Разработка распределенных отказоустойчивых систем на платформе Erlang

2012 CEE-SEC(R)
Software Engineering Conference in Russia

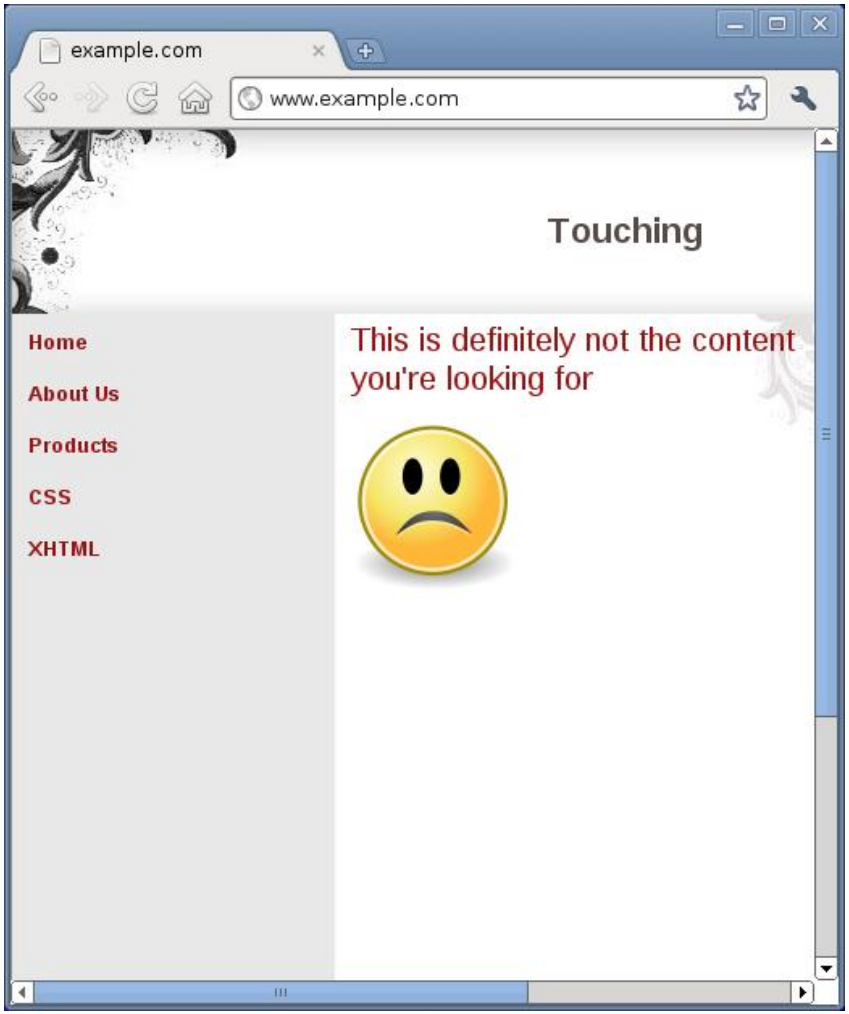
Рексофт: Инновации на заказ

РАССМАТРИВАЕМЫЕ СИСТЕМЫ

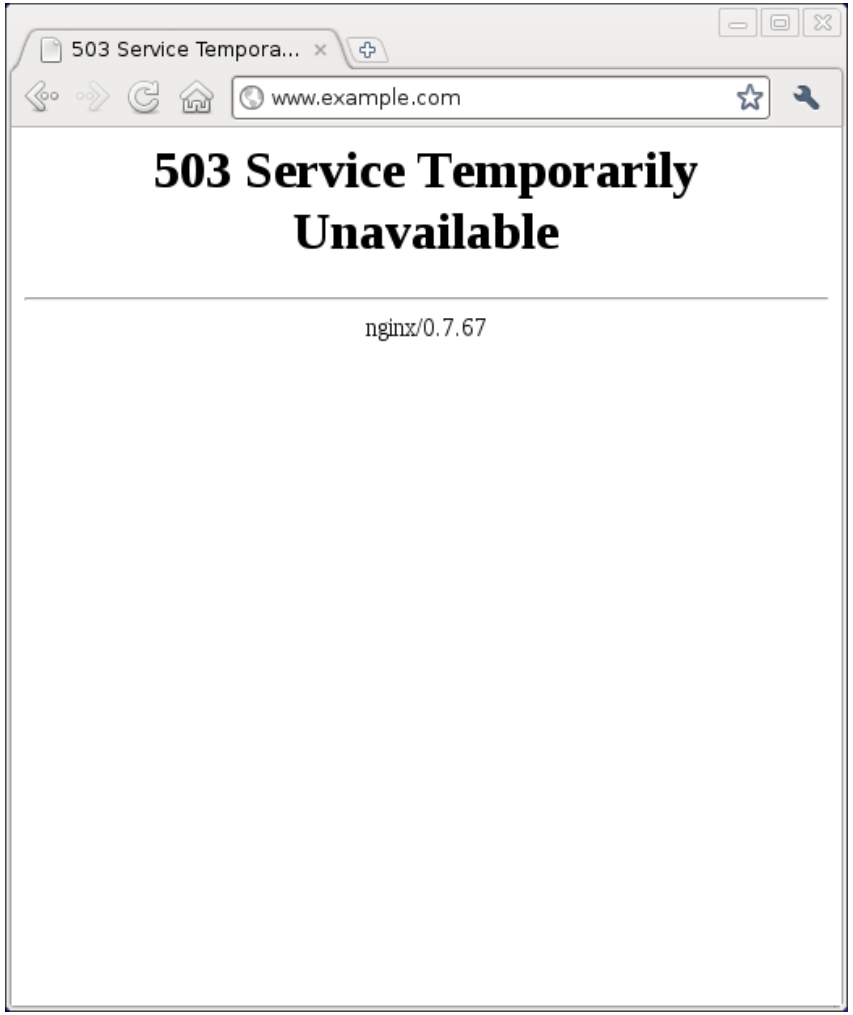
Рассматриваемые системы

- Множество запросов
- Масштабируемость
- Высокая доступность

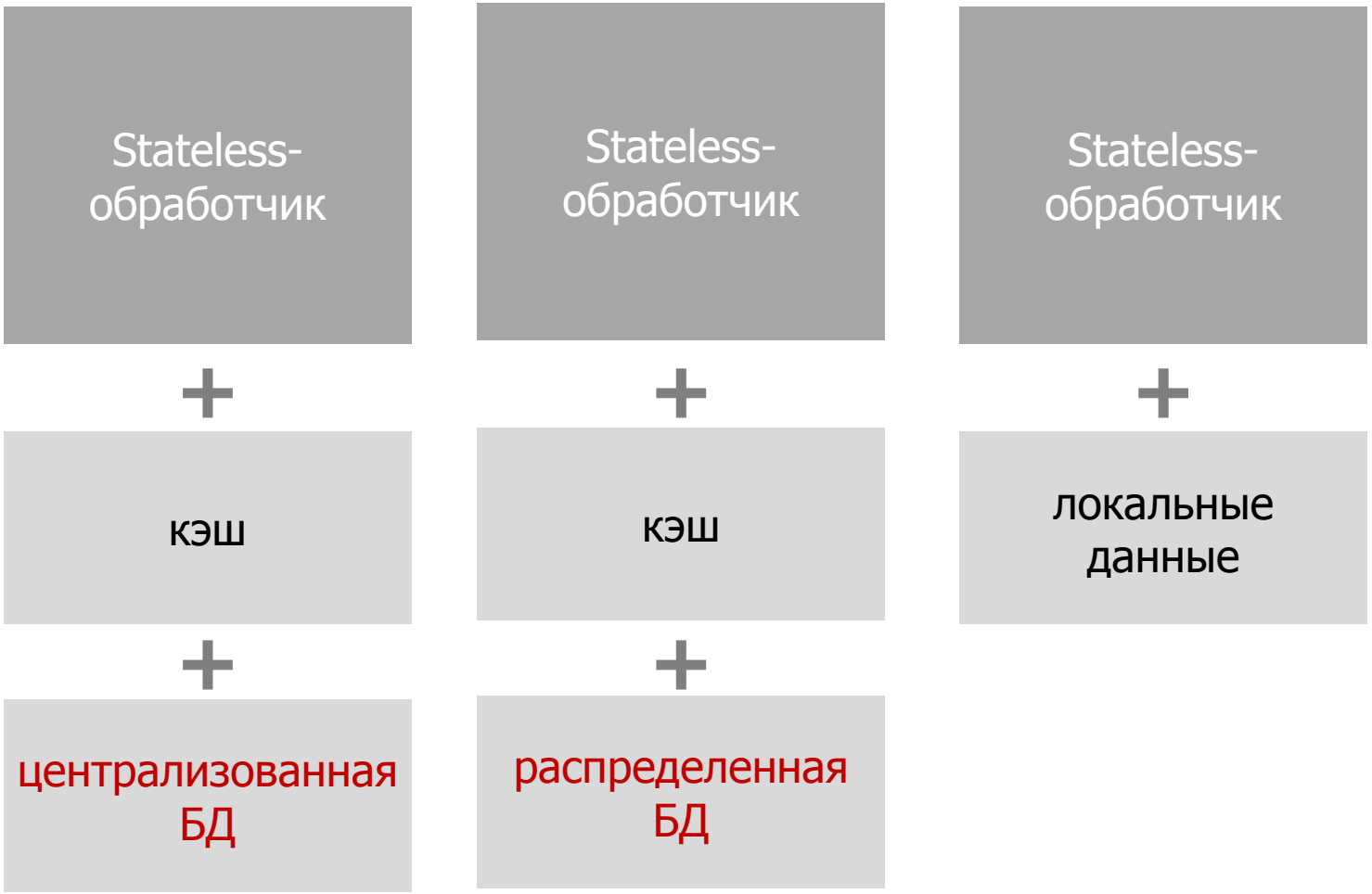
Проблемы: синхронизация



Проблемы: мониторинг



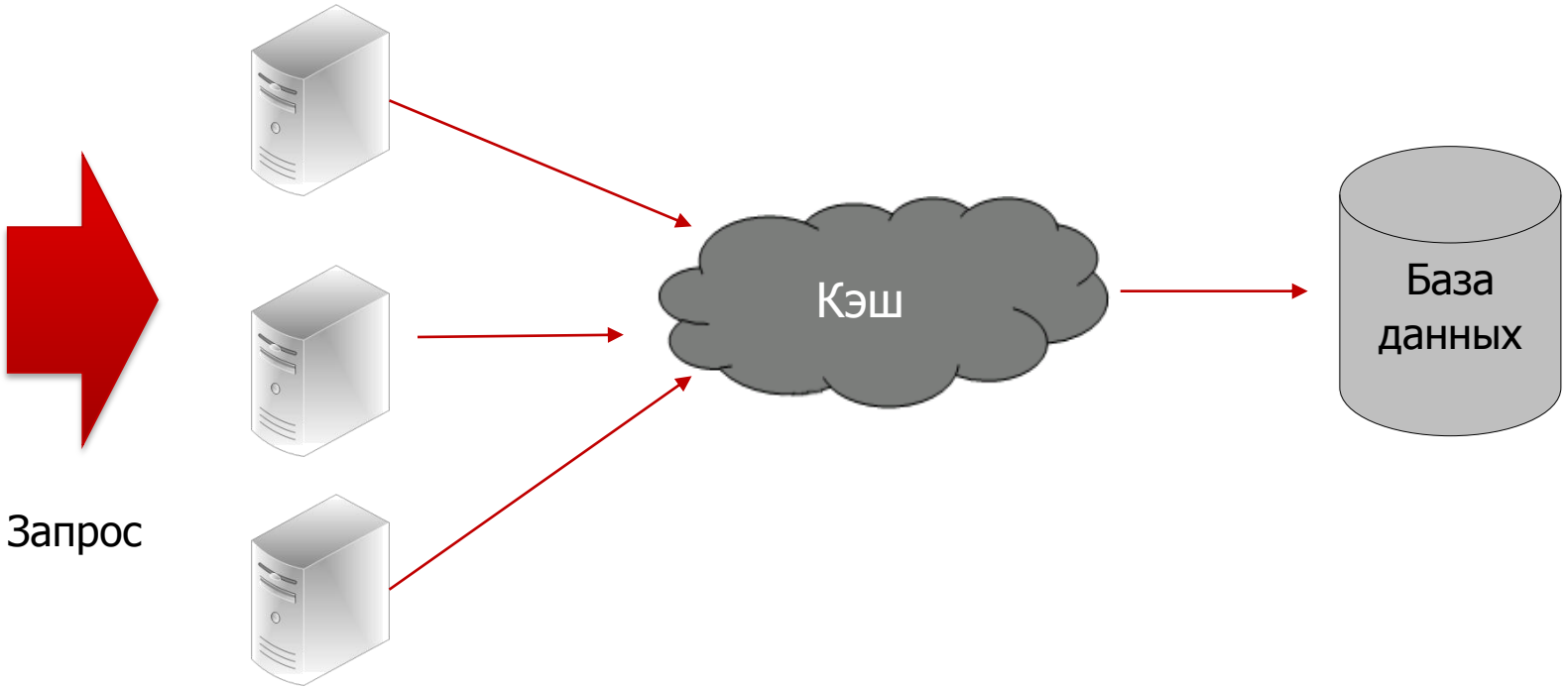
Стандартные решения



Стандартное решение #1
Stateless-обработчик + кэш + централизованная БД

Reksoft

PHP + Memcached + MySQL

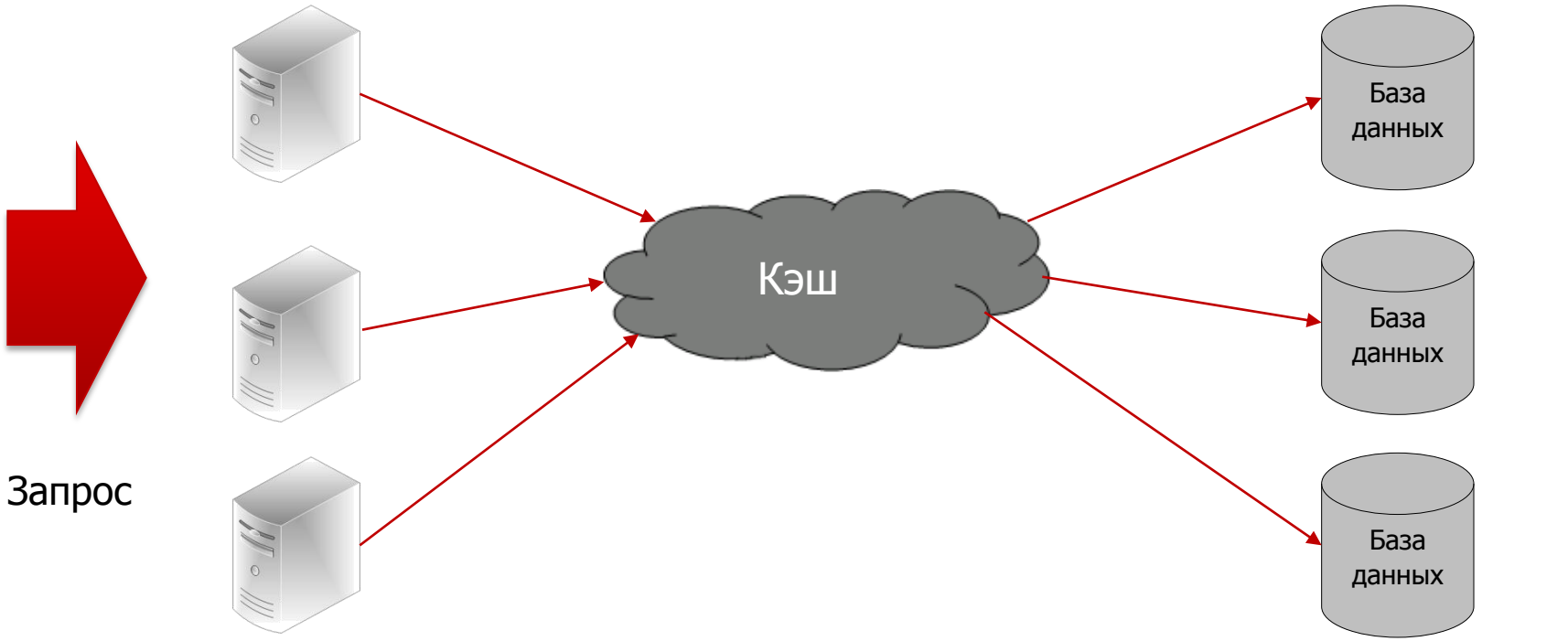


Стандартное решение #1
Stateless-обработчик + кэш + централизованная БД

- Любое изменение состояния – обращение к БД
- Проблемы инвалидации кэша
- Ограниченная масштабируемость
- БД – single-point of failure

Стандартное решение #2
Stateless-обработчик + кэш + распределенная БД

PHP + Memcached + MongoDB



Стандартное решение #2
Stateless-обработчик + кэш + распределенная БД

- Любое изменение состояния – обращение к БД
- Проблемы инвалидации кэша
- Неравномерность распределения данных
- Ограниченные возможности выборки данных

Стандартное решение #3
Stateless-обработчик + локальные данные

Nginx + PHP + Local MySQL или FS

Стандартное решение #3
Stateless-обработчик + локальные данные

- Ограниченная применимость
- Отсутствие стандартных решений для детектирования отказов
- Неравномерность нагрузки

Reksoft

ПЛАТФОРМА ERLANG

Язык Erlang

Появился в: 1987

Авторы: Ericsson Computer Science Laboratory

Назван в честь датского математика и инженера Агнера Эрланга, основателя научного направления по изучению сетевого трафика в телекоммуникационных системах. По другой версии название языка изначально было сокращением от «ericsson language»

Особенности платформы Erlang

- Функциональный стиль
- Отсутствие побочных эффектов
- Интерактивная разработка
- Легковесные процессы
- Изолированность процессов
- Принцип «Let it crash»
- Прозрачная распределенность

Функциональный стиль

Erlang

```
do_fib(_,B,1) -> B;
do_fib(A,B,N) -> do_fib(B,A+B,N-1).
```

C

```
int fibonacci(int is)
{
 int first = 0, second = 1;

 for (int i = 0; i < is - 1; ++i)
 {
  int sum = first + second;
  first = second;
  second = sum;
 }

 return first;
}
```

Лексические переменные

Erlang

```
invalid_fun() ->
  A = 1,
  B = 1,
  A = 2. % Ошибка
```

C

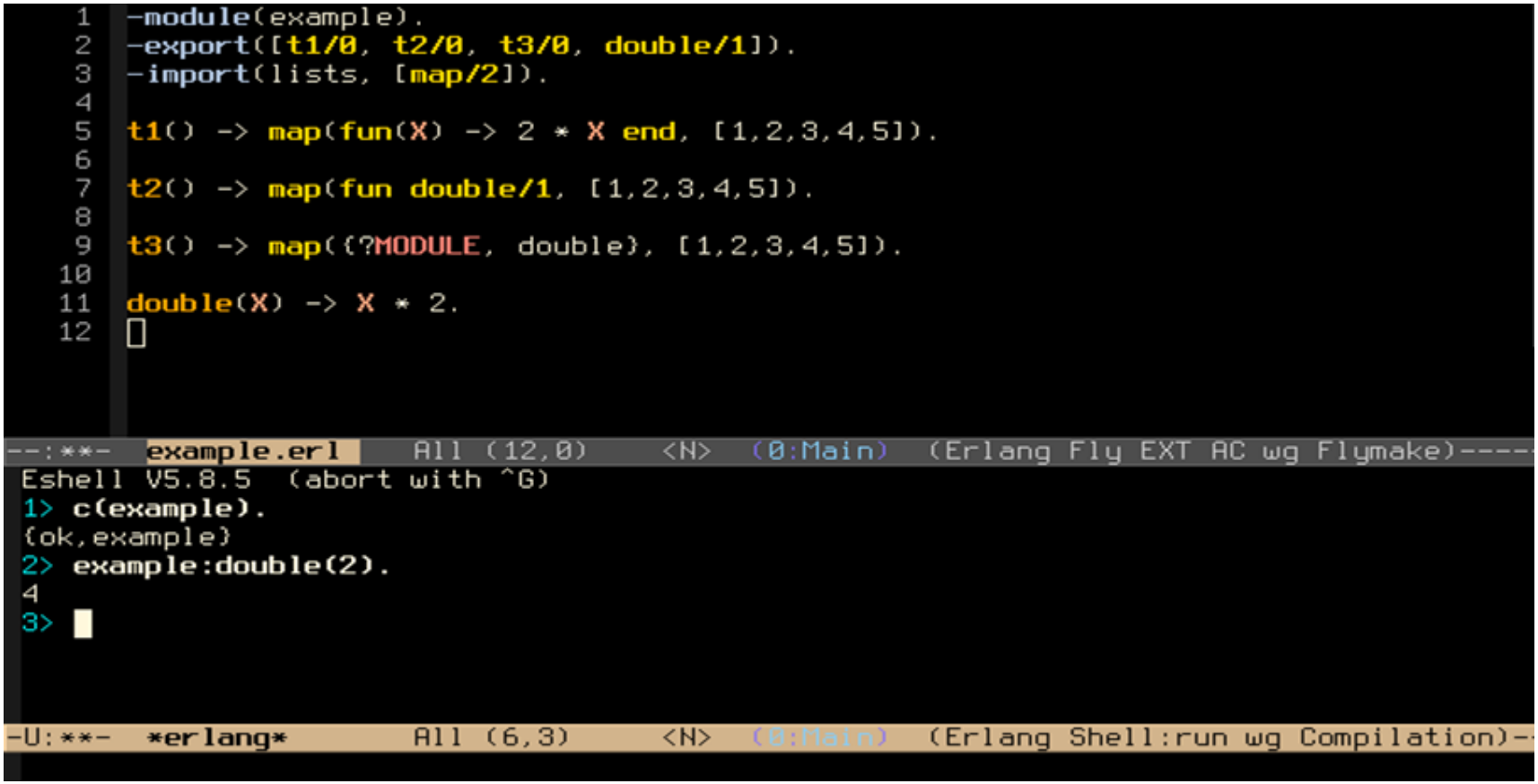
```
int valid_fun() {
  int a = 1;
  int b = 1;
  a = 2;
}
```

Отсутствие побочных эффектов

- Результат функции определяется только её аргументами
- Аргументы передаются по значению
- Функция не может изменять внешнее состояние
- Функции могут вычисляться в любом порядке

Интерактивная разработка

Reksoft



Интерактивная разработка

Reksoft

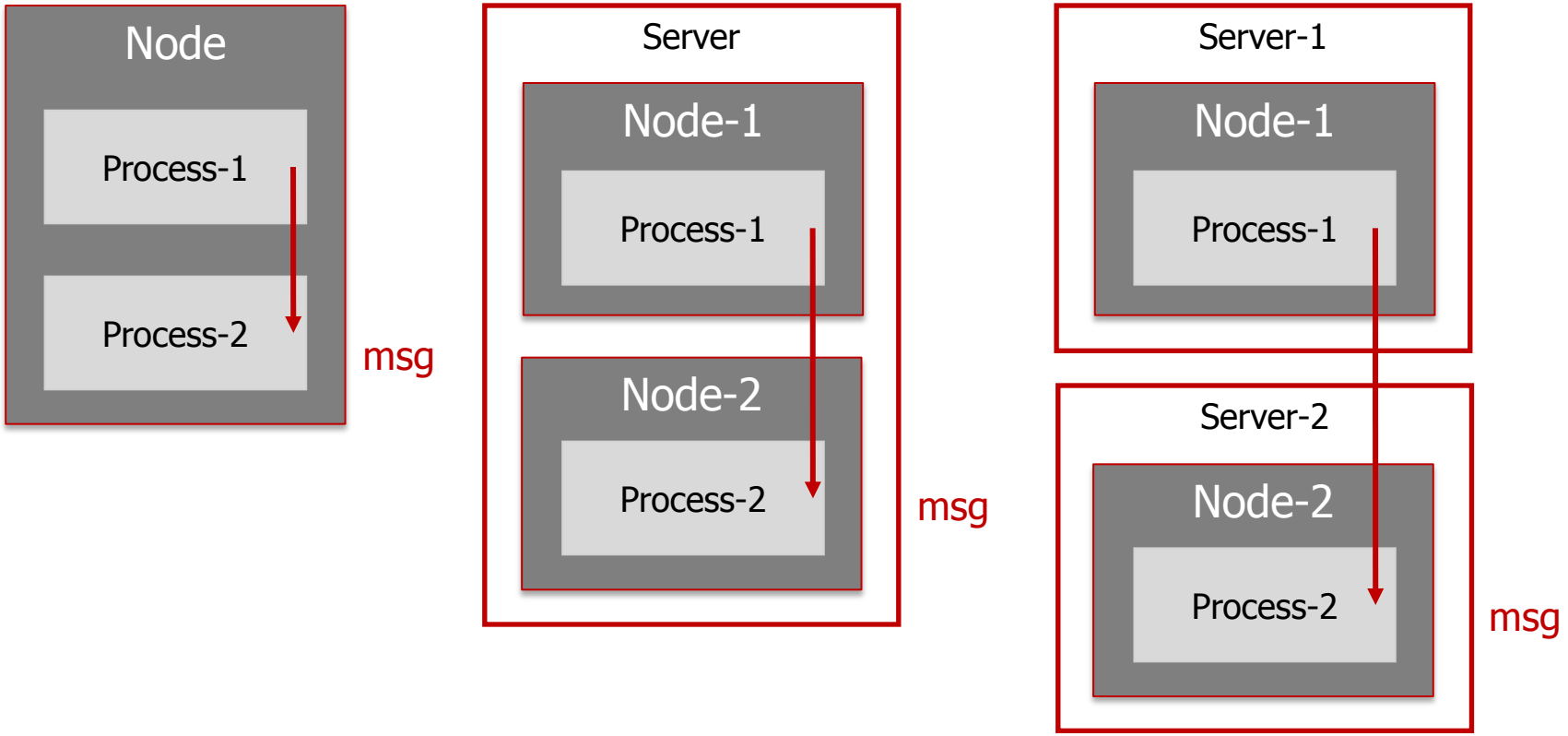
- Горячая замена кода
- Мгновенная обратная связь
- Простота тестирования чистых функций
- Способствует разработке в «состоянии потока»

Процессы

```
ping() ->
  Pid = spawn(fun pong/0, []),
  Pid ! {ping, self()},
  receive
        pong ->
              io:format("Pong!\n")
  end.
```

```
pong() ->
  receive
        {ping, Pid} ->
              io:format("Ping!\n"),
              Pid ! pong
  end.
```

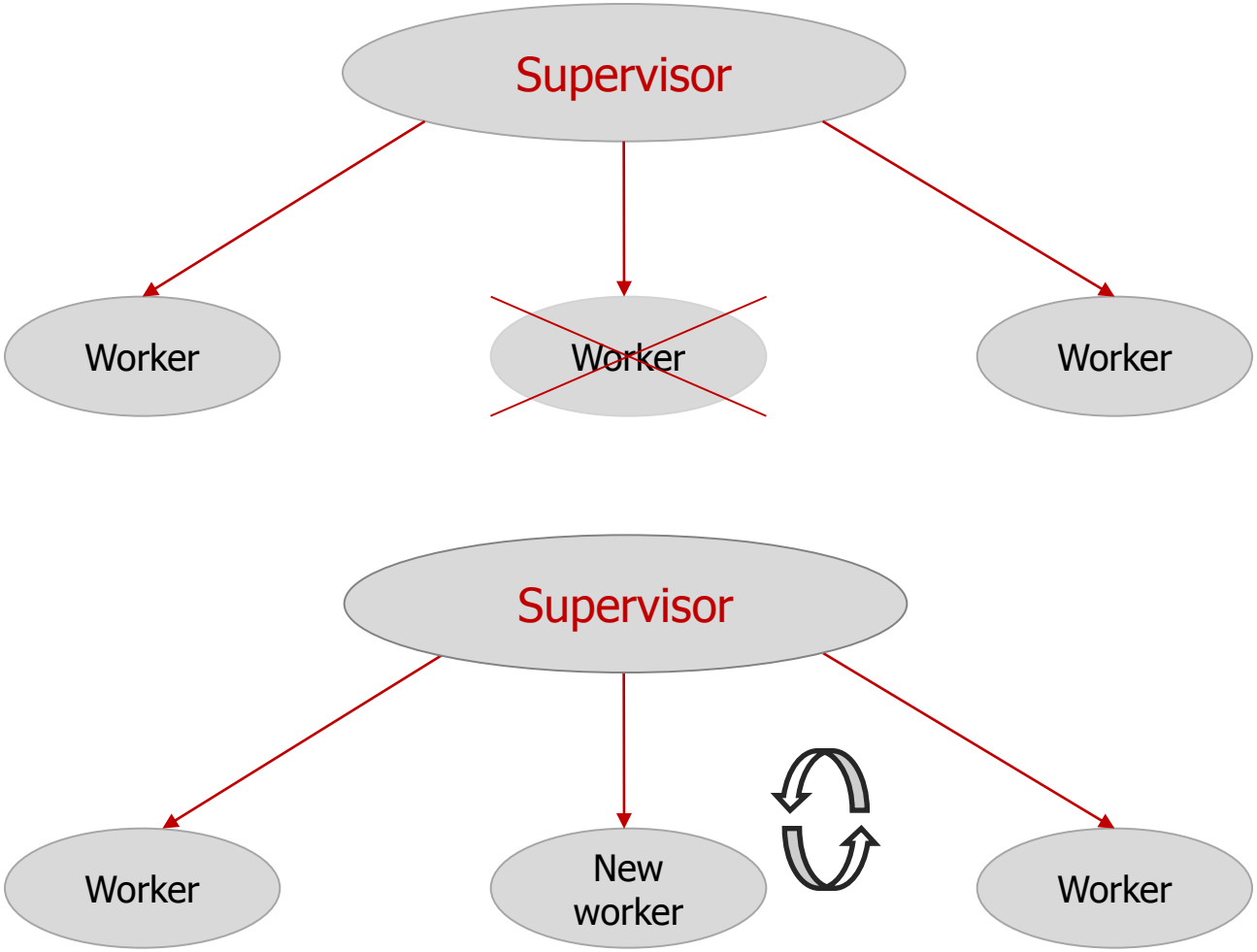
Сетевая прозрачность



Типы отказов

- Отказ аппаратной части
- Зависание
- Аварийное завершение программы
- Ошибки в коде (исключительные ситуации)

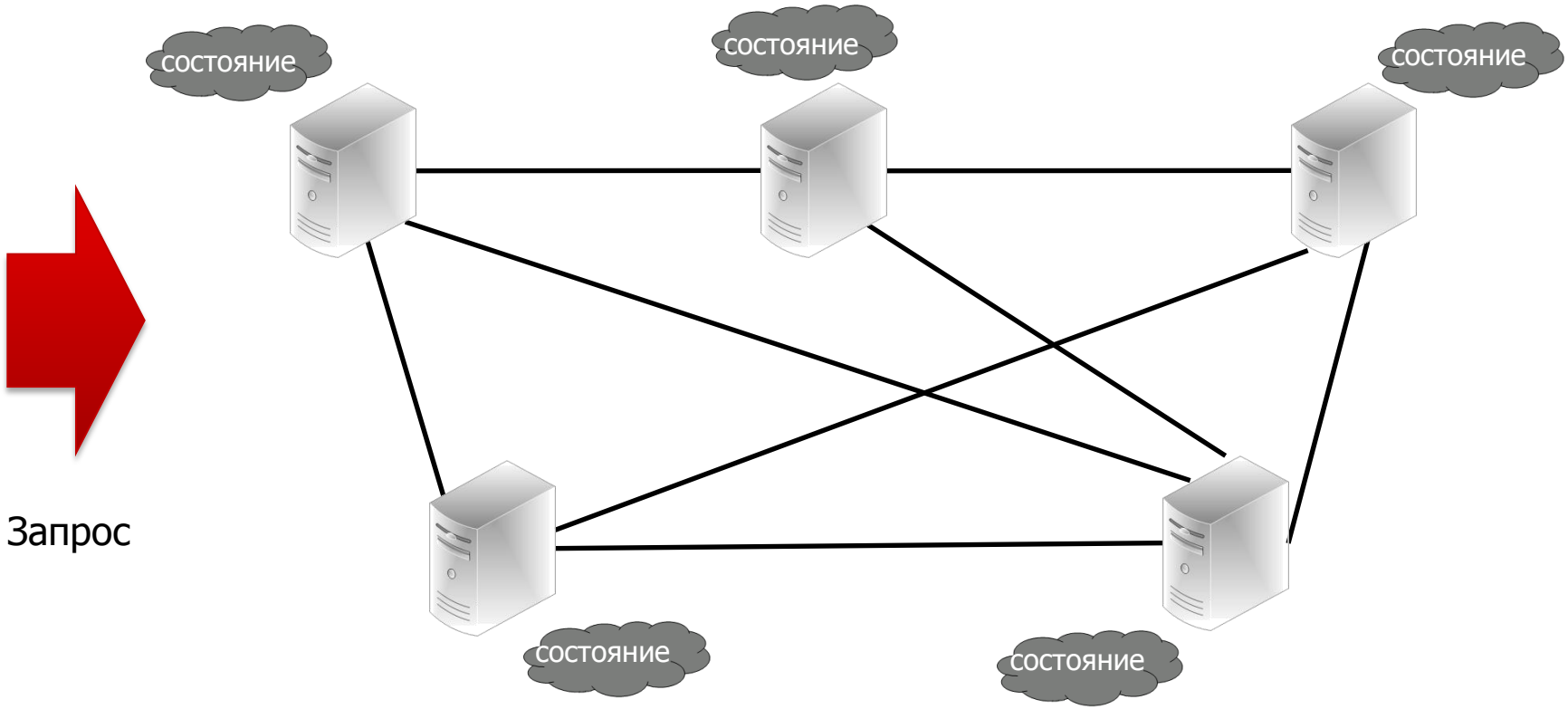
Let it crash


Supervisor
Worker
Worker
Worker
Supervisor
Worker
New worker
Worker

Решение проблем: синхронизация

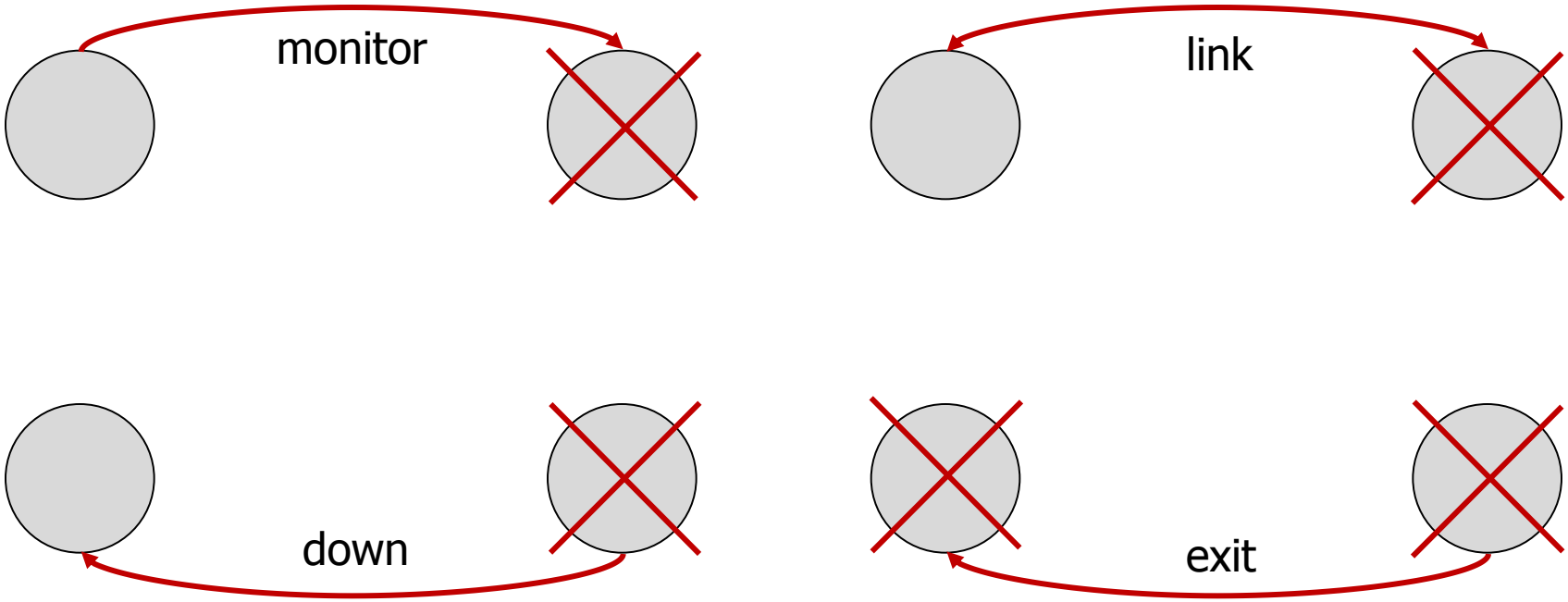
Reksoft

Stateful-процессы и сетевая прозрачность



Решение проблем: мониторинг

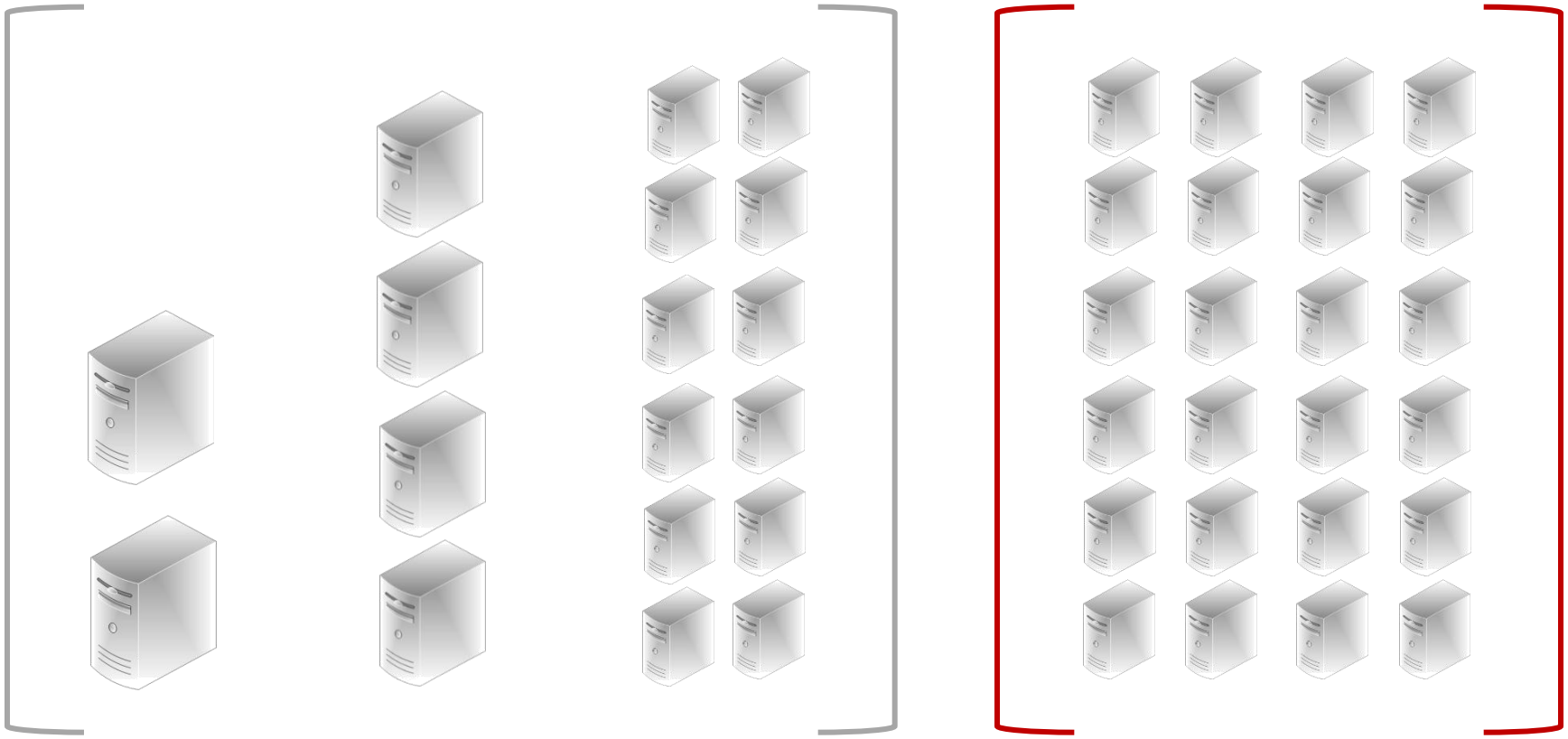
Мониторинг средствами стандартной библиотеки



Применимость Erlang

Reksoft

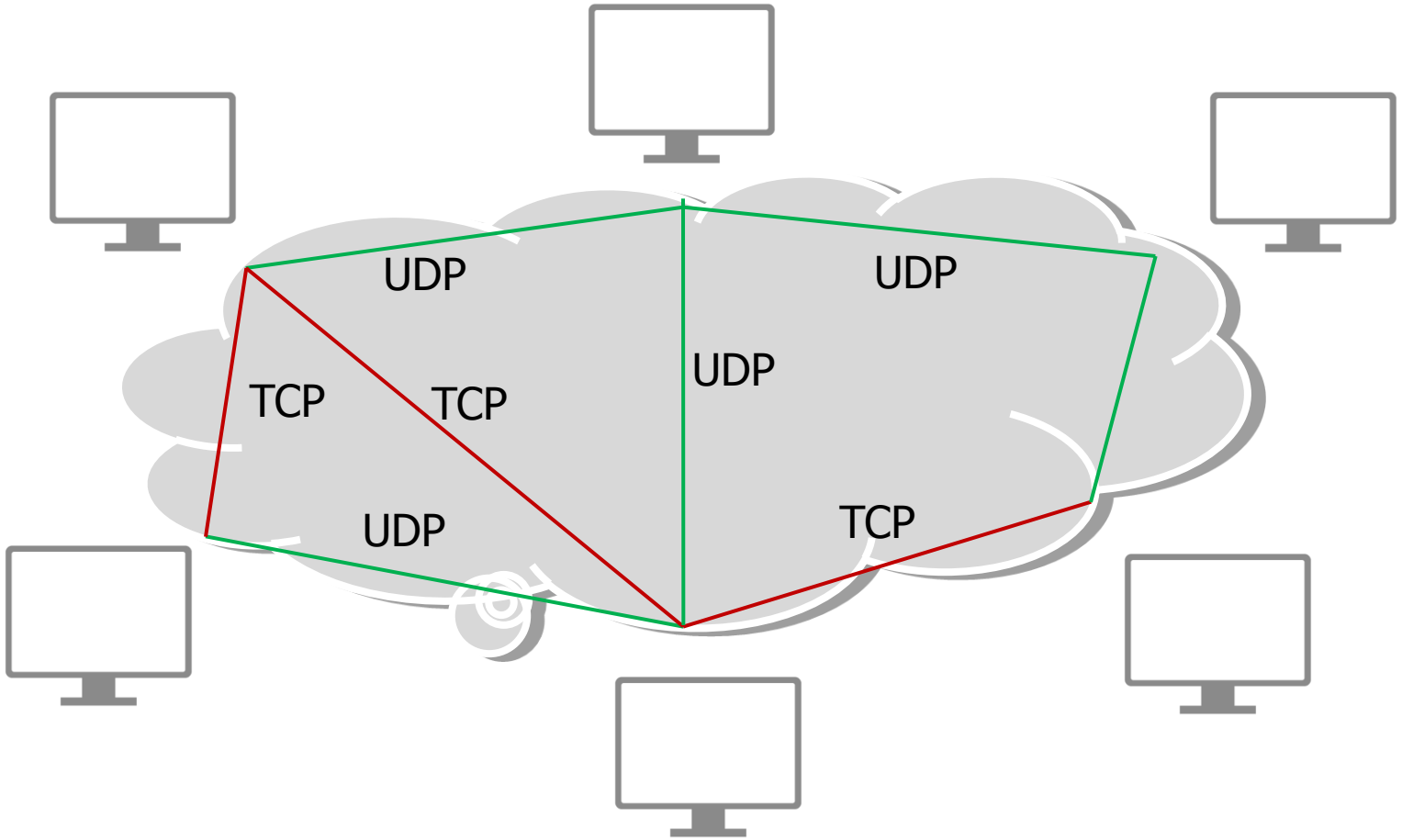
Кластеры



2 10 100 1000

Применимость Erlang

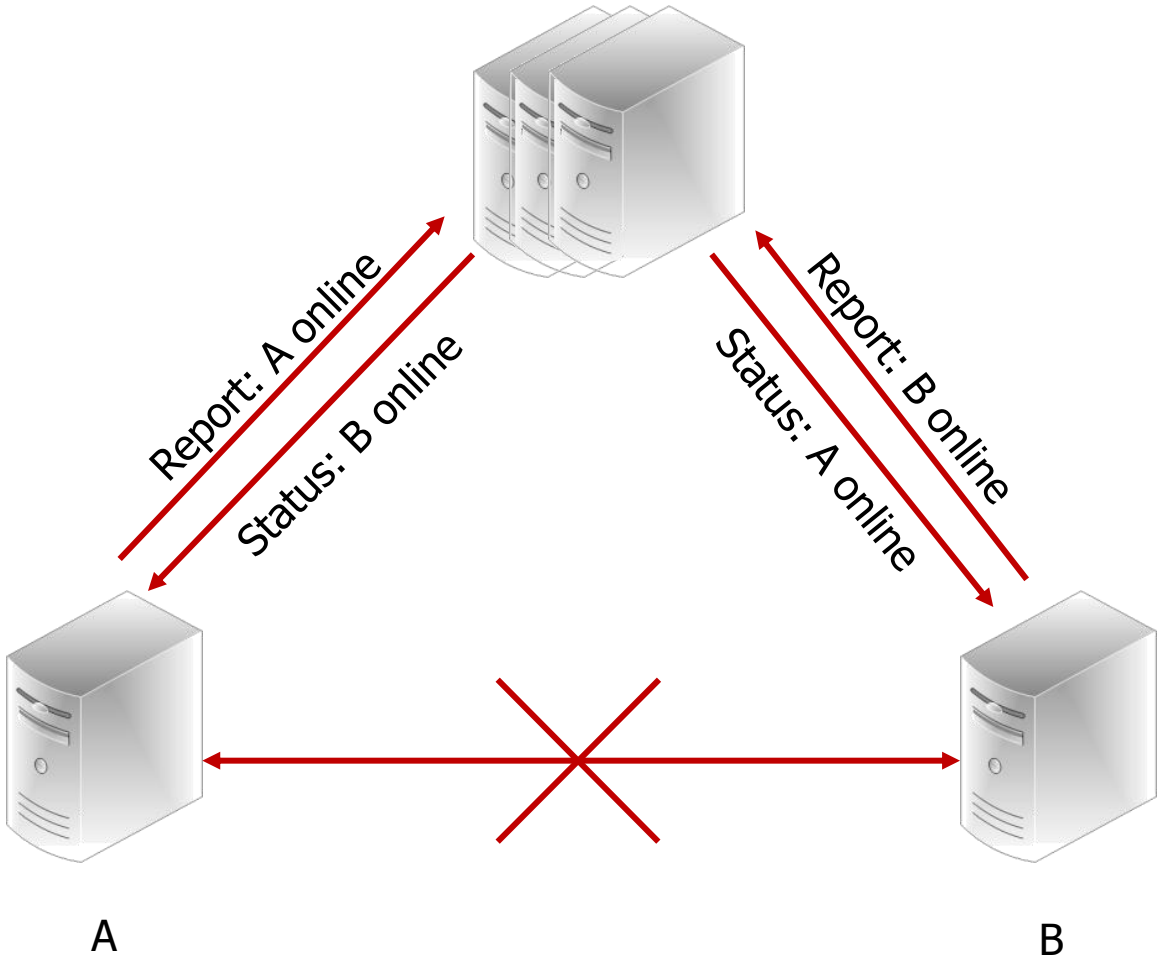
Peer-to-peer



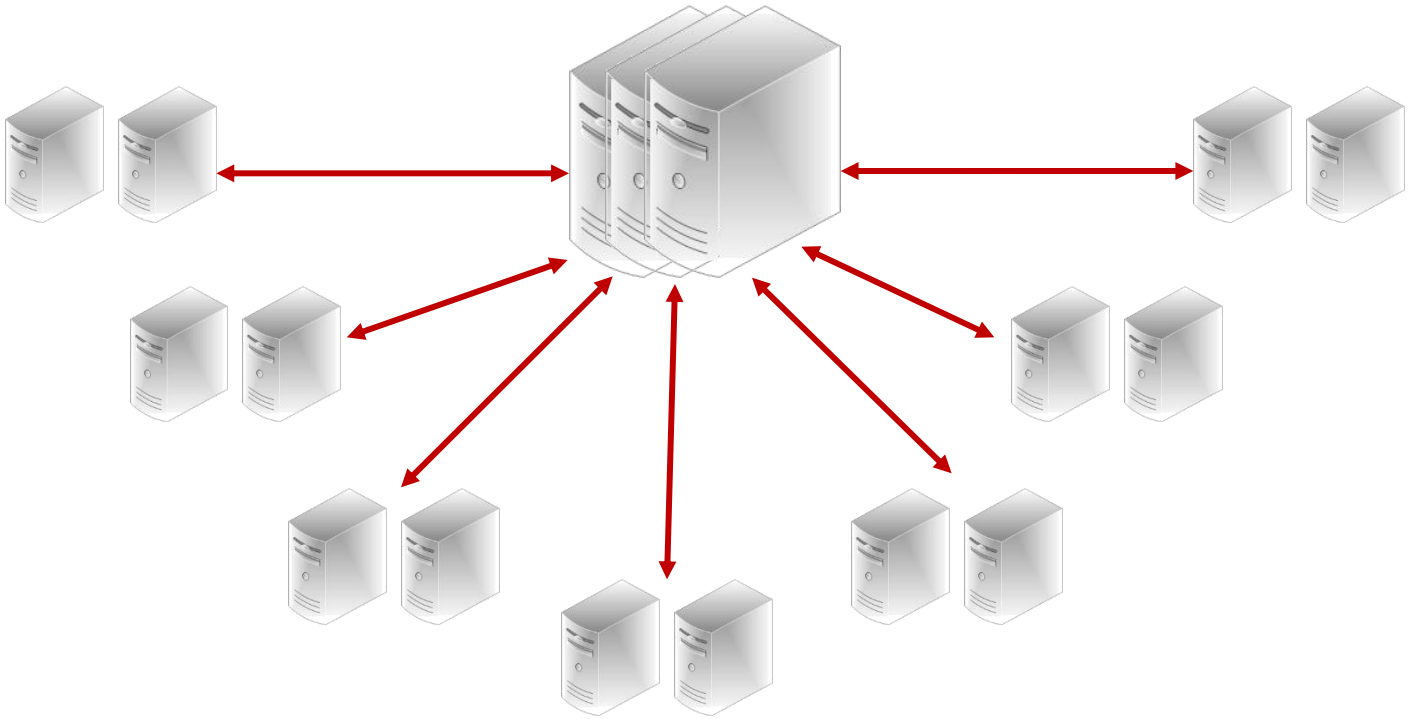
Reksoft

РЕАЛЬНОЕ ПРИМЕНЕНИЕ

Пример реального применения



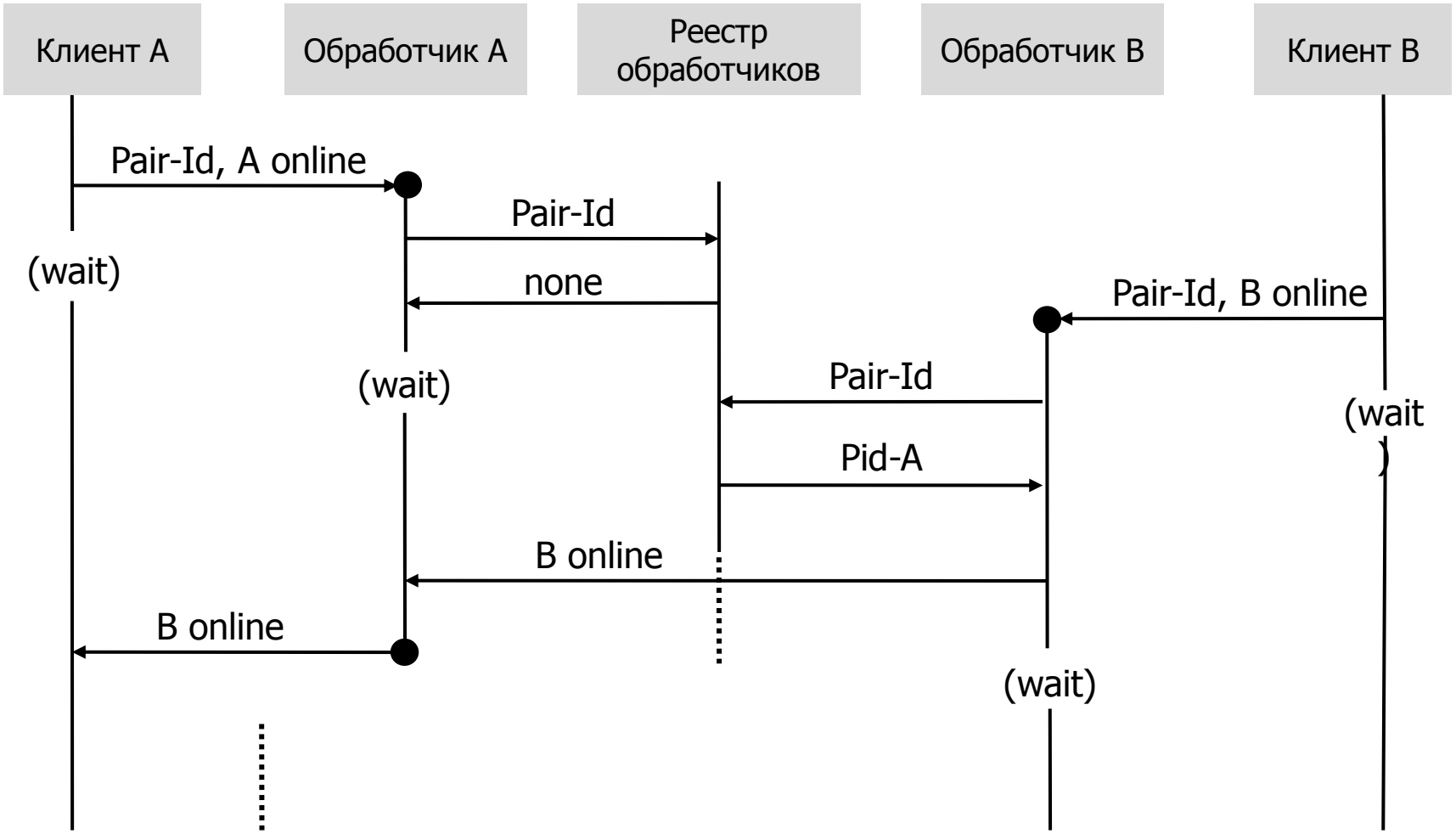
Пример реального применения



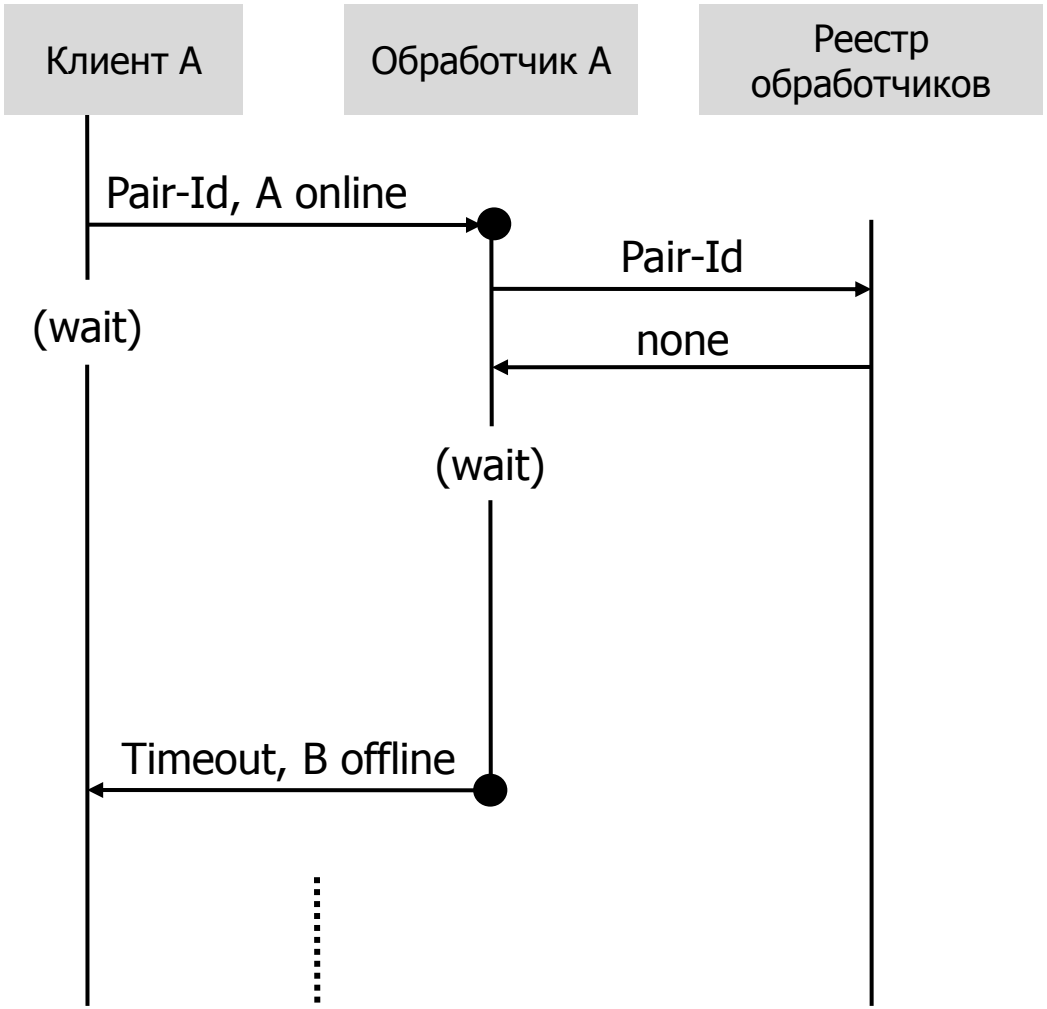
Архитектура решения

- HTTP long polling
- На каждый запрос – отдельный процесс-обработчик
- Обработчики распределены по нескольким нодам
- Локальное хранение состояния в процессе-обработчике

Реализация


Клиент A
Обработчик A
Реестр обработчиков
Обработчик B
Клиент B
Pair-Id, A online
Pair-Id
(wait)
none
Pair-Id, B online
(wait)
Pair-Id
(wait)
Pid-A
B online
B online
(wait)

Реализация


Клиент A
Обработчик A
Реестр обработчиков
Pair-Id, A online
Pair-Id
(wait)
none
(wait)
Timeout, B offline

Реестр обработчиков

Reksoft

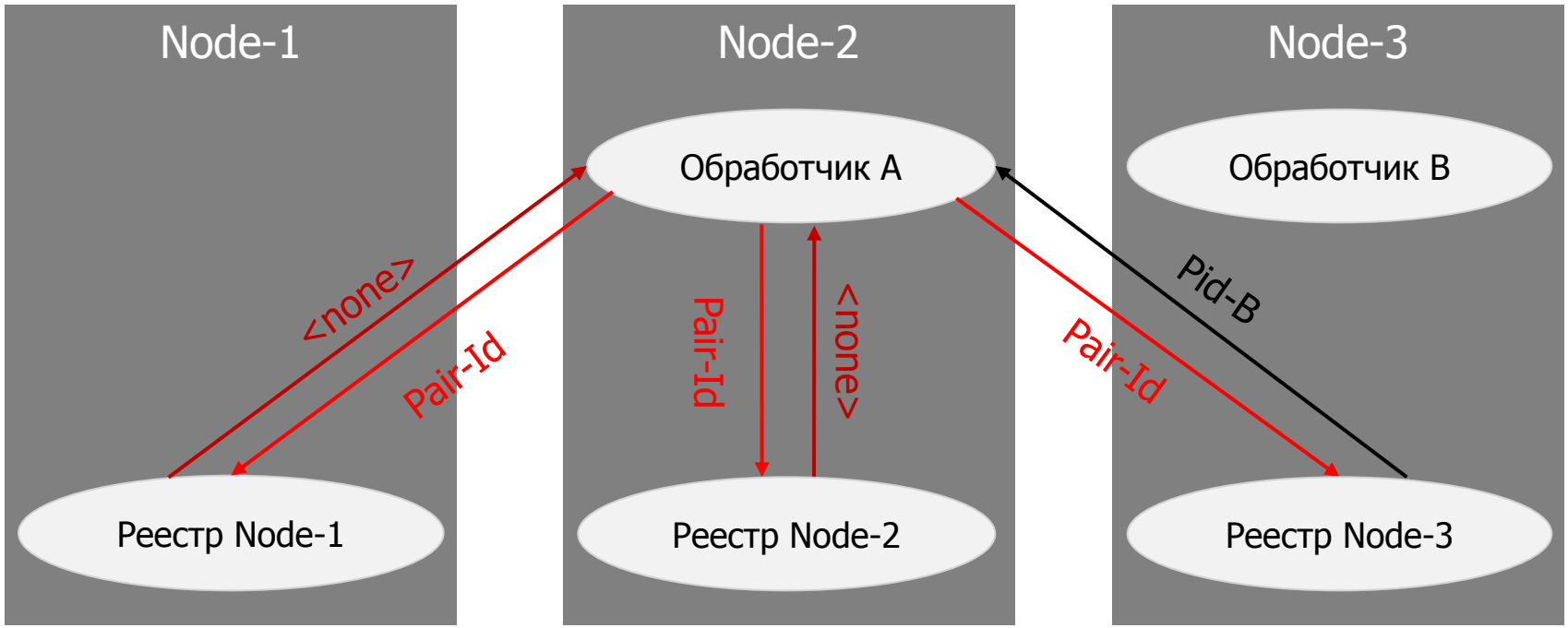
Node-1

Pair-Id-1	Pid-1
Pair-Id-2	Pid-2
Pair-Id-3	Pid-3

Node-2

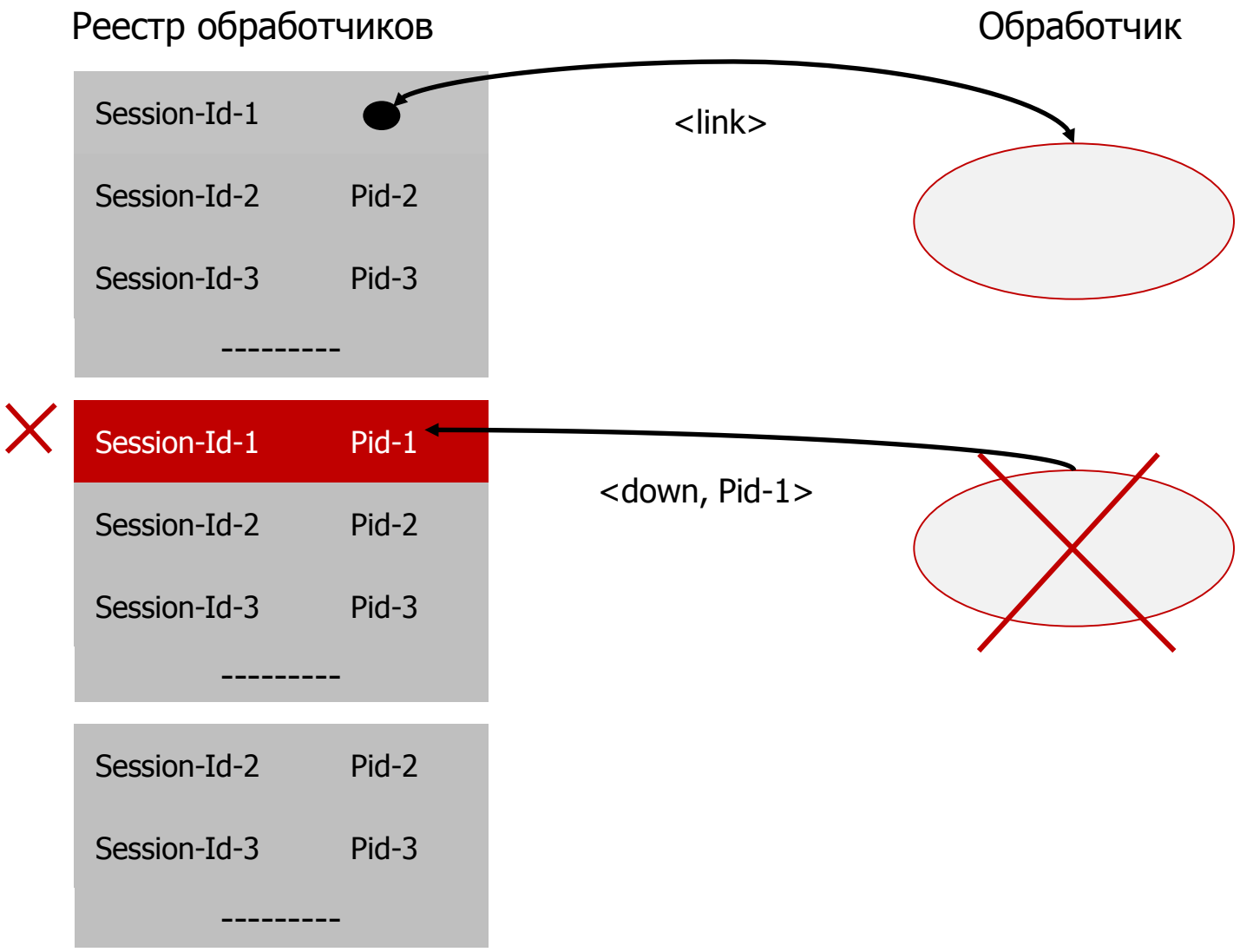
Pair-Id-1	Pid-4
Pair-Id-3	Pid-5
Pair-Id-6	Pid-6

Реестр обработчиков



rpc:multicall(handler_registry, get_pid_by_id, [SessionId])

Применение мониторинга



Спасибо за внимание!

Андрей Смирнов

asmirnov@reksoft.com

Санкт-Петербург, Россия
Тел.: +7 812 325 2100
www.reksoft.com/ru