Групповые взаимодействия и рассылка

Олег Сухорослов

Распределенные системы

Факультет компьютерных наук НИУ ВШЭ

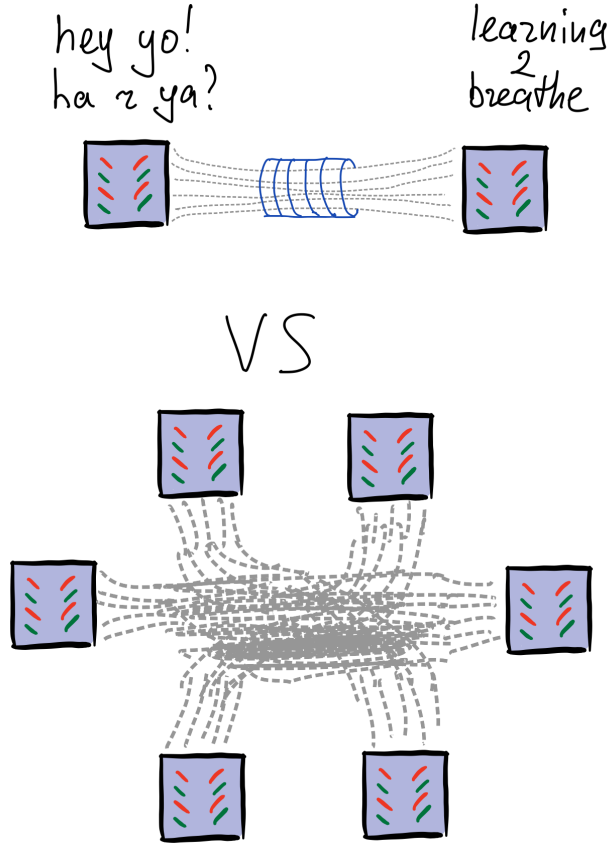
27.09.2021

План

- Групповые взаимодействия
- Реализация рассылки сообщений в группе
- Масштабируемые подходы к распространению информации

Взаимодействия: число процессов

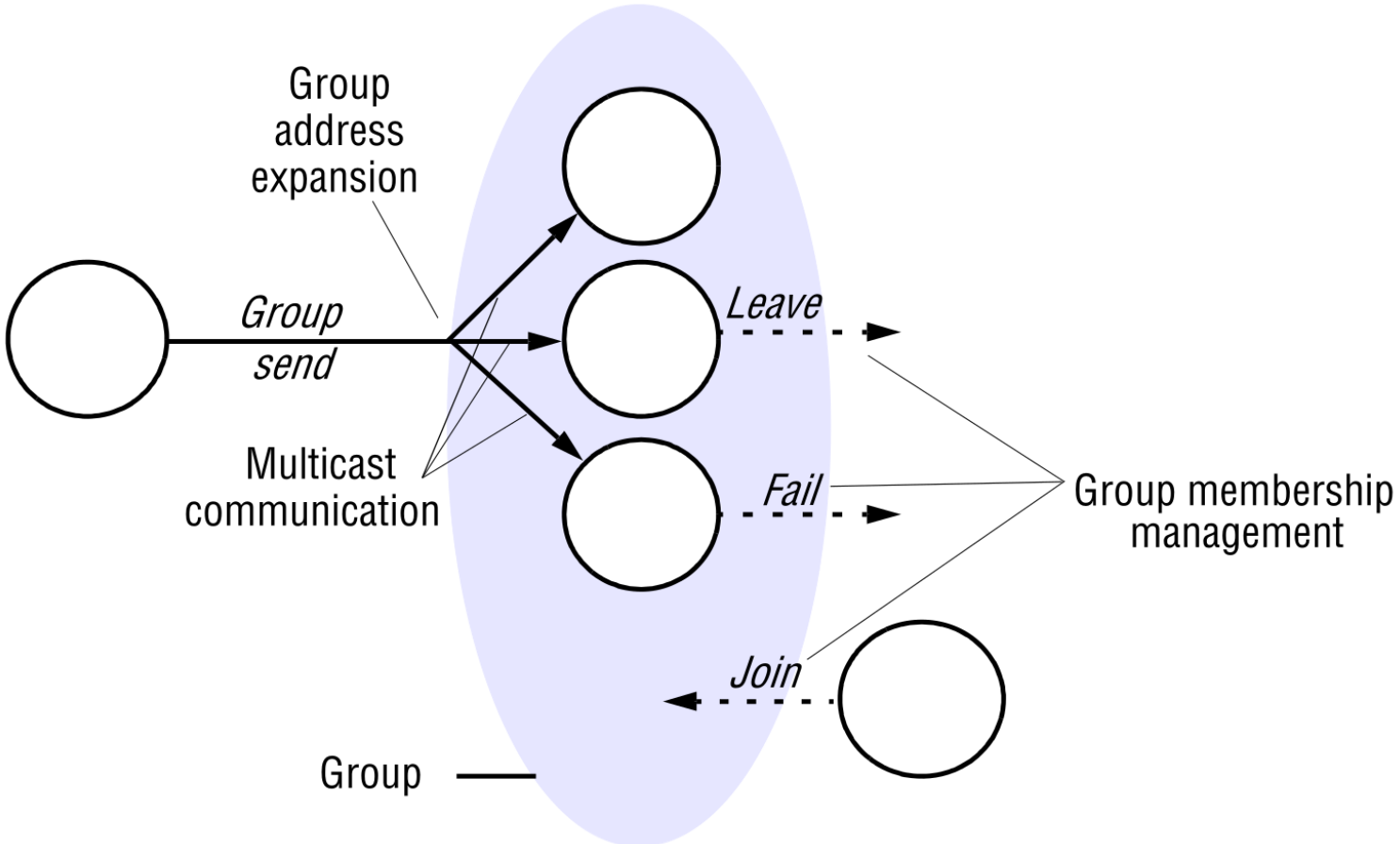
- Парные взаимодействия
  - point-to-point, one-to-one
  - TCP, RPC, HTTP
- Групповые взаимодействия
  - one-to-many, many-to-many
  - ???



Применение

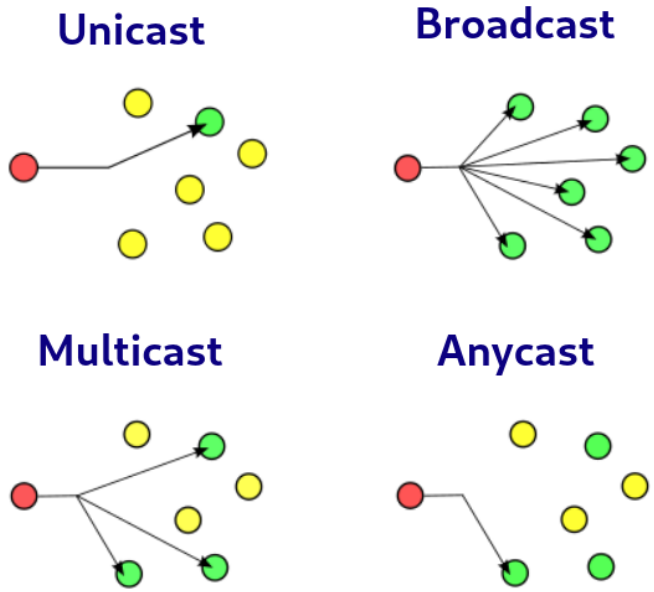
- рассылка уведомлений о событиях
- доставка контента и потоковое вещание
- поиск сервисов и разрешение имен
- синхронизация времени
- поиск данных
- параллельные вычисления
- репликация сервисов или данных

Реализация группового взаимодействия

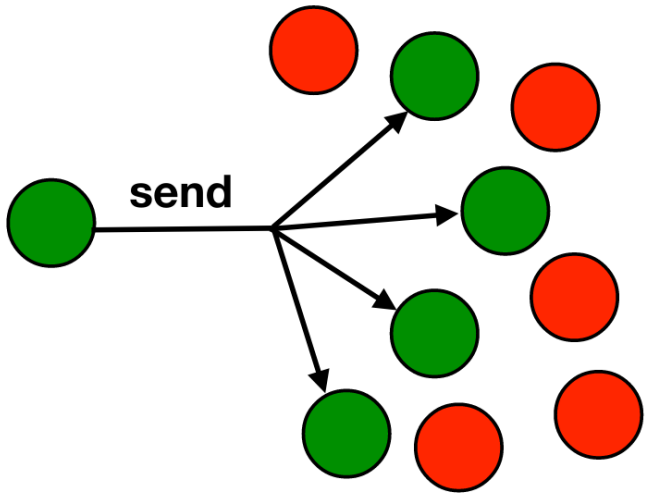


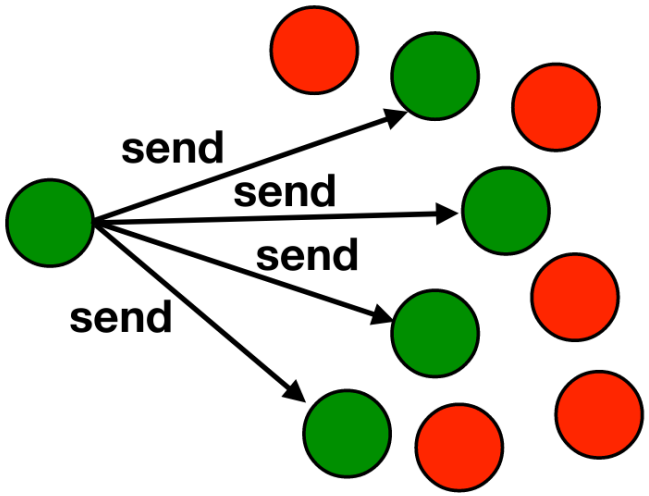
Схемы передачи сообщений

- Unicast (point-to-point)
  - одноадресная передача
- Broadcast
  - широковещательная рассылка
- Multicast
  - многоадресная рассылка
  - source-specific multicast (one-to-many)
  - any-source multicast (many-to-many)
- Anycast
  - передача кому угодно

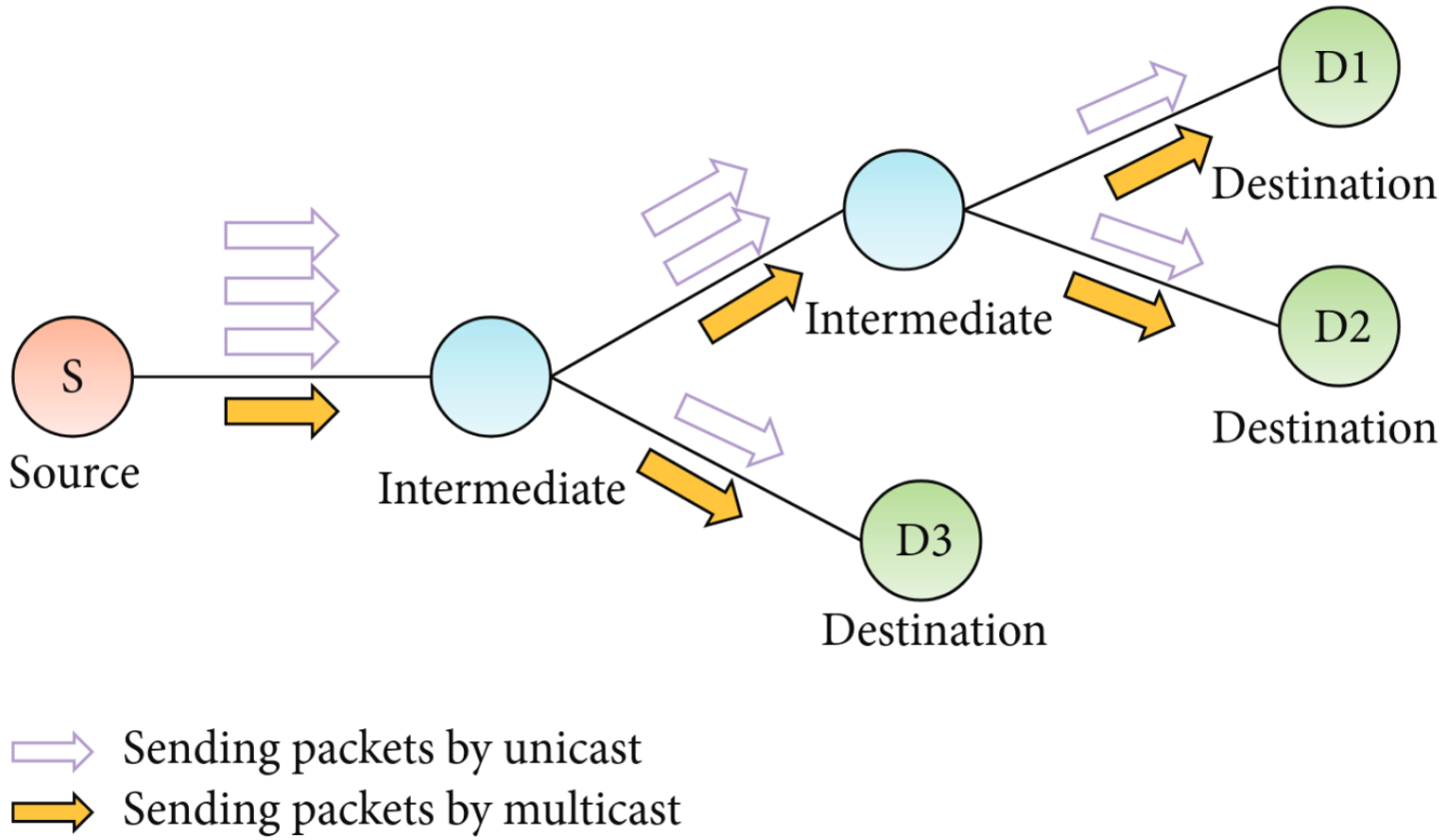


Multicast vs Unicast



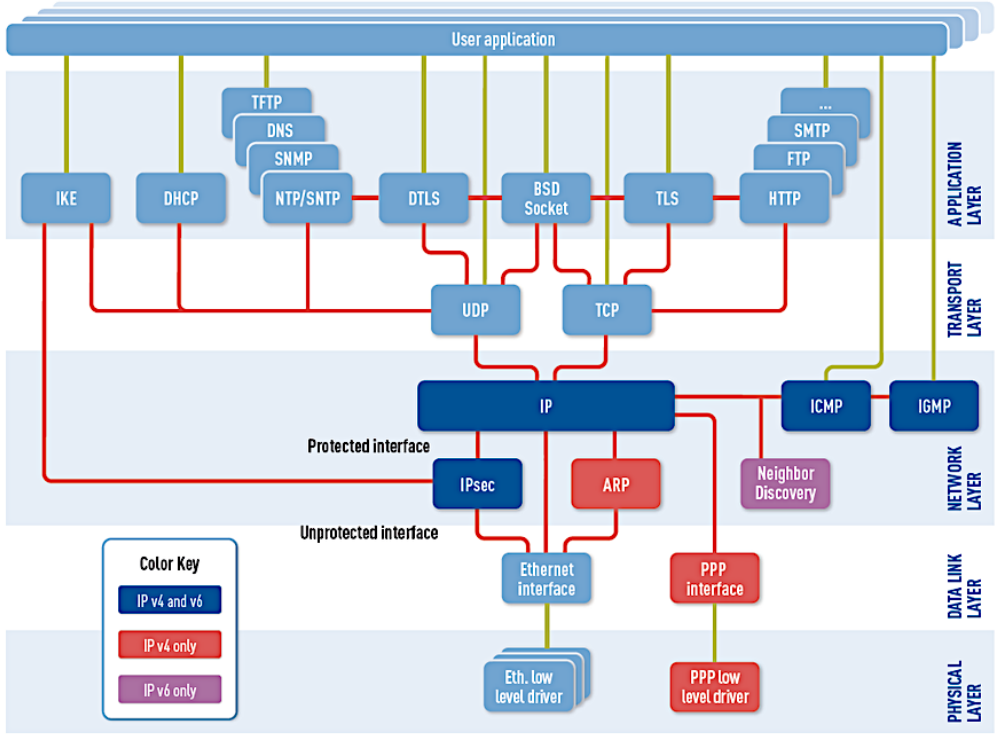


Multicast vs Unicast



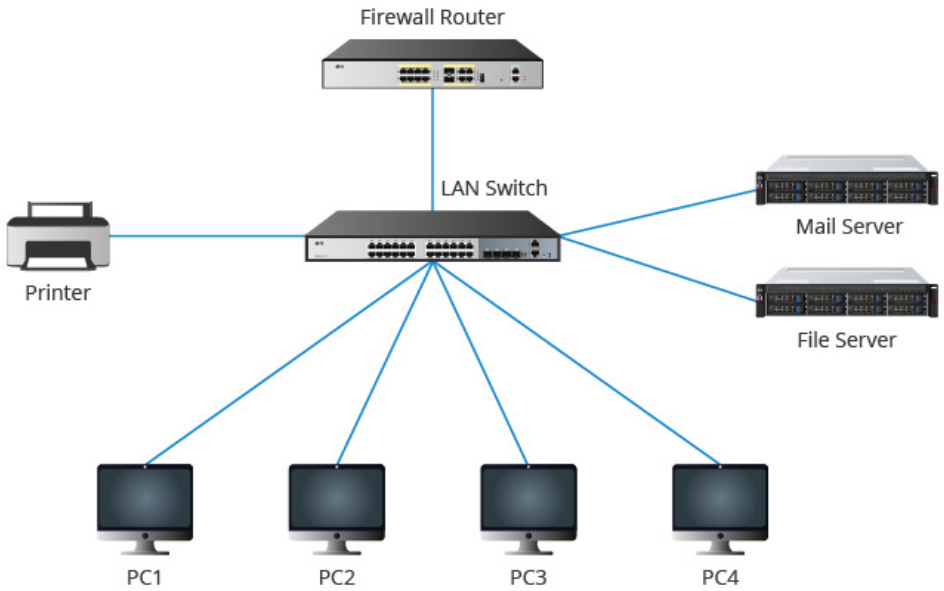
Реализация рассылки

- На уровне сети
  - канальный уровень (Ethernet)
  - сетевой уровень (IP)
- На уровне приложения



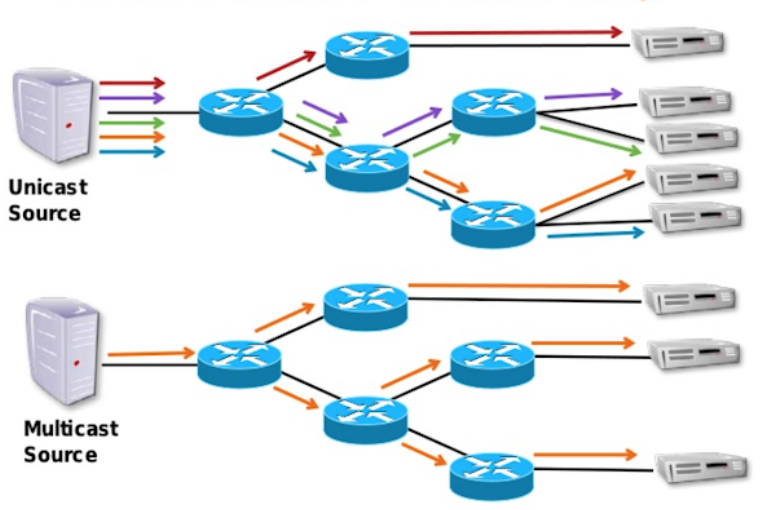
Ethernet

- Выделенный диапазон MAC-адресов
- Рассылка по всем устройствам в сети



IP Multicast

- Позволяет отправить один пакет сразу всем участникам multicast-группы
- Группа идентифицируется с помощью уникального IP-адреса
- Машины в сети могут динамически вступать и выходить из групп
- Для отправки данных не требуется быть участником группы
- Доступ на уровне приложений чаще всего реализован через протокол UDP

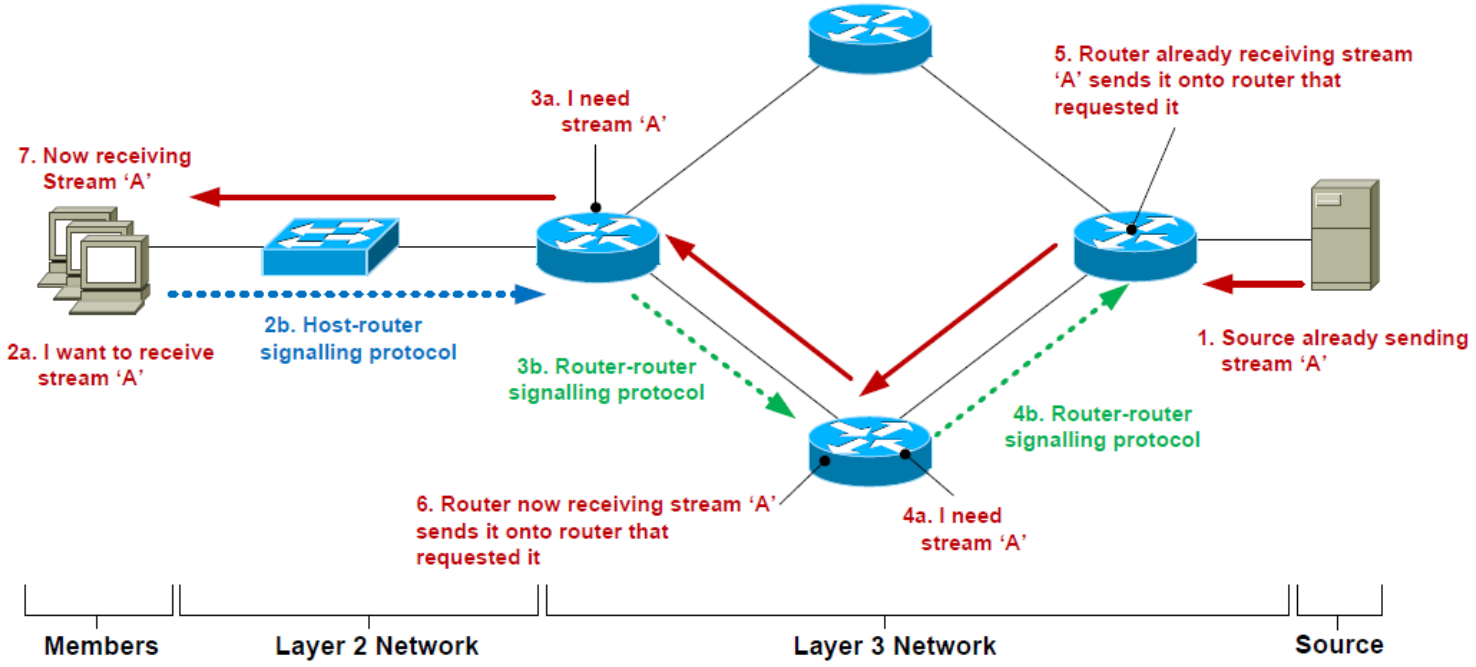


Гарантии IP Multicast

- Мультикаст на базе UDP
  - контроль целостности
  - доставка не гарантируется
  - сохранение порядка сообщений не гарантируется
- Протокол Pragmatic General Multicast (PGM)
  - IETF experimental protocol
  - надежная доставка и сохранение порядка сообщений
  - использует отрицательные подтверждения (NAKs)

IP Multicast в глобальной сети

- Требуется поддержка со стороны маршрутизаторов
- Распространение данных контролируется с помощью TTL (time to live)
- Основные протоколы: IGMP, PIM



Рассылка на уровне приложения

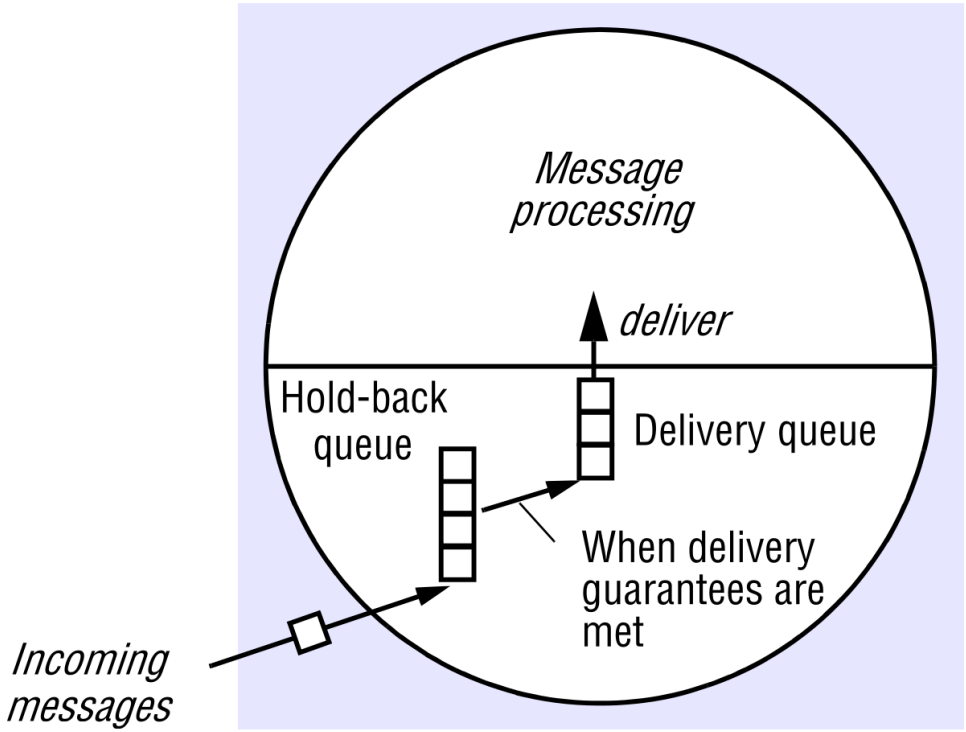
- Отсутствует поддержка со стороны сети
- Недостаточно предоставляемых возможностей и гарантий

Важные моменты

- Адресация участников группы
- Надежность и семантика доставки
- Порядок доставки
- Семантика ответа
- Состав и открытость группы

Интерфейс

- *join(group)*
- *leave(group)*
- *multicast(group, message)*
  - внутри сообщения указываются sender и group
- *receive(group) -> message*
- обратный вызов *deliver(message)*
- ...



Предположения

Далее рассмотрим несколько реализаций рассылки, использующих следующие предположения:

- группа закрытая, состав участников группы зафиксирован
- все процессы в группе знают адреса друг друга
- каналы между процессами надежные (см. следующий слайд)
- процессы могут отказывать только путем полной остановки
  - процесс, который не отказал в ходе выполнения рассылки, будем называть *корректным*

Надежная доставка (point-to-point)

- **Validity**: каждое сообщение будет доставлено
  - если корректный процесс p отправил сообщение m корректному процессу q, то q в конце концов доставит m
- **No Duplication**: отсутствуют повторы сообщений
  - никакое сообщение не доставляется процессом более одного раза
- **No Creation**: сообщения доставляются без искажений
  - если некоторый процесс q доставил сообщение m от процесса p, то m было ранее отправлено от p к q
- **Integrity**: No Duplication + No Creation
- Это свойства безопасности (safety) и живучести (liveness)

Basic Multicast: Свойства

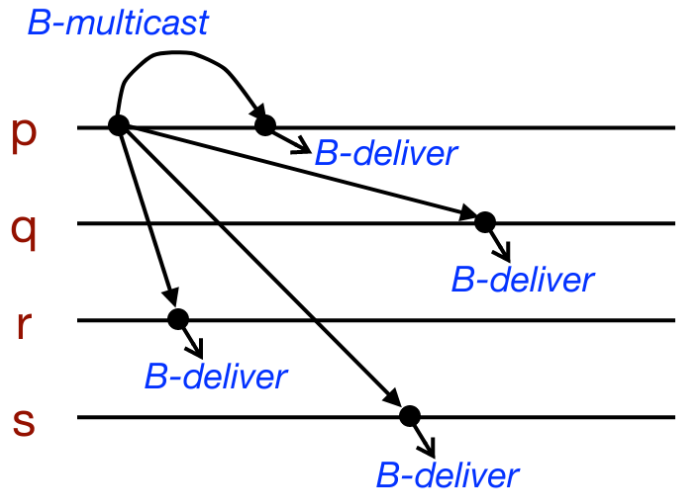
- **Validity**: если корректный процесс разослал сообщение m, то каждый корректный процесс в конце концов доставит m
- **No Duplication**: корректный процесс p доставляет сообщение m не более одного раза
- **No Creation**: если корректный процесс p доставил сообщение m с отправителем s, то m было ранее разослано s

  также называется *best-effort multicast*

Basic Multicast: Реализация

To *B-multicast(g, m)*:
for each process p ∈ g, *send(p, m)*

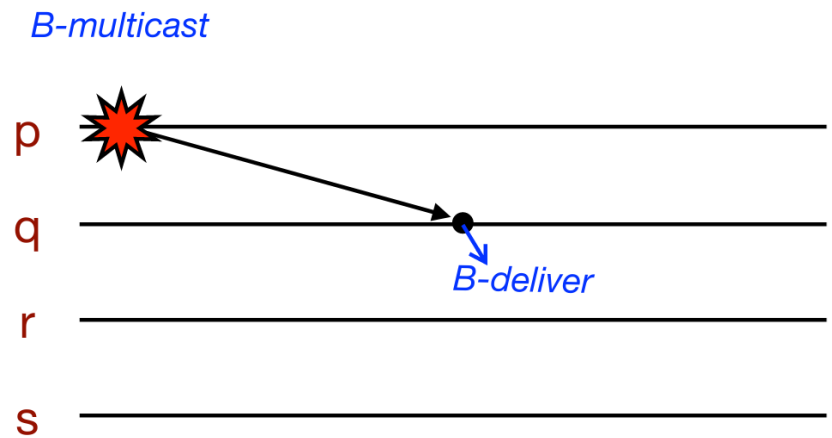
On *receive(m)* at p:
B-deliver(m) at p



- Использует надежный point-to-point канал в виде операции *send()*
- Выполнение свойств следует из свойств надежного канала
- Подвержена Ack-Implosion Problem

Отказ отправителя

- B-multicast надежен только если отправитель корректный
- При отказе отправителя часть процессов в группе может получить и доставить сообщение, а часть нет
- Отсутствует *согласие* между процессами относительно доставки сообщения



Reliable Multicast: Свойства

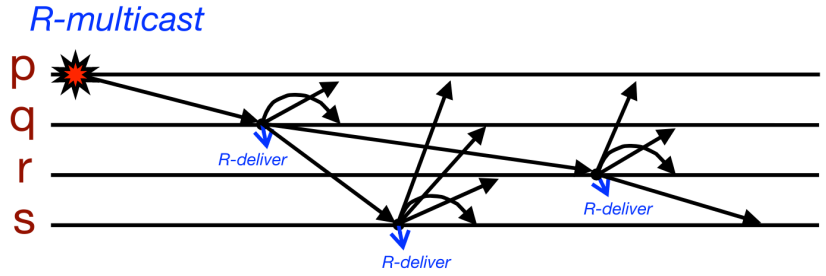
- **No Duplication**: корректный процесс p доставляет сообщение m не более одного раза
- **No Creation**: если корректный процесс p доставил сообщение m с отправителем s, то m было ранее разослано s
- **Validity**: если корректный процесс разослал сообщение m, то он в конце концов доставит m
- **Agreement**: если некоторый корректный процесс доставил сообщение m, то все остальные корректные процессы в группе в конце концов доставят m

Eager Reliable Multicast

On initialization
Received := {};

For process p to R-multicast message m to group g
B-multicast(*g*, *m*); // $p \in g$ is included as a destination

On B-deliver(*m*) *at process q with g = group*(*m*)
if ($m \notin$ *Received*)
then
Received := *Received* $\cup$ {*m*};
if ($q \neq p$) *then B-multicast*(*g*, *m*); *end if*
R-deliver m;
end if



- Упражнение: показать, что выполняются свойства Reliable Multicast
- Низкая эффективность - $O(N^2)$ сообщений!

Другие реализации Reliable Multicast

- Gossip (см. далее в лекции)
- Lazy вариант с детектором отказов (см. далее в курсе)
- IP multicast + подтверждения

Reliable Multicast поверх IP Multicast (1)

- Используем IP multicast и подтверждения
  - подтверждения отправляются вместе с рассылаемыми сообщениями (piggyback)
  - отдельное сообщение в случае обнаружения пропуска сообщения (negative ack)
- Каждый процесс p хранит локально
  - S_p^g - sequence number группы g, в начале 0
  - R_q^g - номер последнего доставленного им сообщения от q в g
- Отправка сообщения
  - к сообщению добавляются значение S_p^g и подтверждения $\langle q, R_q^g \rangle$
  - сообщение с добавкой рассылается через IP multicast
  - значение S_p^g увеличивается на 1

Reliable Multicast поверх IP Multicast (2)

- Получение сообщения с номером S от p
  - если $S = R_p^g + 1$, то сообщение доставляется и R_p^g увеличивается на 1
  - если $S \leq R_p^g$, то сообщение было получено ранее и отбрасывается
  - если $S > R_p^g + 1$, то сообщение помещается в hold-back queue
  - если $S > R_p^g + 1$ или $R > R_q^g$ для подтверждения $\langle q, R \rangle$ из сообщения, то какие-то сообщения еще не получены и возможно потеряны при рассылке
  - процесс запрашивает недостающие сообщения от их отправителей или других процессов, который получали эти сообщения, путем отправки negative acknowledgement
- Особенности
  - требуется постоянная (бесконечная) рассылка сообщений
  - необходимо (вечное) хранение доставленных сообщений на всех процессах
  - попутно получили сохранение порядка сообщений

Uniform Agreement

- Расширение свойства Agreement с корректных до всех процессов
  - Если некоторый процесс доставил сообщение m, то все корректные процессы в группе в конце концов доставят m
- Даже если процесс отказал после доставки сообщения, корректные процессы также должны доставить это сообщение
- Удовлетворяют ли этому свойству наши реализации Reliable Multicast?
- Как обеспечить данное свойство? (см. домашнее задание)

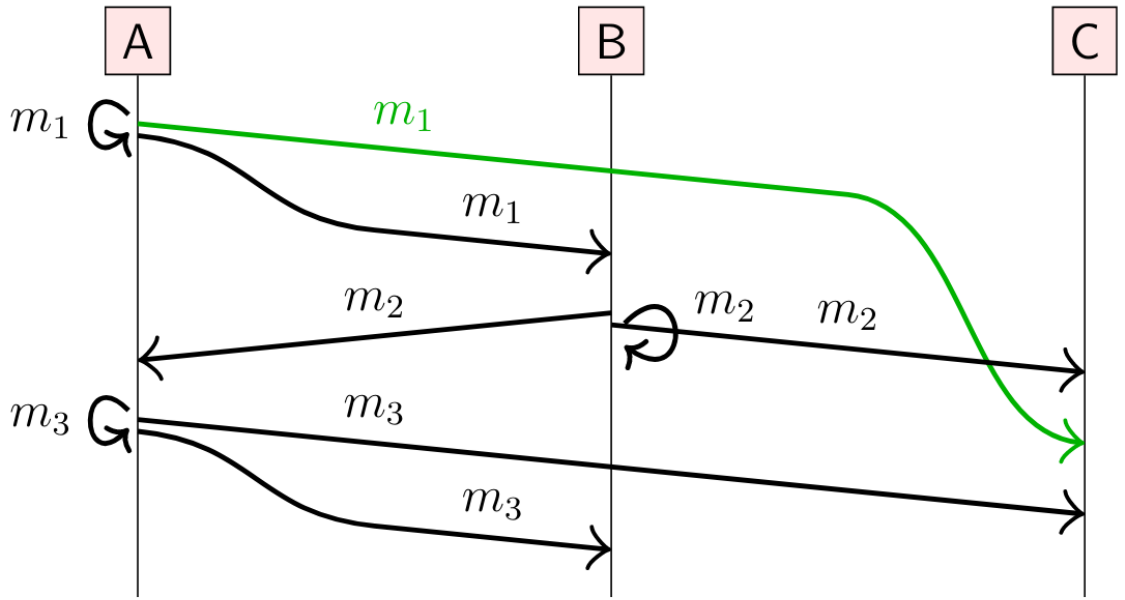
Порядок доставки сообщений

- Произвольный
- FIFO Order
- Causal Oder
- Total Order

Гарантии порядка рассматриваются отдельно от надежности

FIFO Order

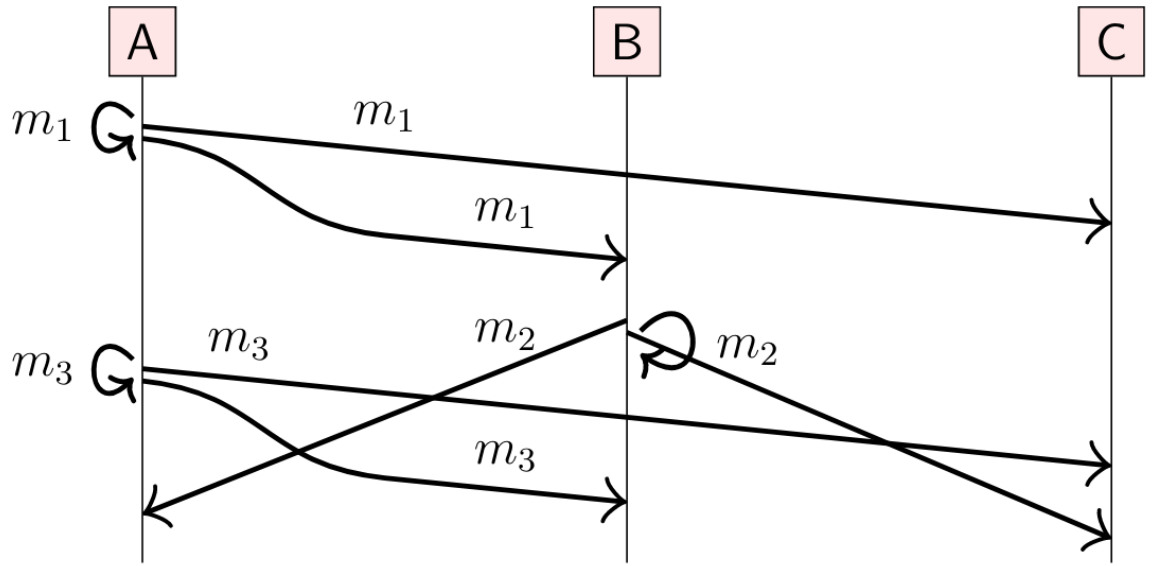
Если корректный процесс сначала разослал m а потом m', то любой корректный процесс, который доставил m', доставит m до m'



Source: Martin Kleppmann Distributed Systems

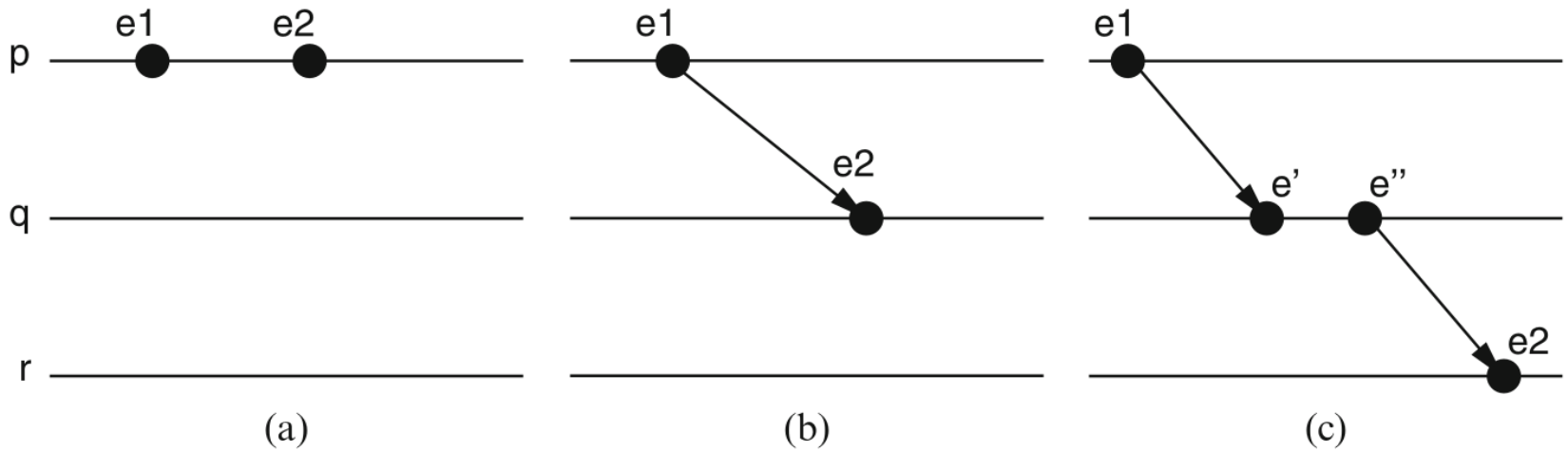
Causal Order

Если рассылка m произошла до (happened-before) рассылки m', то любой корректный процесс, который доставил m', доставит m до m'



Source: Martin Kleppmann Distributed Systems

Отношение happened-before

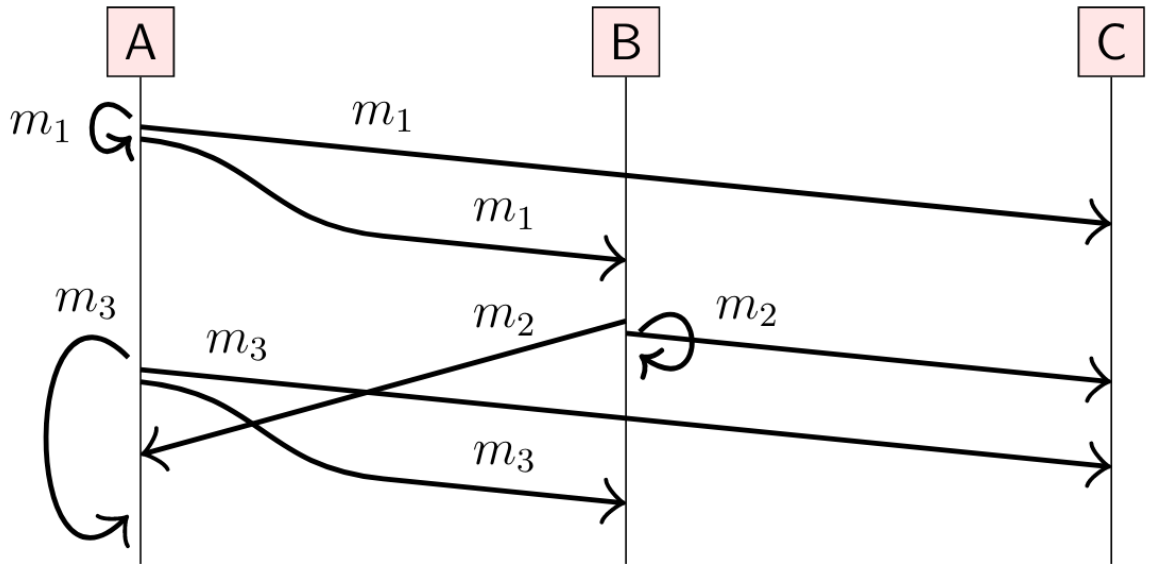


Событие a произошло до события b ($a \rightarrow b$) если выполняется одно из условий:

- a и b произошли на одном процессе, и a произошло раньше
- a является отправкой сообщения m, а b является получением того же сообщения m
- существует событие c такое что $a \rightarrow c$ и $c \rightarrow b$

Total Order

Если некоторый корректный процесс доставил m до m', то любой другой корректный процесс, который доставил m', доставит m до m'



Source: Martin Kleppmann Distributed Systems

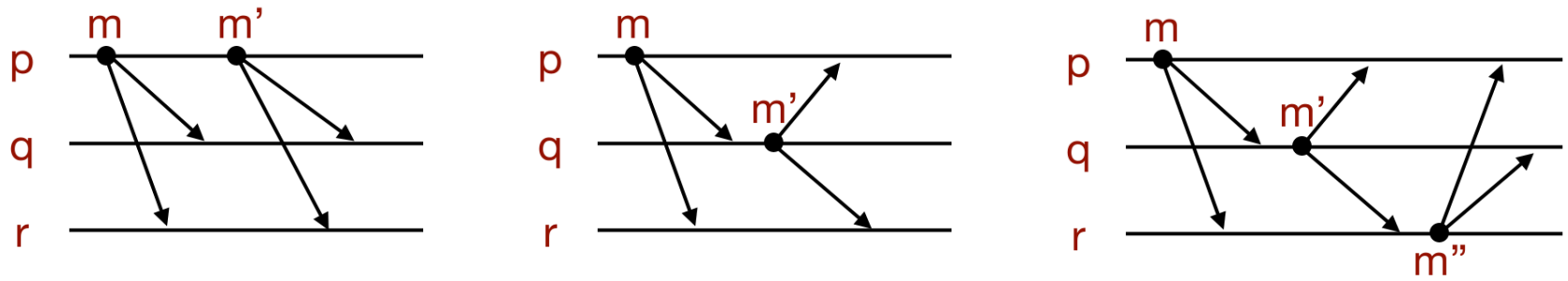
Порядок доставки сообщений

- FIFO Order
  - частичный порядок
- Causal Oder
  - частичный порядок
  - включает в себя FIFO
- Total Order
  - полный порядок
  - не гарантирует FIFO или Causal порядки
  - возможные комбинации: FIFO-Total, Causal-Total
  - FIFO-Total включает в себя Causal

FIFO Order: Реализация

- Основана на использовании sequence numbers
- Каждый процесс p хранит локально
  - S_p^g - сколько сообщений p отправил в группу
  - R_q^g - номер последнего сообщения от q в g, которое доставил p
- При отправке процесс добавляет к сообщению S_p^g и затем увеличивает S_p^g на 1
- При получении сообщения с номером S от процесса q
  - если $S = R_q^g + 1$, то сообщение доставляется
  - если $S > R_q^g + 1$, то сообщение добавляется в hold-back queue
- Для рассылки достаточно использовать B-Multicast
  - если использовать R-Multicast, то получим Reliable FIFO Multicast

Causal Order: Реализация



- Каждый процесс p поддерживает локально вектор размера N
  - j-я компонента вектора равна числу сообщений, которые p доставил от j
- Векторы рассылаются вместе с сообщениями и используются для упорядочивания сообщений
- Вариант *векторных часов*, рассматриваемых далее в курсе
- Для рассылки можно использовать B-multicast или R-multicast

Causal Order: Реализация

Algorithm for group member p_i $(i = 1, 2\ldots, N)$

On initialization

$V_i^g[j] := 0$ $(j = 1, 2\ldots, N)$;

To CO-multicast message m to group g

$V_i^g[i] := V_i^g[i] + 1$;

B-multicast$(g, <V_i^g, m>)$;

On B-deliver$(<V_j^g, m>)$ *from* p_j $(j \neq i)$, *with* $g = group(m)$

place $<V_j^g, m>$ in hold-back queue;

wait until $V_j^g[j] = V_i^g[j] + 1$ and $V_j^g[k] \leq V_i^g[k]$ $(k \neq j)$;

CO-deliver m; // after removing it from the hold-back queue

$V_i^g[j] := V_i^g[j] + 1$;

Total Order: Реализация

- Основная идея: назначить каждому сообщению уникальный номер
  - sequence numbers на уровне всей группы
  - каждый процесс может локально упорядочить сообщения
- Возможные подходы
  - централизованный - выделенный процесс-лидер (sequencer)
  - распределенный - процессы согласуют номера друг с другом, логические часы
  - см. литературу

Рассмотренные реализации рассылки

- Basic (aka best-effort)
- Reliable
- (Reliable) FIFO
- (Reliable) Causal
- Total Ordered

- Что насчёт Reliable + Total Ordered = Atomic Multicast?
  - Эквивалентен задаче консенсуса, рассматриваемой позже в курсе

Рассмотренные реализации рассылки

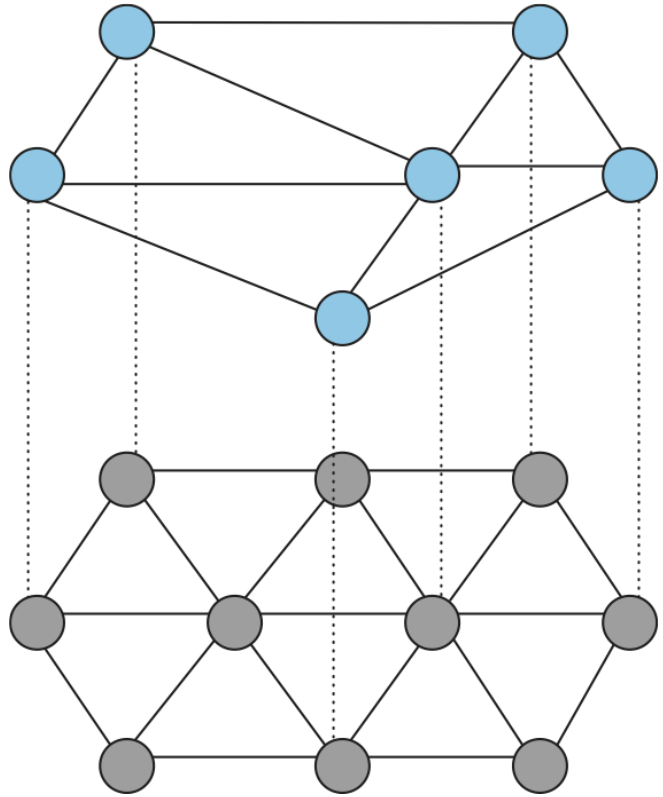
- Акцент на гарантиях надежности и порядка
- Упрощающие предположения
  - отказы только типа полной остановки
  - все процессы знают друг друга
  - состав групп зафиксирован
- Как обеспечить масштабируемость?
  - участников очень много
  - они могут находиться в разных частях Интернета и не знать друг о друге
  - классические подходы не работают или создают большую нагрузку на сеть

Масштабируемые подходы

- Топологии на базе оверлейных сетей
  - рассылки по дереву или mesh-сети
- Распространение информации с помощью epidemic protocols
  - gossip, анти-энтропия, rumor spreading

Оверлейная сеть (overlay network)

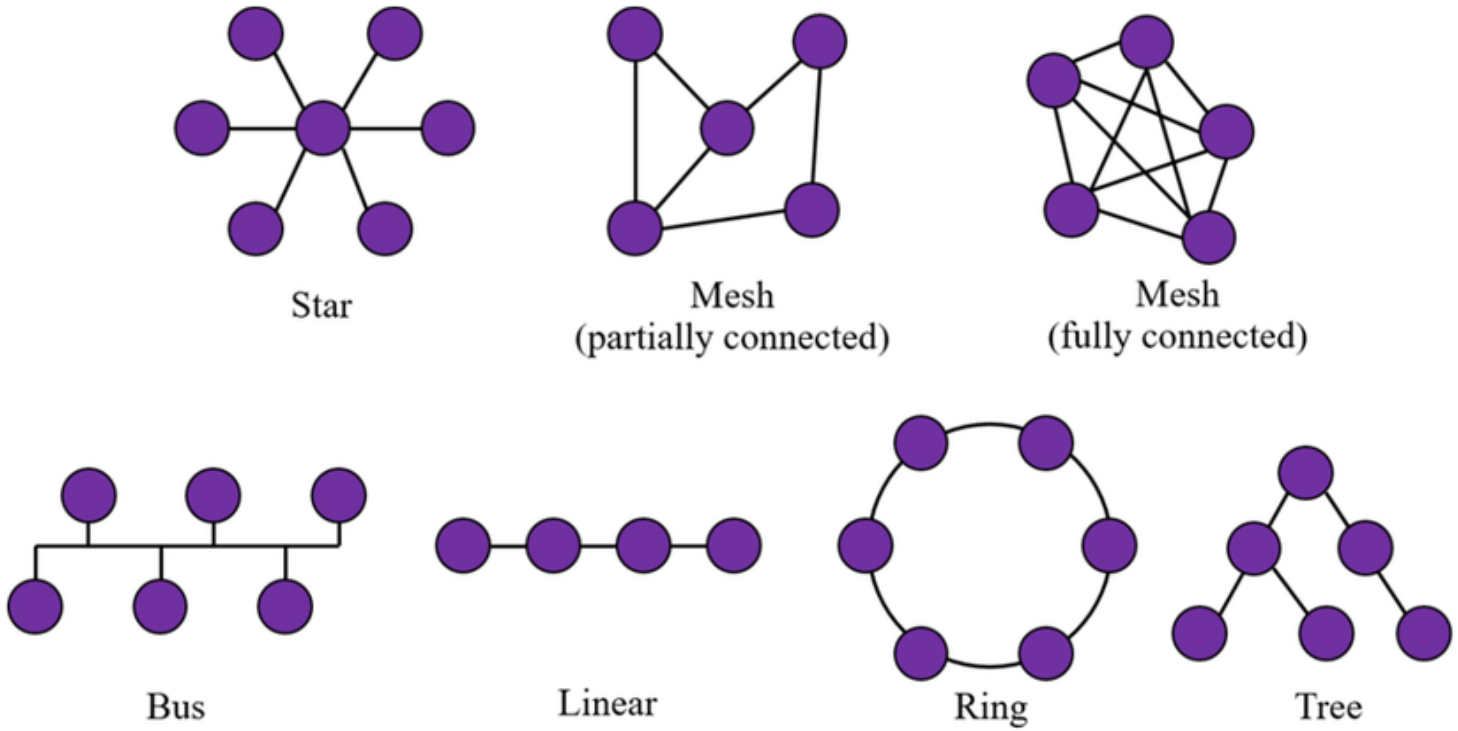
- "Виртуальная" сеть поверх физической сети
- Реализует набор сервисов
  - специфичных для приложения
  - более эффективных, чем доступные в обычной сети
  - недоступных в обычной сети
- Основные элементы
  - Топология
  - Адресация узлов
  - Протоколы
  - Алгоритмы маршрутизации



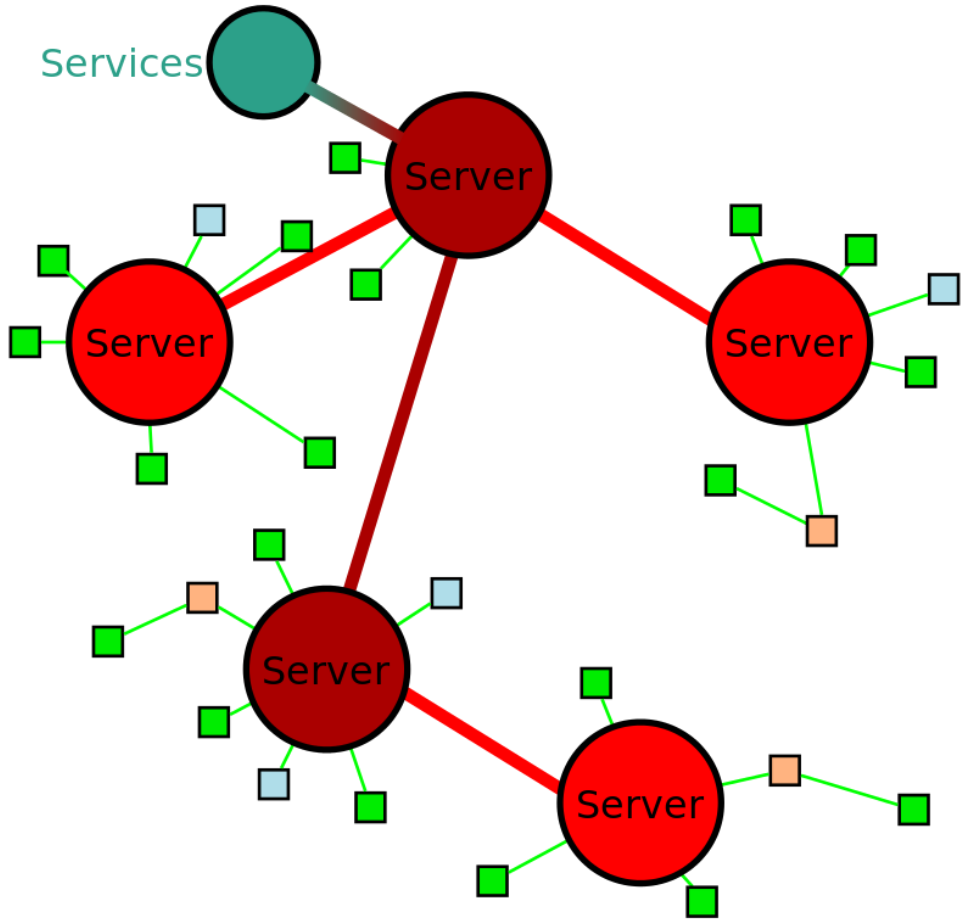
Применение оверлейных сетей

- Мультикаст
- Доставка контента, VoIP, потоковое видео
- Улучшенная маршрутизация в Интернете
- Именование и поиск (peer-to-peer сети)
- Беспроводные и самоорганизующиеся сети
- Обспечение безопасности (VPN)

Возможные топологии



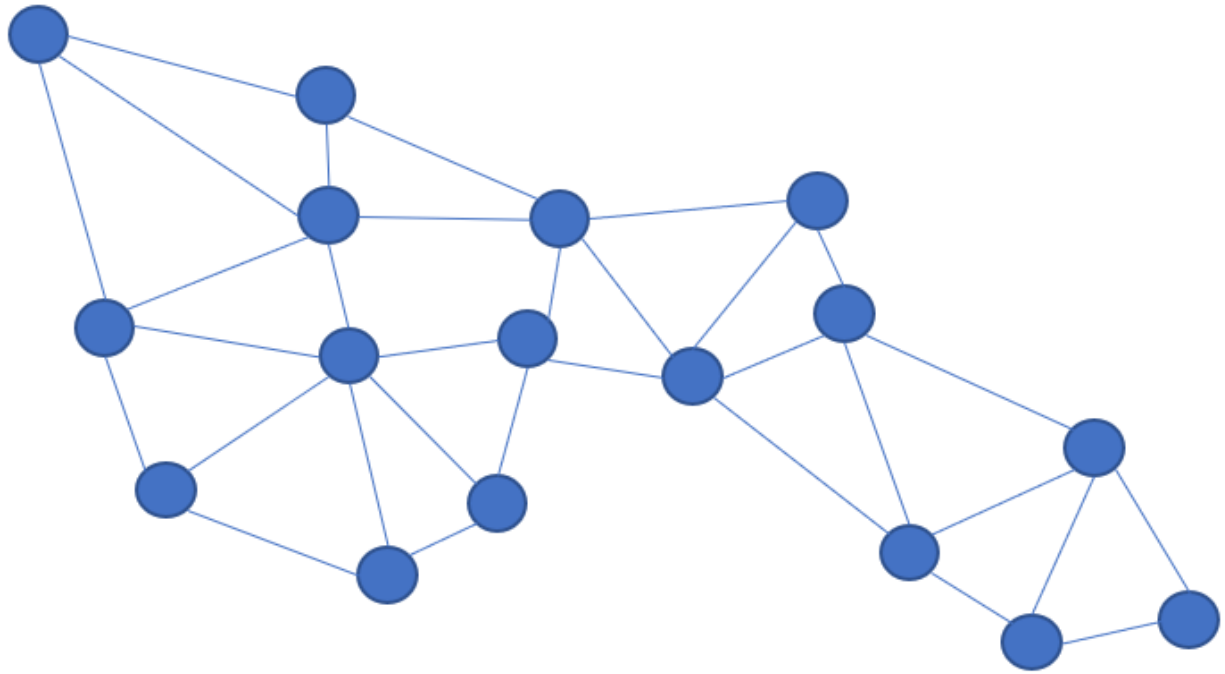
Дерево: Internet Relay Chat



Дерево: Особенности

- Построение эффективного остовного дерева
  - Корнем является источник мультикаста
  - За основу можно взять существующую mesh-сеть
  - Или динамически определять "близость" узлов
- Добавление нового узла
  - Выбор родителя для нового узла
  - Баланс между минимизацией длин путей и нагрузкой на узлы
  - Может потребоваться переконфигурация дерева
- Починка дерева в случае отказа
- Примеры: PlumTree (НИС), switch-trees (литература)

Mesh-сеть: Flooding и Pull

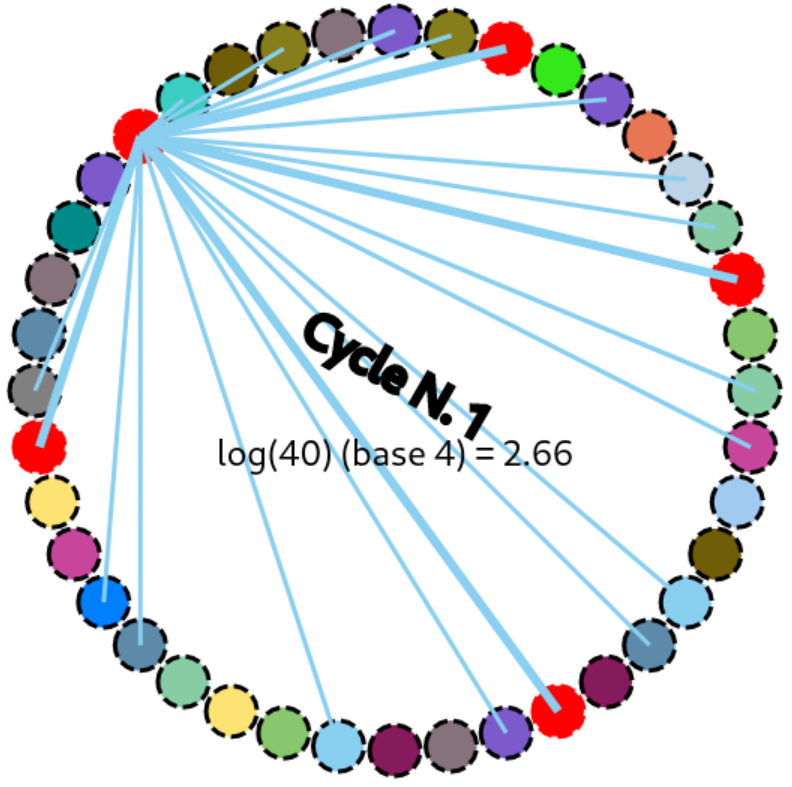


Mesh-сеть: Особенности

- Лучшая устойчивость к отказам, чем дерево
- Лучше приспособлены к динамическому составу участников (churn rate)
- Сложнее организовать эффективную рассылку
- Может требоваться буферизация полученных данных (pull)

Gossip

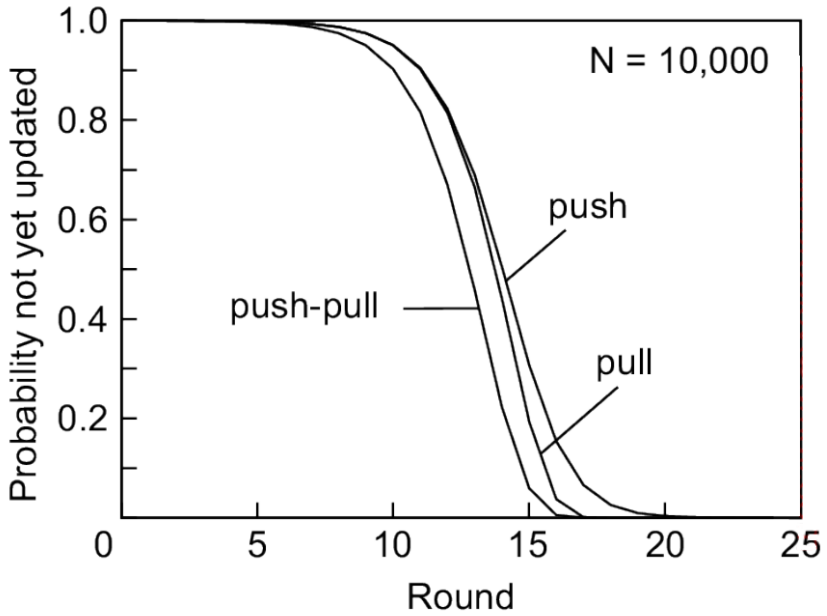
- Подход к распространению информации на основе локальных связей
- Аналогии с распространением слухов или болезней (epidemic protocols)
- Возможные состояния узла: infected, susceptible, removed
- В каждом раунде узел взаимодействует с одним или несколькими соседями (fanout)
- Для распространения данных на все узлы требуется *O(logN)* раундов



https://flopezluis.github.io/gossip-simulator/

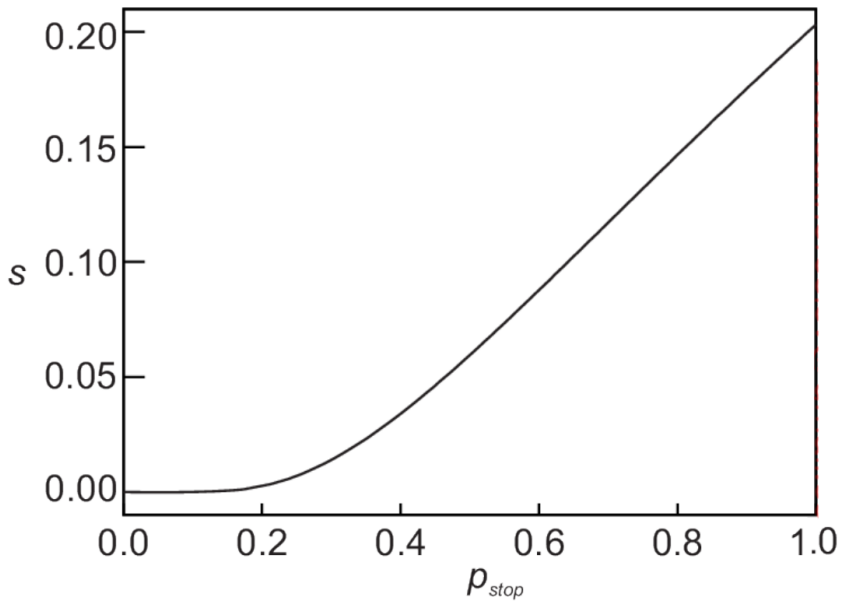
Анти-энтропия

- Узел *P* выбирает случайным образом другой узел *Q*
  - Push: *P* отправляет *Q* известную ему информацию (обновления)
  - Pull: *P* запрашивает у *Q* известную тому информацию
  - Push-Pull: *P* и *Q* обмениваются известной им информацией



Rumor spreading

- Если сосед уже имеет информацию, то узел перестает распространять ее с вероятностью p_{stop}
- Не гарантирует распространение информации до всех узлов



Литература

- Coulouris G.F. et al. Distributed Systems: Concepts and Design. Pearson, 2011 (разделы 4.4, 4.5, 6.2, 15.4)
- Kleppmann M. Distributed Systems (разделы 4.2-4.3)
- van Steen M., Tanenbaum A.S. Distributed Systems: Principles and Paradigms. Pearson, 2017. (раздел 4.4)

Литература (дополнительно)

- Peterson L., Davie B. Computer Networks: A Systems Approach (разделы 4.3, 9.4)
- Сети для самых маленьких. Часть девятая. Мультикаст.