Клиент-серверная и пиринговая (P2P) модели: обзор и классификация

Дунайцев Р.А.

Кафедра сетей связи и передачи данных СПбГУТ
им. проф. М.А. Бонч-Бруевича

roman.dunaytsev@spbgut.ru

Лекция № 1

Содержание

1. Введение
2. Клиент-серверная модель
   - Классификация серверов
   - Классификация клиентов
   - Логические уровни
   - Физические уровни
3. Пиринговая модель
   - Децентрализованная P2P-система
   - Гибридная P2P-система
4. Выводы

Содержание

1. Введение
2. Клиент-серверная модель
   - Классификация серверов
   - Классификация клиентов
   - Логические уровни
   - Физические уровни
3. Пиринговая модель
   - Децентрализованная P2P-система
   - Гибридная P2P-система
4. Выводы

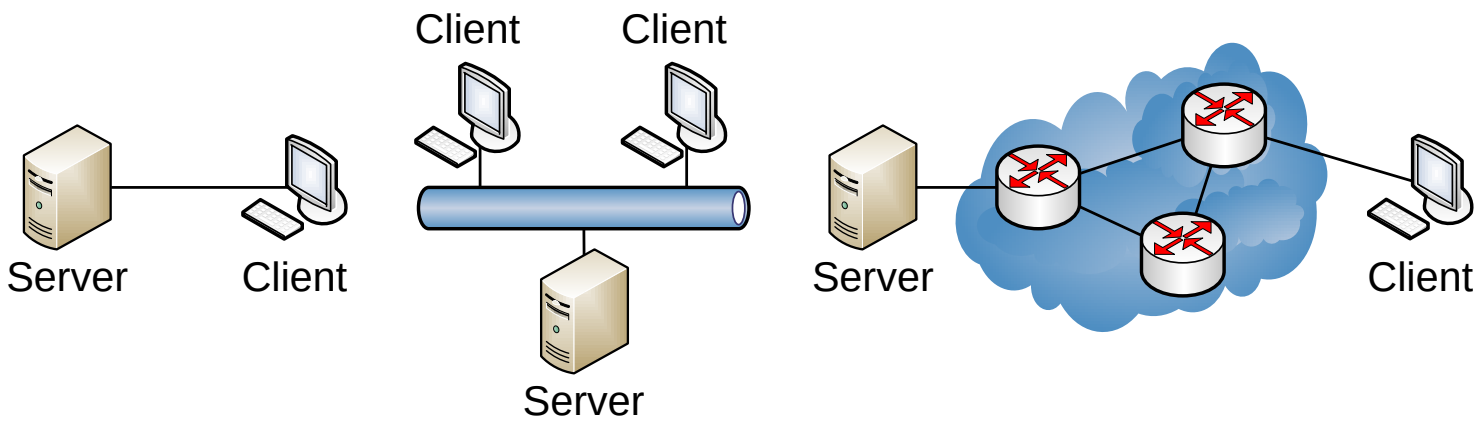
Введение

- **Сеть связи** – совокупность оконечных устройств связи, объединенных каналами передачи данных и коммутирующими устройствами, обеспечивающими обмен сообщениями между всеми оконечными устройствами
  - Размер сети может быть любым, от 2 до 1000000'ов узлов
- **Коммутирующие устройства**: хабы, мосты, коммутаторы, маршрутизаторы, шлюзы, ...
- **Оконечные устройства**: ПК, ноутбуки, планшеты, смартфоны, DNS-серверы, Web-серверы, почтовые серверы, ...

Введение

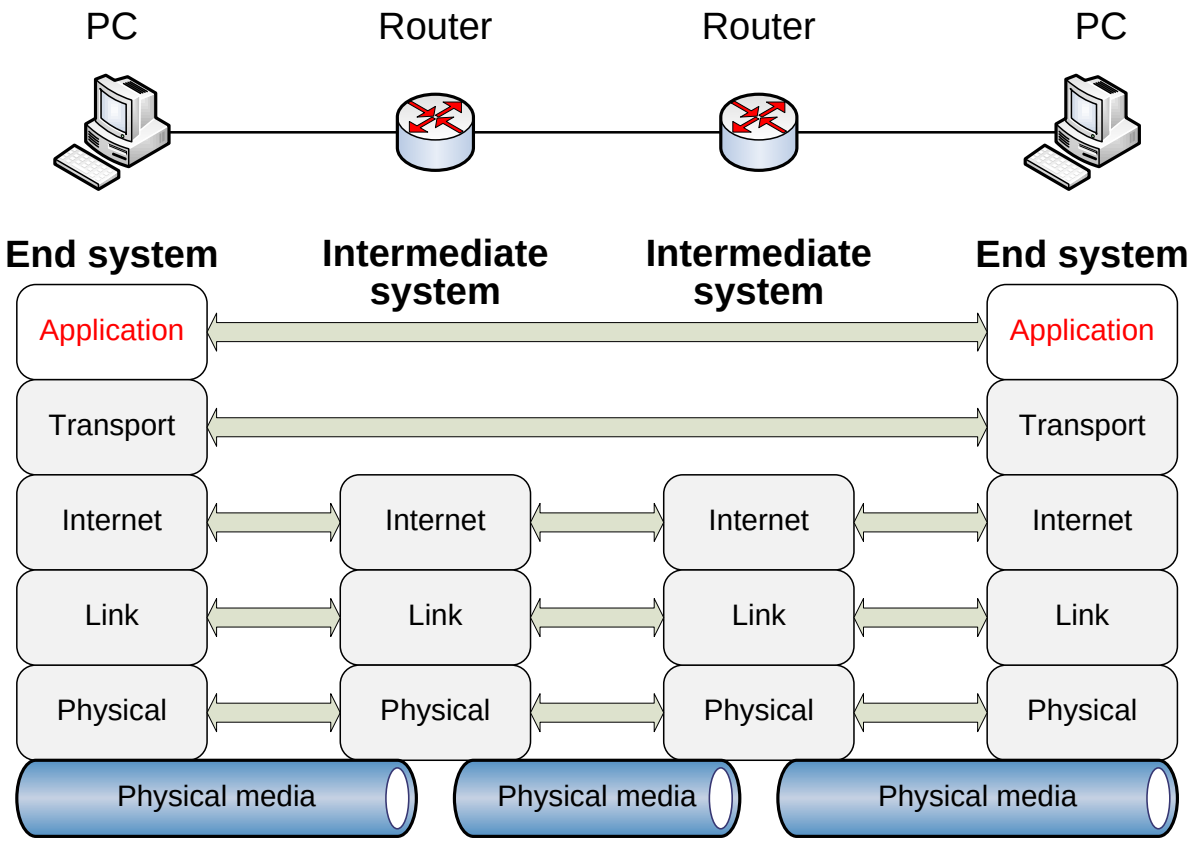
- **2 модели взаимодействия оконечных устройств**:
  - Клиент-серверная (client/server)
  - Одноранговая или пиринговая (peer-to-peer, P2P)

- Эти модели описывают взаимодействие оконечных устройств, безотносительно способа их соединения!



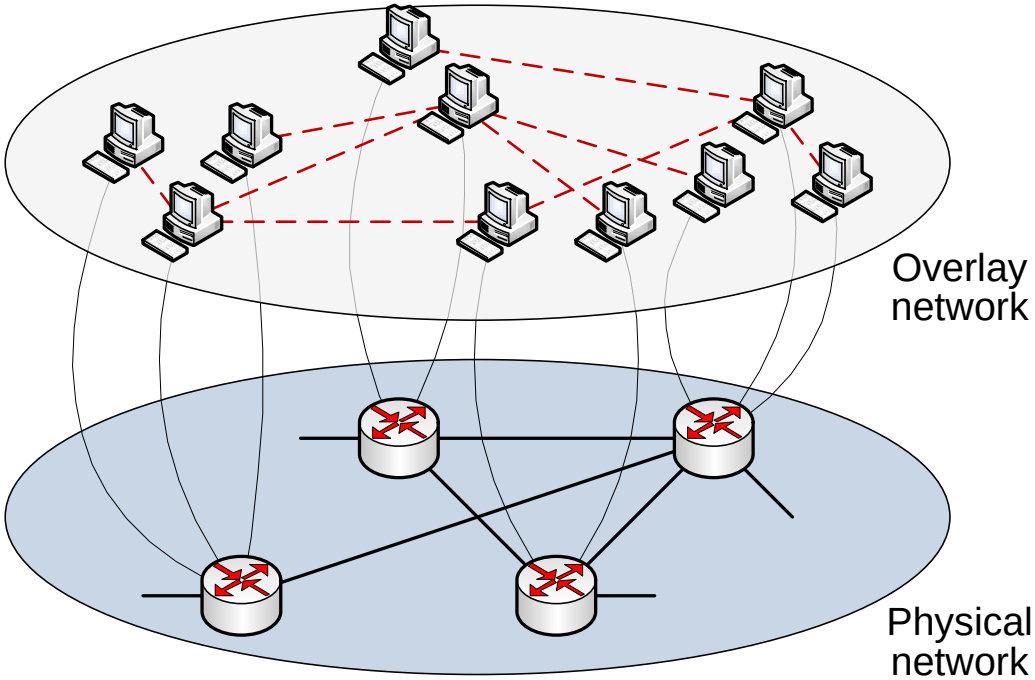
Введение

- Клиент-серверные и пиринговые файлообменные протоколы реализуются на прикладном уровне стека TCP/IP
  - Протоколы прикладного уровня являются **сквозными** (end-to-end)



Введение

- Клиент-серверные и пиринговые системы могут быть представлены в виде **виртуальных наложенных сетей**, состоящих из оконечных узлов и логических каналов между ними, реализуемых поверх физической инфраструктуры
  - Топология наложенной сети может не совпадать с топологией физической



Содержание

1. Введение
2. Клиент-серверная модель
   - Классификация серверов
   - Классификация клиентов
   - Логические уровни
   - Физические уровни
3. Пиринговая модель
   - Децентрализованная P2P-система
   - Гибридная P2P-система
4. Выводы

Клиент-серверная модель

- В клиент-серверной модели все оконечные узлы делятся на клиенты и серверы, каждый из которых выполняет свои функции
- **Клиенты** – играют **активную** роль и инициируют обмен информацией путем обращения к серверу
  - Клиенты должны иметь информацию о доступных серверах и предоставляемых ими услугах
  - Клиенты контактируют лишь с серверами, друг друга они не 'видят'
- **Серверы** – играют **пассивную** роль, ожидая запросов от клиентов на выполнение каких-либо действий
  - Назначением сервера является предоставление услуги клиентам
  - Обычно одним сервером обслуживается множество клиентов

Клиент-серверная модель

- Как правило, клиент и сервер – это 2 отдельных устройства/программы, приспособленные для выполнения определенных задач
  - Например, Web-клиент (браузер) лучше всего работает на большом экране, в то время Web-сервер монитора вовсе не требует
- Однако, одно и то же устройство может функционировать как клиент для одного приложения/сервиса, а для другого – как сервер
- Более того, в некоторых приложениях/сервисах одно и то же устройство может функционировать и как клиент, и как сервер
- 'Клиент' и 'сервер' – многозначные термины!

Клиент-серверная модель

- **Аппаратные роли**:
  - 'Клиент' – устройство, предназначенный для эксплуатации одним пользователем
  - 'Сервер' – устройство, выделенное и/или специализированное для работы на нем серверного программного обеспечения
  - Как правило, отличаются габаритами, надежностью, производительностью, оснащением, стоимостью

Клиент-серверная модель

- **Программные роли**:
  - 'Сервер' – программа, выполняющая обслуживающие функции по запросу 'клиента', предоставляя ему доступ к определенным ресурсам или услугам
  - Обычно клиентские программы устанавливаются на клиентских устройствах и наоборот
  - Но возможна установка клиентских и серверных программ на одном устройстве

- **Web-клиенты**: Google Chrome, Mozilla Firefox, Internet Explorer
  - См. 'Web Statistics': www.w3schools.com/browsers/default.asp
- **Web-серверы**: Apache, Microsoft IIS, GWS
  - См. 'Web Server Survey': news.netcraft.com/archives/category/web-server-survey/

Клиент-серверная модель

- **Транзакционные роли**:
  - В любом сеансе связи 'клиентом' является тот, кто его инициирует, т.е. посылает запрос на предоставление услуги
  - Соответственно, 'сервером' в сеансе связи является тот, кто принимает запрос и предоставляет услугу
  - В основном запросы посылаются клиентскими программами, установленными на клиентских устройствах
  - Но так бывает не всегда: когда 2 SMTP-сервера пересылают электронную почту, один из них инициирует сеанс связи, выступая в роли 'клиента', хотя оба являются серверными программами, установленными на серверных устройствах

Клиент-серверная модель

- Клиент-серверная модель vs. модель ведущий/ведомый
- **Модель ведущий/ведомый** – одно устройство (master) управляет одним или несколькими другими устройствами (slaves)
  - Например, при установке 2 жестких дисков на ПК один из них назначается ведущим, а другой – ведомым

	Client/server	**Master/slave**
Происхождение	Телекоммуникации	Инженерия
Предмет	Услуга (запрашивается / предоставляется)	Контроль (управляющий / управляемый)
Активная роль	Клиент (client)	Ведущий (master)
Пассивная роль	Сервер (server)	Ведомый (slave)
Соотношение	$N_{clients} > N_{servers}$	$N_{masters} < N_{slaves}$

Серверы

- **2 типа серверов** :
  - Итерационный (iterative)
  - Параллельный (concurrent)

- **Итерационный сервер**
  1. Ожидание поступления запроса от клиента
  2. Обслужить запрос
  3. Вернуться на шаг 1

- Т.е. итерационный сервер обслуживает клиентов **по одному**

Серверы

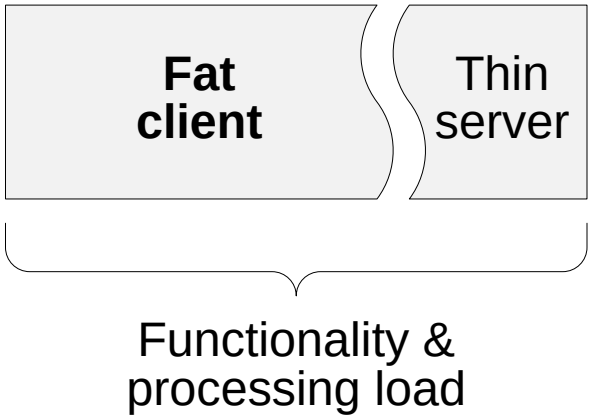
- **Параллельный сервер**
  1. Ожидание поступления запроса от клиента
  2. Использовать/создать процесс (process/task/thread) для обслуживания запроса
  3. Вернуться на шаг 1
- Т.е. параллельный сервер обслуживает множество клиентов **одновременно**

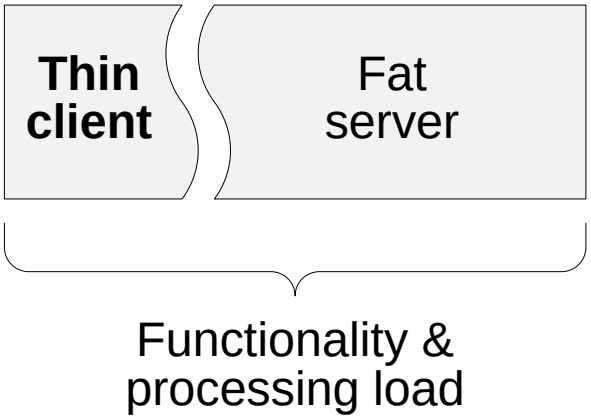
Серверы

- Итерационный или параллельный?
- Итерационный сервер проще и больше подходит для обслуживания запросов с небольшим временем выполнения
  - Если время выполнения запросов велико и/или меняется в широких пределах, время ожидания обслуживания другими клиентами может оказаться чрезмерно большим
  - Интернет-сервисы, такие как **echo** (RFC 862) и **daytime** (RFC 867), обычно реализуются на базе итерационных серверов
- Параллельный сервер сложнее, но имеет более высокую производительность
  - По сравнению с итерационным сервером, время ожидания обслуживания клиентами оказывается меньше
  - Интернет-сервисы, такие как **HTTP**, **telnet** и **FTP**, обычно реализуются на базе параллельных серверов

Клиенты

- Клиенты - устройства/программы, запрашивающие предоставление услуги у серверов
- Клиенты (и, соответственно, серверы) различаются по своему функционалу и объему выполняемой работы
- **2 типа клиентов**:
  - Толстые (fat/thick/full)
  - Тонкие (thin/slim/lean)





Клиенты

- **Толстые клиенты** – устройства/программы выполняющие все или большую часть задач по обработке информации
- **Толстые клиенты как устройства** – пользовательское устройство достаточно мощное, чтобы работать независимо от сервера
  - Примеры: ПК, ноутбук

- **Толстые клиенты как программы** – программа, предоставляющая пользователю расширенный функционал и выполняющая больший объем вычислений
  - Примеры: клиенты Lineage II (объем дистрибутива более 2 ГБ)

Клиенты

- **Тонкие клиенты** – устройства/программы, переносящие все или большую часть задач по обработке информации на сервер
- **Тонкие клиенты как устройства** – пользовательское устройство с полностью пассивным охлаждением и минимальной аппаратной конфигурацией
  - Примеры: тонкие клиенты в компьютерных классах

- **Тонкие клиенты как программы** – программа, реализующая лишь пользовательский интерфейс
  - Примеры: клиенты OnLive (объем дистрибутива менее 2 МБ)

Клиенты

- **Тонкие клиенты как устройства**: LT310 Jupiter ($\sim$ 180 евро)
  - Полностью пассивное охлаждение
  - Загрузка: с LAN, USB или внутреннего SSD (опционально)
  - Процессор: 1 ГГц
  - Оперативная память: 1 ГБ
  - Сетевой интерфейс: Fast Ethernet

Клиенты

- **Маленький – не значит тонкий**: HP Z2 Mini (∼ 1000 евро)
  - Активное охлаждение с пониженным уровнем шума
  - Графика: NVIDIA Quadro
  - Процессор: Intel Core i7 или Intel Xeon
  - Оперативная память: от 8 до 32 ГБ
  - Жесткий диск: от 256 ГБ до 1 ТБ

Клиенты

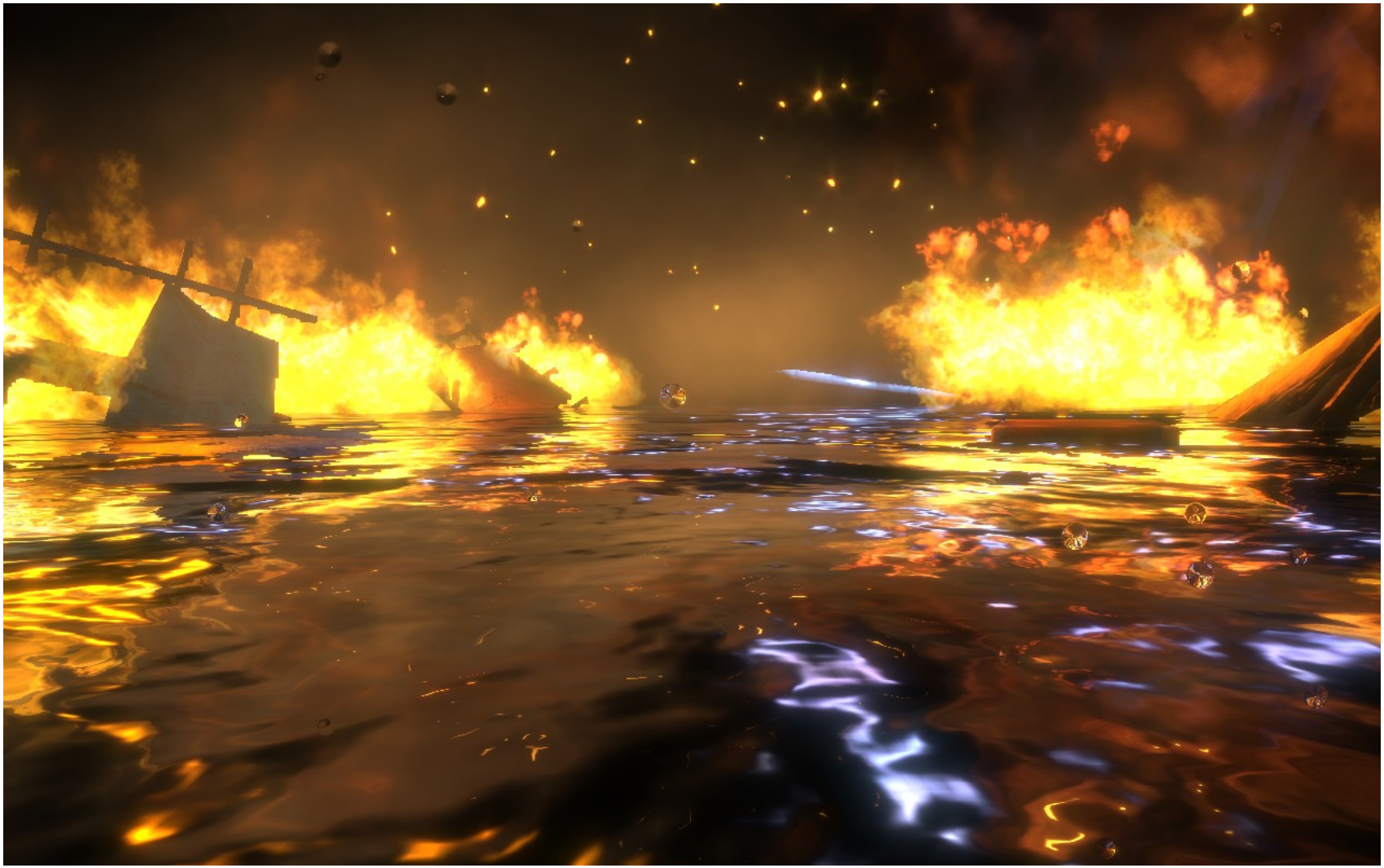
- Толстый или тонкий?
- Обе технологии имеют сильные и слабые стороны
- **Плюсы систем с тонкими клиентами**:
  - Безопаснее в плане вирусов, спама, кражи
  - Проще администрирование и обновление ПО
  - Меньше общая стоимость владения (Total Cost of Ownership, TCO)
  - Меньше отказов оборудования
- **Минусы систем с тонкими клиентами**:
  - Работоспособность всей системы зависит от работы сервера
  - В основном проприетарные системы
  - Требовательны к пропускной способности сети

Клиенты

- **Тонкие клиенты как программы** : www.onlive.com
- OnLive – система, использующая концепцию облачных вычислений
  - **OnLive App** – компьютерные игры
  - **OnLive Desktop** – офисные приложения
  - **OnLive Files** – хранение данных
- OnLive PlayPack Bundle
  - Более 100 игр
  - $9.99 в месяц
- 02.04.2015 компания Sony выкупила патенты компании OnLive и 30.04.2015 сервис завершил свою работу

Клиенты

- BioShock (2007) с максимальными настройками: **lenovo X201i**

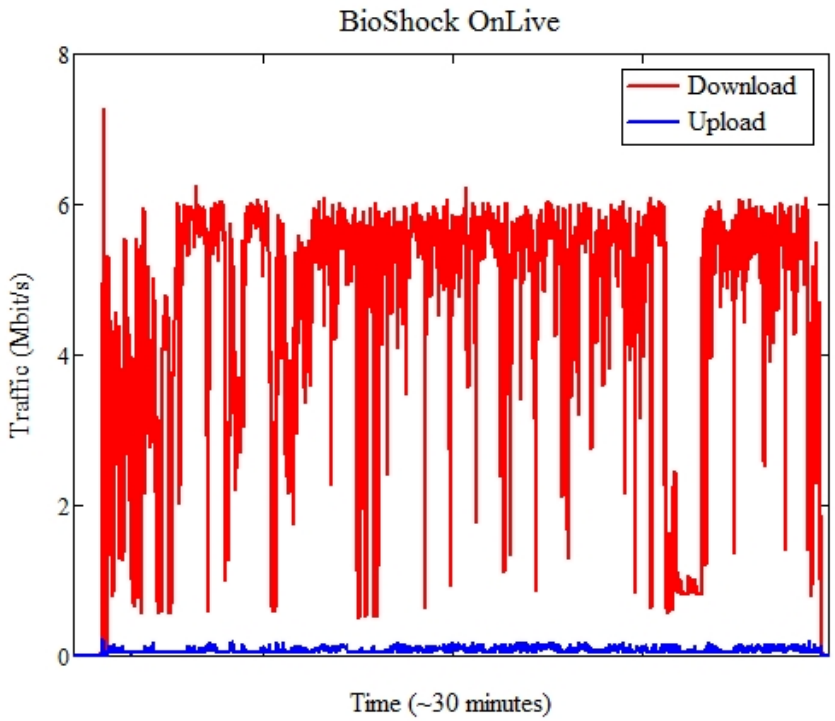


Клиенты

- BioShock (2007) с максимальными настройками: **OnLive**

Клиенты

- Интернет-доступ по ADSL: 24/1 Мбит/с
  - Максимальная скорость 'вниз' во время игры = 7,2 Мбит/с
  - Максимальная скорость 'вверх' во время игры = 0,2 Мбит/с



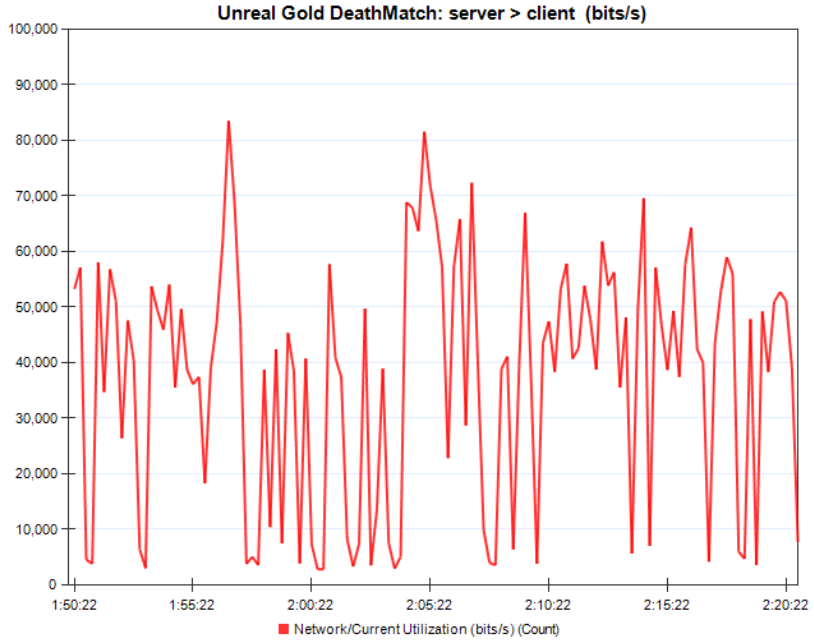
Клиенты

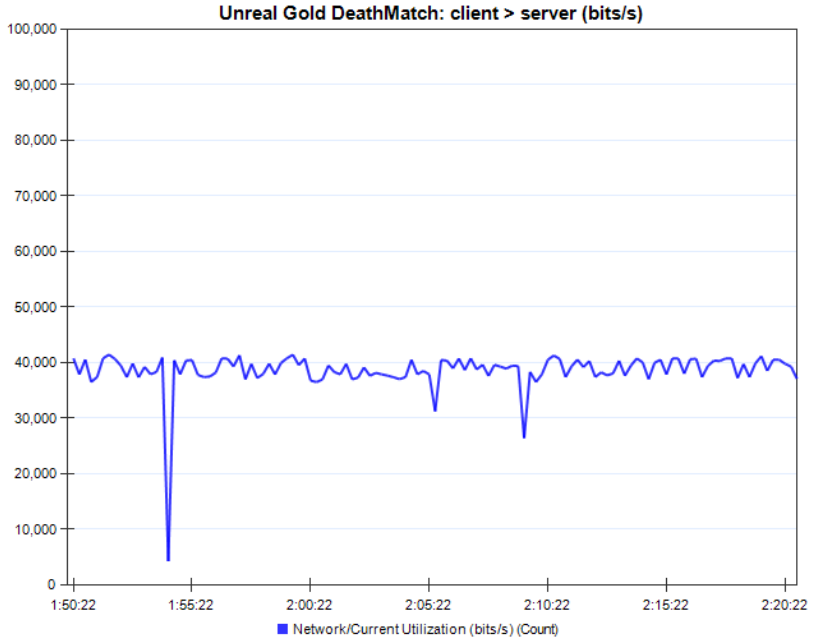
- BioShock: ПК vs. OnLive

	ПК	**OnLive**
Дистрибутив	~ 2 ГБ	~ 2 МБ
Минимальные системные требования	CPU: Pentium 4 RAM: 1 GB Video card: 128 MB HDD: 8 GB free space	Internet connection: 2 Mbit/s Host: Most PCs & netbooks OS: 7, Vista, XP, Mac 10.5.8 Screen resolution: 1024x576
Рекомендуемые системные требования	CPU: Intel Core 2 Duo RAM: 2 GB Video card: 512 MB HDD: 8 GB free space	Internet connection: 5 Mbit/s Host: Dual-core PCs OS: 7, Vista, XP, Mac 10.6 Screen resolution: 1280x720
Стоимость	$10-40 (Amazon.com)	$10 (PlayBack Bundle)

Клиенты

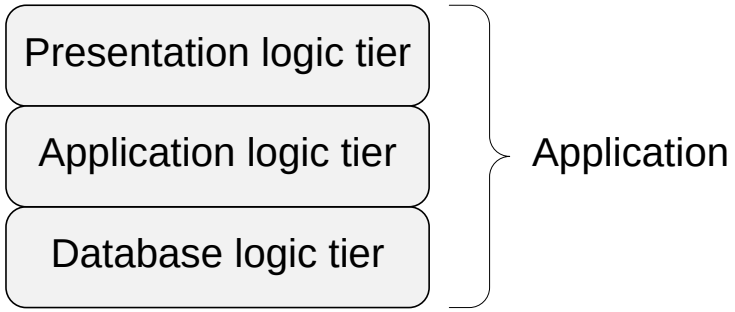
- **Толстые клиенты как программы**: Unreal Gold DeathMatch
  - Потоки данных примерно одинаковы





Логические уровни

- **3 логических уровня в сетевом ПО**:
  - Логика представления (presentation logic)
  - Логика приложения (application logic)
  - Логика базы данных (database logic)
- Данная модель является эталонной, на практике границы между уровнями могут быть размыты
- Логическое разделение на уровни может отличаться от физического

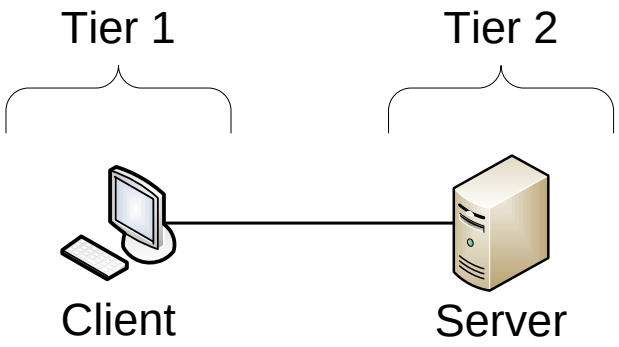


Логические уровни

- **Логика представления** – отвечает за отображение информации и взаимодействие с пользователем (т.е. пользовательский интерфейс)
  - Она также отвечает за преобразование информации поступающей к/от пользователя
- **Логика приложения** – отвечает за обработку команд, принятие решений, выполнение вычислений и координацию работы приложения
  - Она также отвечает за передачу и обработку данных между уровнем представления и уровнем базы данных
- **Логика базы данных** – отвечает за управление базой данных
  - Она также отвечает за хранение и извлечение данных согласно требованиям уровня логики приложения

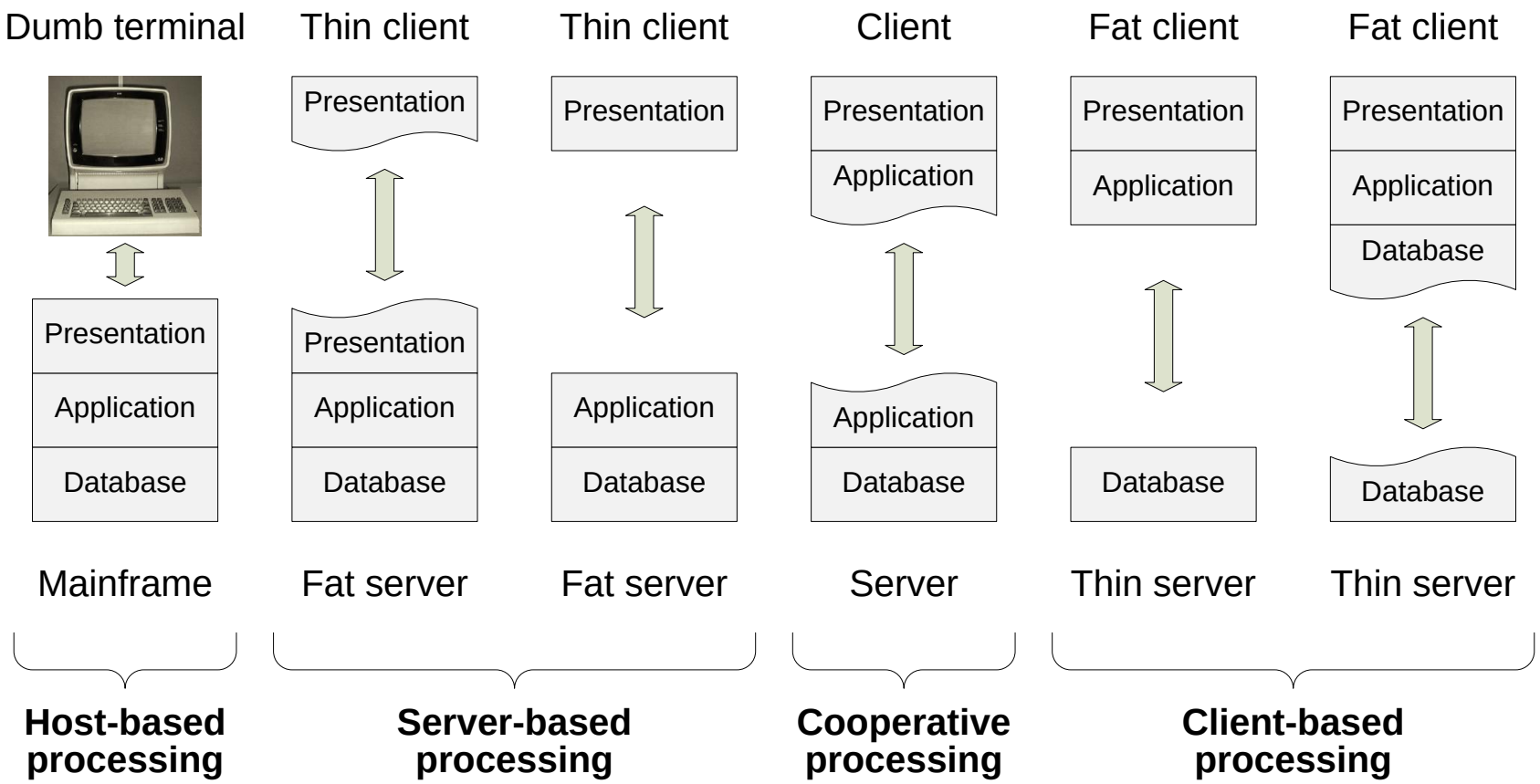
Физические уровни

- **1-уровневая архитектура** – централизованная клиент-серверная система
  - Пользователи обращаются к такой системе (т.н. мэйнфрейму) через неинтеллектуальный терминал (dumb terminal), а выводимая информация полностью контролируется мэйнфреймом
- **2-уровневая архитектура** – клиент-серверная система, в которой клиенты запрашивают предоставление услуги, а сервер предоставляет ее, используя лишь собственные ресурсы



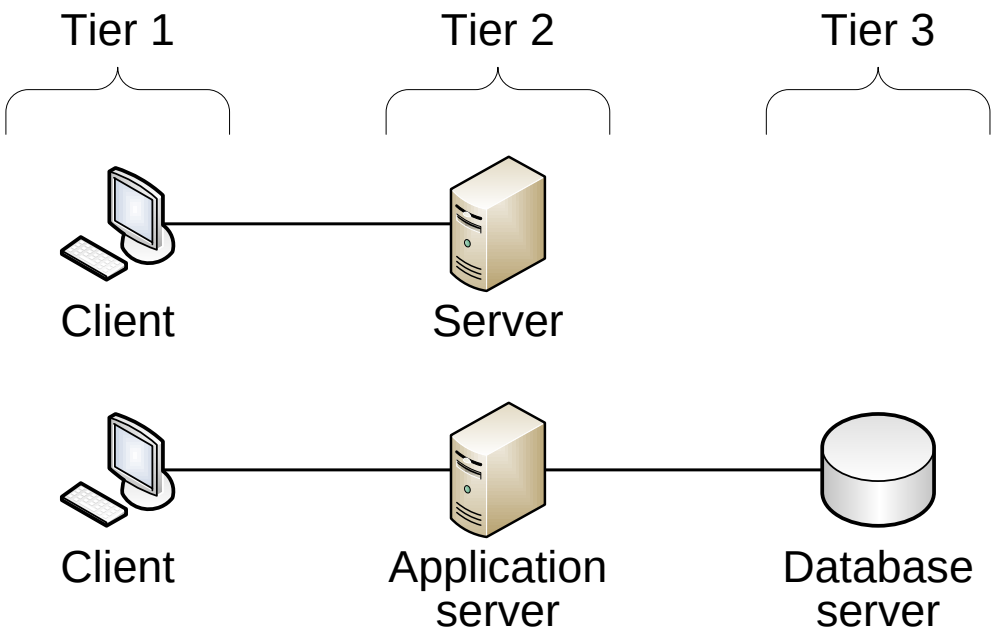
Физические уровни

- Распределение 3 логических уровней между 2 физическими



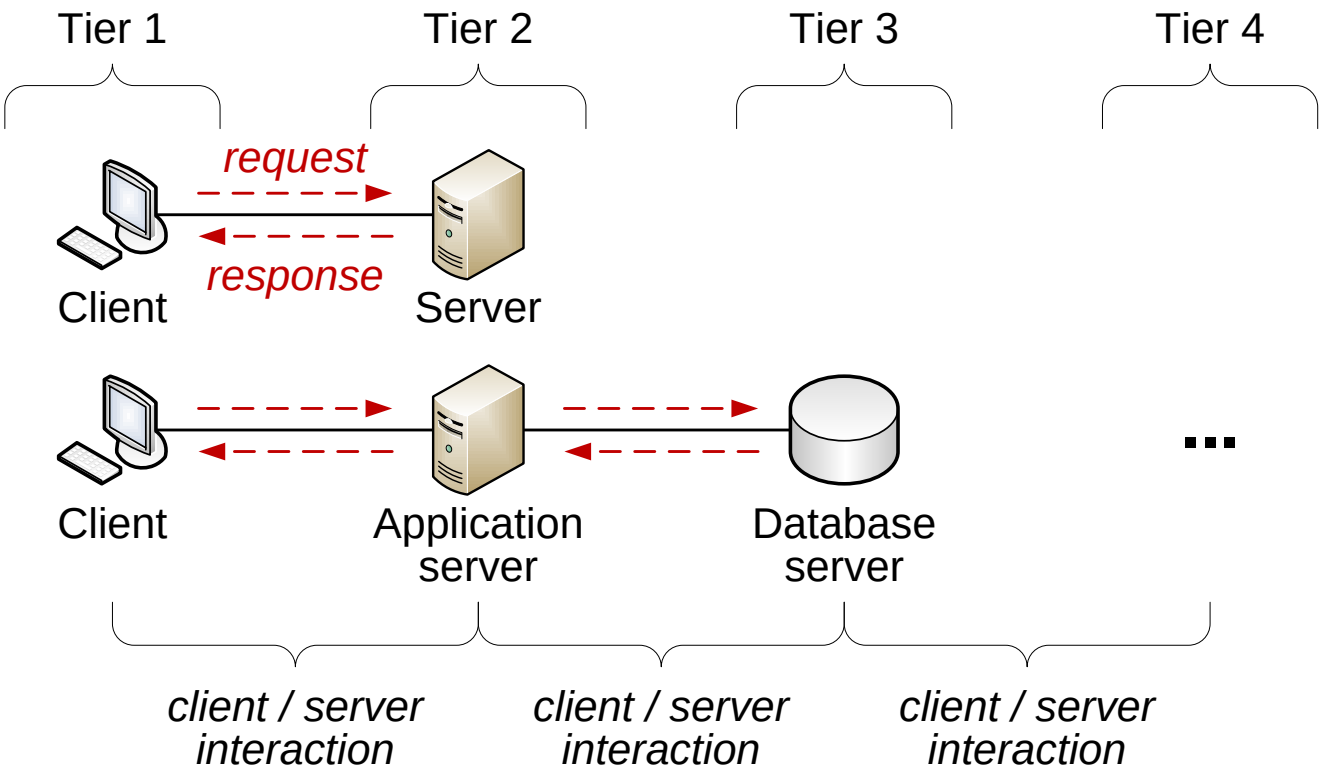
Физические уровни

- **3-уровневая архитектура** – клиент-серверная система, состоящая из:
  - **Клиентов**, которые запрашивают предоставление услуги
  - **Сервера приложений**, который предоставляет услугу, обращаясь при этом к базе данных
  - **Сервера базы данных**, который хранит данные, необходимые для обслуживания клиентов



Физические уровни

- **N-уровневая архитектура** – клиент-серверная система, состоящая более чем из 3 уровней



Физические уровни

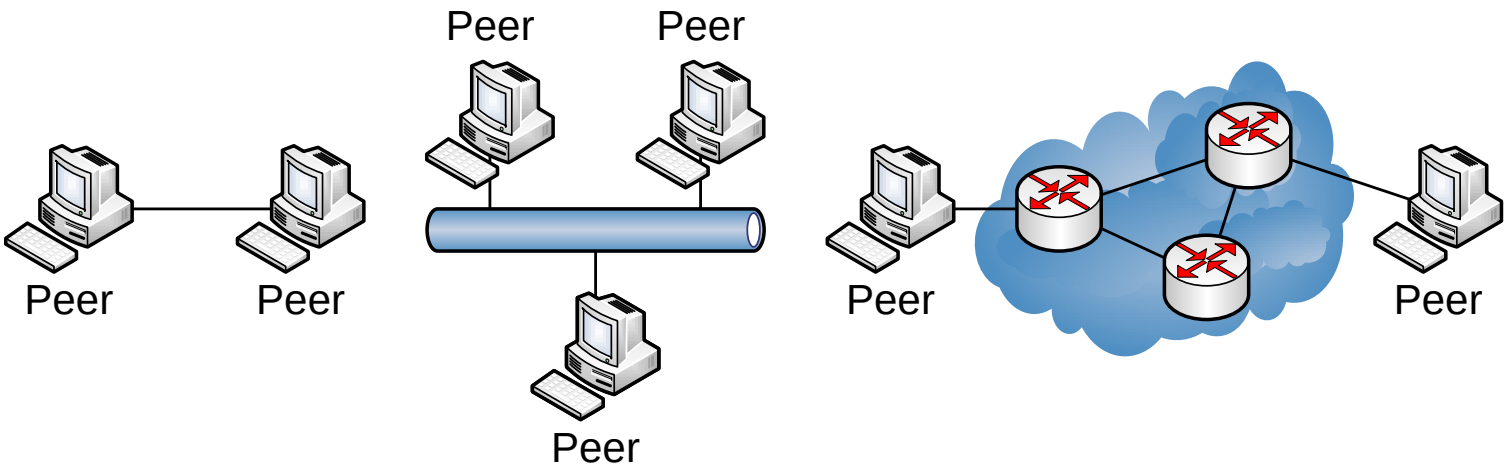
- **Плюсы многоуровневой архитектуры**:
  - Более гибкая, так как изменения на одном уровне не затрагивают остальные уровни
  - Более безопасная, так как меры безопасности могут быть реализованы на каждом уровне
  - Выше производительность, так как нагрузка распределяется между уровнями
- **Минусы многоуровневой архитектуры**:
  - Более сложная
  - Требуются меры по балансировке нагрузки и повышению отказоустойчивости

Содержание

1. Введение
2. Клиент-серверная модель
   - Классификация серверов
   - Классификация клиентов
   - Логические уровни
   - Физические уровни
3. Пиринговая модель
   - Децентрализованная P2P-система
   - Гибридная P2P-система
4. Выводы

Пиринговая модель

- В пиринговой моделе отсутствует разделение на клиентов и серверов: все узлы равны (peers) и функционируют как клиенты и серверы одновременно (т.н. servent = SERVer + cliENT)
  - Любой узел может послать запрос, но в каждом сеансе связи можно по-прежнему выделить 'клиента' и 'сервера' согласно их транзакционным ролям
  - Потоки трафика (теоретически) симметричны, в отличие от клиент-серверных систем, где трафик крайне асимметричен (т.е. в одном направлении передается данных больше, чем в другом)



Пиринговая модель

- **Плюсы**:
  - Не требуется сервер и наличие обслуживающего персонала
  - Выше отказоустойчивость
- **Минусы**:
  - Проблемы с безопасностью из-за отсутствия централизованного управления
  - Повышенная нагрузка на оборудование пользователей

Децентрализованная P2P-система

- **2 типа пиринговых систем**:
  - Децентрализованная (pure P2P)
  - Гибридная (hybrid P2P)

- **Децентрализованная P2P-система** – все узлы равны и одновременно выполняют функции клиентов и серверов
  - Серверы или выделенные узлы с особыми функциями отсутствуют
- Примеры: рабочие группы в Microsoft Windows NT, Gnutella v0.4
- **Плюсы**:
  - Высокая отказоустойчивость
- **Минусы**:
  - Неэффективный поиск контента в больших P2P-системах

Гибридная P2P-система

- **Гибридная пиринговая система** – P2P-система, в которой имеются серверы или выделенные узлы с особыми функциями
  - **Серверы** – содержат информацию о хранимых данных и месте их размещения
  - **Выделенные узлы** – управляют поиском контента
- Т.о., клиент-серверные и пиринговые системы не являются взаимоисключающими!
- Примеры: Usenet, Napster, Gnutella v0.6, eDonkey2000, BitTorrent
- **Плюсы**:
  - Более эффективный поиск контента, чем в децентрализованных системах
- **Минусы**:
  - Работоспособность всей системы зависит от работы сервера

Содержание

1. Введение
2. Клиент-серверная модель
   - Классификация серверов
   - Классификация клиентов
   - Логические уровни
   - Физические уровни
3. Пиринговая модель
   - Децентрализованная P2P-система
   - Гибридная P2P-система
4. Выводы

Выводы

- Клиент-серверное vs. пиринговое взаимодействие

	Клиент-серверное	**Пиринговое**
Участники	Клиенты и серверы	Одноранговые
ПО	Различно для клиентов и серверов	Одинаково для всех участников
Активная роль	Клиент	Любой участник
Пассивная роль	Сервер	Любой участник
Взаимодействие	Клиенты с серверами	Произвольное
Провайдеры услуг/ контента/ресурсов	Серверы	Активные участники
Потоки данных	Асимметричны	Симметричны

Выводы

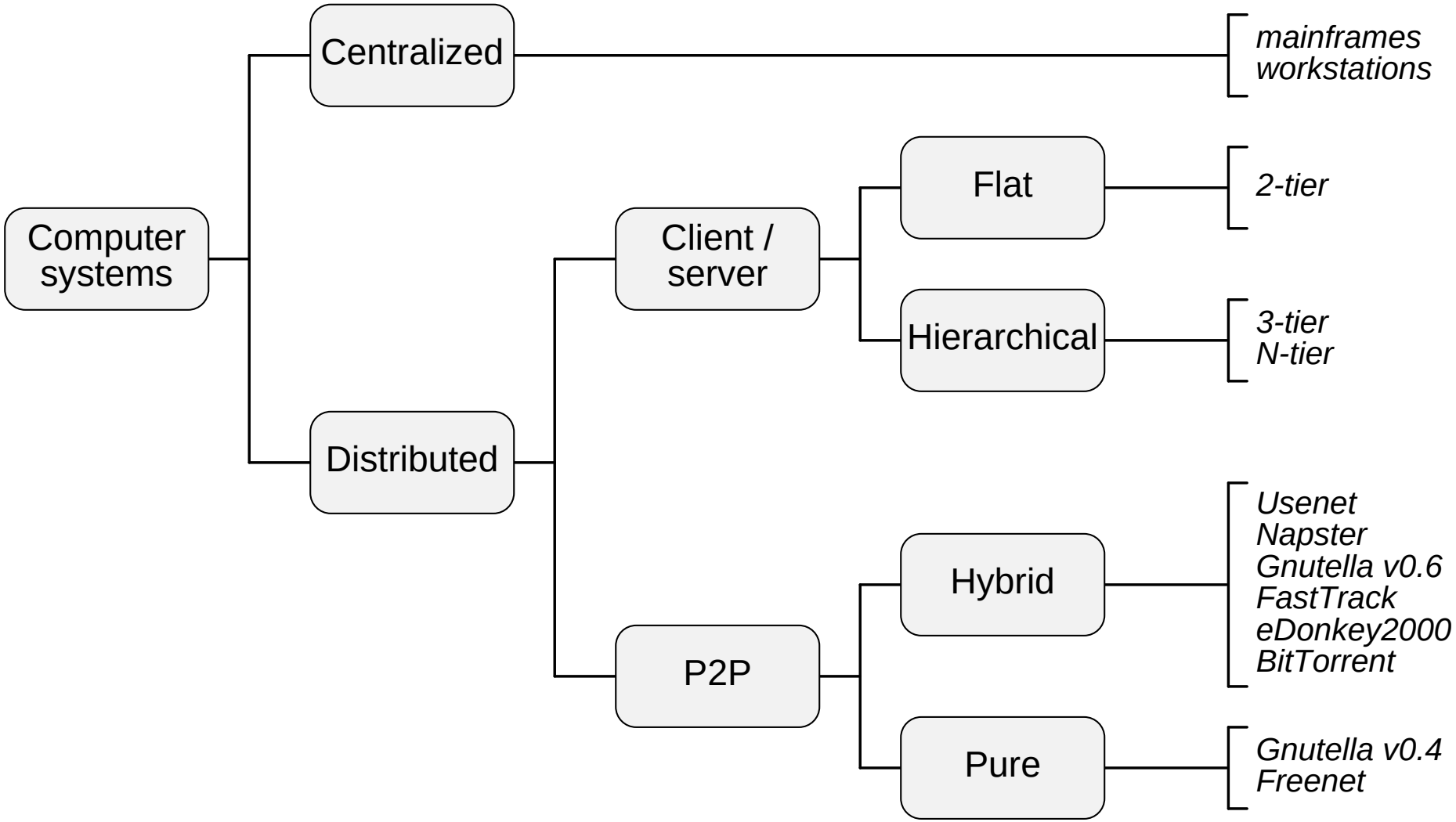
- **Клиент-серверное vs. пиринговое взаимодействие**: лекция vs. совместная работа над проектом

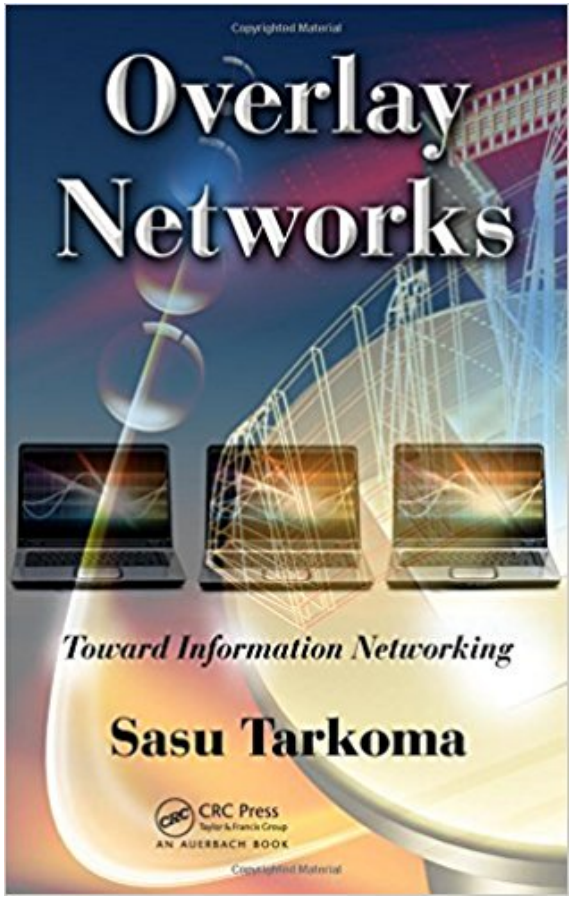
Выводы

- **Клиент-серверное vs. пиринговое взаимодействие**: посещение ресторана vs. приготовление еды дома

Выводы

- Классификация архитектур компьютерных систем



Выводы

- Sasu Tarkoma, 'Overlay Networks: Toward Information Networking', CRC Press, 2010