DOT NEXT

Межпроцессные разговоры: причины и способы

Игорь Лабутин, 20.05.2017
ilabutin@gmail.com

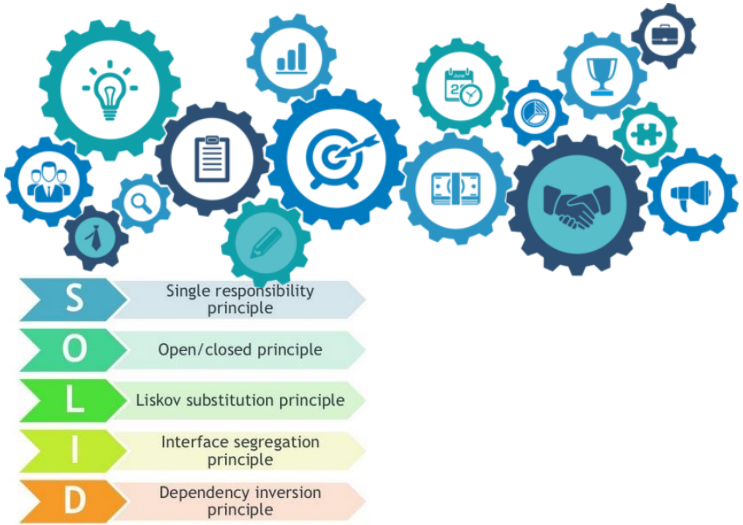
О себе

- 16 лет в разработке ПО
  - C/C++, Linux, QNX, Embedded
  - Последние 9 лет – .NET (C#)
- Интересы
  - Сети, протоколы обмена данными
  - Проблемы производительности
  - Сборка проектов и удобство разработчиков

О докладе

О докладе

О докладе

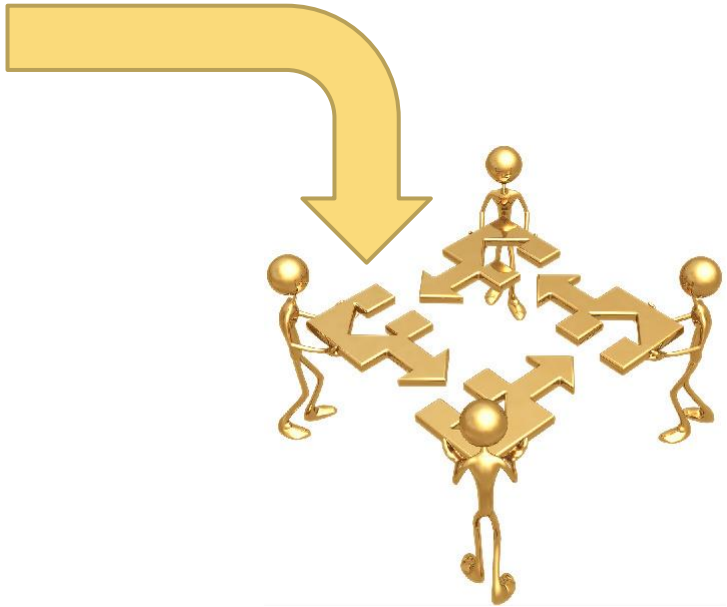

S
Single responsibility principle
O
Open/closed principle
L
Liskov substitution principle
I
Interface segregation principle
D
Dependency inversion principle

О докладе

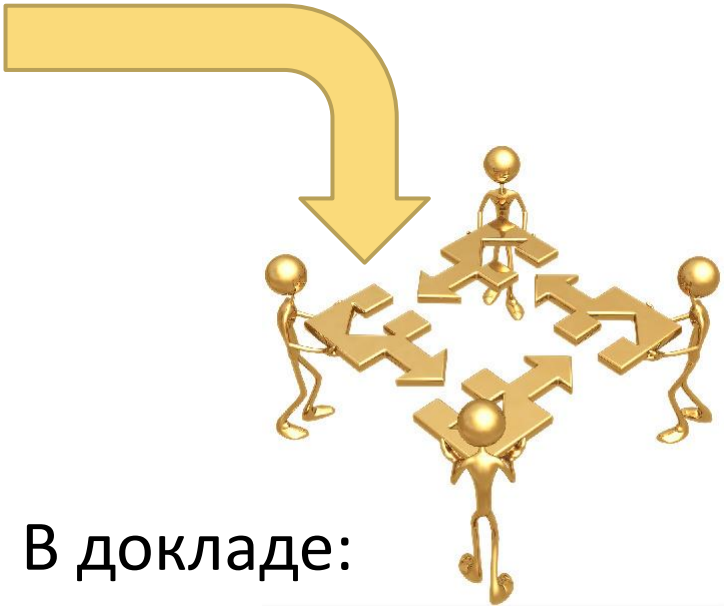

S
Single responsibility principle
O
Open/closed principle
L
Liskov substitution principle
I
Interface segregation principle
D
Dependency inversion principle

О докладе


S
Single responsibility principle
O
Open/closed principle
L
Liskov substitution principle
I
Interface segregation principle
D
Dependency inversion principle



О докладе



В докладе:

- Причины и методы выбора
- Сравнение характеристик

Зачем?

Зачем?

- Изолировать

Зачем?

- Изолировать
- Legacy

Зачем?

- Изолировать
- Legacy
- Масштабируемость

Зачем?

- Изолировать
- Legacy
- Масштабируемость
- Безопасность

Как?

- Командная строка
- Файлы
- IPC
  - Локально
  - По сети

Не все методы одинаковы

- Удобство
- Производительность
- Переносимость
- Гарантии доставки
- Одно- или двунаправленность

Сериализация

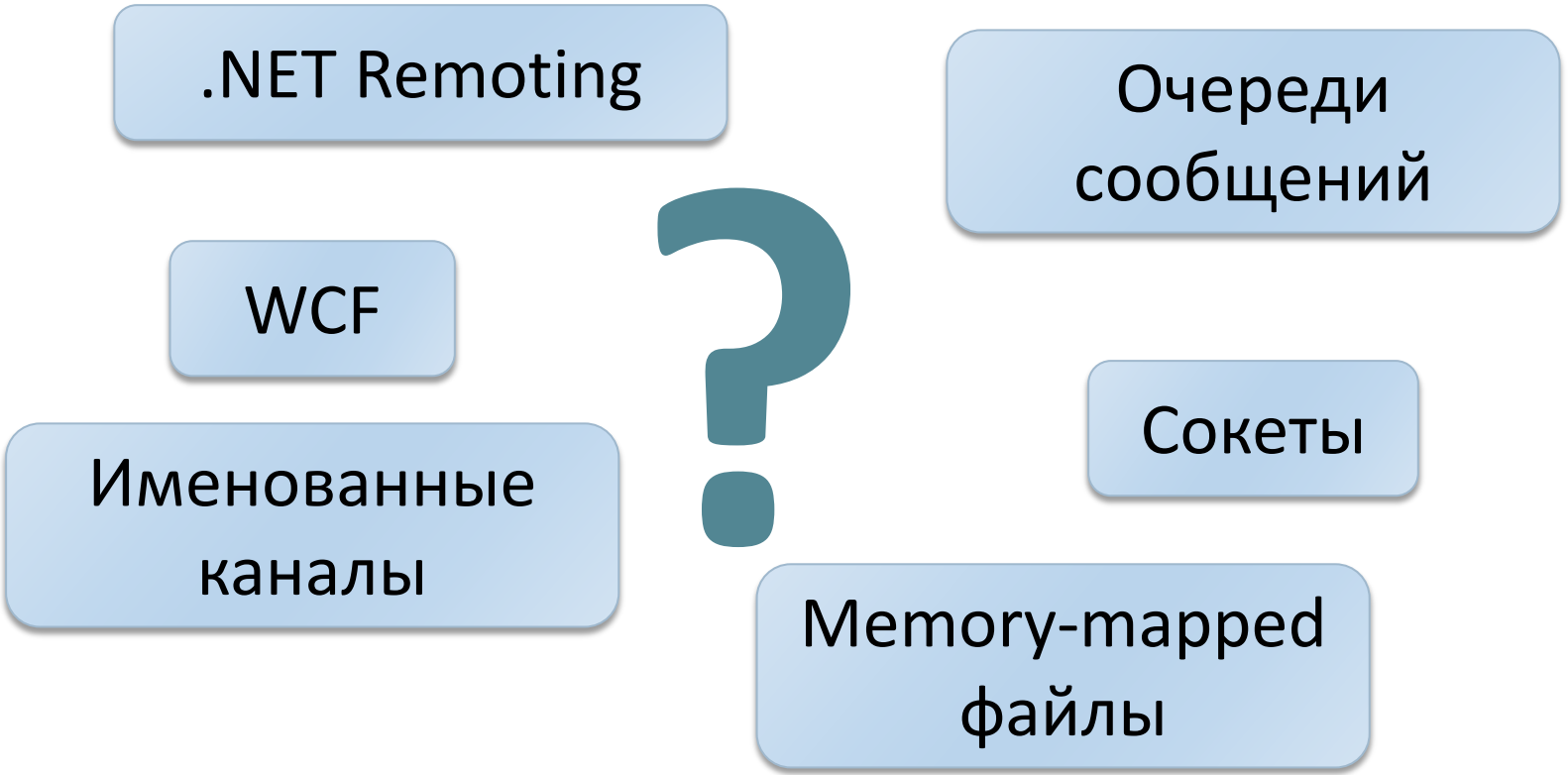
- JSON, бинарная, SOAP, etc
- Влияет на скорость и память

Сериализация

- JSON, бинарная, SOAP, etc
- Влияет на скорость и память

- Выбор: out of scope
- В примерах использовался DataContractSerializer + XmlBinaryReader/Writer

Что выбрать?


.NET Remoting
Очереди сообщений
WCF
?
Сокеты
Именованные каналы
Memory-mapped файлы

Измерение производительности

Время отклика
Количество запросов в секунду

Измерение производительности

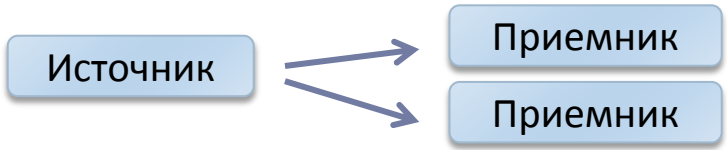
Время отклика
Количество запросов в секунду

Задержка
Количество сообщений в секунду
Пропускная способность (Мб/с)

Измерение производительности

Время отклика
Количество запросов в секунду

Задержка
Количество сообщений в секунду
Пропускная способность (Мб/с)

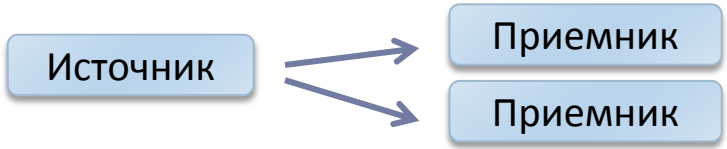


Отправляемых сообщений в секунду
Обрабатываемых сообщений в секунду

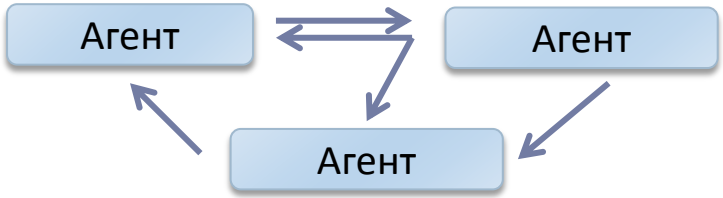
Измерение производительности

Время отклика
Количество запросов в секунду

Задержка
Количество сообщений в секунду
Пропускная способность (Мб/с)



Отправляемых сообщений в секунду
Обрабатываемых сообщений в секунду

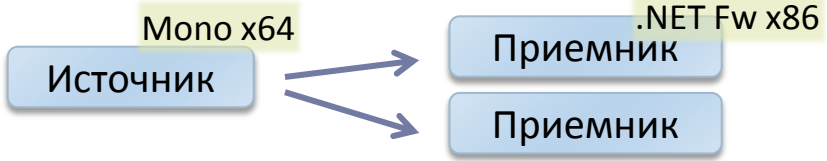


???

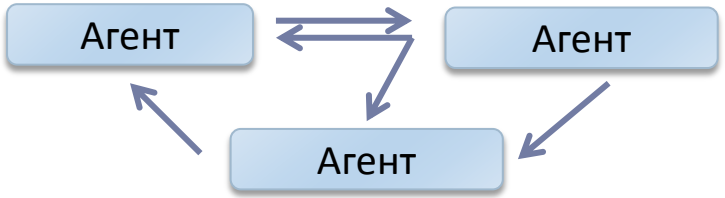
Измерение производительности

Время отклика
Количество запросов в секунду

Задержка
Количество сообщений в секунду
Пропускная способность (Мб/с)



Отправляемых сообщений в секунду
Обрабатываемых сообщений в секунду

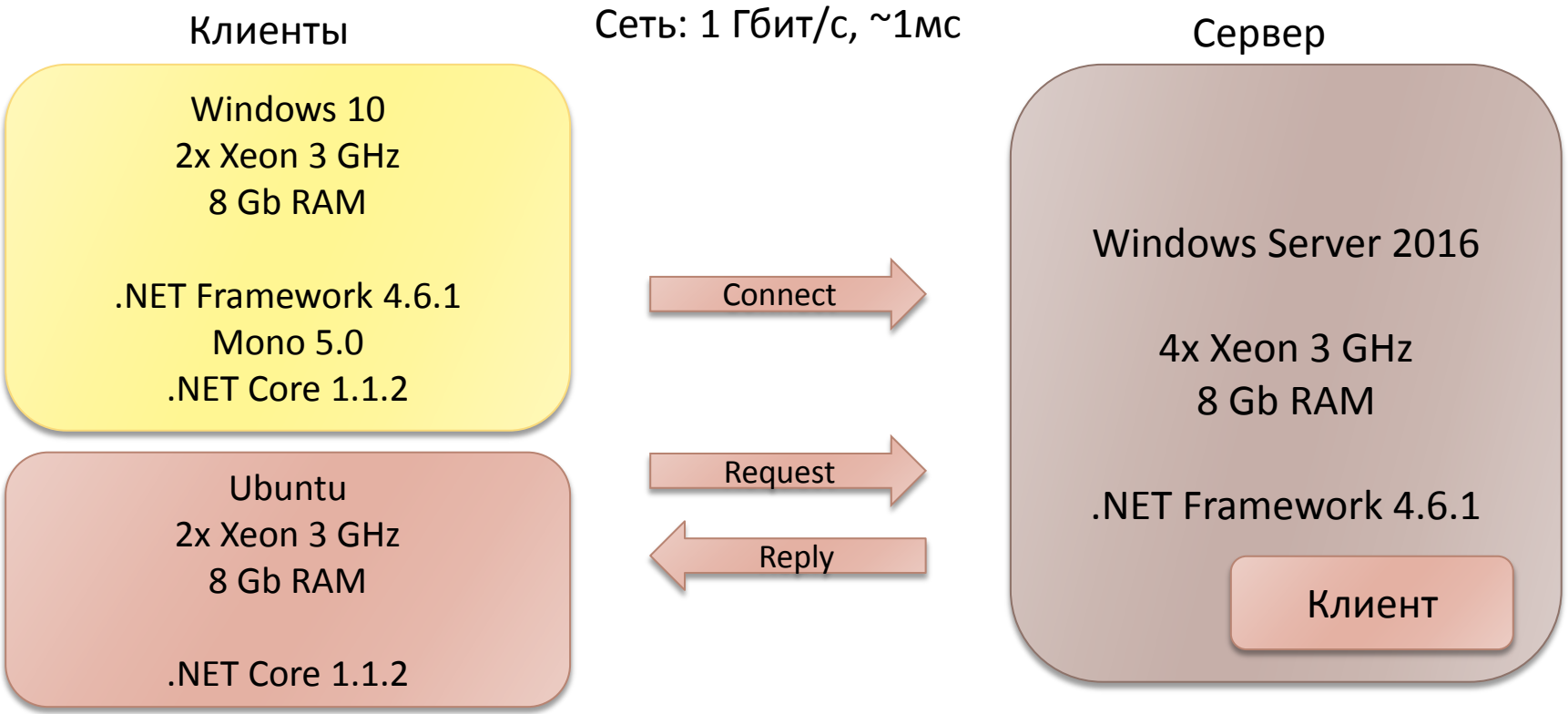


???

Измерение производительности

- Используем request/reply
  - Время отклика (время на 1-байтный запрос/ответ)
  - Кол-во запросов в секунду для 1 Кб, 10 Кб и 100 Кб данных
- Память – сильно зависит от сериализатора

Окружение


Клиенты
Сеть: 1 Гбит/с, ~1мс
Сервер
Windows 10
2x Xeon 3 GHz
8 Gb RAM
.NET Framework 4.6.1
Mono 5.0
.NET Core 1.1.2
Ubuntu
2x Xeon 3 GHz
8 Gb RAM
.NET Core 1.1.2
Connect
Request
Reply
Windows Server 2016
4x Xeon 3 GHz
8 Gb RAM
.NET Framework 4.6.1
Клиент

Окружение

Клиенты

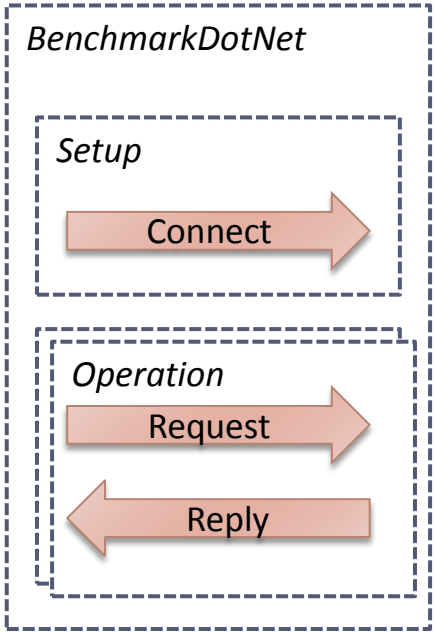
Windows 10
2x Xeon 3 GHz
8 Gb RAM

.NET Framework 4.6.1
Mono 5.0
.NET Core 1.1.2

Ubuntu
2x Xeon 3 GHz
8 Gb RAM

.NET Core 1.1.2

Сеть: 1 Гбит/с, ~1мс



Сервер

Windows Server 2016

4x Xeon 3 GHz
8 Gb RAM

.NET Framework 4.6.1

Клиент

Низкоуровневые методы

- Сокеты TCP/UDP
- Именованные каналы
- Memory-mapped файлы

Метод	Все платформы	Сетевой	⇄
TCP	+	+	+
UDP	+	+	-
Named pipes	+	+/-	+
Memory-mapped file	+	-	-

Низкоуровневые методы: время отклика

Локальный клиент, мкс

	.NET Fx	Mono	.NET Core
TCP	64	90	62
UDP	51	82	72
Named pipes	27	41	25
MMF	31	-	26

Низкоуровневые методы: время отклика

Локальный клиент, мкс

	.NET Fx	Mono	.NET Core
TCP	64	90	62
UDP	51	82	72
Named pipes	27	41	25
MMF	31	-	26

Сетевой клиент, мкс

	.NET Fx	Mono	.NET Core	Linux
TCP	114000	125000	123000	
UDP	1276	1049	1425	
Named pipes	6532		6488	-

Время отклика: причина

```
var client = new System.Net.Sockets.TcpClient();
client.Connect(new IPEndPoint(...));
```

```
var networkStream = client.GetStream();
networkStream.Send(data);
var reply = networkStream.Receive<ReplyData>();
```

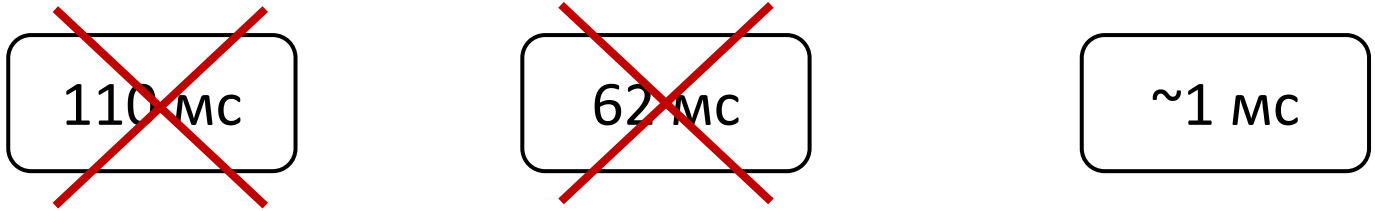
110 мс

Время отклика: причина

```
var client = new System.Net.Sockets.TcpClient();
client.Connect(new IPEndPoint(...));

client.NoDelay = true;

var networkStream = client.GetStream();
networkStream.Send(data);
var reply = networkStream.Receive<ReplyData>();
```

Время отклика: причина

```
var client = new System.Net.Sockets.TcpClient();
client.Connect(new IPEndPoint(...));

client.NoDelay = true;

var networkStream = client.GetStream();
networkStream.Send(data);
var reply = networkStream.Receive<ReplyData>();
```




```
var server = listeningSocket.AcceptTcpClient();
server.NoDelay = true;
```

Низкоуровневые методы: время отклика

Локальный клиент, мкс

	.NET Fx	Mono	.NET Core
TCP	64	90	62
UDP	51	82	72
Named pipes	27	41	25
MMF	31	-	26

Сетевой клиент, мкс

	.NET Fx	Mono	.NET Core	Linux
TCP	1540	1435	1691	1753
UDP	1276	1049	1425	1647
Named pipes	6532		6488	-

Низкоуровневые методы: время отклика

Локальный клиент, мкс

	.NET Fx	Mono	.NET Core
TCP	64	90	62
UDP	51	82	72
Named pipes	27	41	25
MMF	31	-	26

Сетевой клиент, мкс

	.NET Fx	Mono	.NET Core	Linux
TCP	1540	1435	1691	1753
UDP	1276	1049	1425	1647
Named pipes	6532		6488	-

Низкоуровневые методы: время отклика

Локальный клиент, мкс

	.NET Fx	Mono	.NET Core
TCP	64	90	62
UDP	51	82	72
Named pipes	27	41	25
MMF	31	-	26

Сетевой клиент, мкс

	.NET Fx	Mono	.NET Core	Linux
TCP	1540	1435	1691	1753
UDP	1276	1049	1425	1647
Named pipes	6532		6488	-

Низкоуровневые методы: запросы

Локальный клиент, запросов в секунду

	.NET Fx	Mono	.NET Core
TCP	14.8K	10.7K	14.9K
UDP	16.6K	11.6K	13.5K
Named pipes	31.3K	20.2K	25.6K
MMF	18K	-	19.6K

Сетевой клиент, запросов в секунду

	.NET Fx	Mono	.NET Core	Linux
TCP	664	702	591	554
UDP	796	953	701	580
Named pipes	165		154	-

Низкоуровневые методы: запросы

Локальный клиент, запросов в секунду

	.NET Fx	Mono	.NET Core
TCP	14.8K	10.7K	14.9K
UDP	16.6K	11.6K	13.5K
Named pipes	31.3K	20.2K	25.6K
MMF	18K	-	19.6K

Сетевой клиент, запросов в секунду

	.NET Fx	Mono	.NET Core	Linux
TCP	664	702	591	554
UDP	796	953	701	580
Named pipes	165		154	-

Низкоуровневые методы: особенности

- UDP/MMF
  - Двунаправленность вручную
  - Потеря данных
  - Ограниченный размер сообщения
- Возможность выбора сериализатора

Сообщения

- Шаблоны общения из коробки:
  - Pub/Sub
  - Pipeline
  - Request/reply
  - …
- Возможность выбора сериализатора
- Кроссплатформенность

Сообщения: время отклика

Проще API => медленнее чем TCP

Локальный клиент, мкс

	.NET Fx	Mono	.NET Core
TCP	64	90	62
ZeroMQ	142	175	130

Сетевой клиент, мкс

	.NET Fx	Mono	.NET Core	Linux
TCP	1540	1435	1691	1753
ZeroMQ	1947	1793	1831	1881

Сообщения: время отклика

Проще API => медленнее чем TCP

Локальный клиент, мкс

	.NET Fx	Mono	.NET Core
TCP	64	90	62
ZeroMQ	142	175	130

Сетевой клиент, мкс

	.NET Fx	Mono	.NET Core	Linux
TCP	1540	1435	1691	1753
ZeroMQ	1947	1793	1831	1881

Сообщения: запросы

Локальный клиент, запросов в секунду

	.NET Fx	Mono	.NET Core
TCP	14.8K	10.7K	14.9K
ZeroMQ	6.5K	5.5K	7K

Сетевой клиент, запросов в секунду

	.NET Fx	Mono	.NET Core	Linux
TCP	664	702	591	554
ZeroMQ	513	557	546	532

Сообщения: запросы

Локальный клиент, запросов в секунду

	.NET Fx	Mono	.NET Core
TCP	14.8K	10.7K	14.9K
ZeroMQ	6.5K	5.5K	7K

Сетевой клиент, запросов в секунду

	.NET Fx	Mono	.NET Core	Linux
TCP	664	702	591	554
ZeroMQ	513	557	546	532

Сообщения с гарантией

- Очереди
  - Гарантия доставки
  - Требует отдельного брокера
- Примеры

- Возможность выбора сериализатора

Сообщения с гарантией: время отклика

Локальный клиент, мкс

	.NET Fx	Mono	.NET Core
TCP	64	90	62
ZeroMQ	142	175	130
RabbitMQ	386	431	395

Сетевой клиент, мкс

	.NET Fx	Mono	.NET Core	Linux
TCP	1540	1435	1691	1753
ZeroMQ	1947	1793	1831	1881
RabbitMQ	2634	2777	2472	2762

Сообщения с гарантией: время отклика

Локальный клиент, мкс

	.NET Fx	Mono	.NET Core
TCP	64	90	62
ZeroMQ	142	175	130
RabbitMQ	386	431	395

Сетевой клиент, мкс

	.NET Fx	Mono	.NET Core	Linux
TCP	1540	1435	1691	1753
ZeroMQ	1947	1793	1831	1881
RabbitMQ	2634	2777	2472	2762

Сообщения с гарантией: запросы

Локальный клиент, запросов в секунду

	.NET Fx	Mono	.NET Core
TCP	14.8K	10.7K	14.9K
ZeroMQ	6.5K	5.5K	7K
RabbitMQ	2.6K	2.3K	2.5K

Сетевой клиент, запросов в секунду

	.NET Fx	Mono	.NET Core	Linux
TCP	664	702	591	554
ZeroMQ	513	557	546	532
RabbitMQ	380	360	404	362

RPC: Вызов метода

- [Почти] прозрачный вызов метода

.NET Remoting

- Сериализатор встроен, но иногда можно менять
- Не полностью кроссплатфоменен

RPC: время отклика

Локальный клиент, мкс

	.NET Fx	Mono	.NET Core
TCP	64	90	62
.NET Remoting IPC/TCP	112/234	-	-
WCF HTTP/TCP	240/159	560/-	363/162
WebAPI	385	480	382

Сетевой клиент, мкс

	.NET Fx	Mono	.NET Core	Linux
TCP	1540	1435	1691	1753
.NET Remoting TCP	1750	-	-	-
WCF HTTP/TCP	3334/1938	2954/-	2787/1907	4967/1669
WebAPI	3370	2789	2519	3365

RPC: время отклика

Локальный клиент, мкс

	.NET Fx	Mono	.NET Core
TCP	64	90	62
.NET Remoting IPC/TCP	112/234	-	-
WCF HTTP/TCP	240/159	560/-	363/162
WebAPI	385	480	382

Сетевой клиент, мкс

	.NET Fx	Mono	.NET Core	Linux
TCP	1540	1435	1691	1753
.NET Remoting TCP	1750	-	-	-
WCF HTTP/TCP	3334/1938	2954/-	2787/1907	4967/1669
WebAPI	3370	2789	2519	3365

RPC: время отклика

Локальный клиент, мкс

	.NET Fx	Mono	.NET Core
TCP	64	90	62
.NET Remoting IPC/TCP	112/234	-	-
WCF HTTP/TCP	240/159	560/-	363/162
WebAPI	385	480	382

Сетевой клиент, мкс

	.NET Fx	Mono	.NET Core	Linux
TCP	1540	1435	1691	1753
.NET Remoting TCP	1750	-	-	-
WCF HTTP/TCP	3334/1938	2954/-	2787/1907	4967/1669
WebAPI	3370	2789	2519	3365

RPC: запросы

Локальный клиент, запросов в секунду

	.NET Fx	Mono	.NET Core
TCP	14.8K	10.7K	14.9K
.NET Remoting IPC/TCP	8.4K/4.3K	-	-
WCF HTTP/TCP	4K/6.3K	1.8K/-	2.7K/6.2K
WebAPI	2.6K	2.1K	2.6K

Сетевой клиент, запросов в секунду

	.NET Fx	Mono	.NET Core	Linux
TCP	664	702	591	554
.NET Remoting TCP	572	-	-	-
WCF HTTP/TCP	299/516	338/-	359/524	201/600
WebAPI	279	339	397	205

RPC: запросы

Локальный клиент, запросов в секунду

	.NET Fx	Mono	.NET Core
TCP	14.8K	10.7K	14.9K
.NET Remoting IPC/TCP	8.4K/4.3K	-	-
WCF HTTP/TCP	4K/6.3K	1.8K/-	2.7K/6.2K
WebAPI	2.6K	2.1K	2.6K

Сетевой клиент, запросов в секунду

	.NET Fx	Mono	.NET Core	Linux
TCP	664	702	591	554
.NET Remoting TCP	572	-	-	-
WCF HTTP/TCP	299/516	338/-	359/524	201/600
WebAPI	279	339	397	205

RPC: Особенности WCF

- Позволяет подменять сериализатор
- Можно выбрать любой транспорт
  - TCP/UDP/NamedPipes/MSMQ
  - Даже RabbitMQ
- Сервер только на .NET Fx

Повторим

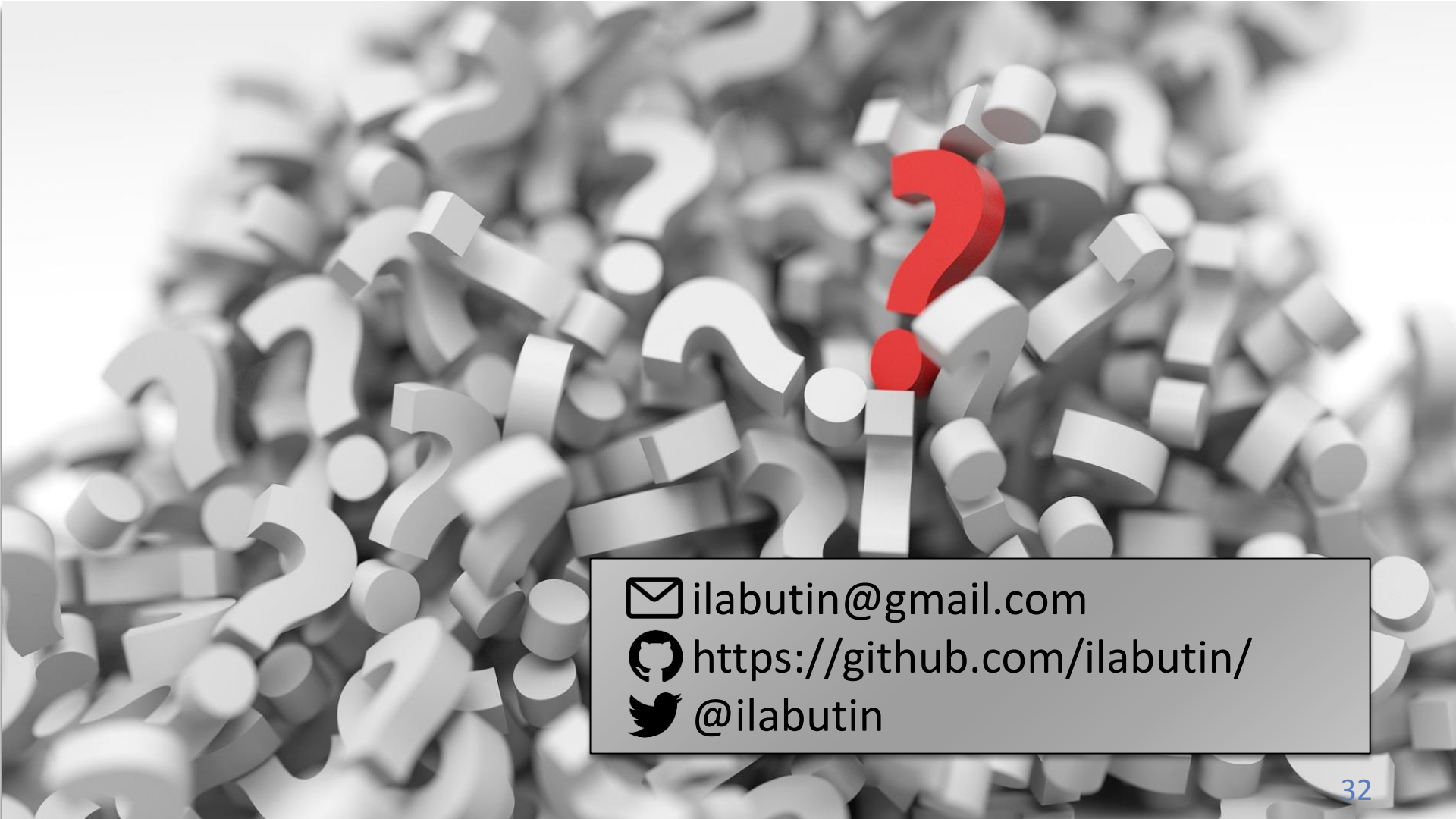
- Сокеты/именованные каналы/MMF
  - Быстро, но не очень удобный API
- ZeroMQ/NetMQ
  - Удобнее, но чуть медленнее
- Полноценные очереди сообщений
  - Гарантии доставки
  - Внешний брокер
- RPC

Заключение

- Выбор метода – баланс скорости и удобства
- Локально или по сети – огромная разница

Ссылки

- Обзор IPC для .NET FW от Ricardo Peres: http://bit.ly/2pblVVm
- Рекомендации Microsoft по выбору метода сериализации: http://bit.ly/2pbg2rB
- Советы по переходу на .NET Core: http://bit.ly/2pbn933
- Примеры и бенчмарки из доклада: https://github.com/ilabutin/dotnext2017spb



ilabutin@gmail.com
https://github.com/ilabutin/
@ilabutin