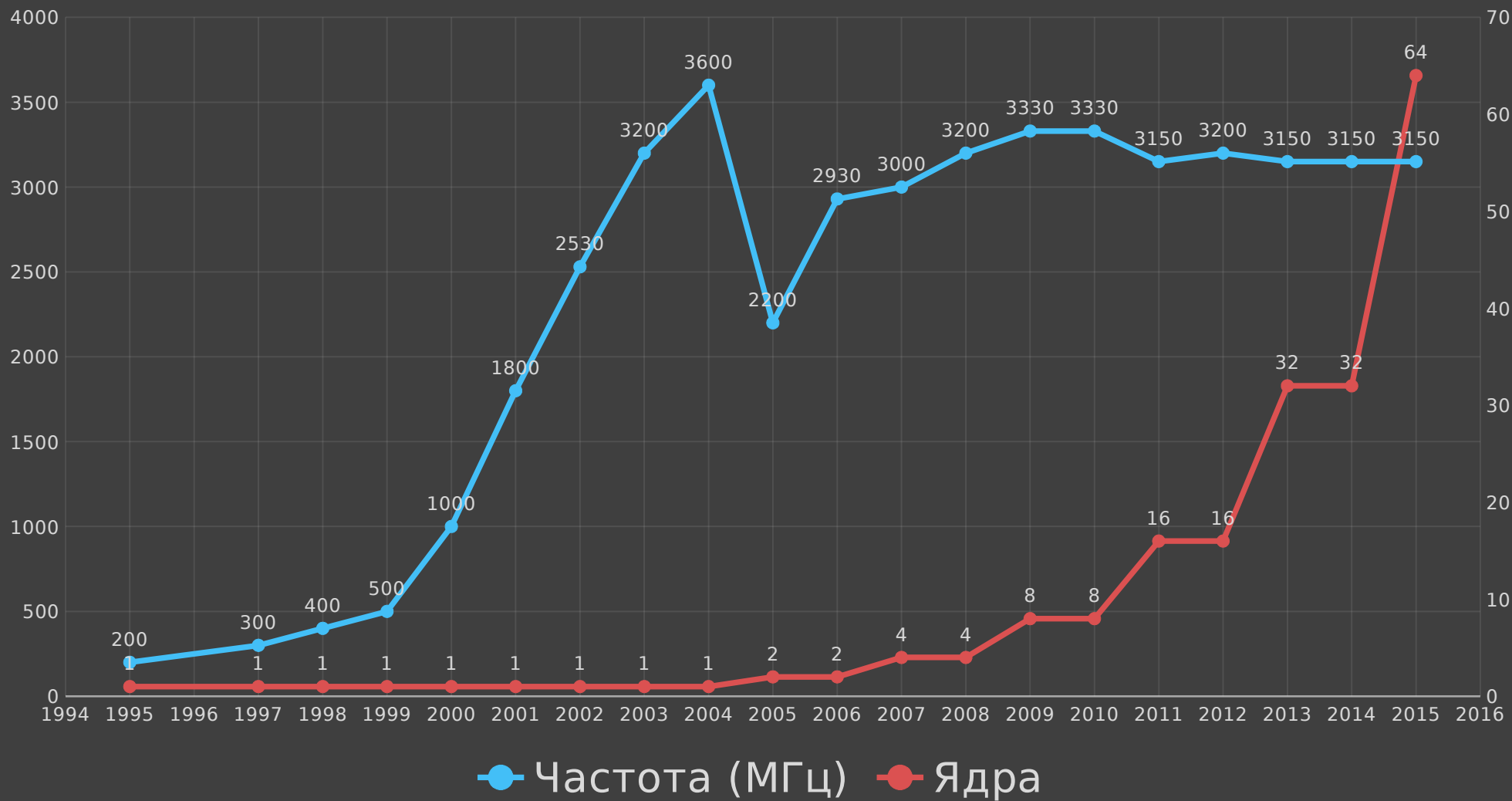
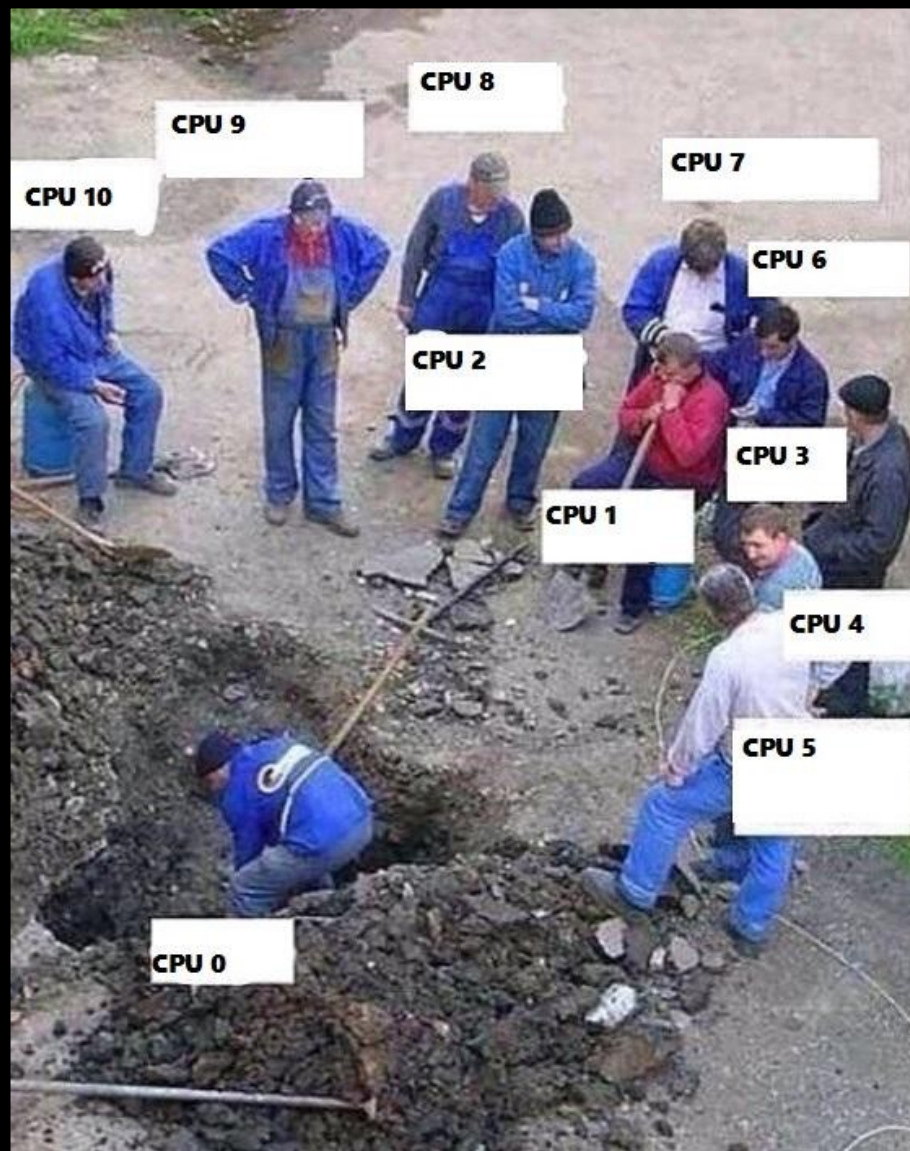




Akka.NET

Тактовая частота и количество ядер по годам





Многопоточность

Ожидания

Реальность



Вертикальное масштабирование



Вертикальное
масштабирование



Горизонтальное
масштабирование



Вертикальное
масштабирование



Горизонтальное
масштабирование



Эластичность



Вертикальное
масштабирование



Горизонтальное
масштабирование



Эластичность



Parallel Linq
TPL – async/await
Потоки

Вертикальное
масштабирование



Горизонтальное
масштабирование



Эластичность



Parallel Linq
TPL – async/await
Потоки

WCF
Web API
ServiceStack
MSMQ

Вертикальное
масштабирование



Горизонтальное
масштабирование



Эластичность



Parallel Linq
TPL – async/await
Потоки

WCF
Web API
ServiceStack
MSMQ



Что такое Актор?

Что такое Актор?

Это — единица организации программного кода

Что такое Актор?

Это — единица организации программного кода

ООП

Поведение
Состояние
Синхронные* вызовы

Акторы

Поведение
Состояние
Асинхронные сообщения

Что такое Актор?

Это — единица организации программного кода

ООП

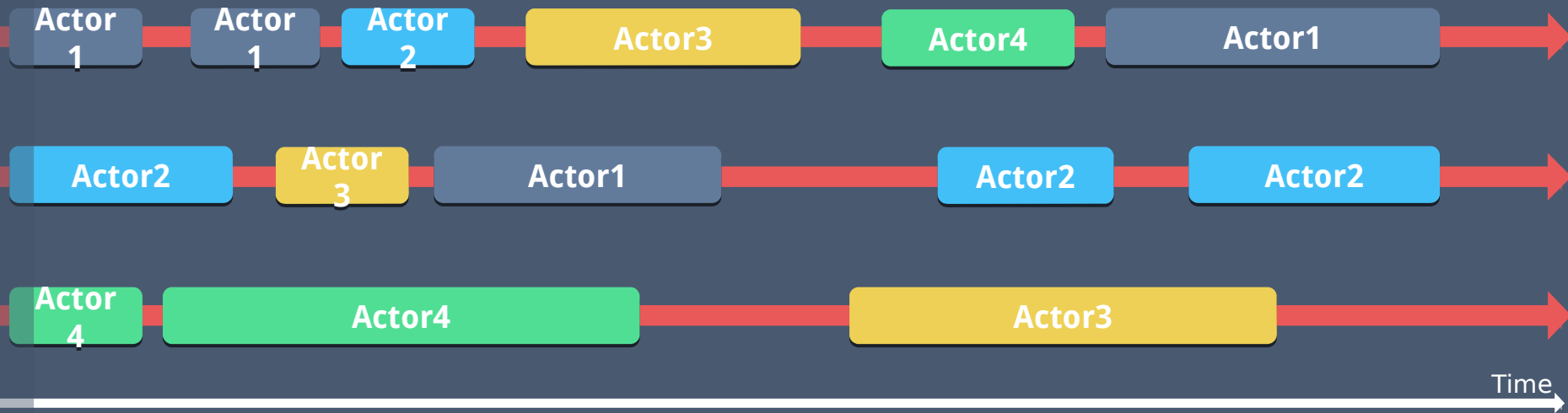
Поведение
Состояние
Синхронные* вызовы

Акторы

Поведение
Состояние
Асинхронные сообщения

Не нужно думать о:

- разделяемом состоянии
- видимости состояния
- потоках, блокировках, конкурентных коллекциях, и т. п.



- Акторы работают в фиксированном пуле потоков, равномерно распределяя нагрузку между ядрами, нет лишних переключений контекста.
- Есть возможность управлять тем, где и как будет выполняться актор.

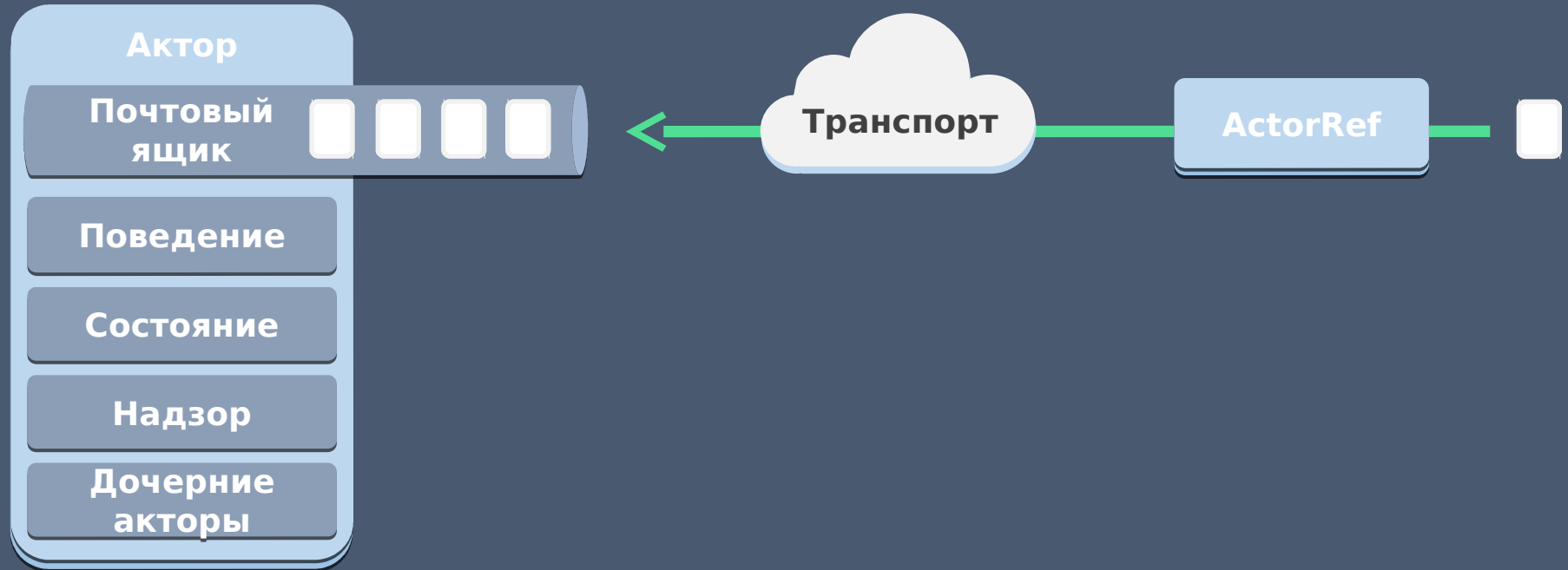
Актор можно использовать как...

- поток
- экземпляр объекта/компонента
- колбек/подписчик
- синглтон или сервис (например, слой работы с бд)
- маршрутизатор, балансировщик, пул
- сервис вне текущего процесса
- конечный автомат

Модель акторов используют:

- Erlang
- Facebook WhatsApp (Erlang)
- RabbitMQ (Erlang)
- CouchDB (Erlang)
- LinkedIn.com (JVM Akka)
- Walmart.com (JVM Akka)
- Blizzard (JVM Akka)

Анатомия актора




Действия с акторами

- 1) Define
- 2) Create
- 3) Send
- 4) Become
- 5) Supervise


```
public class GreetingActor : ReceiveActor
{
    public class Greet
    {
        public string Who { get; private set; }
        public Greet(string who)
        {
            Who = who;
        }
    }

    public GreetingActor()
    {
        Receive<Greet>(greet => Console.WriteLine(greet.Who));
    }
}
```

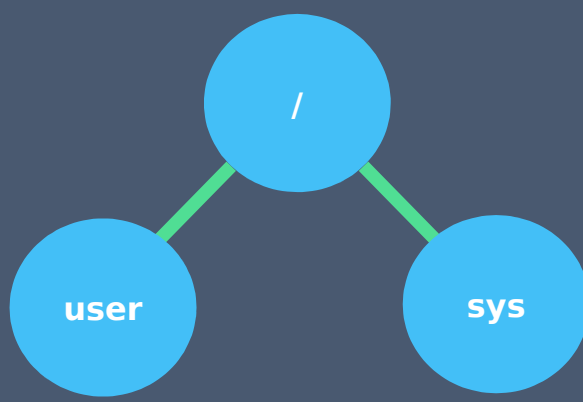
A vertical blue abstract graphic on the left side of the slide, featuring a bright light source at the top that creates a lens flare effect, with rays of light extending downwards and outwards, fading into a darker blue at the bottom.

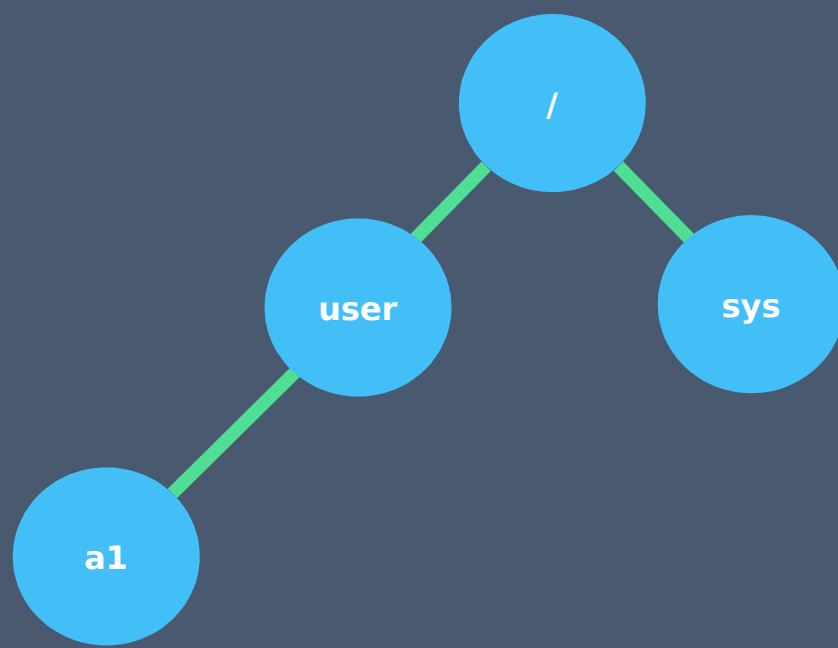
```
var system = ActorSystem.Create("my-system");  
var actorRef = system.ActorOf<GreetingActor>("my-actor");  
actorRef.Tell(new GreetingActor.Greet("World"));
```

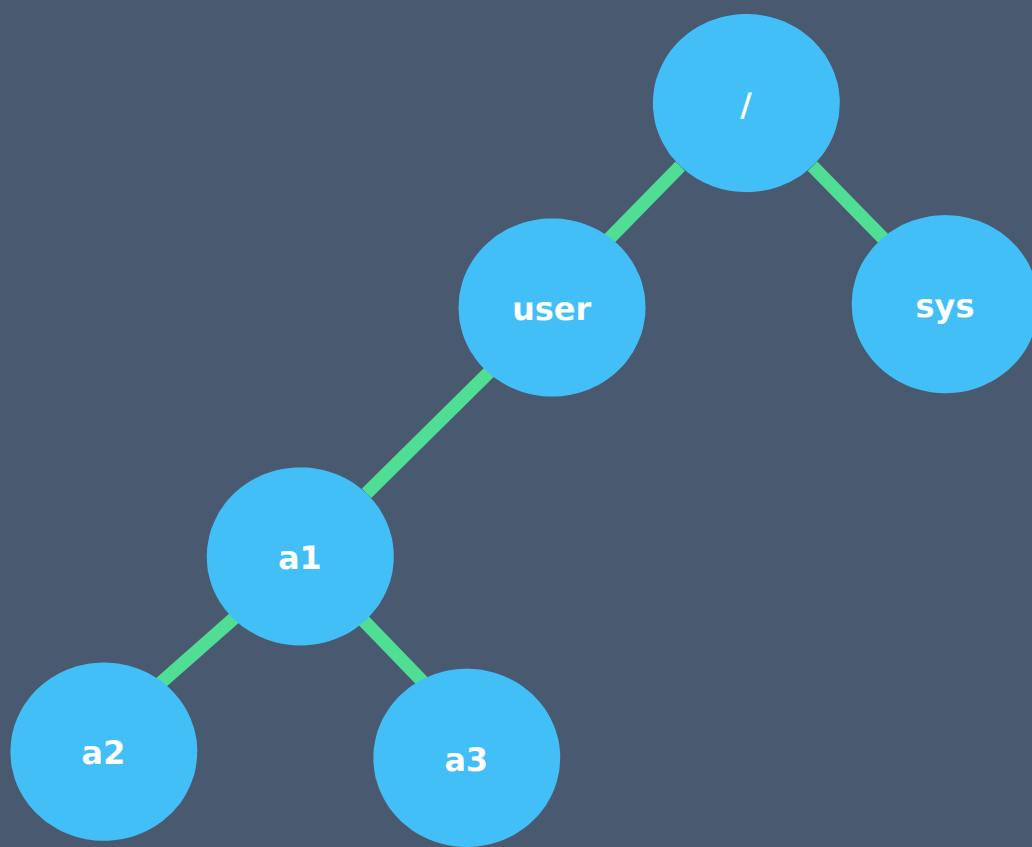
```
var system = ActorSystem.Create("my-system");  
var actorRef = system.ActorOf(Props.Create<GreetingActor>()  
    .WithRouter(new RoundRobinPool(10)),  
    "my-actor");  
actorRef.Tell(new GreetingActor.Greet("World"));
```

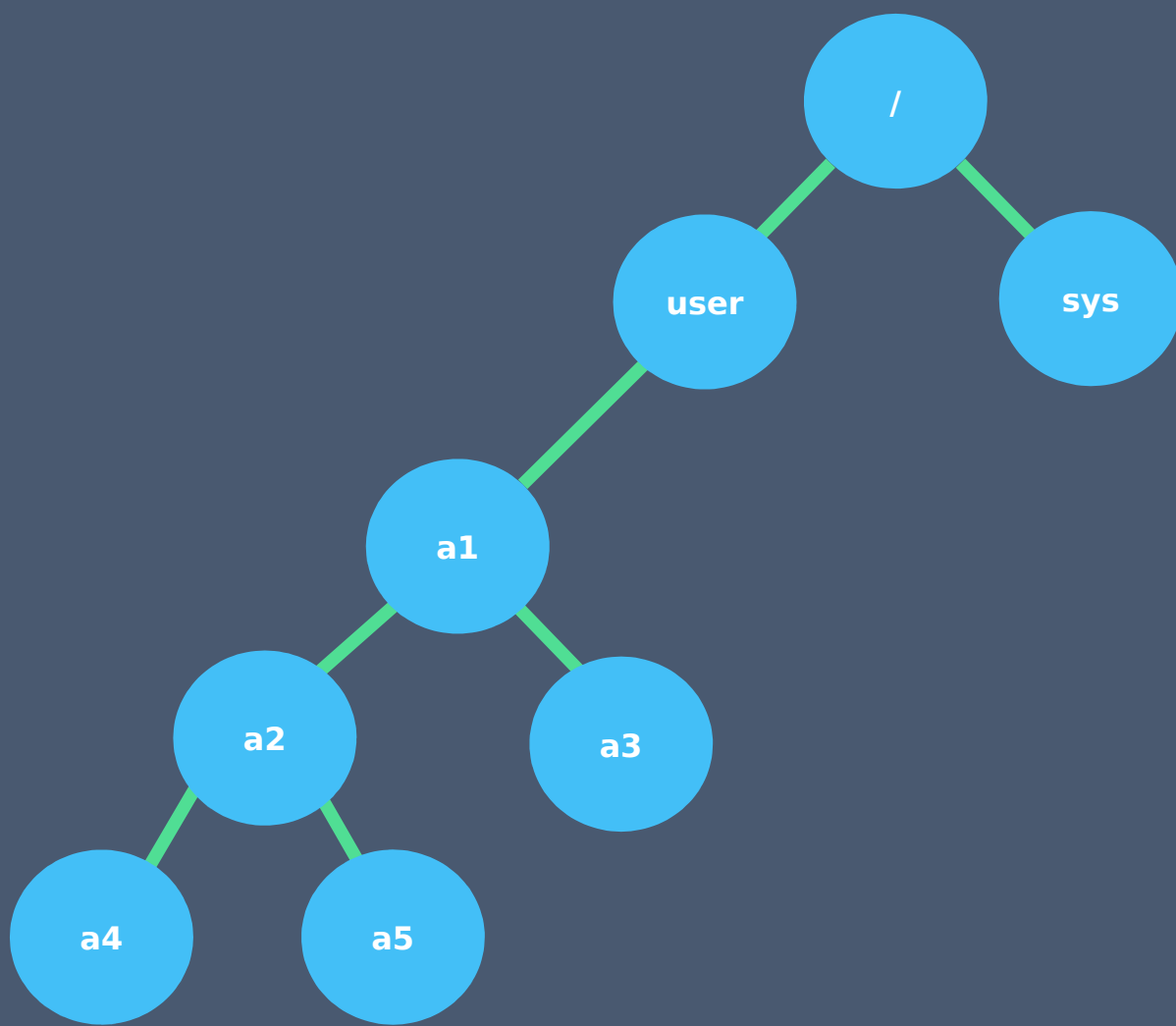
```
var system = ActorSystem.Create("my-system");
var actorRef = system.ActorOf(Props.Create<GreetingActor>()
    .WithDeploy(new Deploy(new RemoteScope(
        Address.Parse("akka.tcp://my-system@192.168.1.15:6001")))),
    "my-actor");

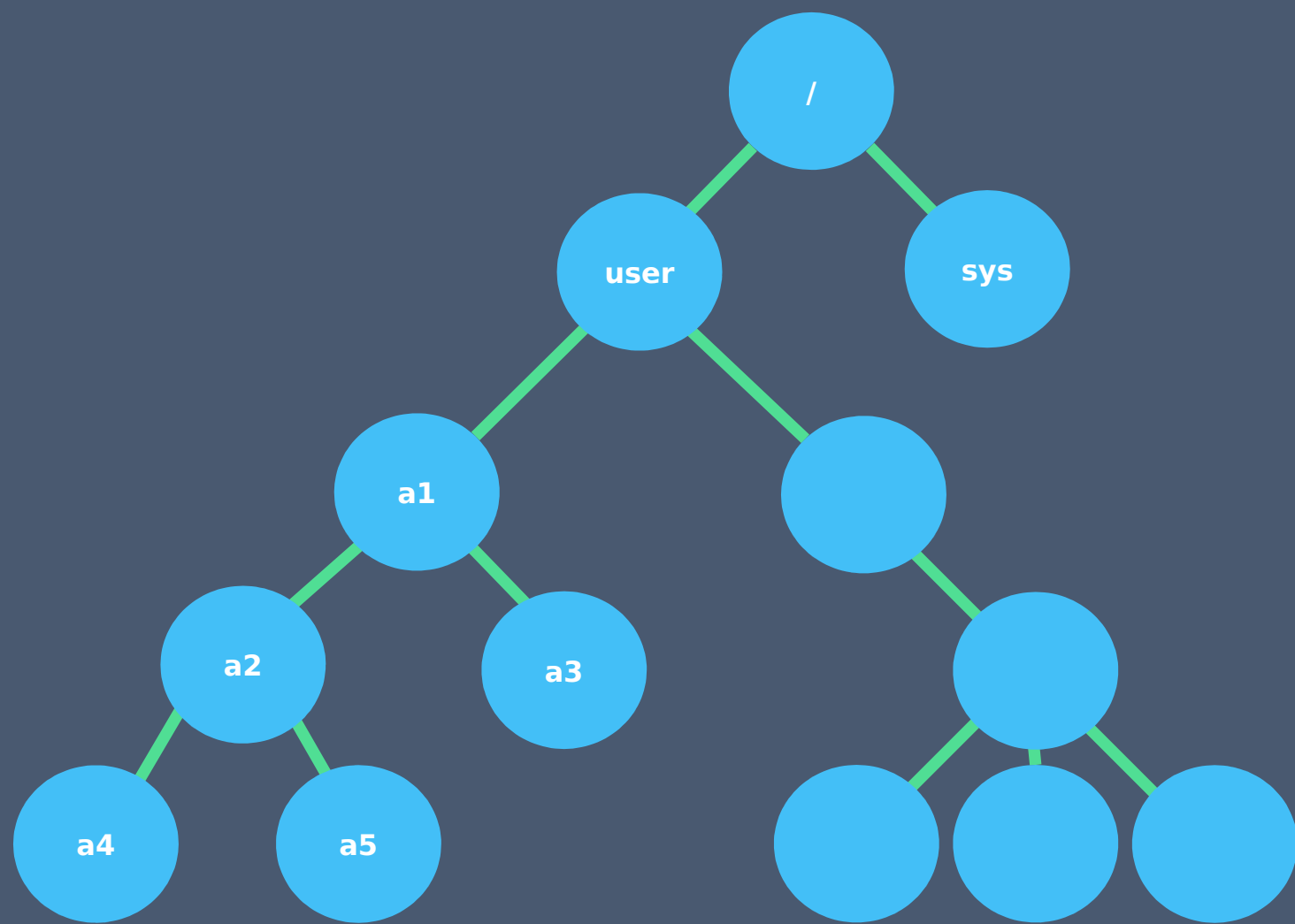
actorRef.Tell(new GreetingActor.Greet("World"));
```

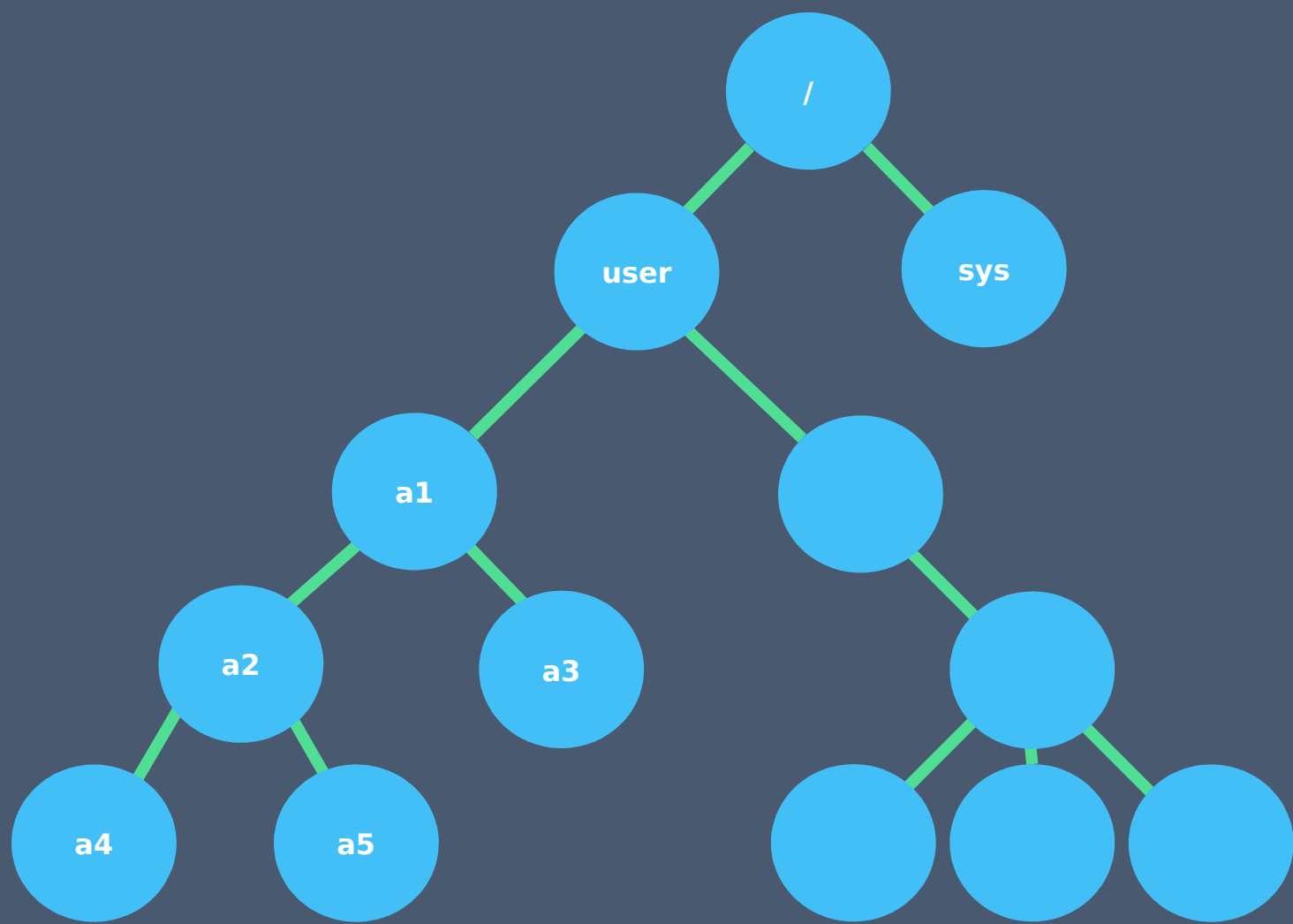




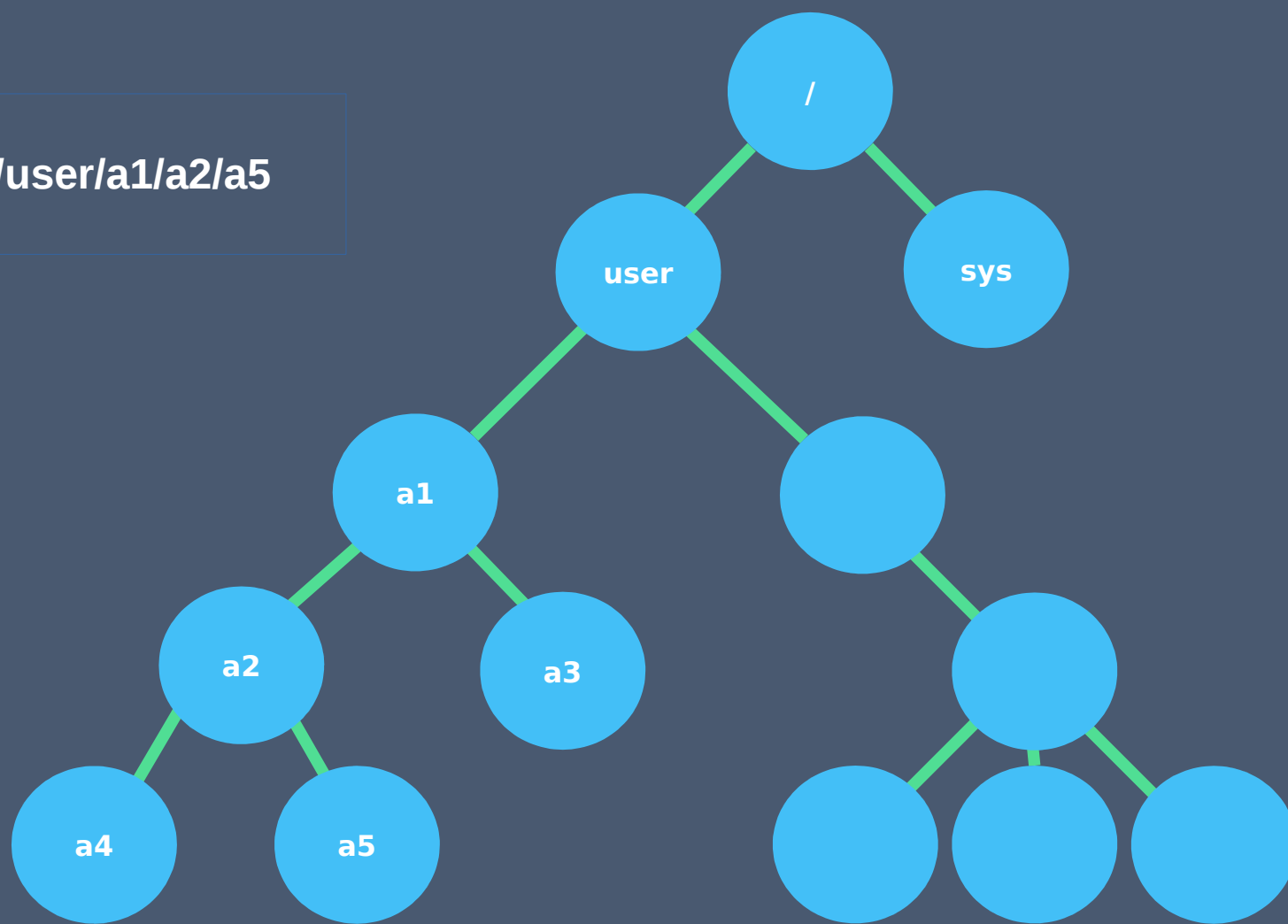






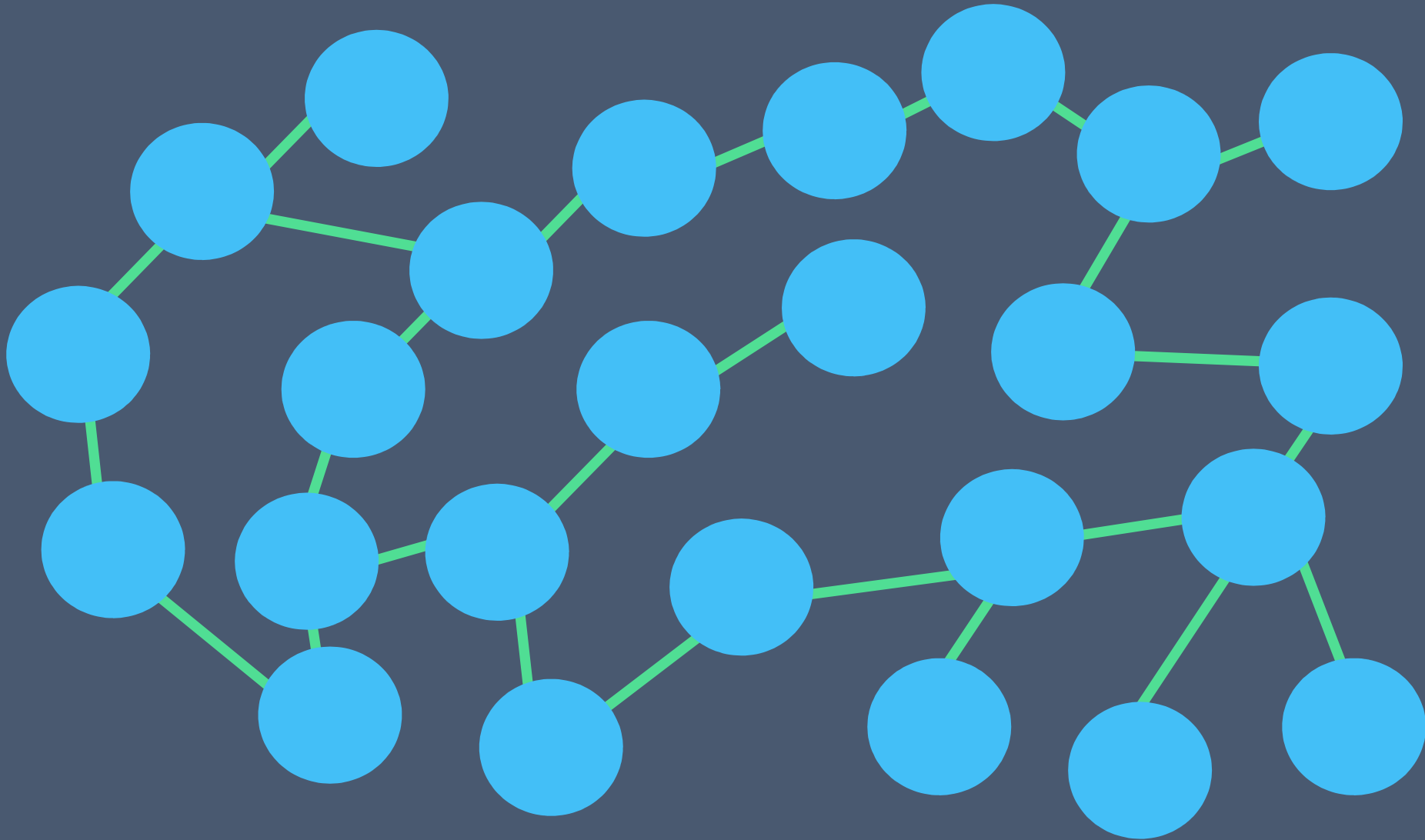


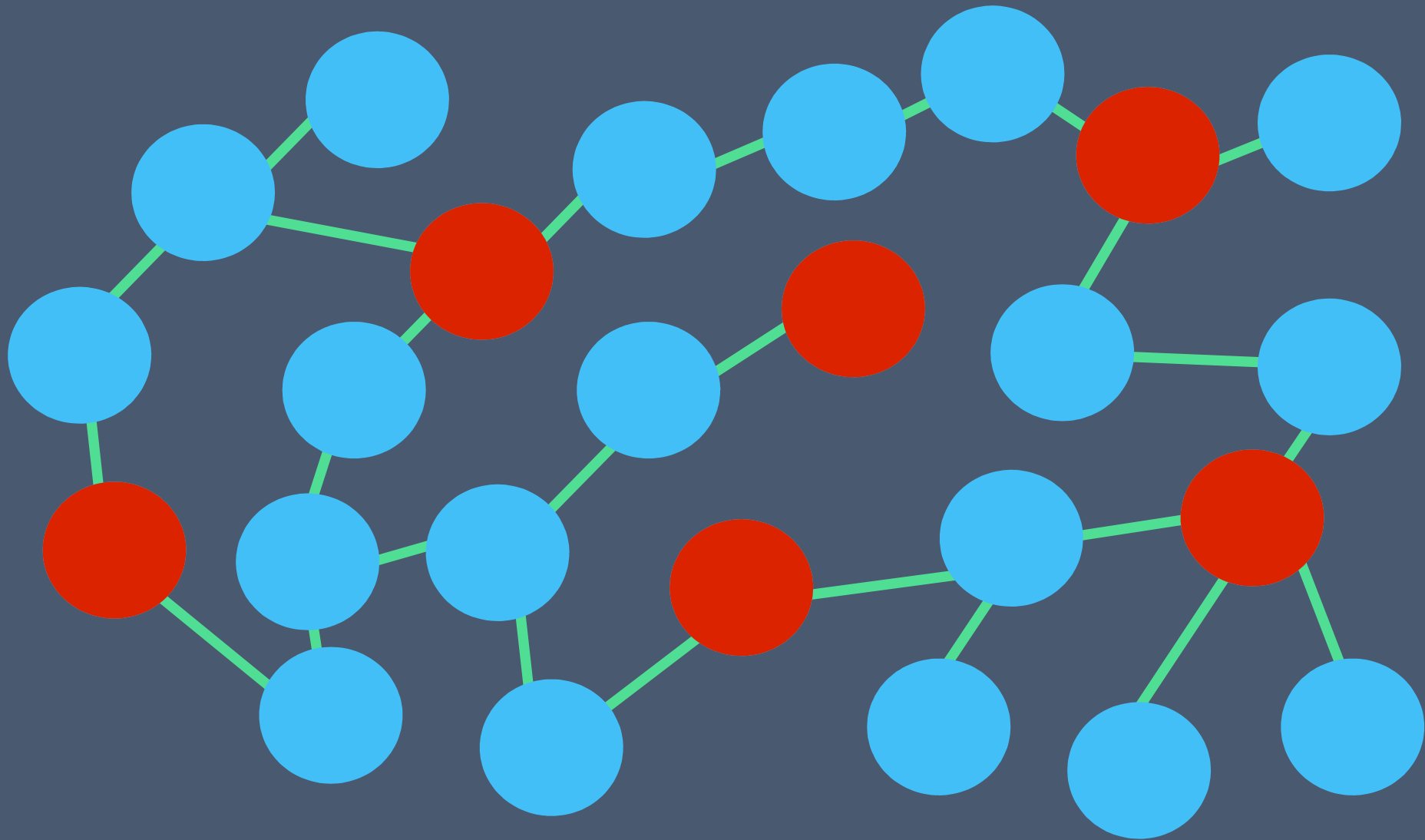
`/user/a1/a2/a5`



Обработка исключений в C, C#, Java...

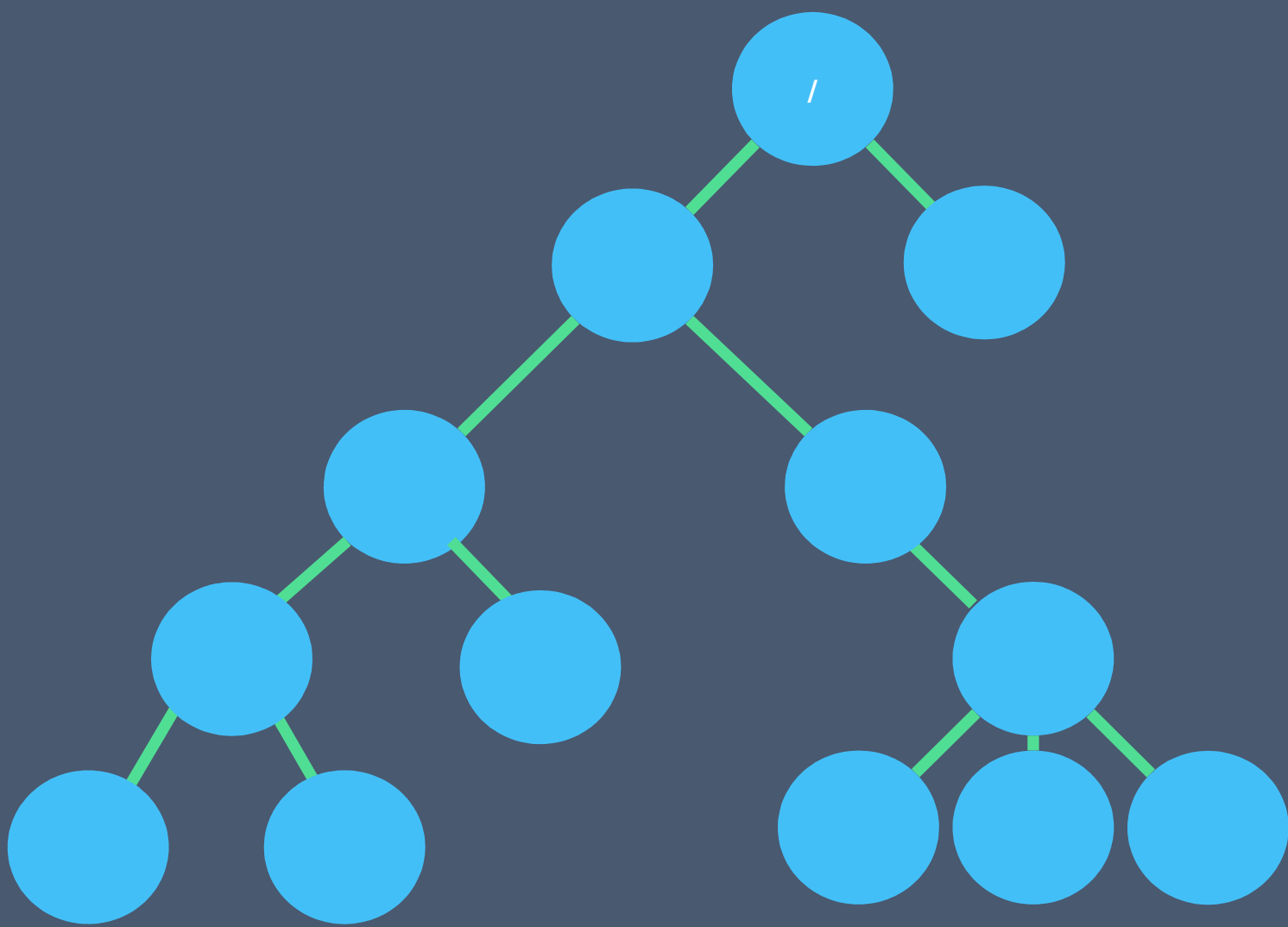


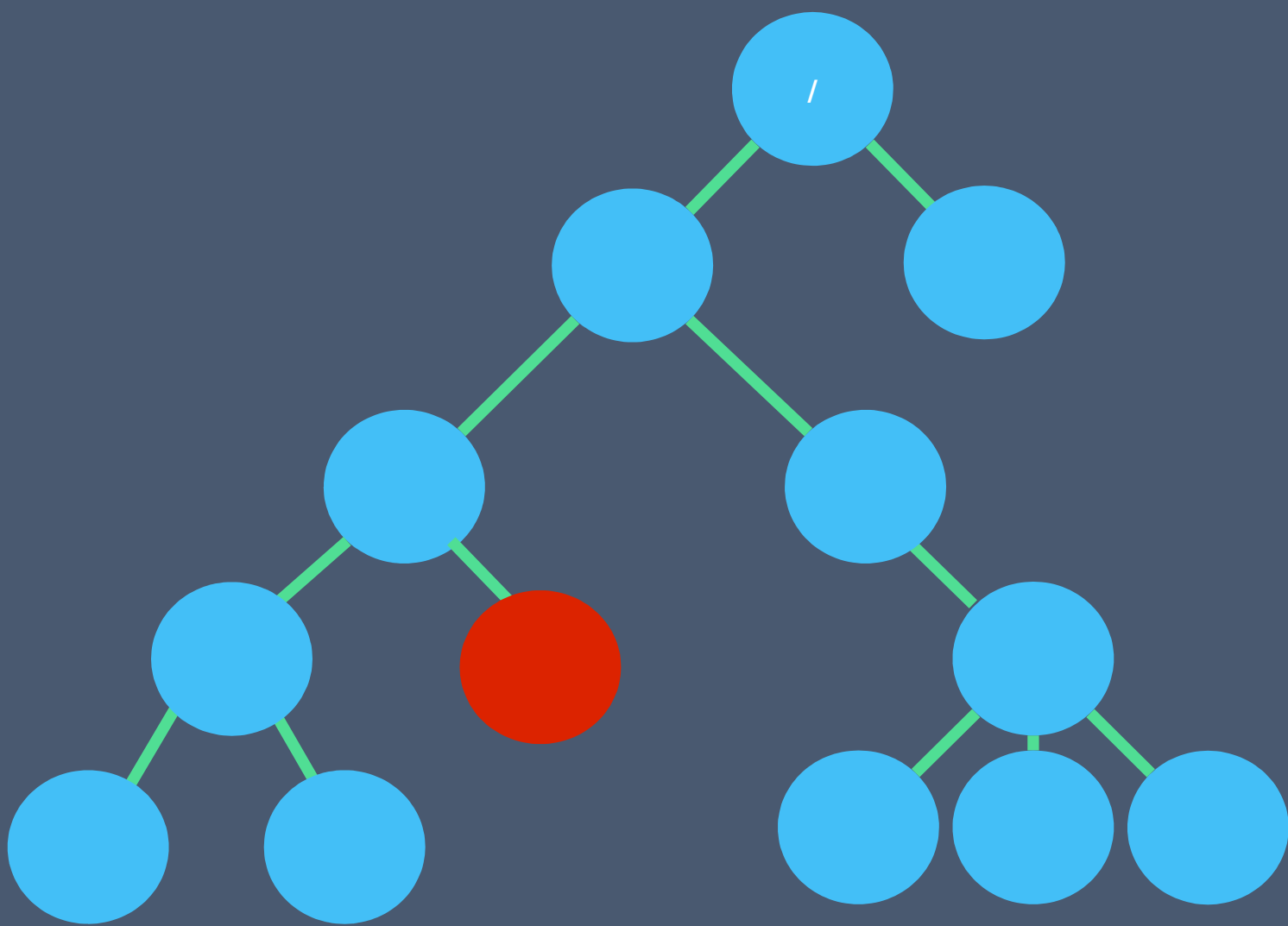


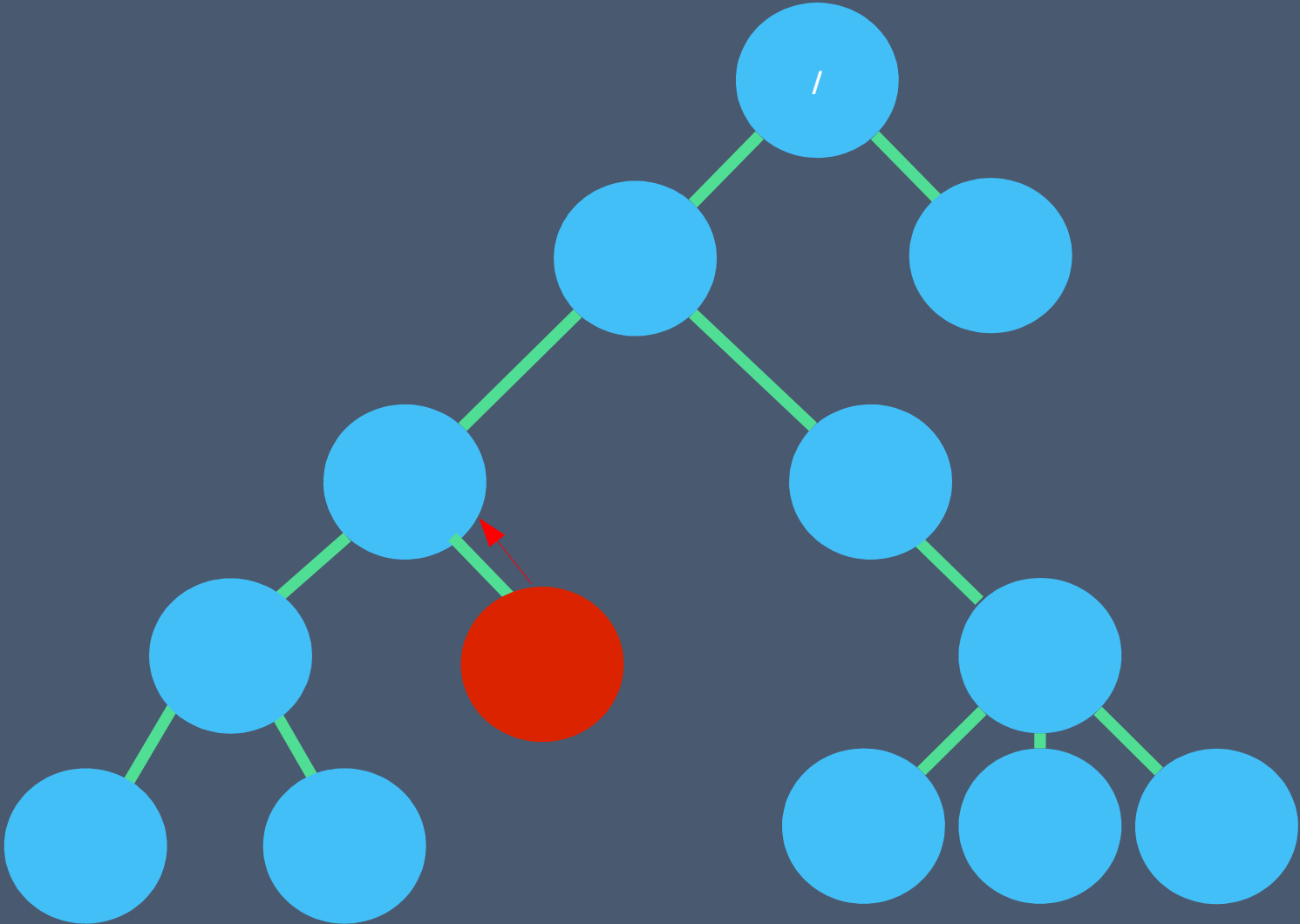


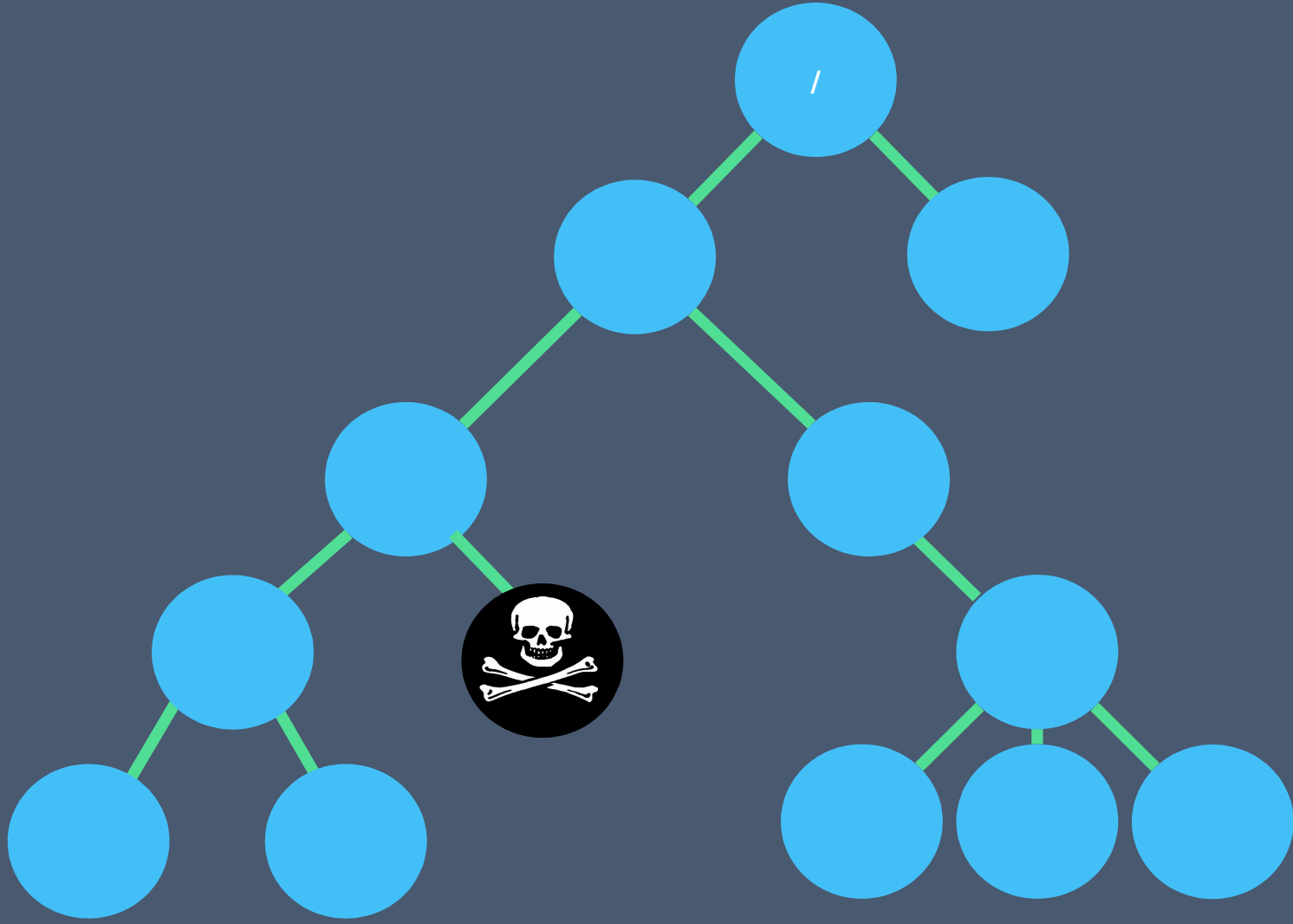
Error Kernel

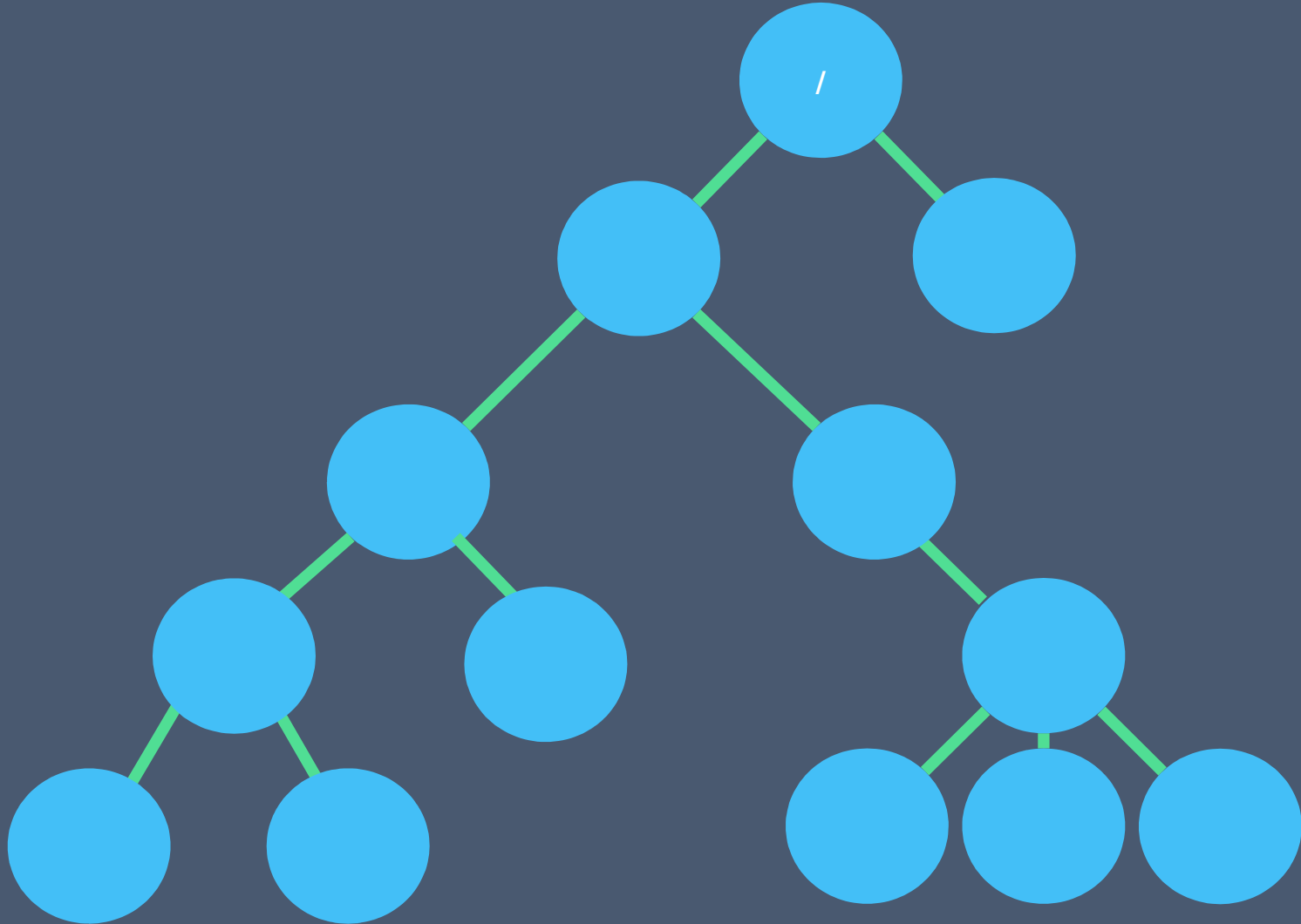


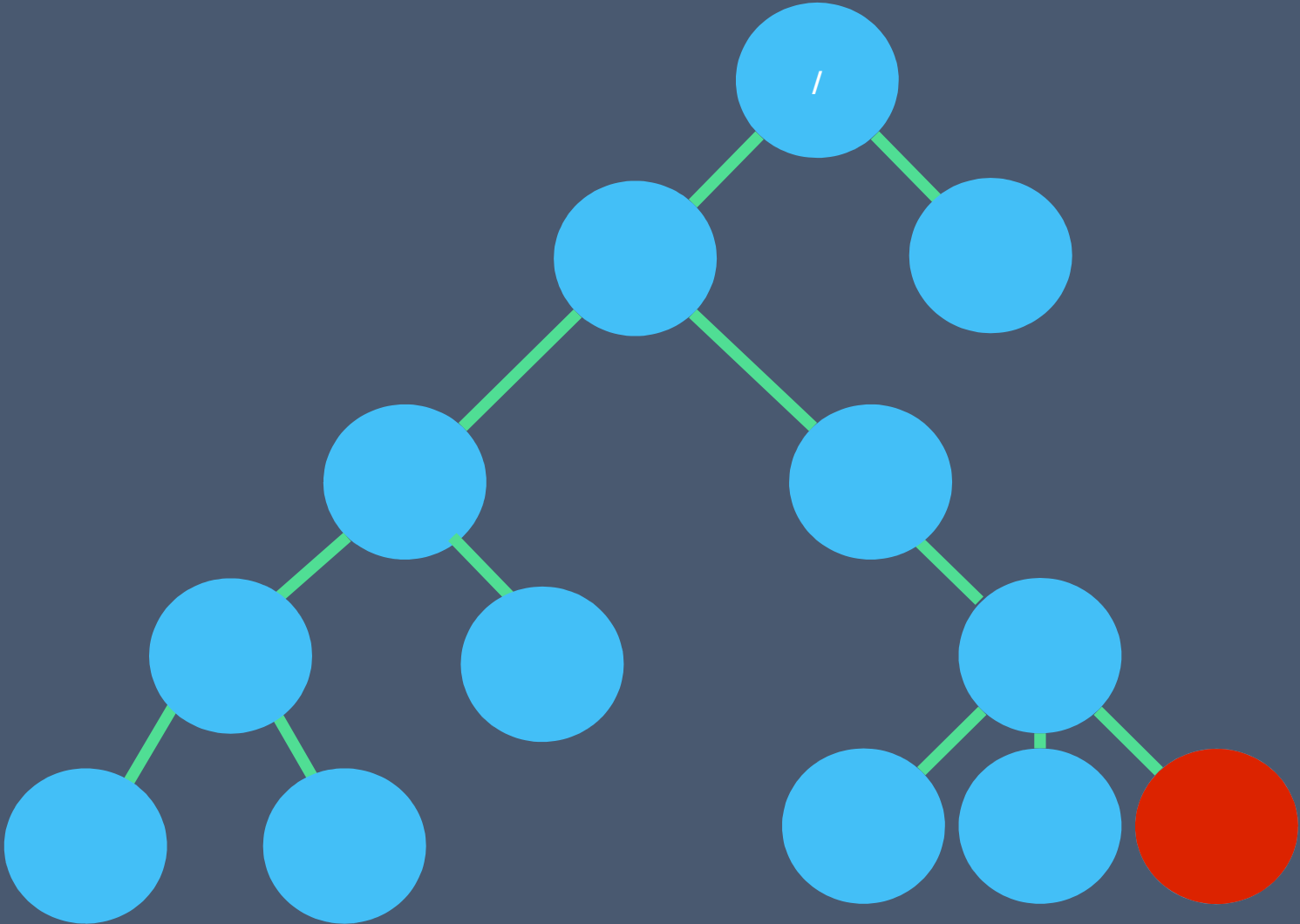


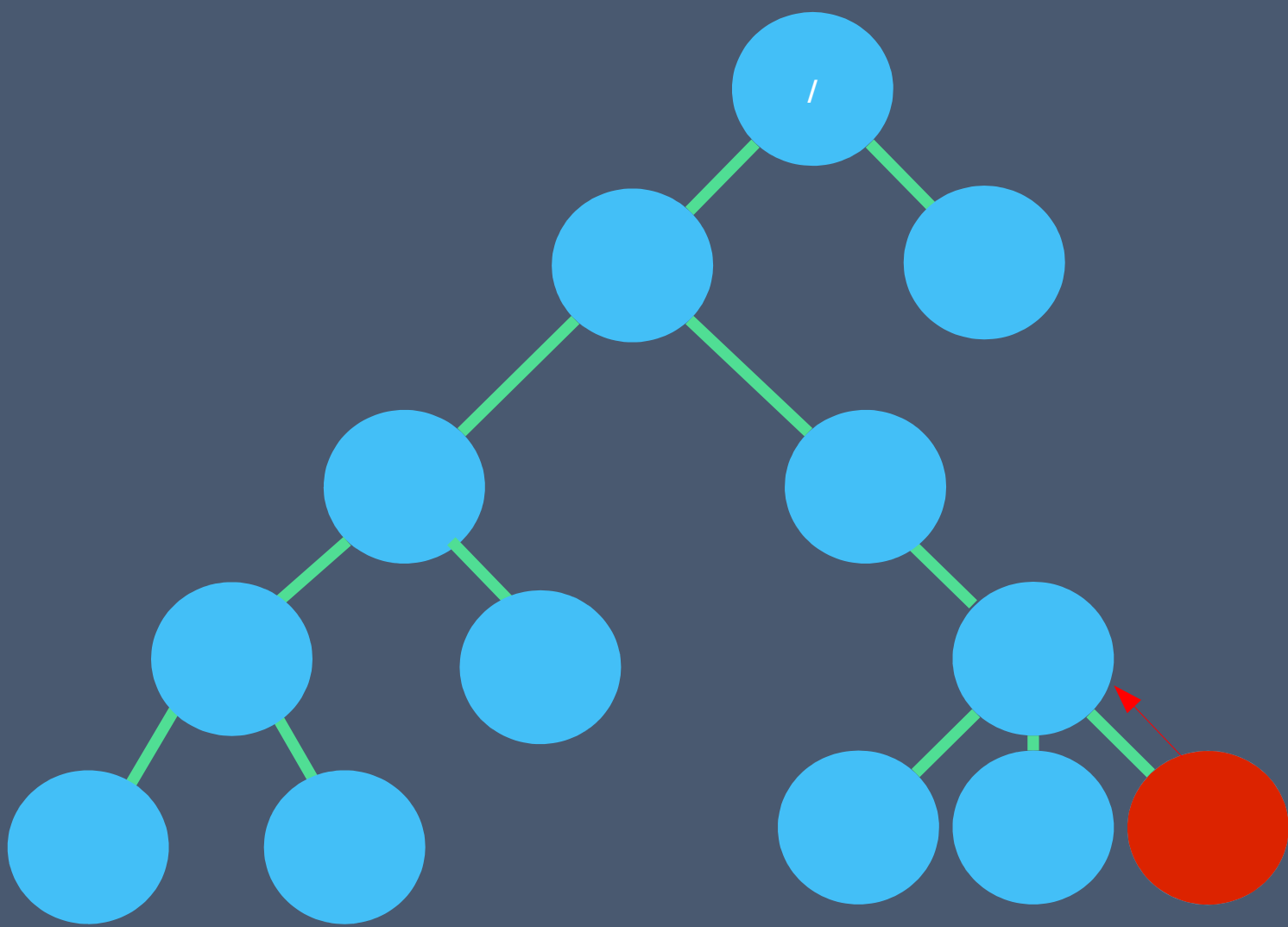


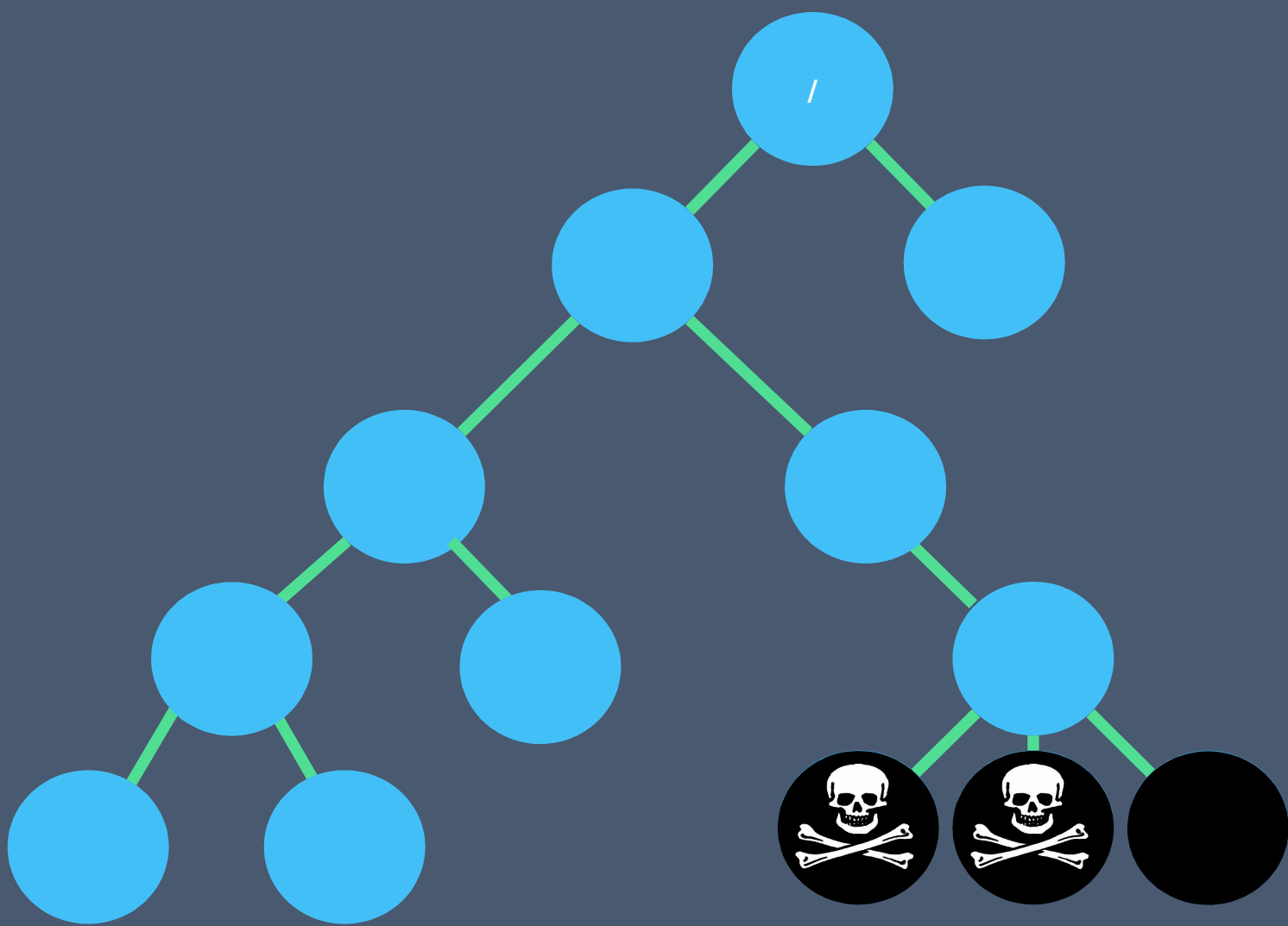


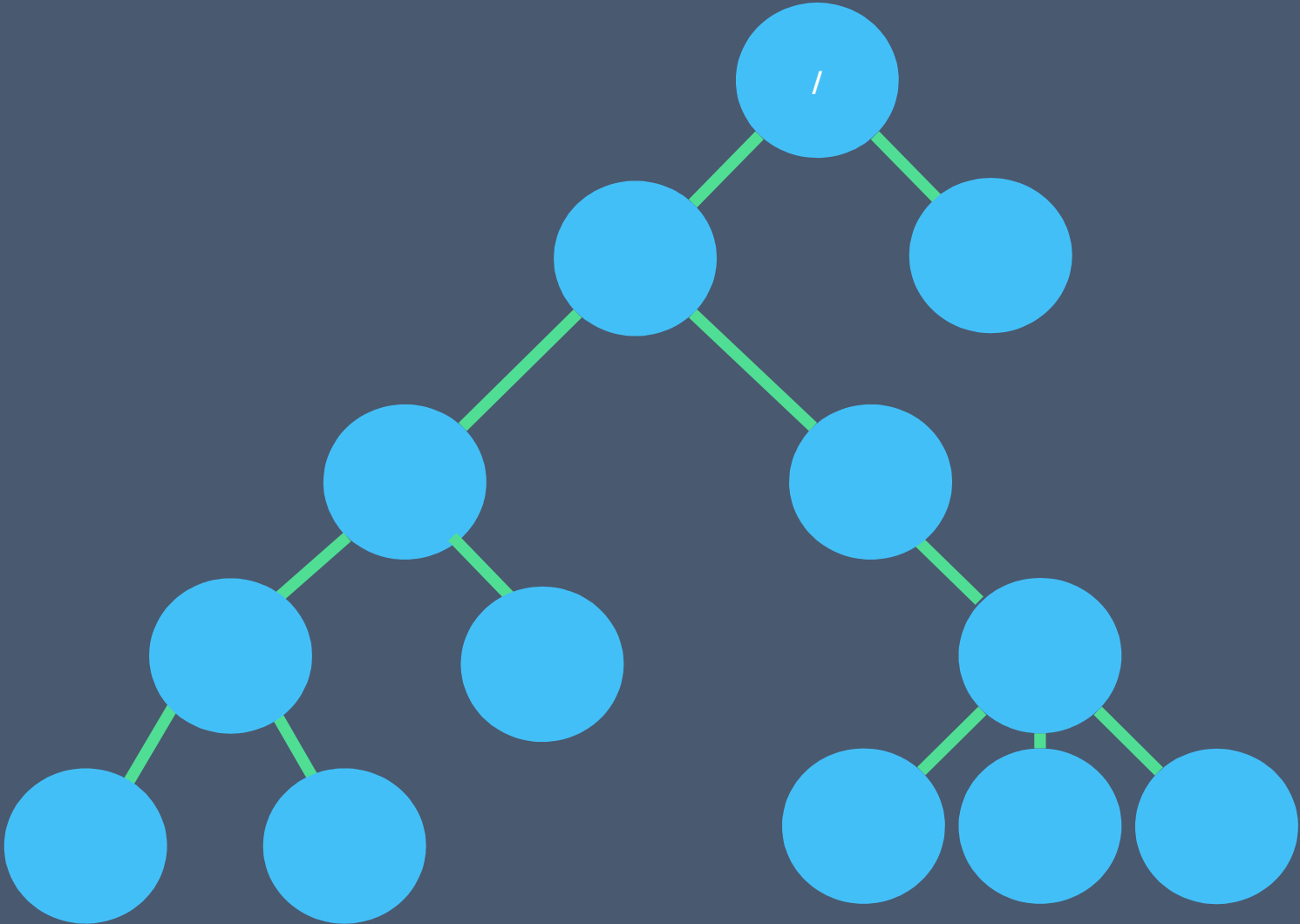


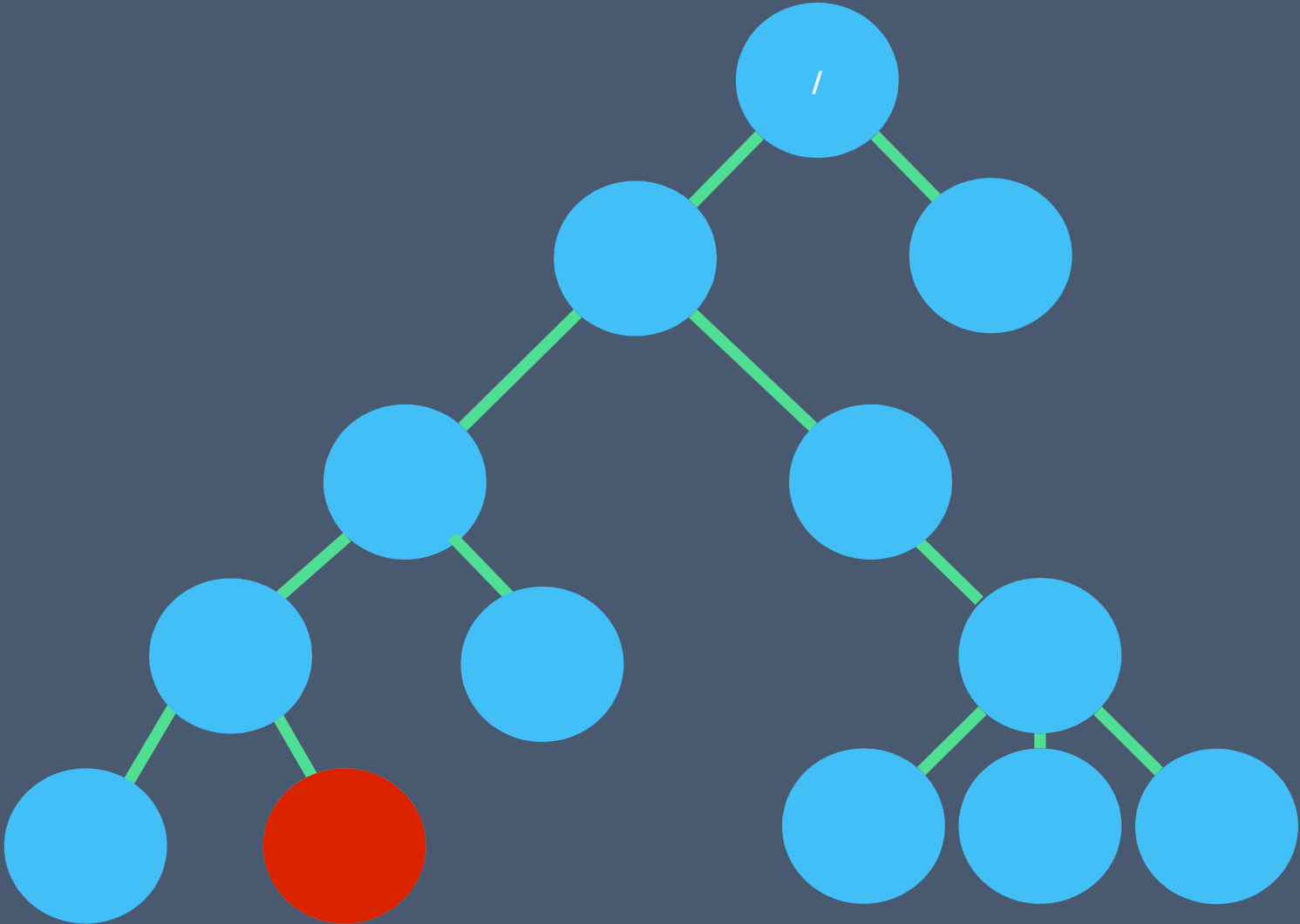


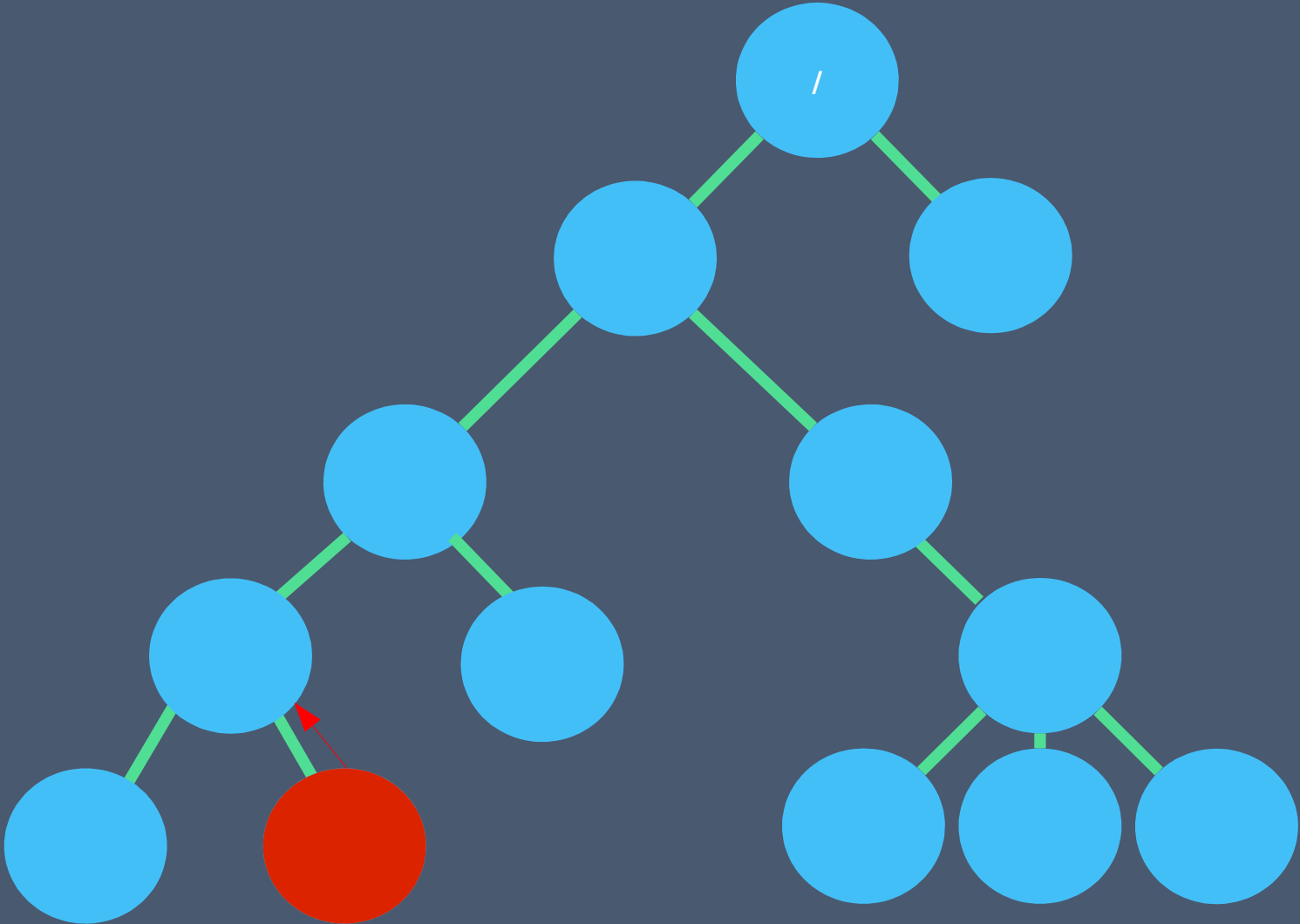


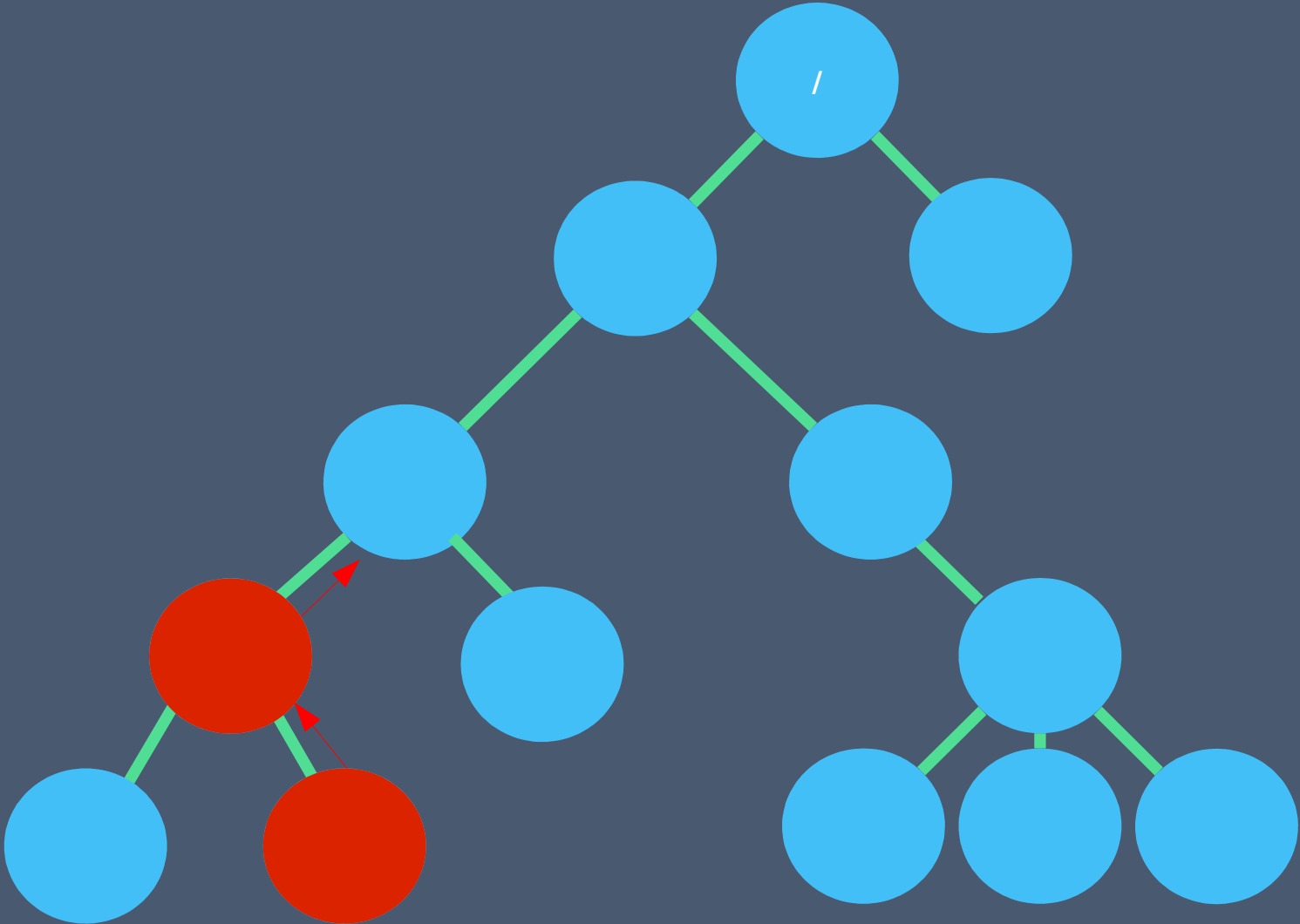


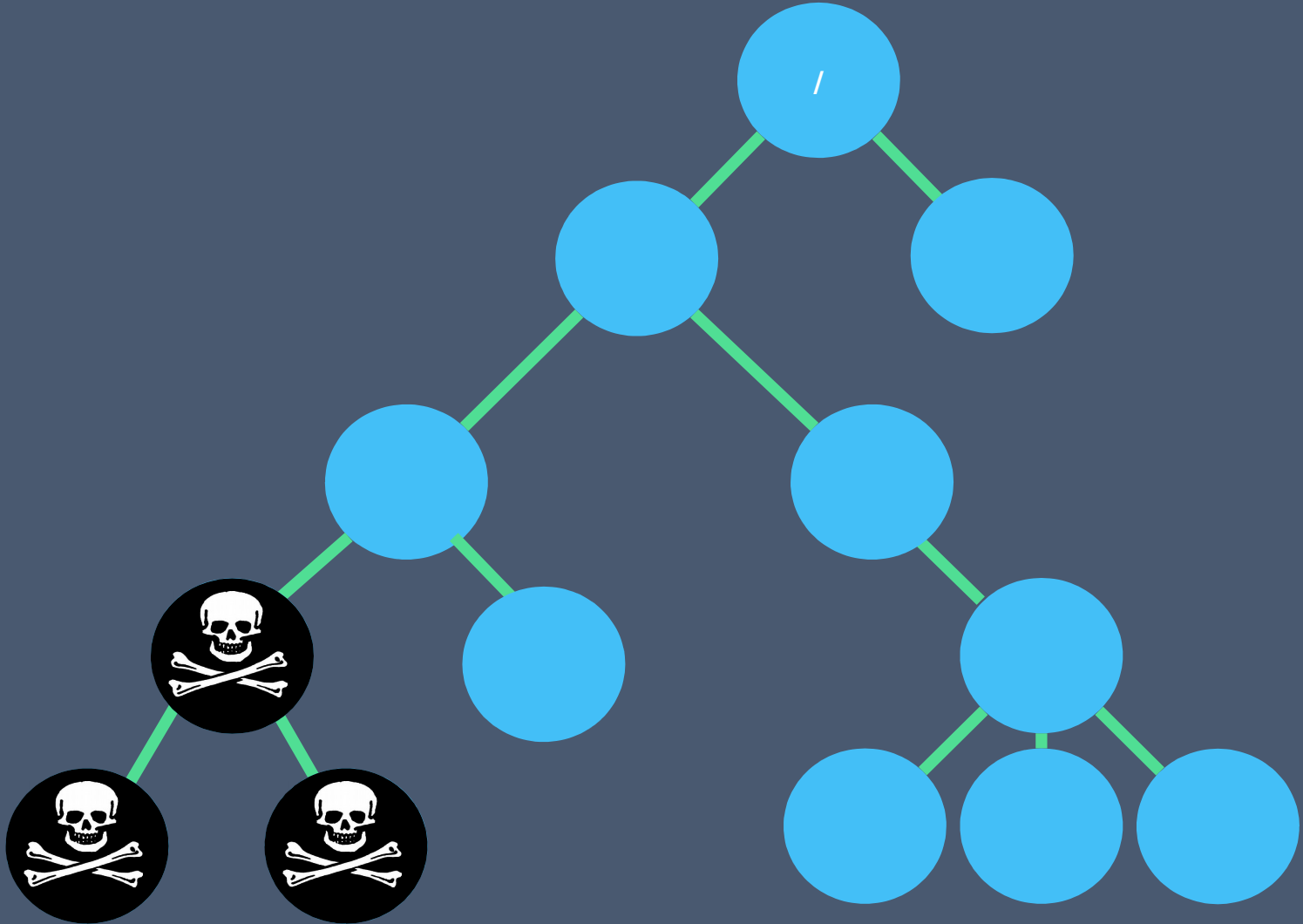


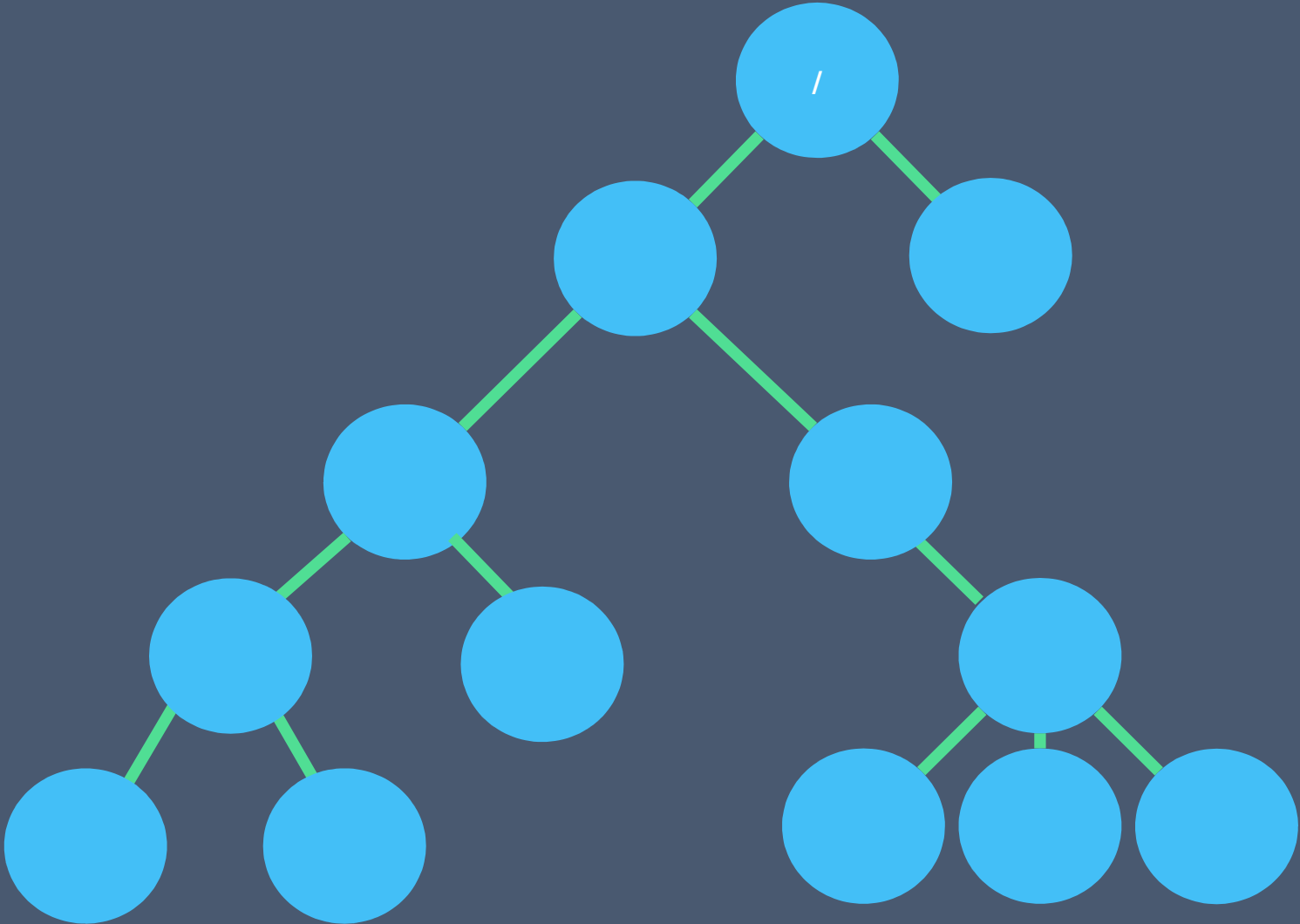


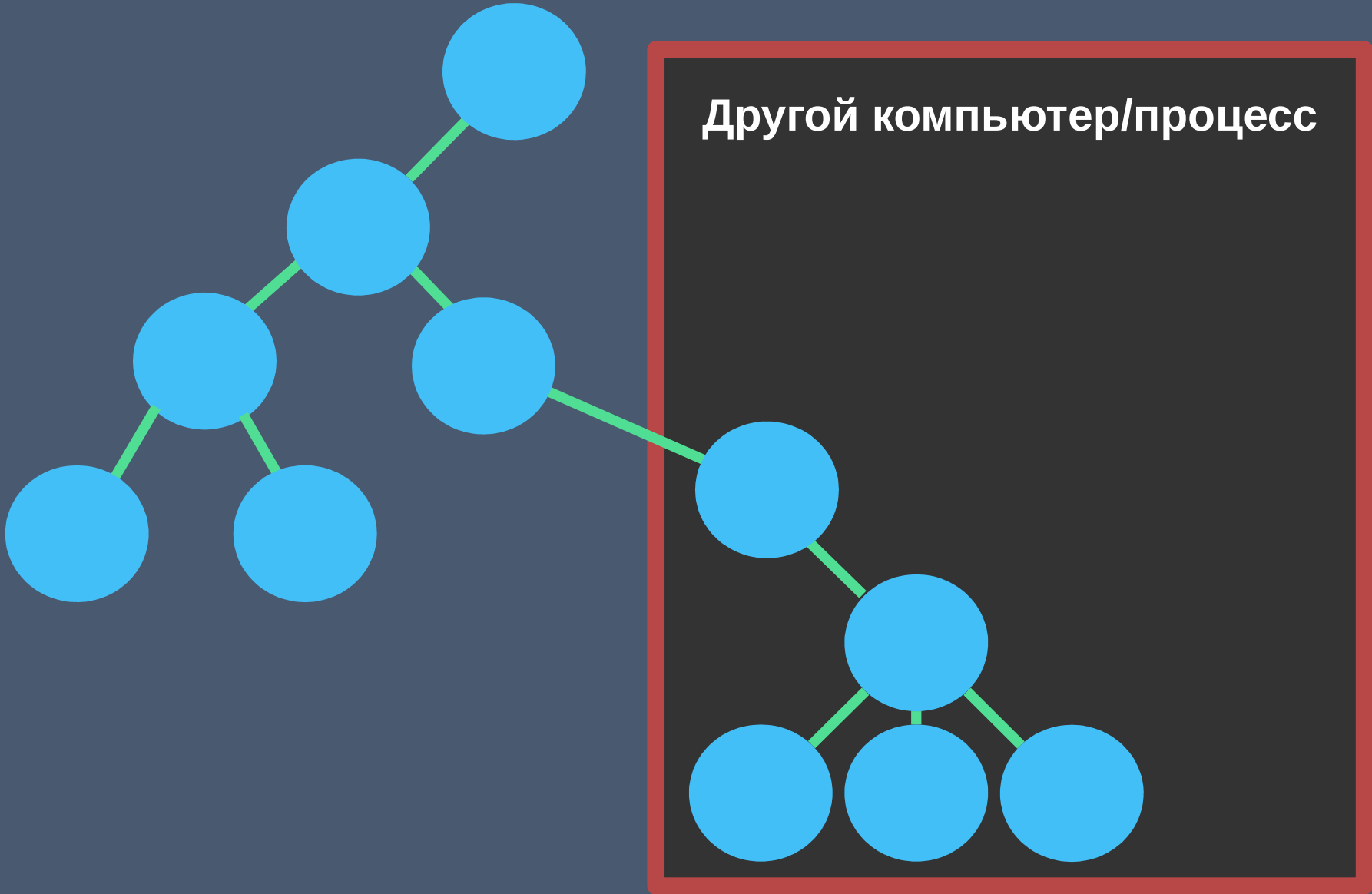












```
protected override SupervisorStrategy SupervisorStrategy()
{
    return new OneForOneStrategy(5, new TimeSpan(0, 1, 0), e =>
        e is ArithmeticException ? Directive.Resume
        : e is IOException ? Directive.Restart
        : Directive.Escalate);
}
```



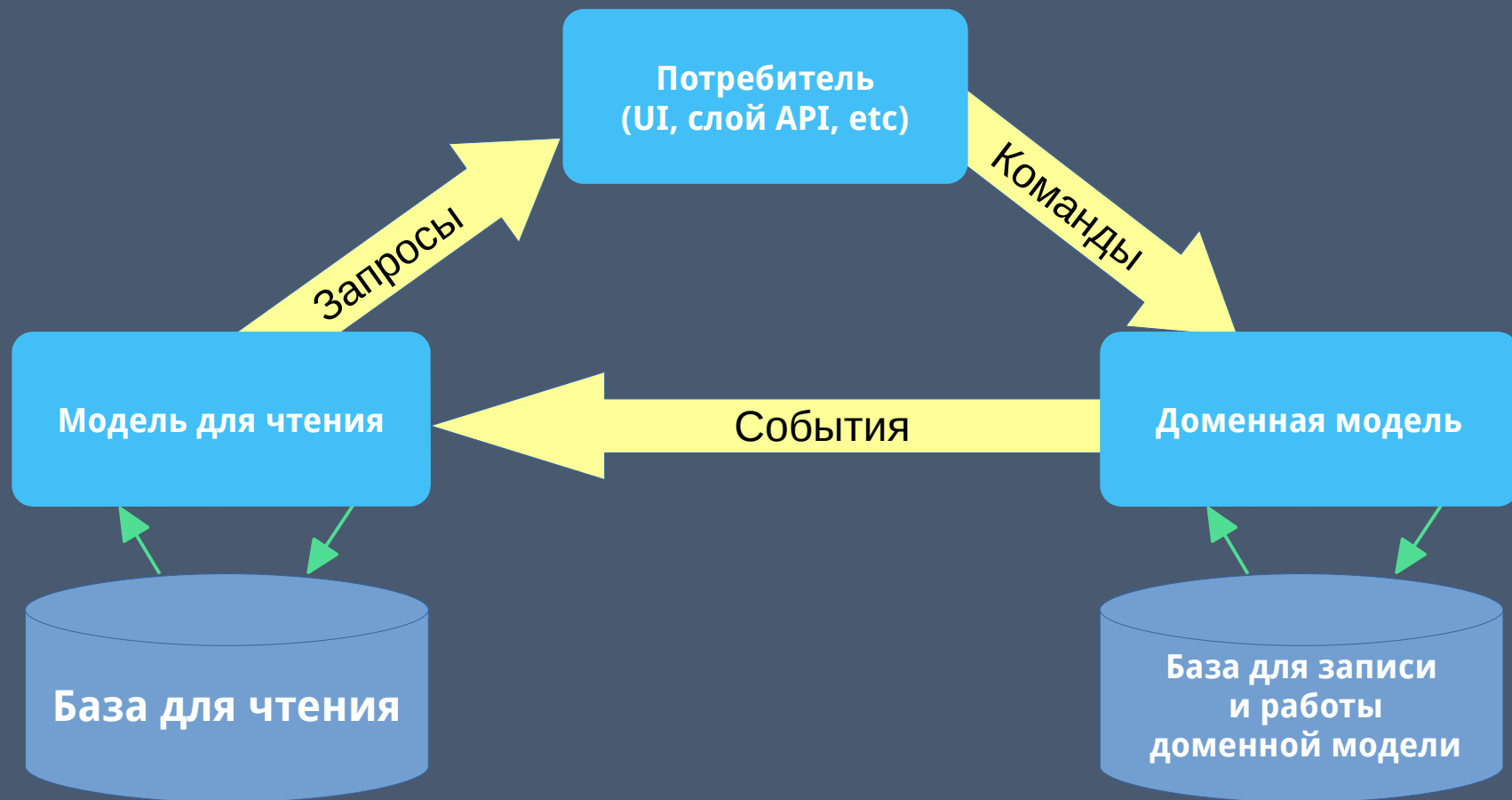
State Machine Demo

ASP.NET Demo



Akka.Persistence

CQRS (Command/Query Request Separation)



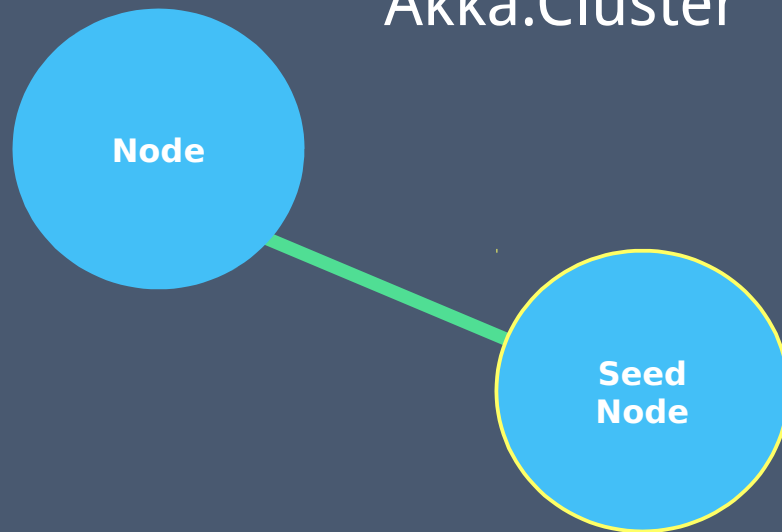
Akka.Cluster



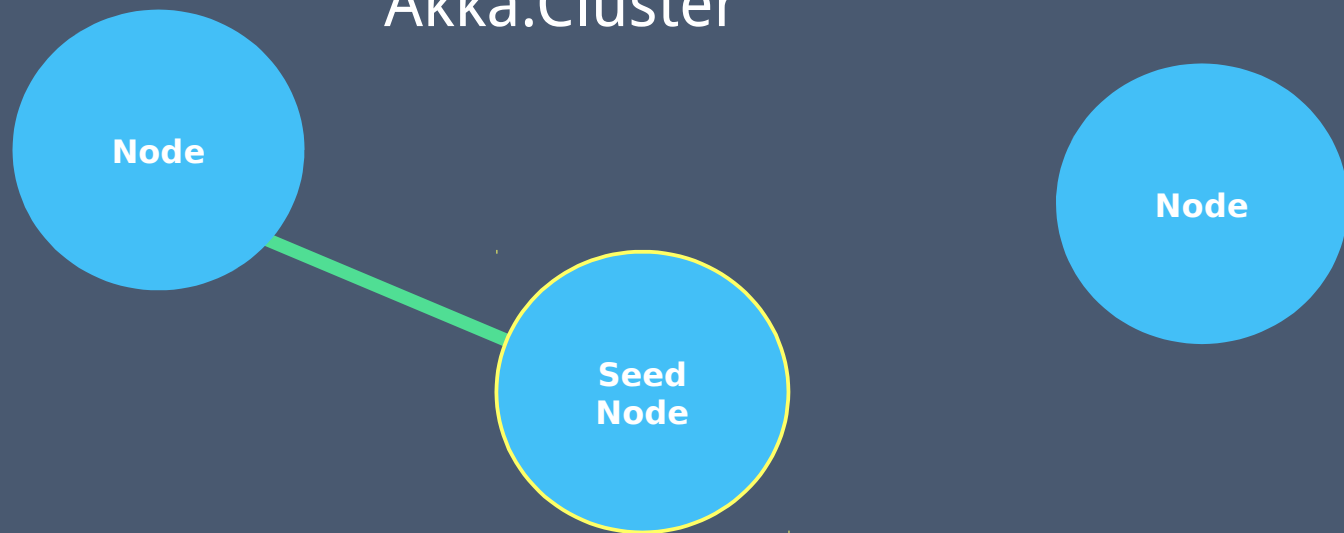
Akka.Cluster



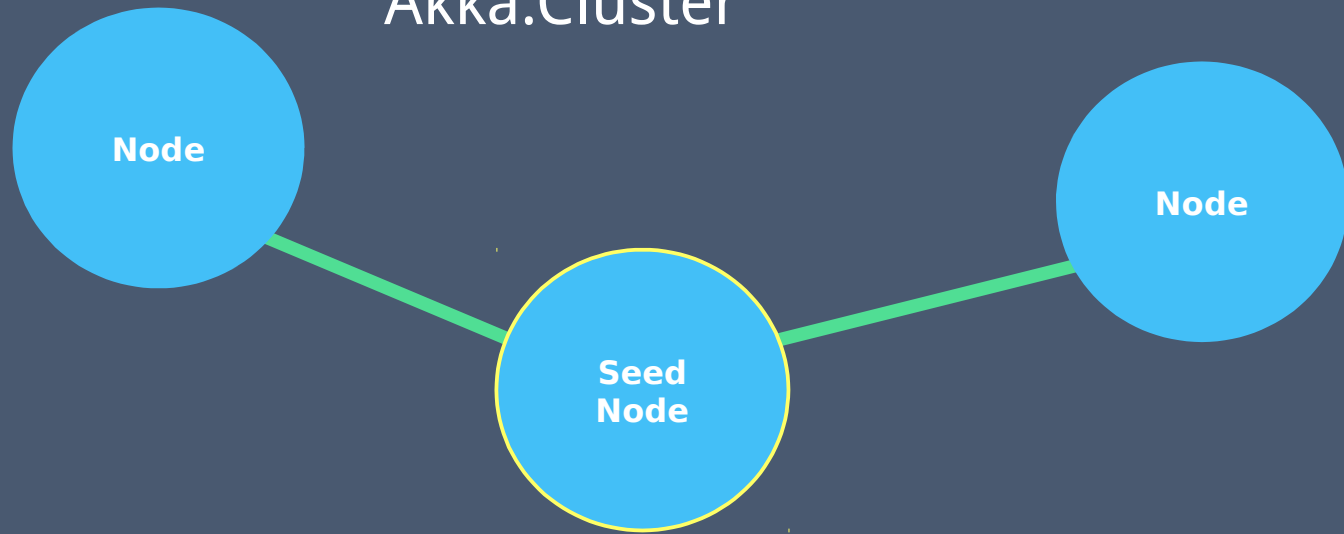
Akka.Cluster



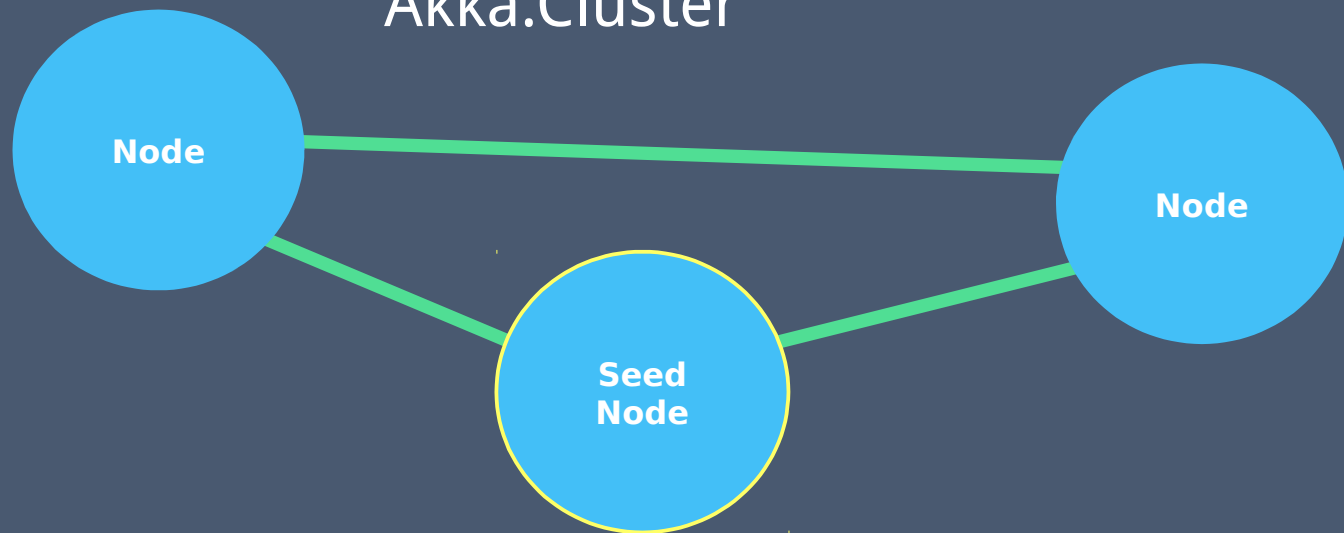
Akka.Cluster



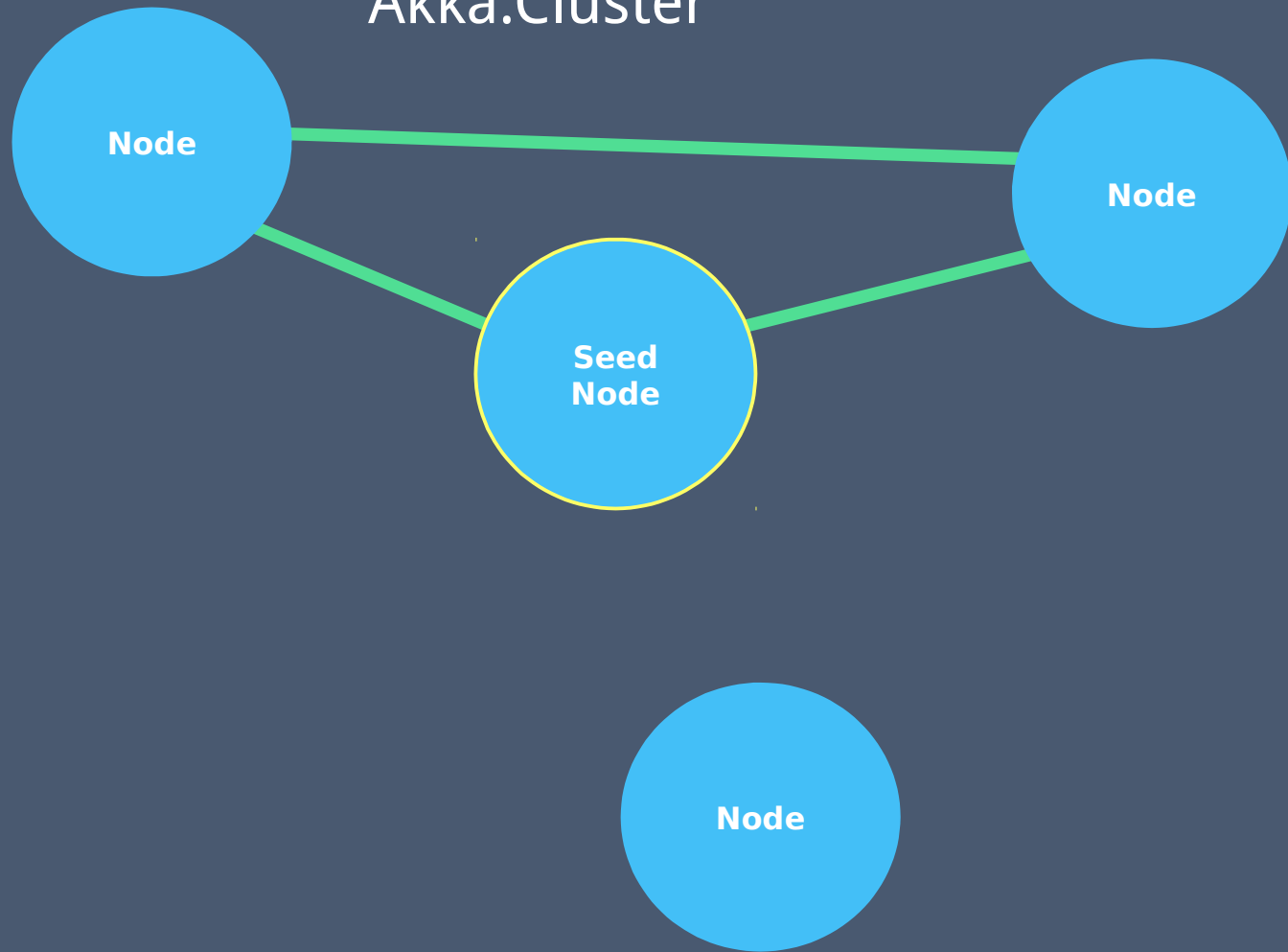
Akka.Cluster



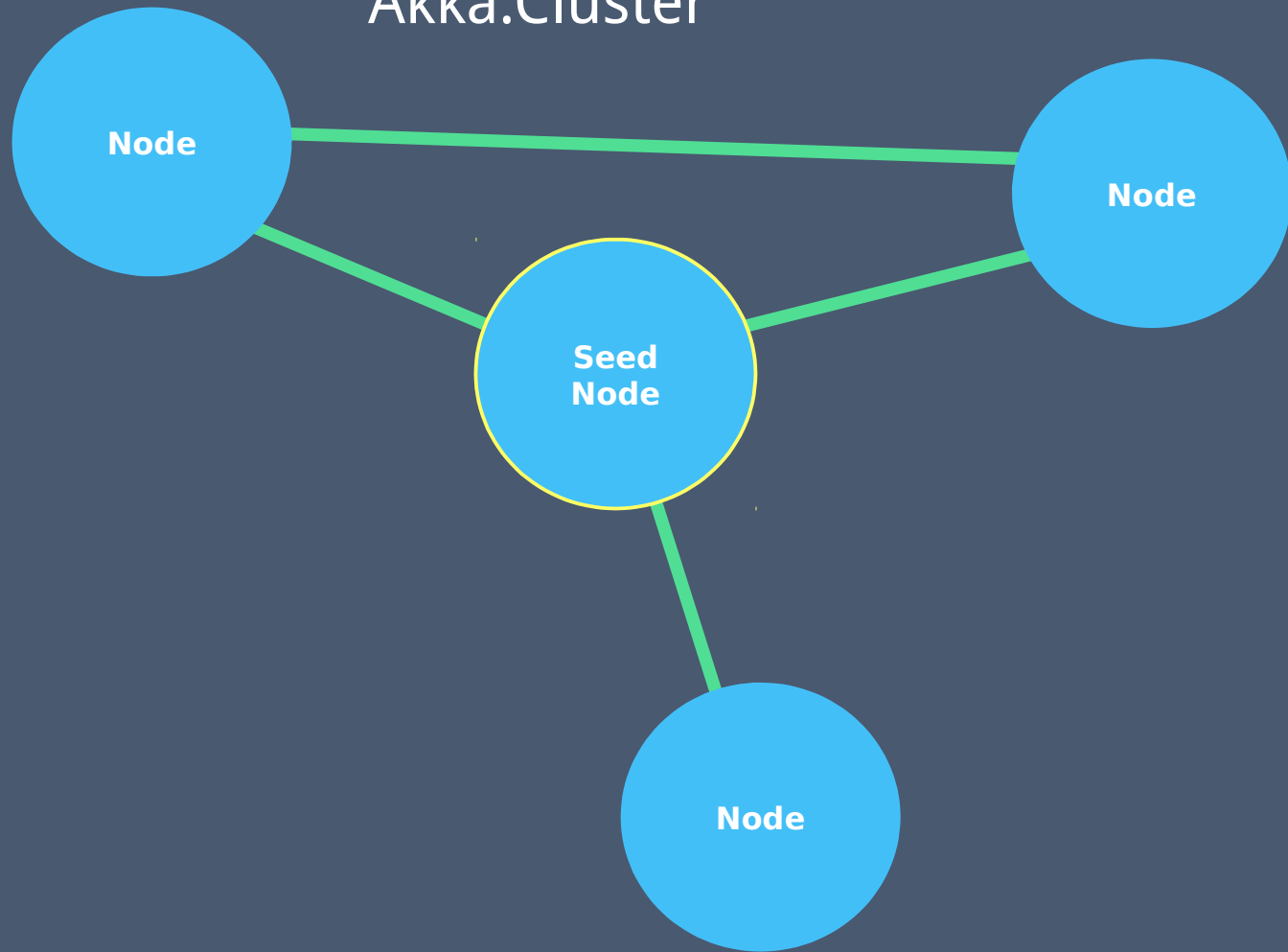
Akka.Cluster



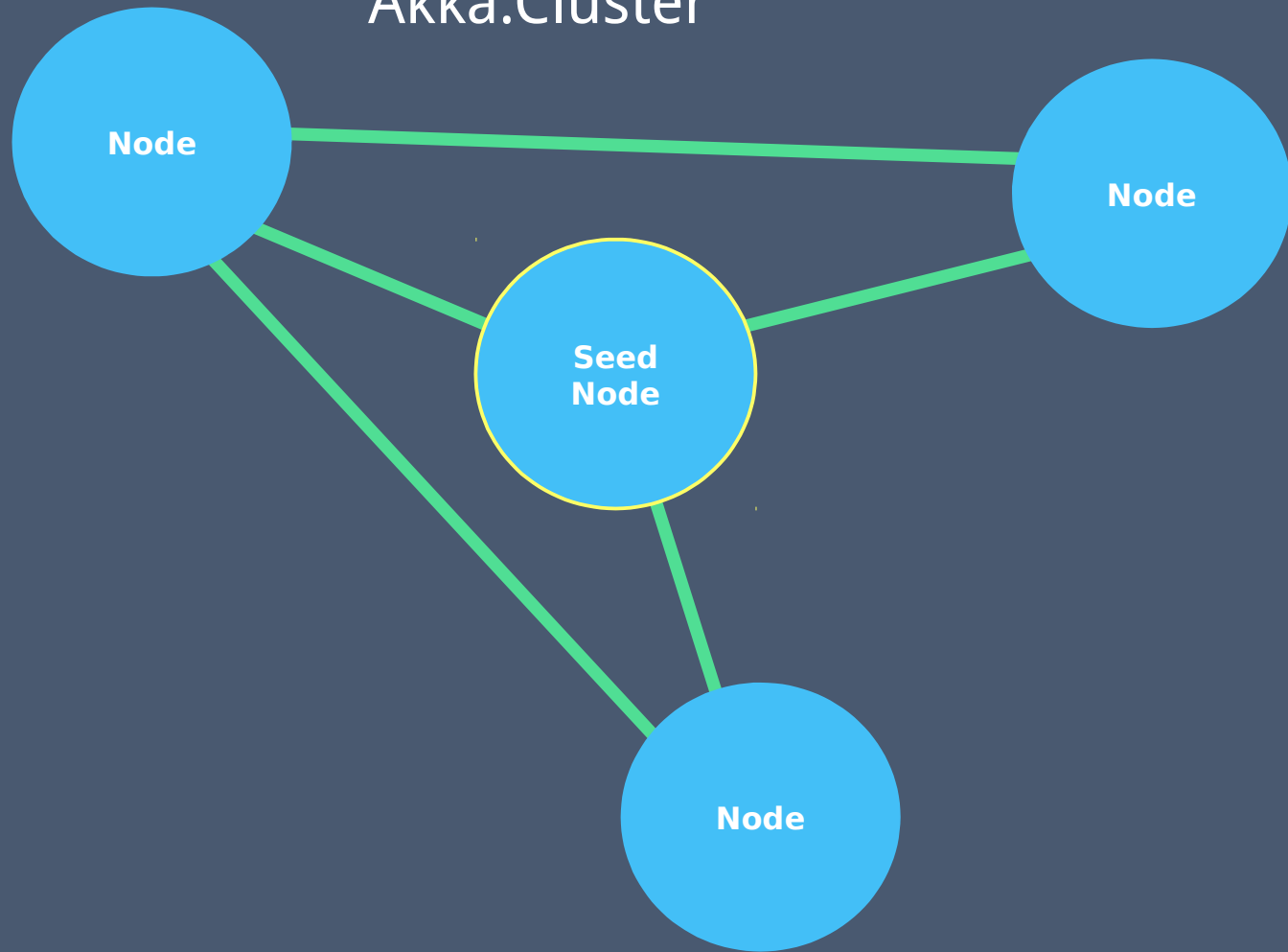
Akka.Cluster



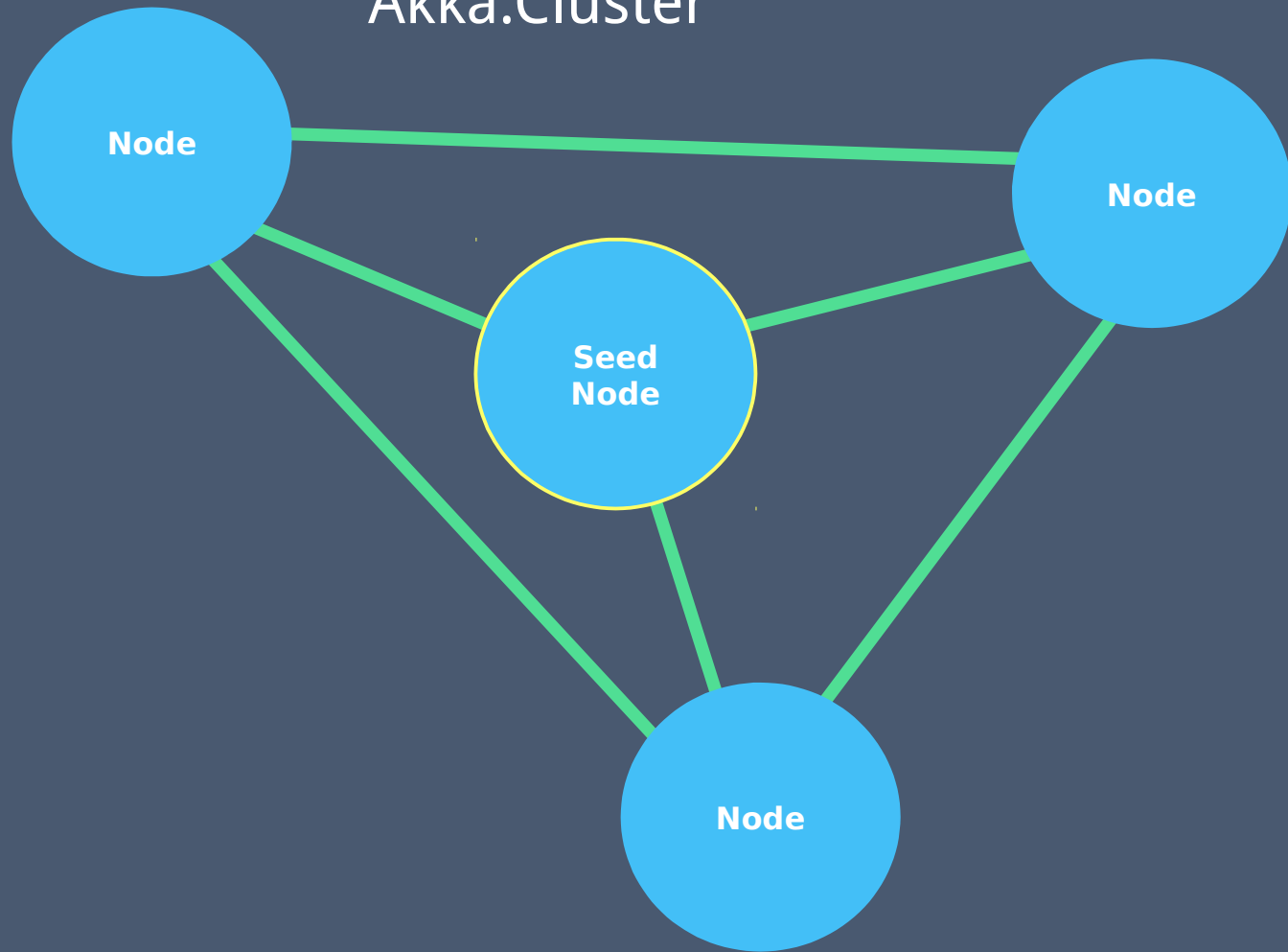
Akka.Cluster



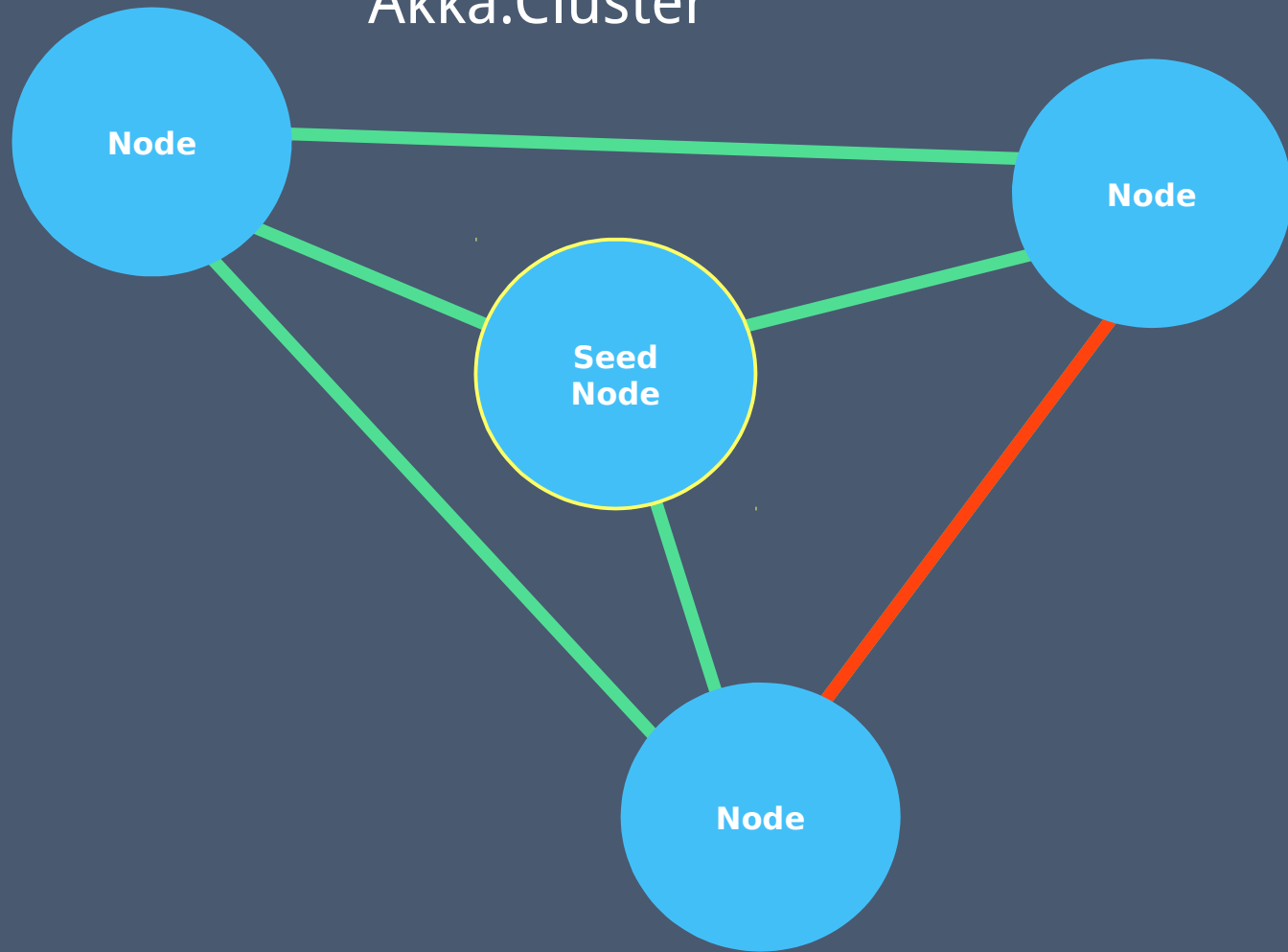
Akka.Cluster



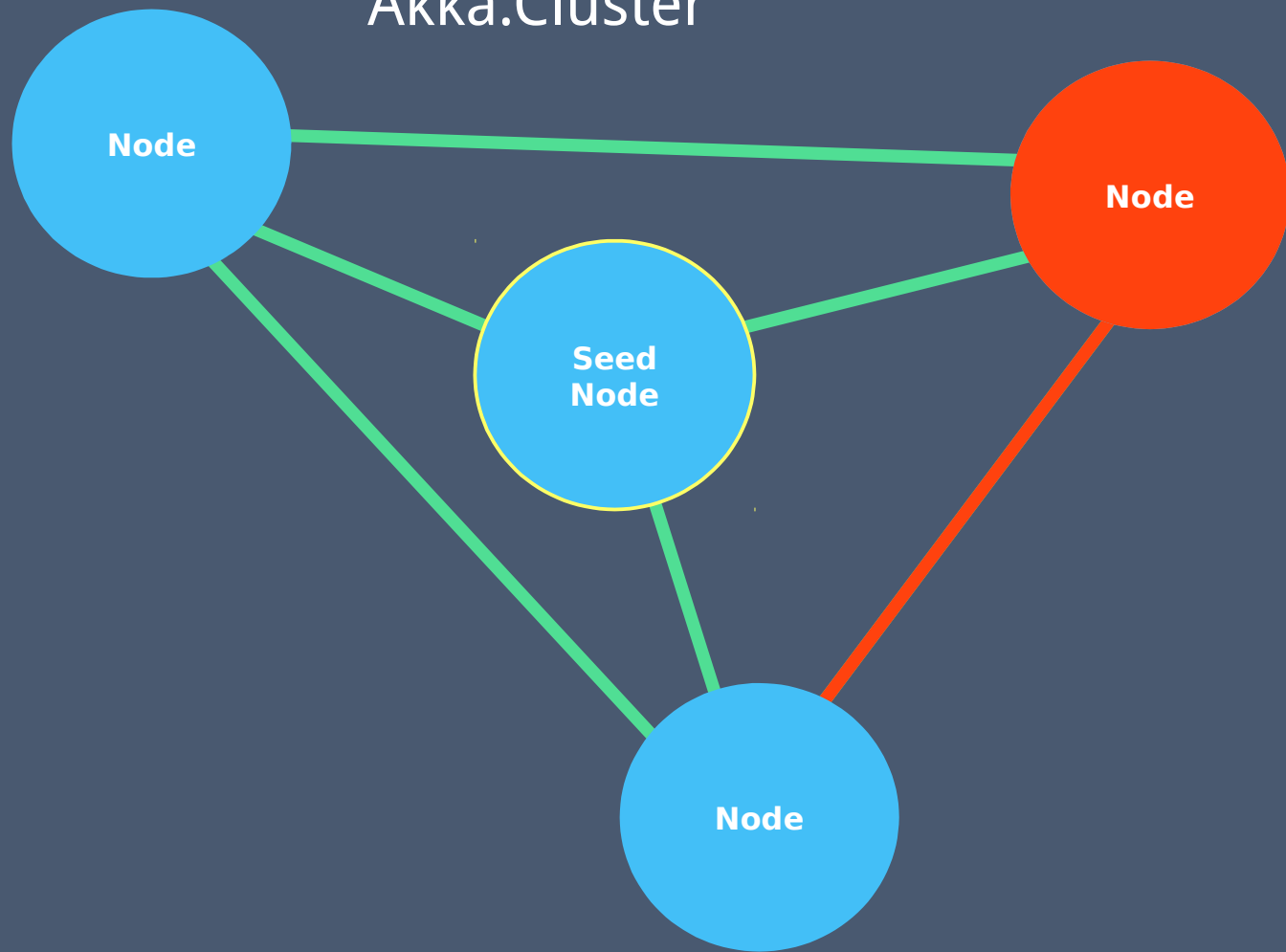
Akka.Cluster



Akka.Cluster



Akka.Cluster



Akka.Routing

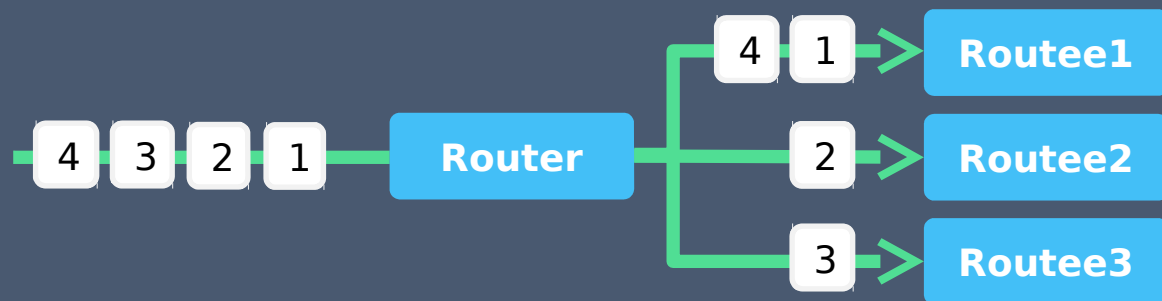
По контролю времени жизни:

- Pool
- Group

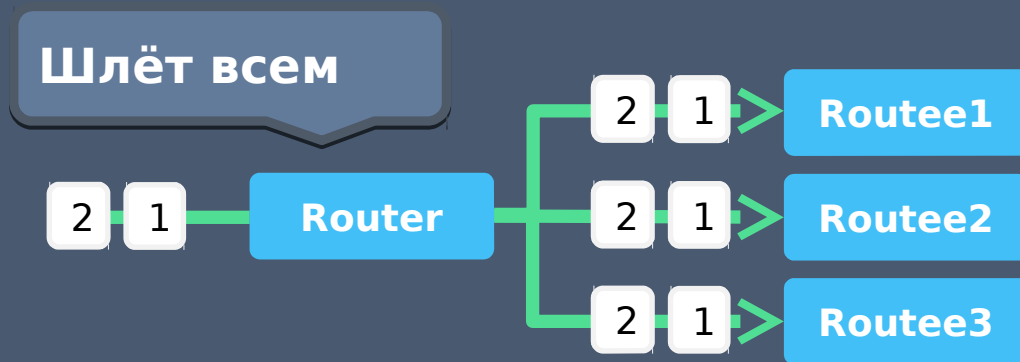
По логике маршрутизации

- RandomRouter
- SmallestMailboxRouter
- BroadcastRouter
- RoundRobinRouter
- ConsistentHashRouter
- ScatterGatherFirstCompletedRouter
- TailChoppingRouter

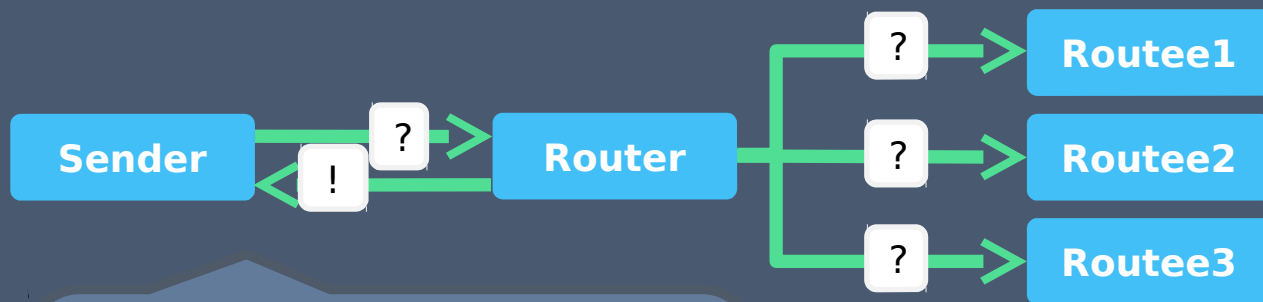
RoundRobinRouter



BroadcastRouter

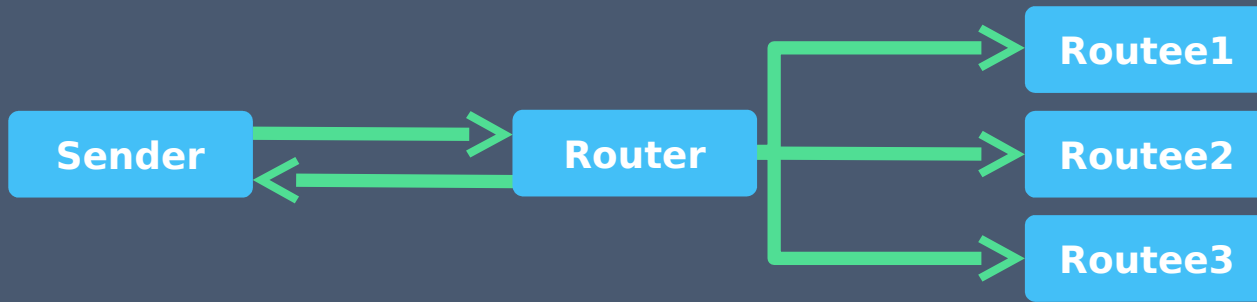


ScatterGatherFirstCompletedRouter

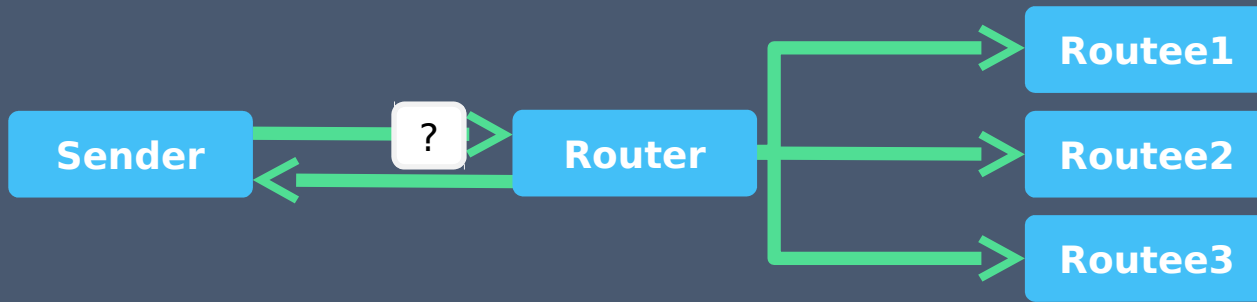


Шлём всем, получаем
первый ответ, отправляем
запрашивающему
(Task.WaitAny)

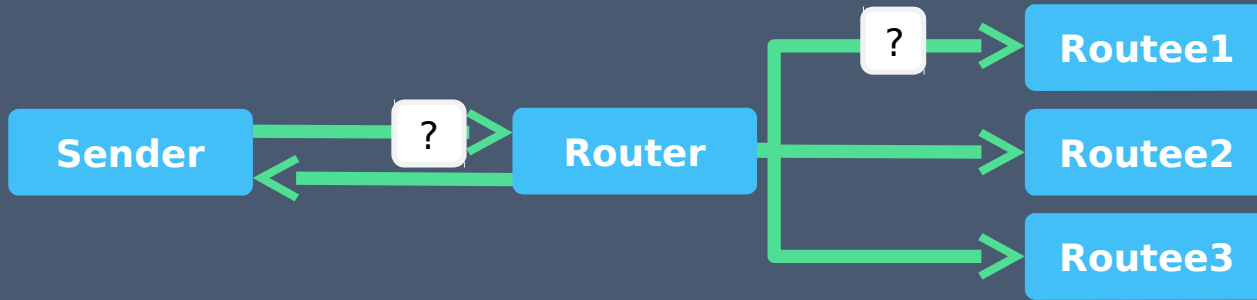
TailChoppingRouter



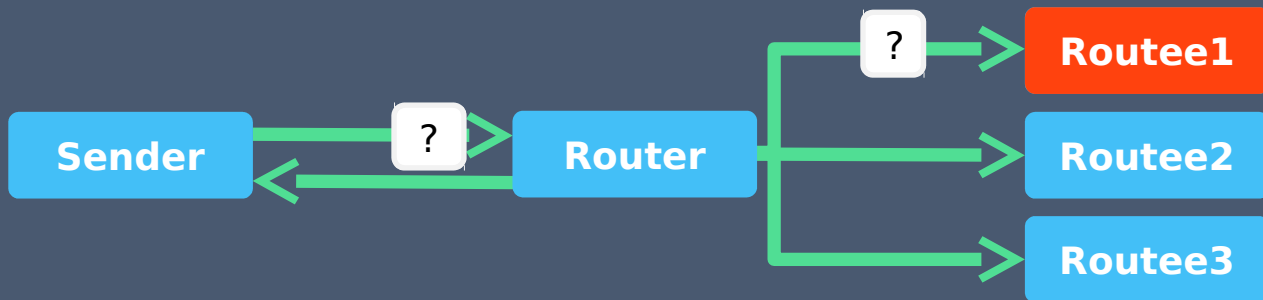
TailChoppingRouter



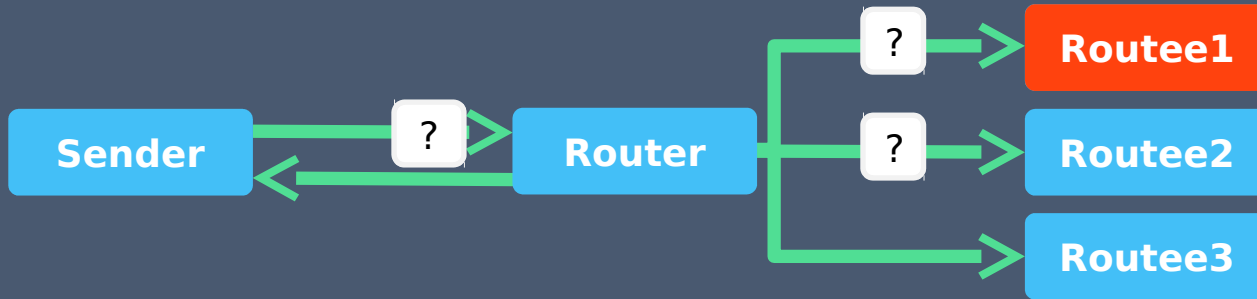
TailChoppingRouter



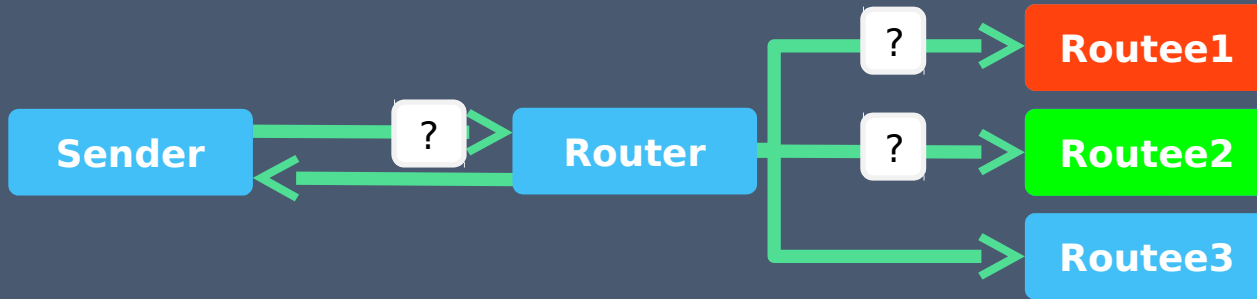
TailChoppingRouter



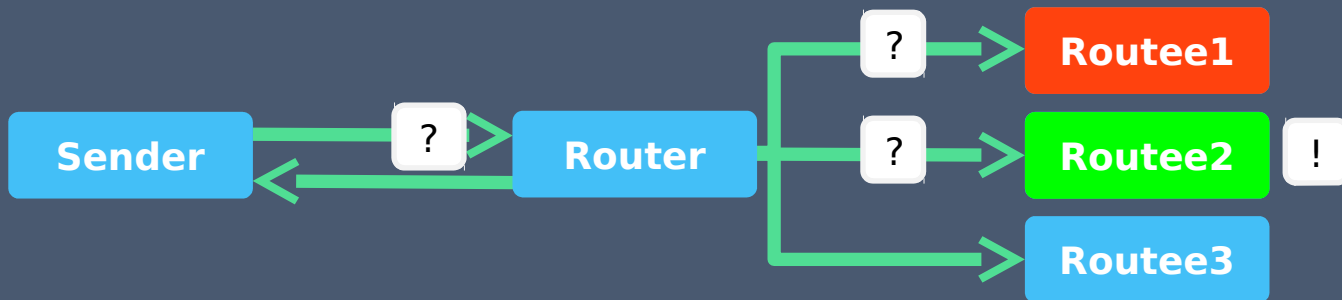
TailChoppingRouter



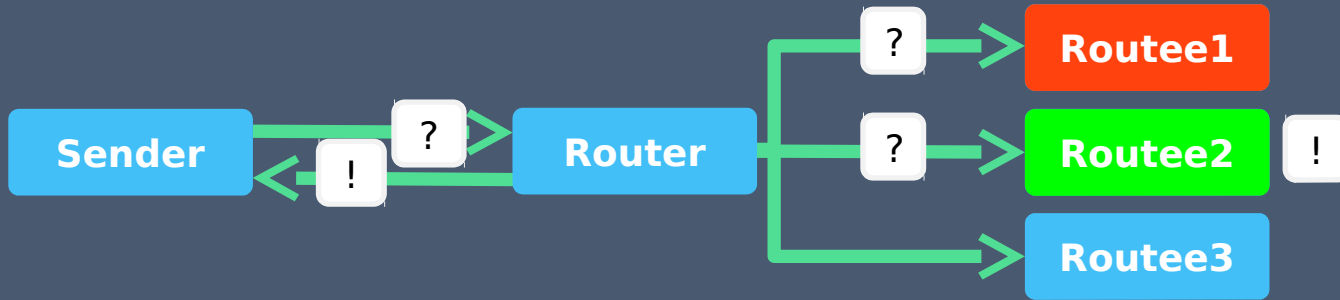
TailChoppingRouter



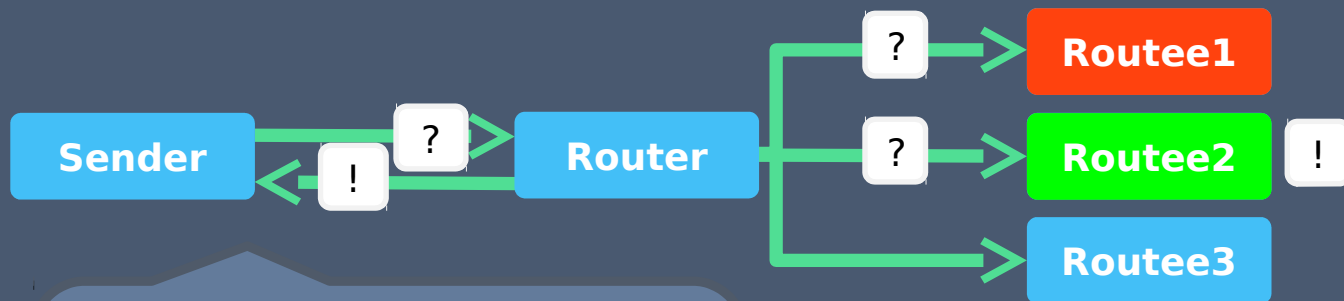
TailChoppingRouter



TailChoppingRouter

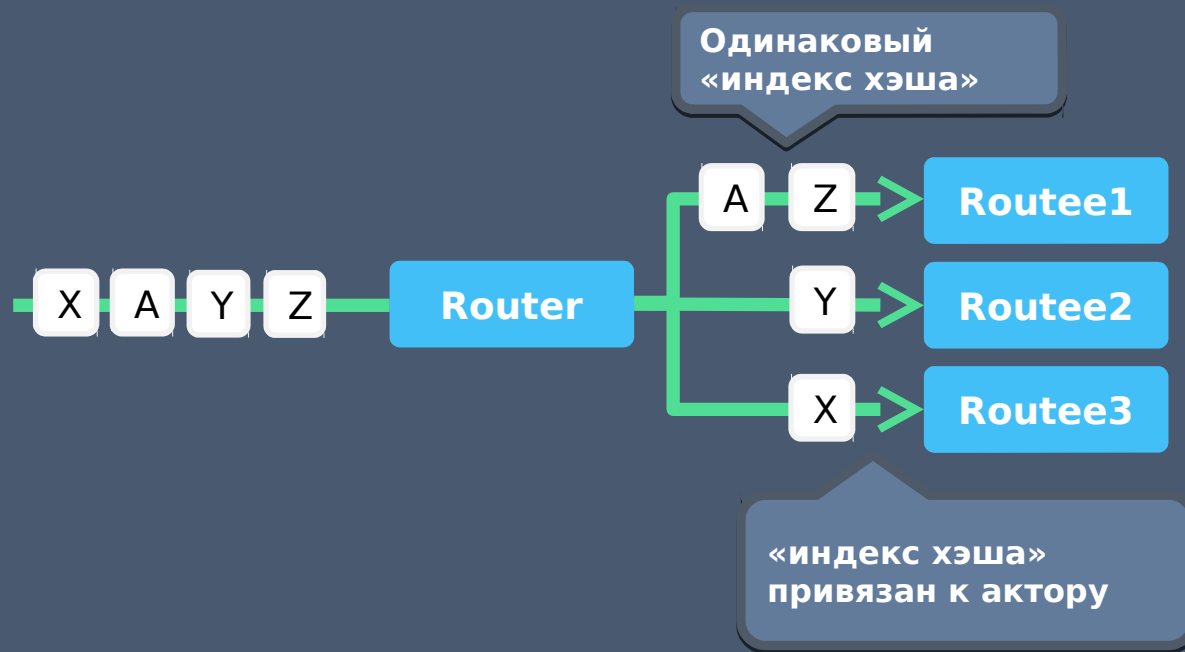


TailChoppingRouter

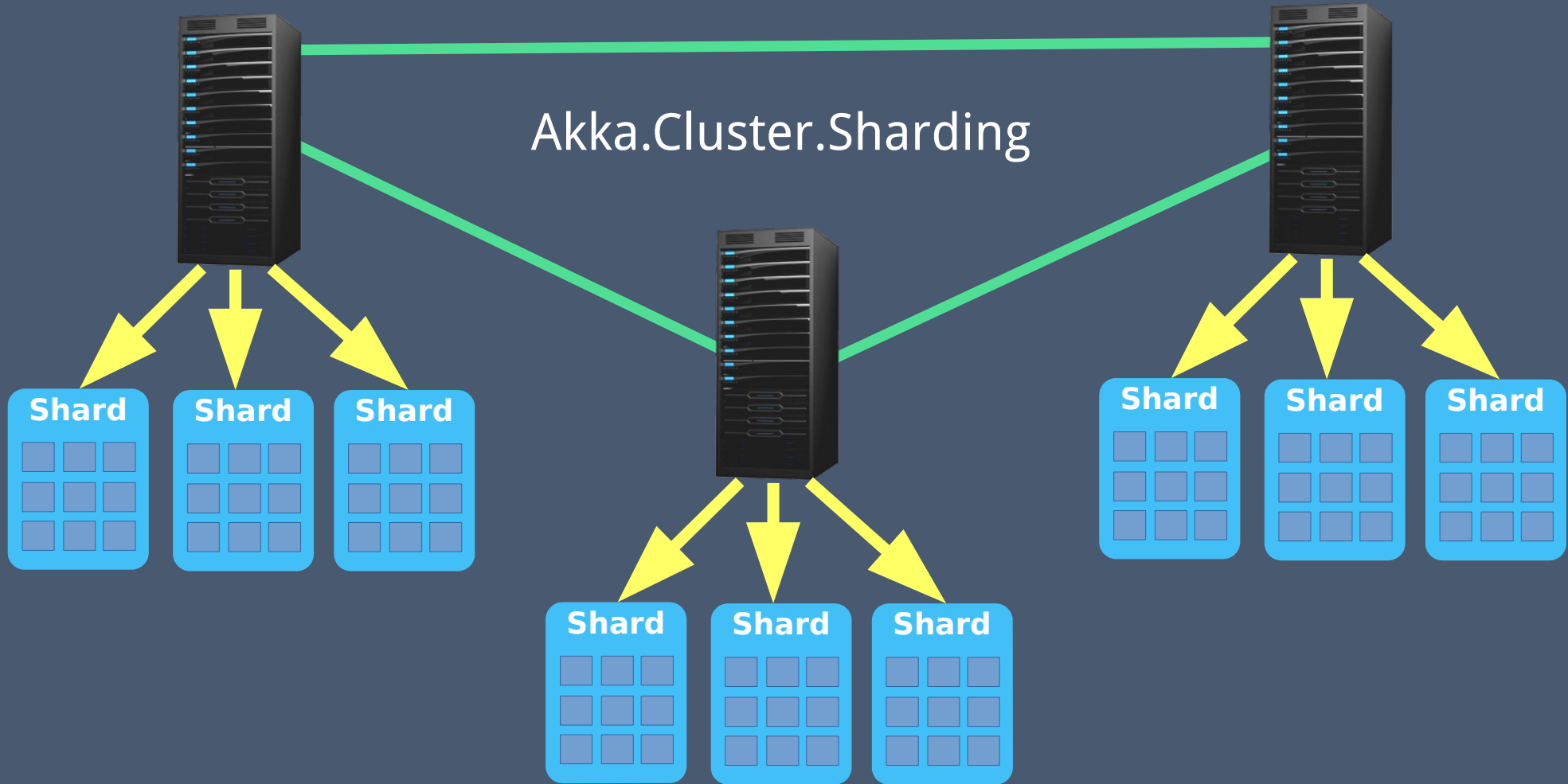


Шлёт случайному, если
не получает ответа, шлёт
следующему

ConsistentHashRouter



Akka.Cluster.Sharding



Акка.NET

Сайт

<http://getakka.net/>

Блог разработчиков

<https://petabridge.com/blog/>

Спикер

Никита Цуканов

Email: nikita.d.tsukanov@gmail.com

Skype: [kekekeks](#)

Код из презентации:

<https://github.com/kekekeks/msk.net-2016-08-11>