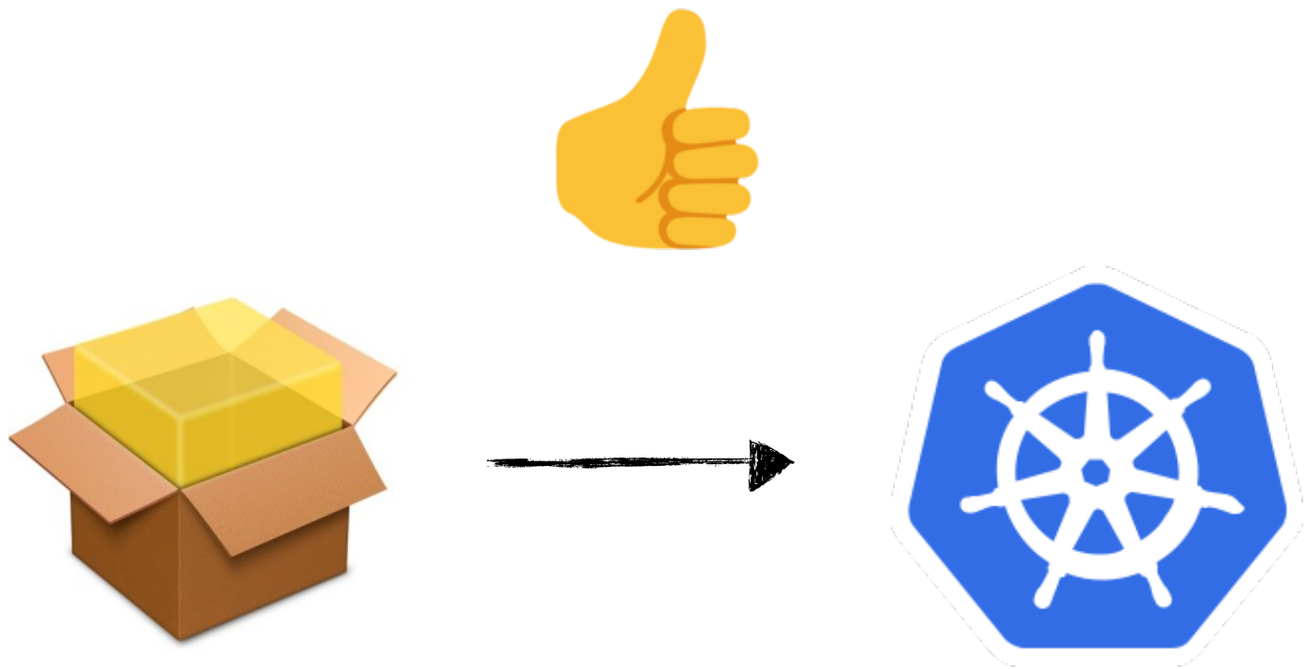

N
W
E
S
name: foo
version: 0.3.19
output:
kind: container
image: units/foo:0.3.19
- default->9000/http
dependencies:
- ref: bar@2.1
- ref: baz@2.3
plans:
- name: default
namespaces:
- name: production
units:
- ref: foo
plans:
- default

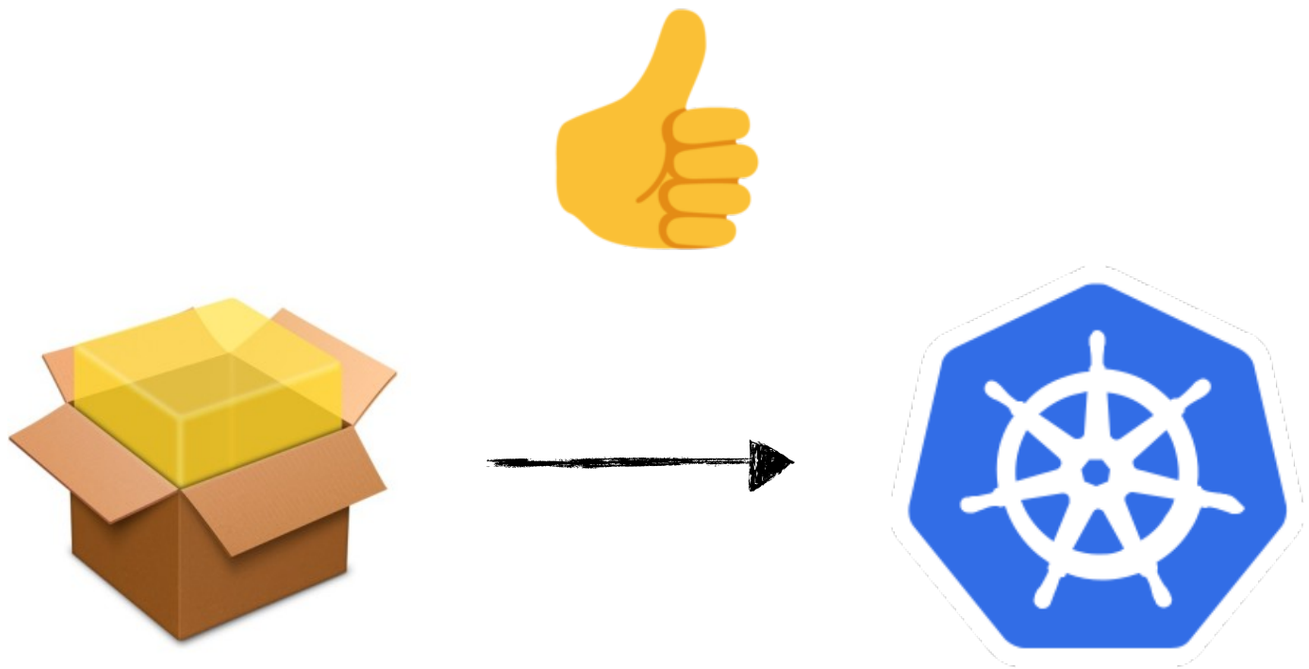

name: foo
version: 0.3.19
output:
kind: container
image: units/foo:0.3.19
- default->9000/http
dependencies:
- ref: bar@2.1
- ref: baz@2.3
plans:
- name: default
namespaces:
- name: production
units:
- ref: foo
plans:
- default
N
W
E
S

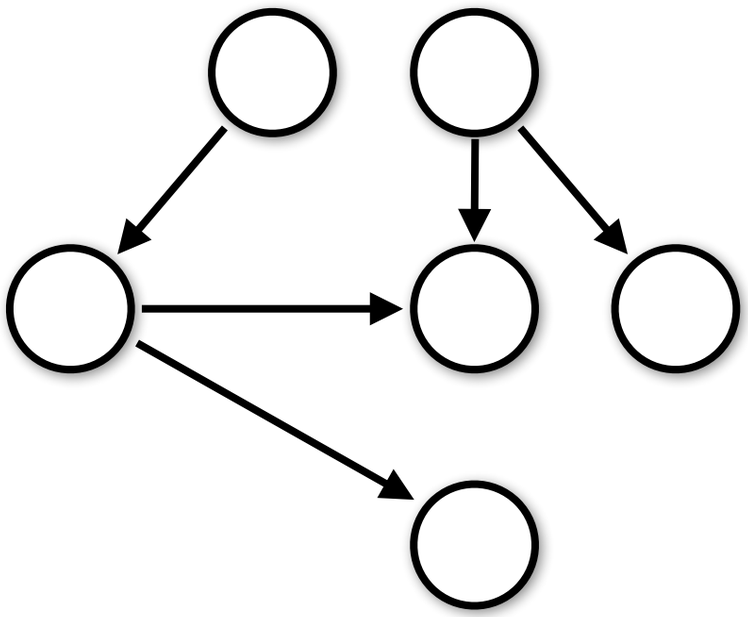

name: foo
version: 0.3.19
output:
kind: container
image: units/foo:0.3.19
- default->9000/http
dependencies:
- ref: bar@2.1
- ref: baz@2.3
plans:
- name: default
namespaces:
- name: production
units:
- ref: foo
plans:
- default
N
W
E
S

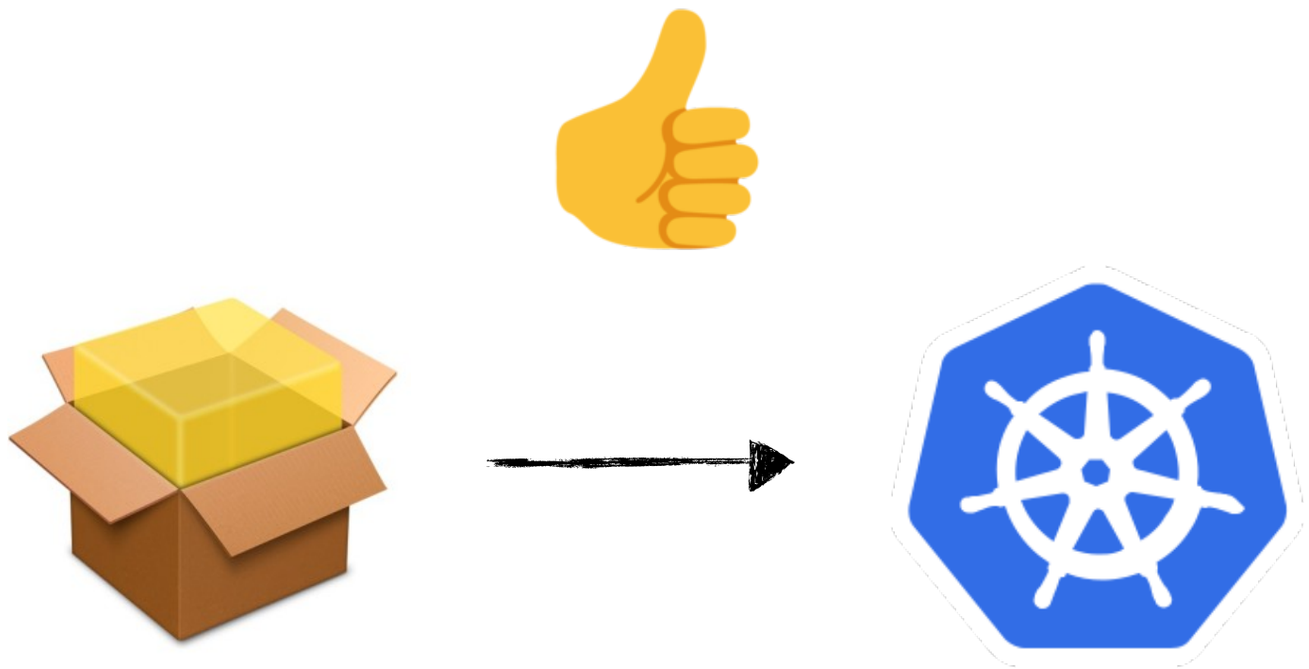

name: foo
version: 0.3.19
output:
kind: container
image: units/foo:0.3.19
- default->9000/http
dependencies:
- ref: bar@2.1
- ref: baz@2.3
plans:
- name: default
namespaces:
- name: production
units:
- ref: foo
plans:
- default
N
W
E
S

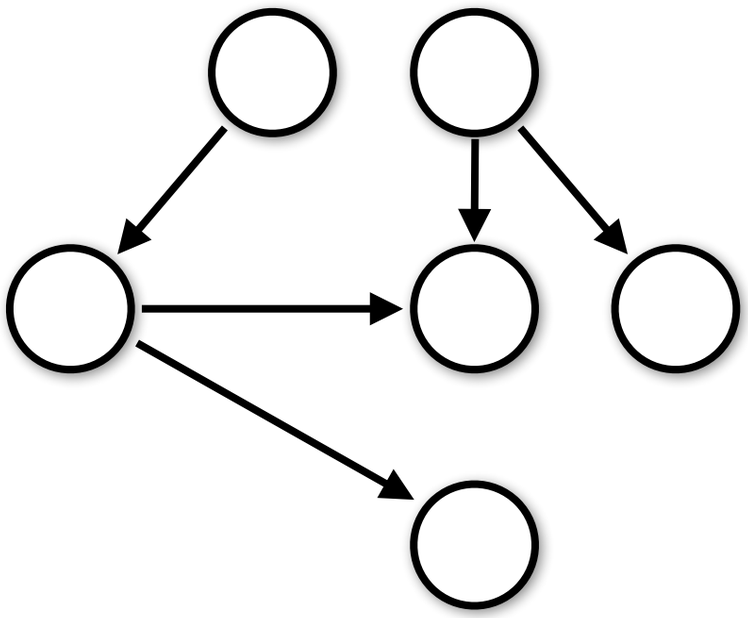

name: foo
version: 0.3.19
output:
kind: container
image: units/foo:0.3.19
- default->9000/http
dependencies:
- ref: bar@2.1
- ref: baz@2.3
plans:
- name: default
namespaces:
- name: production
units:
- ref: foo
plans:
- default
N
W
E
S

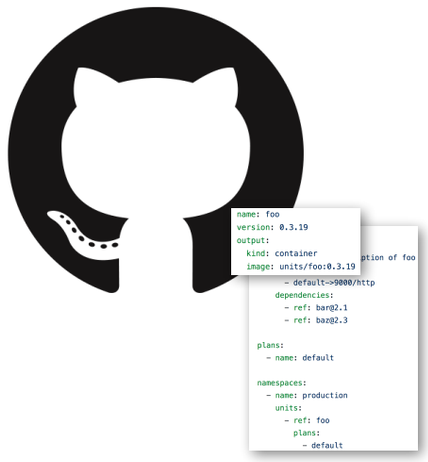


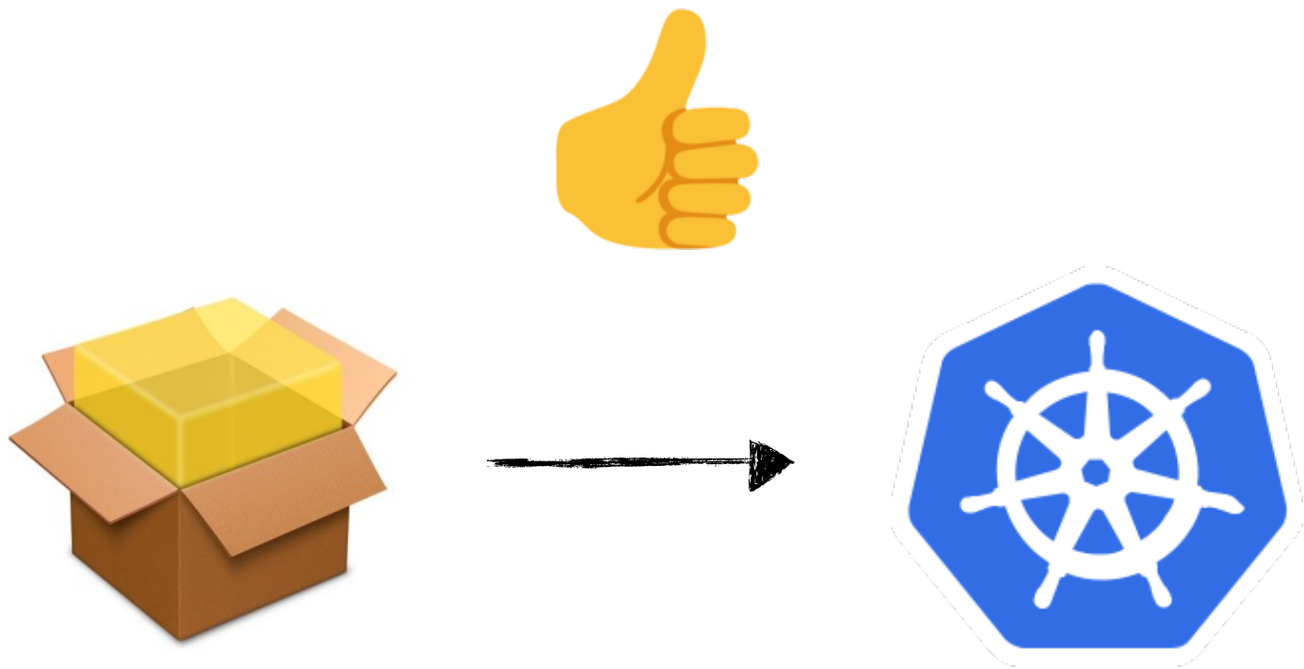


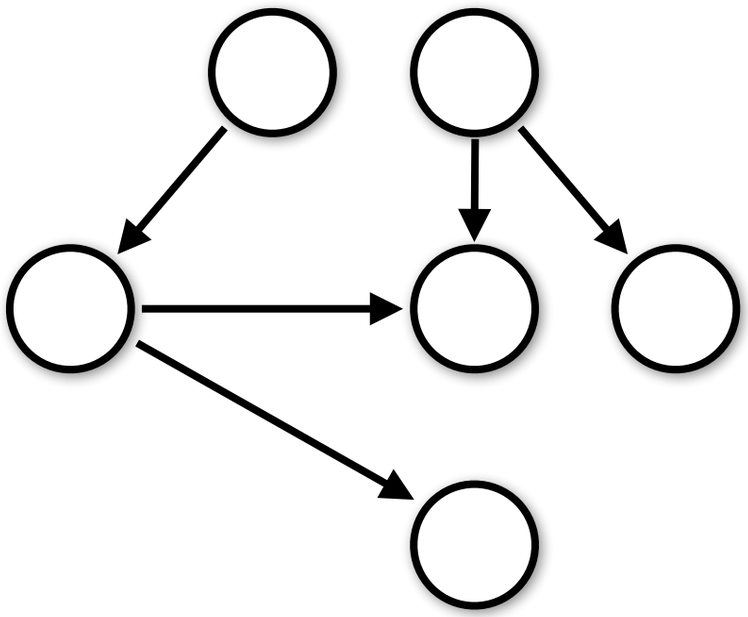


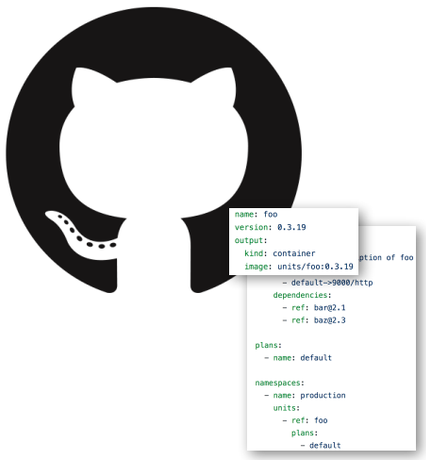


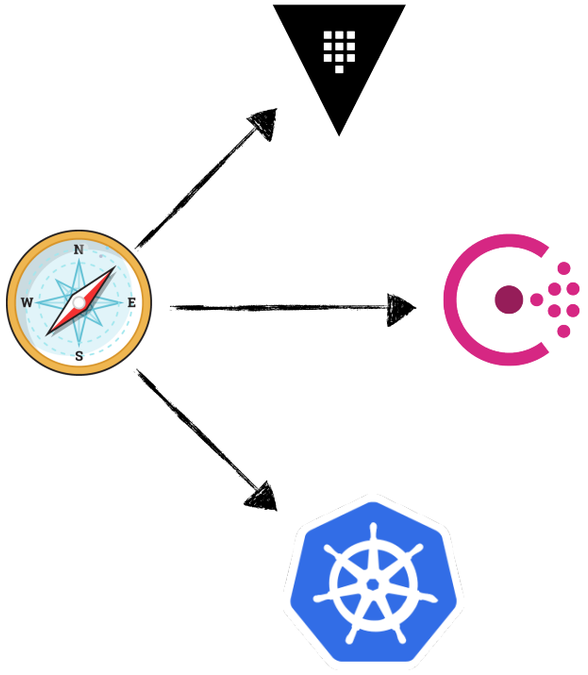



name: foo
version: 0.3.19
output:
kind: container
image: units/foo:0.3.19
- default->9000/http
dependencies:
- ref: bar@2.1
- ref: baz@2.3
plans:
- name: default
namespaces:
- name: production
units:
- ref: foo
plans:
- default






name: foo
version: 0.3.19
output:
kind: container
image: units/foo:0.3.19
dependencies:
- ref: bar@2.1
- ref: baz@2.3
plans:
- name: default
namespaces:
- name: production
units:
- ref: foo
plans:
- default


N
W
E
S

https://getnelson.github.io/

https://github.com/getnelson/

https://gitter.im/getnelson/nelson