

4. Групповые взаимодействия и рассылка

Сухорослов Олег Викторович

25.09.2023

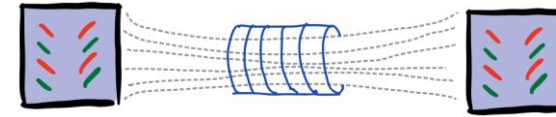
План лекции

- Групповые взаимодействия
- Надежная рассылка сообщений в группе
- Масштабируемые подходы к распространению информации

Взаимодействия: число процессов

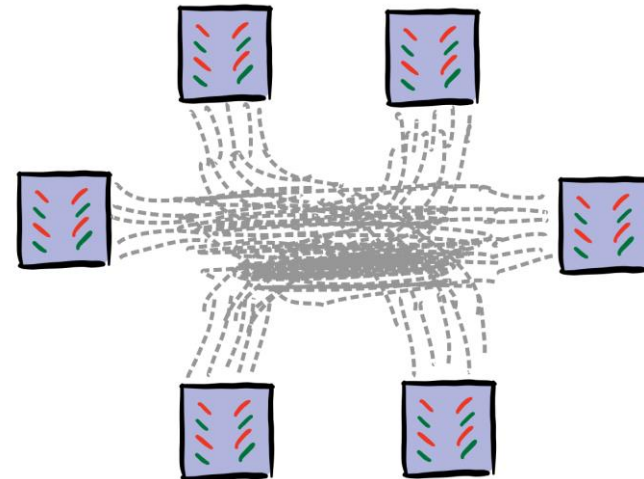
- Парные взаимодействия

- point-to-point, one-to-one
- TCP, RPC, HTTP...



- Групповые взаимодействия (group communication)

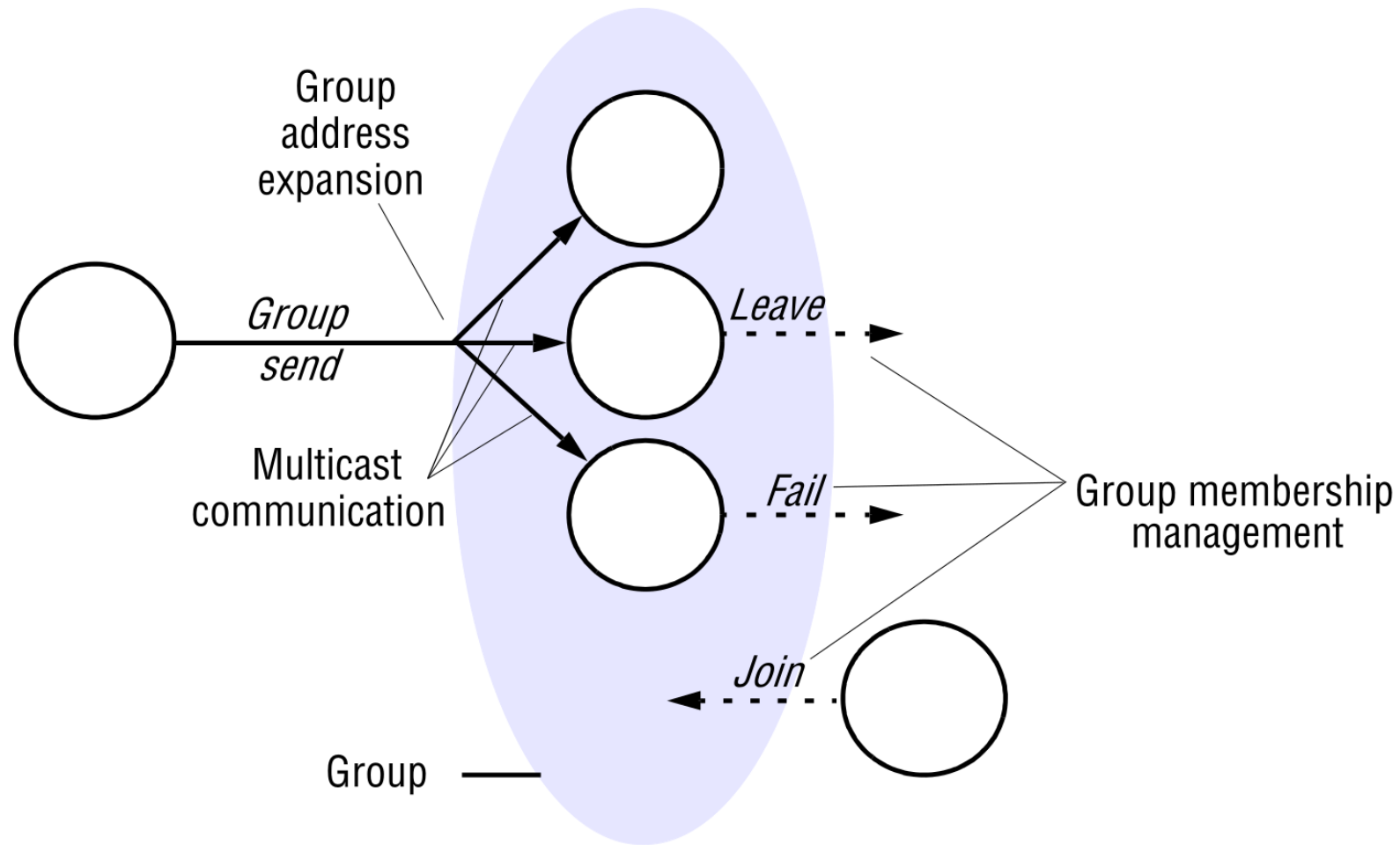
- one-to-many, many-to-many
- ???



Применение групповых взаимодействий

- Рассылка уведомлений о событиях
- Доставка контента и потоковое вещание
- Поиск сервисов и разрешение имен
- Синхронизация времени
- Поиск данных
- Параллельные вычисления
- Репликация сервисов или данных

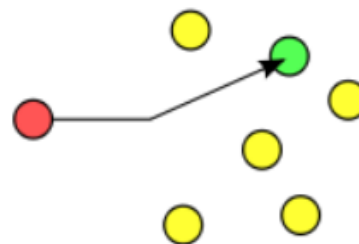
Реализация группового взаимодействия



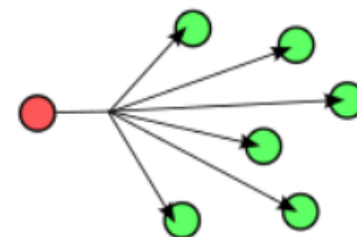
Схемы передачи сообщений

- 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

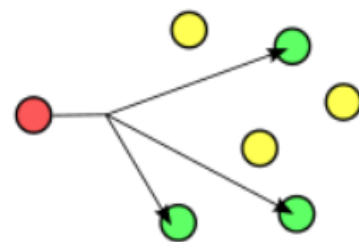
Unicast



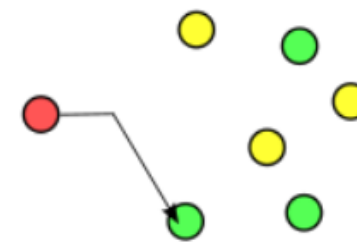
Broadcast



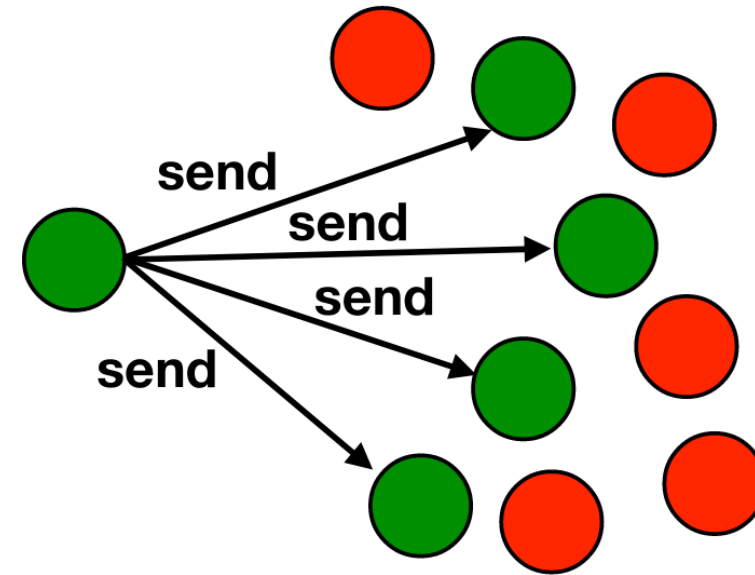
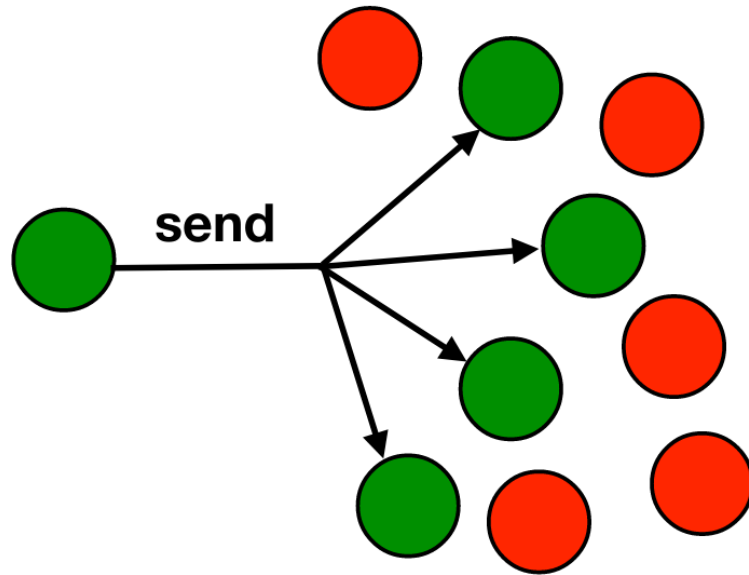
Multicast



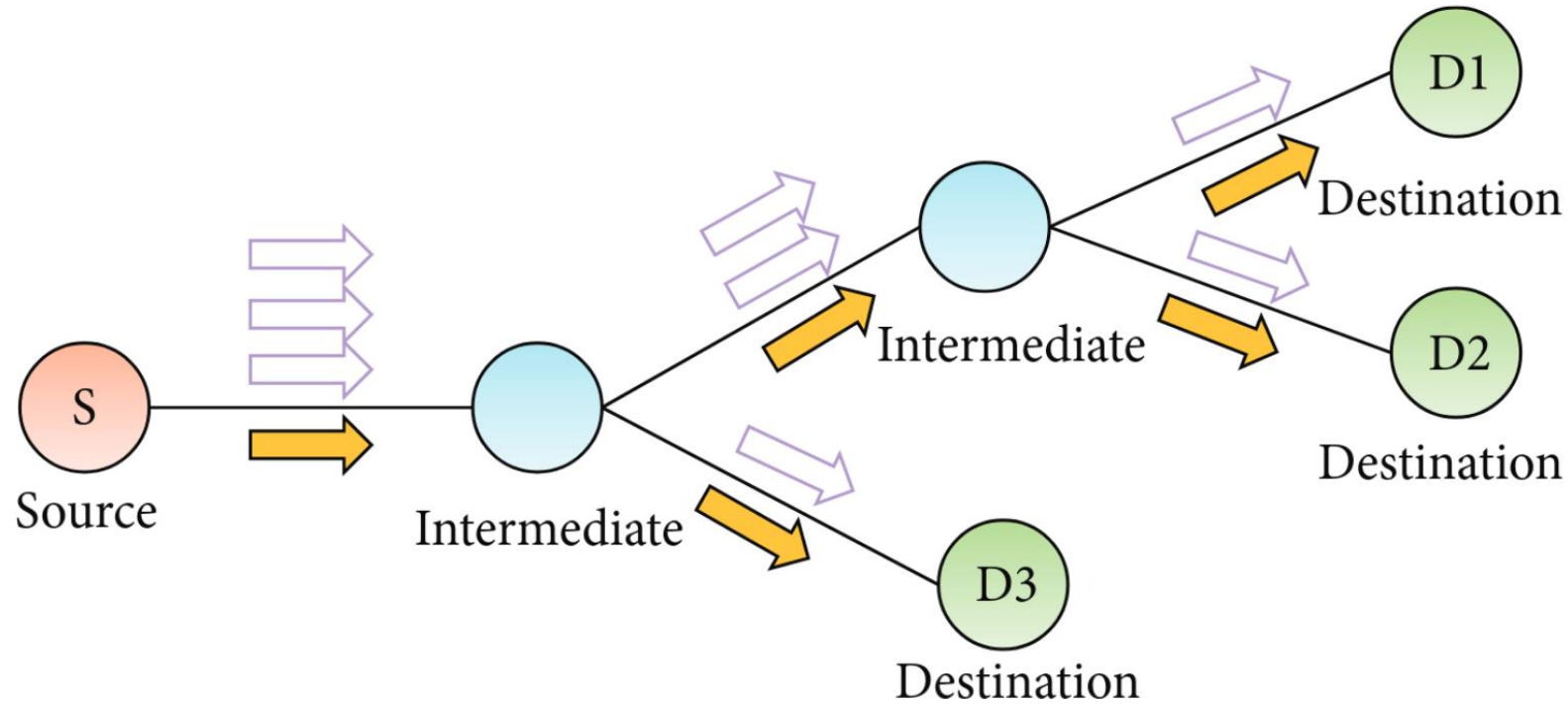
Anycast





Multicast vs Unicast



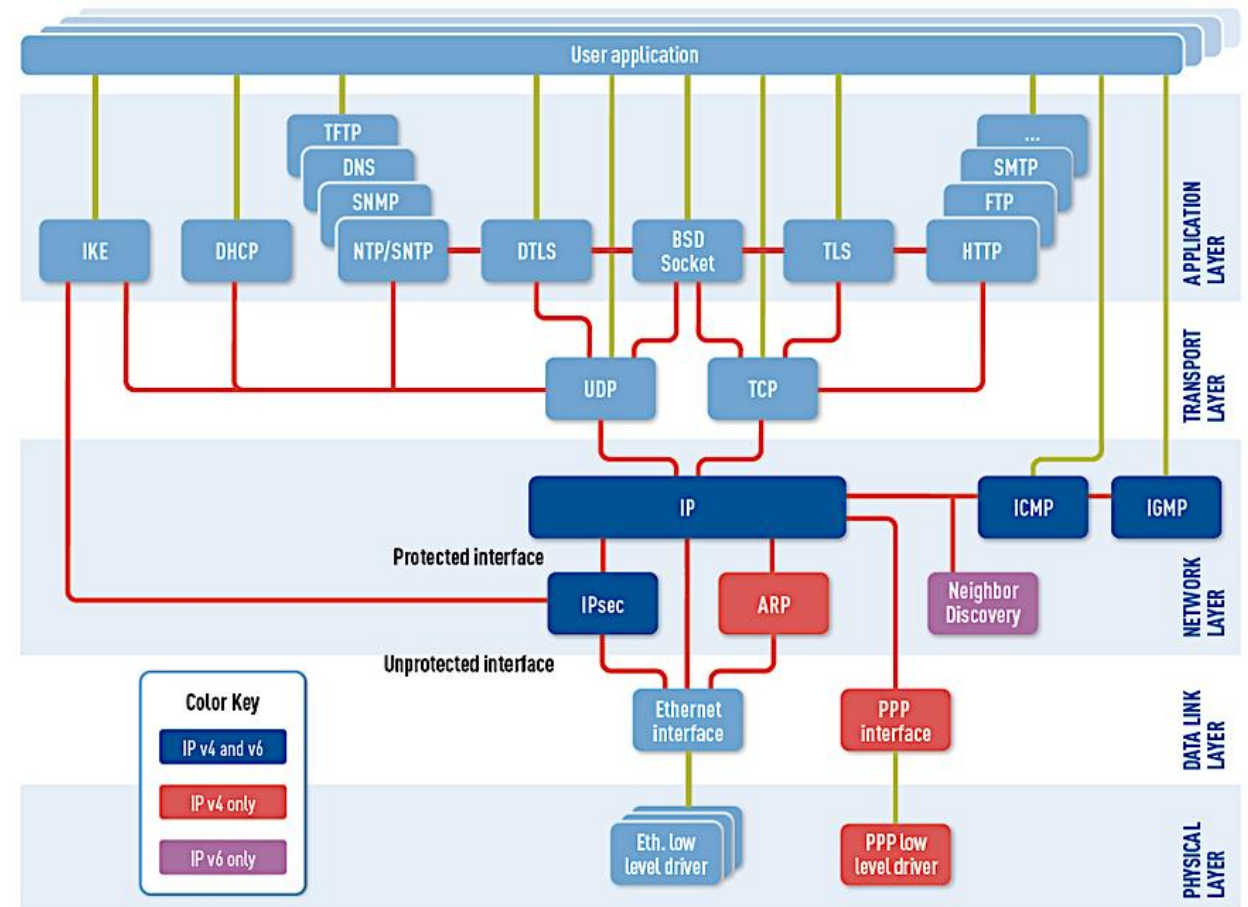
Multicast vs Unicast



-  Sending packets by unicast
-  Sending packets by multicast

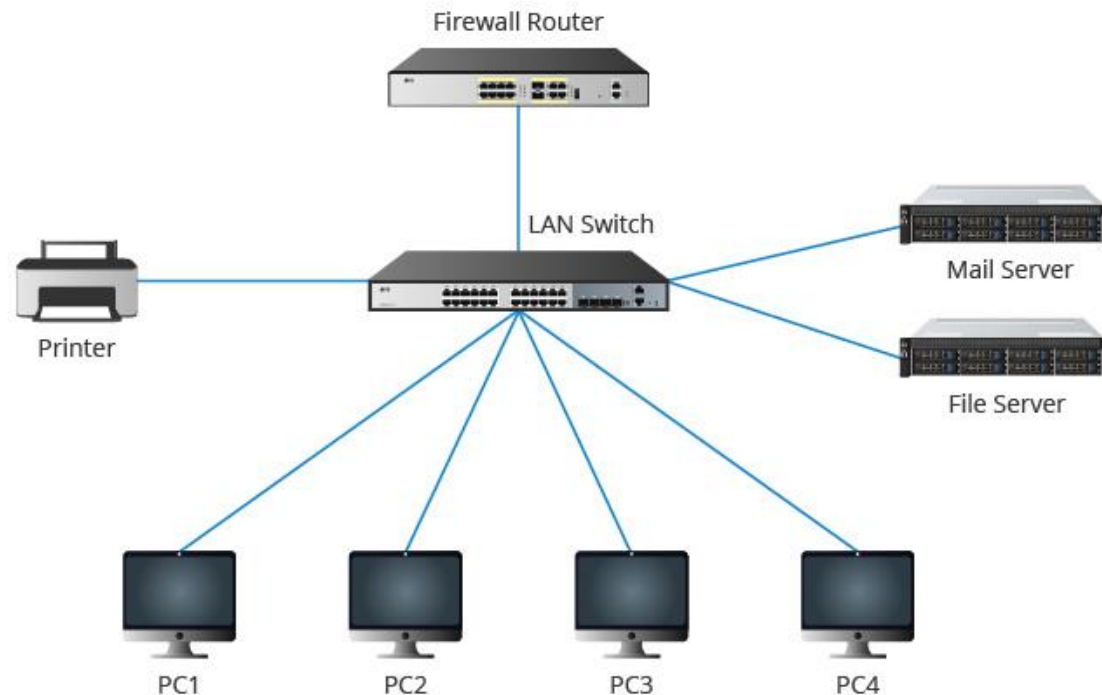
Реализация рассылки

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



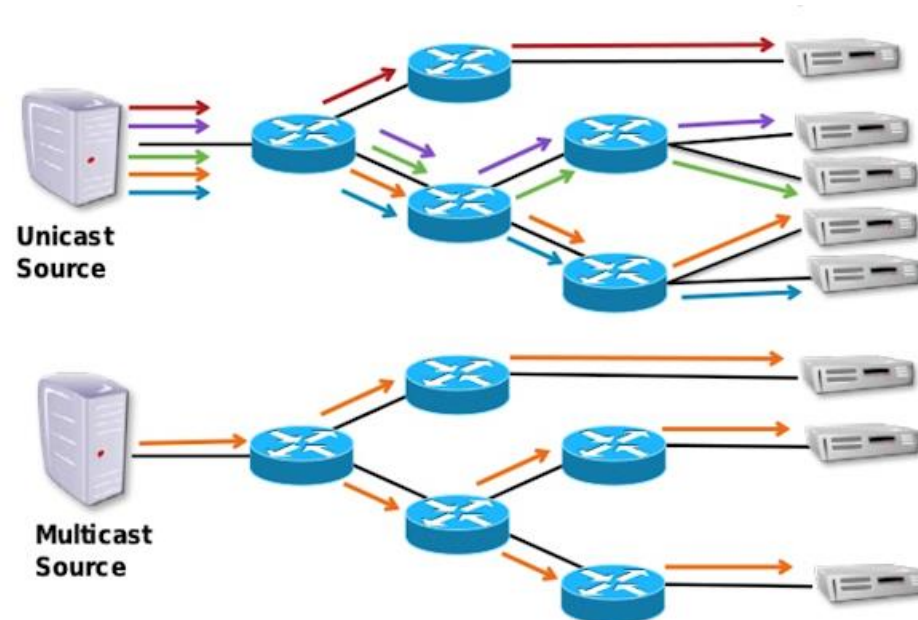
Ethernet

- 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 обычно реализован через протокол UDP

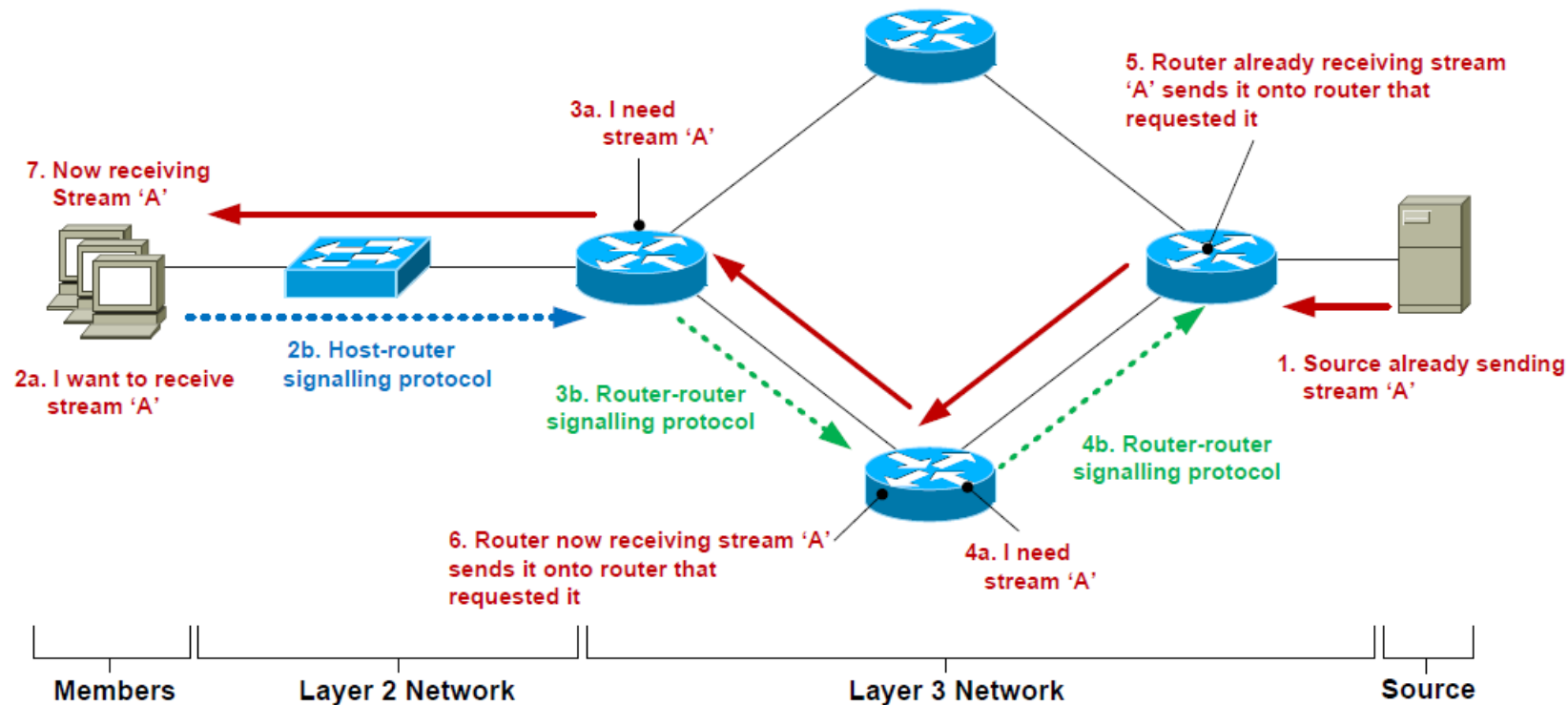


Гарантии IP Multicast

- 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

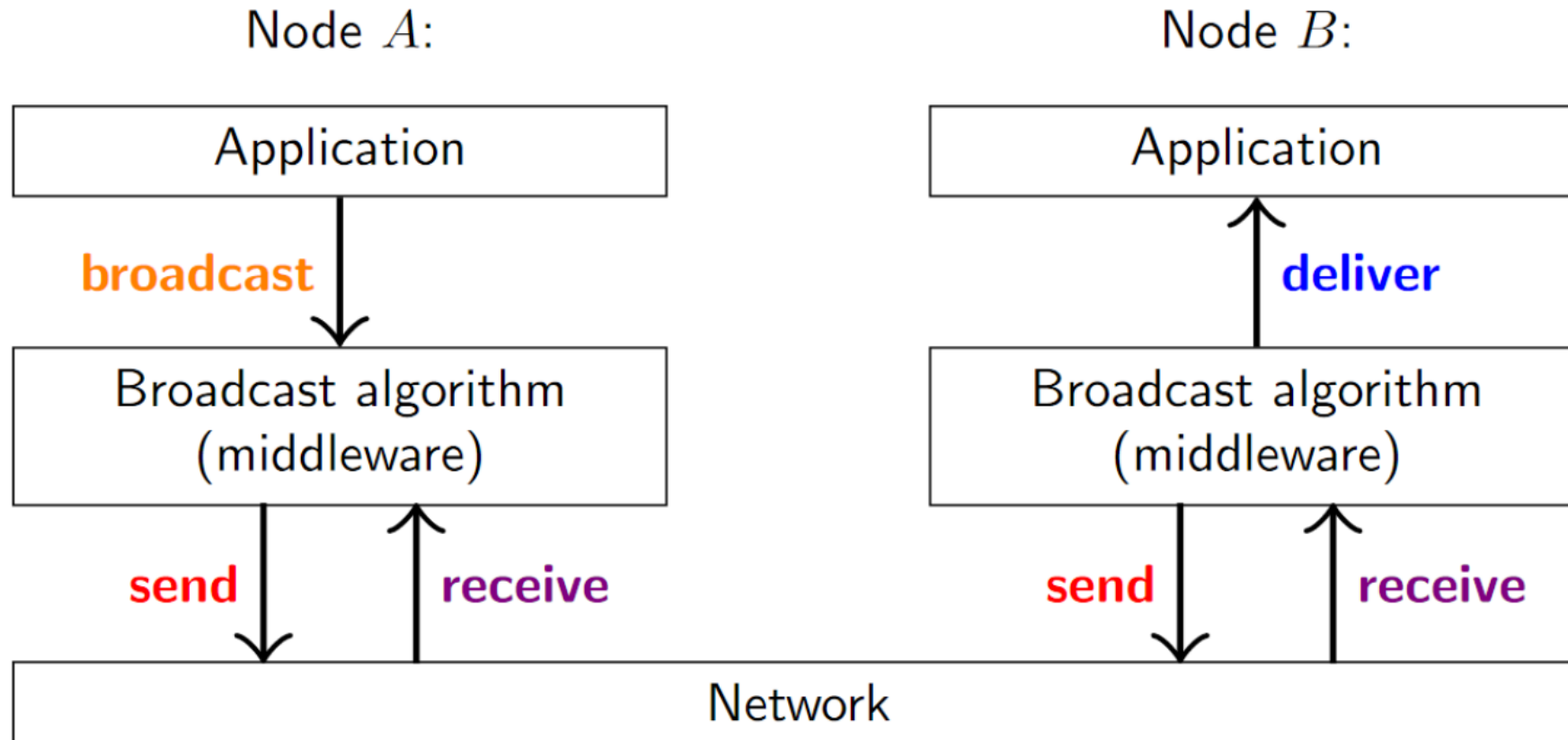
- Требуется поддержка со стороны маршрутизаторов
- Распространение данных контролируется с помощью TTL (time to live)
- Основные протоколы: IGMP, PIM, MSDP



Рассылка на уровне приложения

- Отсутствует поддержка со стороны сети
- Недостаточно предоставляемых возможностей и гарантий
- Важные элементы дизайна
 - Адресация участников группы
 - Надежность и семантика доставки
 - Порядок доставки
 - Семантика ответа
 - Состав и открытость группы

Интерфейс



Предположения

Рассмотрим несколько реализаций рассылки, использующих следующие предположения:

- группа *закрытая*, состав участников группы зафиксирован
- все процессы в группе знают адреса друг друга
- каналы между процессами являются *надежными* (см. далее)
- процессы могут отказывать только путем *полной остановки*
 - отказавший процесс не возвращается в систему
 - процесс, который не отказал в ходе рассылки, будем называть *корректным*

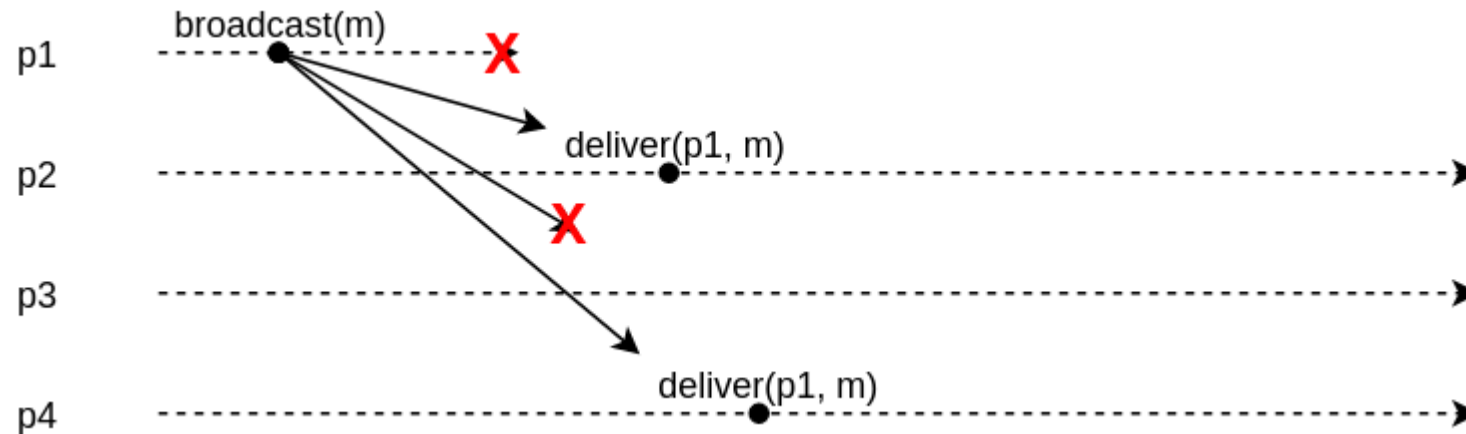
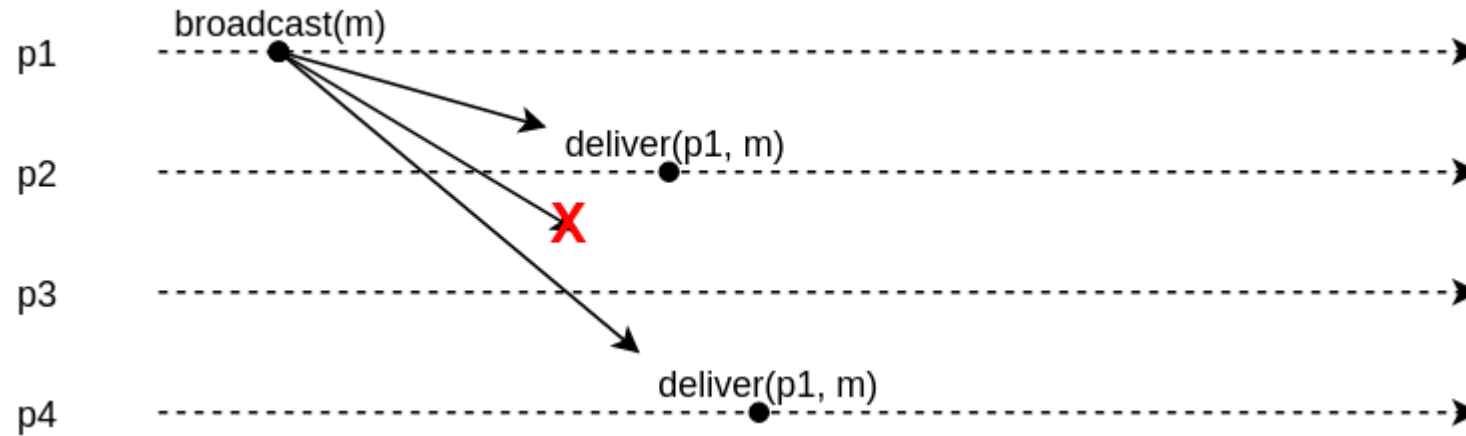
Надежный канал (point-to-point)

- **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

Best-Effort Broadcast (БЕВ): Свойства

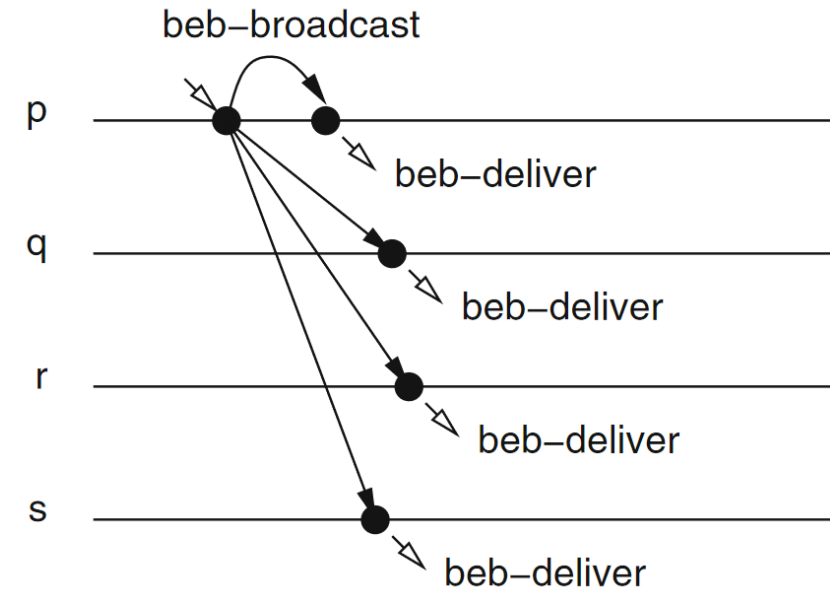
- **Validity:** если корректный процесс разослал сообщение m , то каждый корректный процесс в конце концов доставит m
- **No Duplication:** корректный процесс p доставляет сообщение m не более одного раза
- **No Creation:** если корректный процесс p доставил сообщение m с отправителем s , то m было ранее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

Какие истории удовлетворяют ВЕВ?



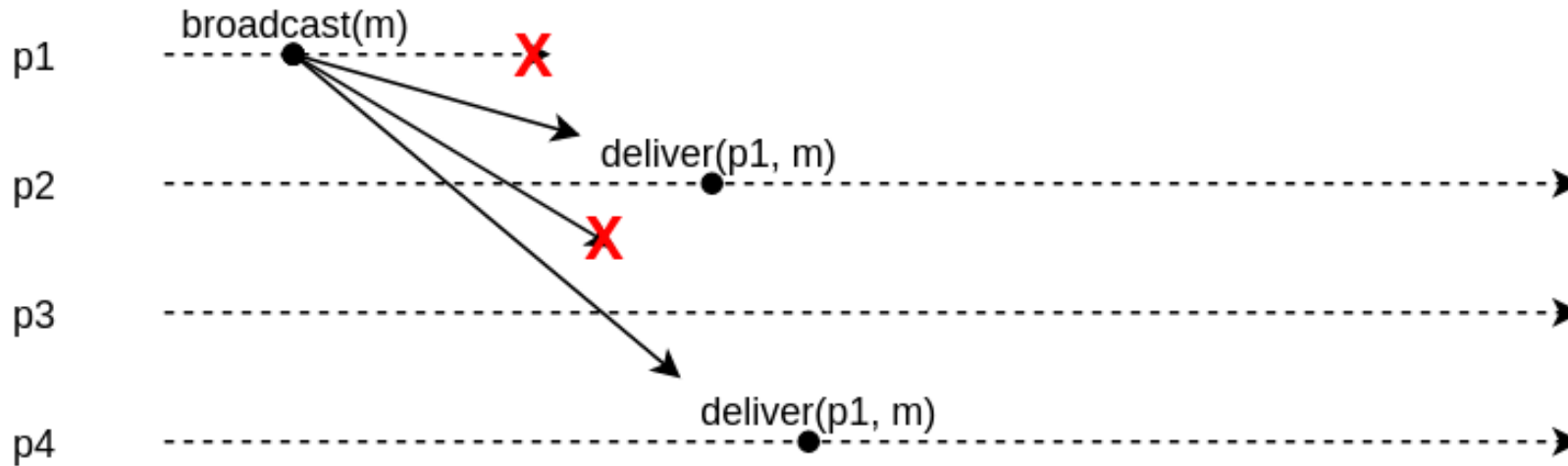
Best-Effort Broadcast: Реализация

```
On broadcast(msg):  
    for proc in group:  
        send(proc, msg)  
  
On receive(msg) at proc:  
    deliver(msg)
```



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

Отказ отправителя

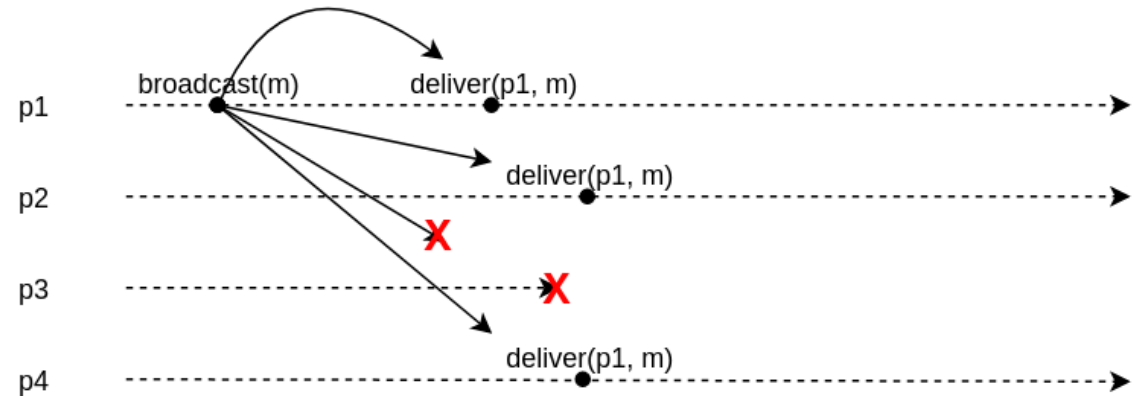
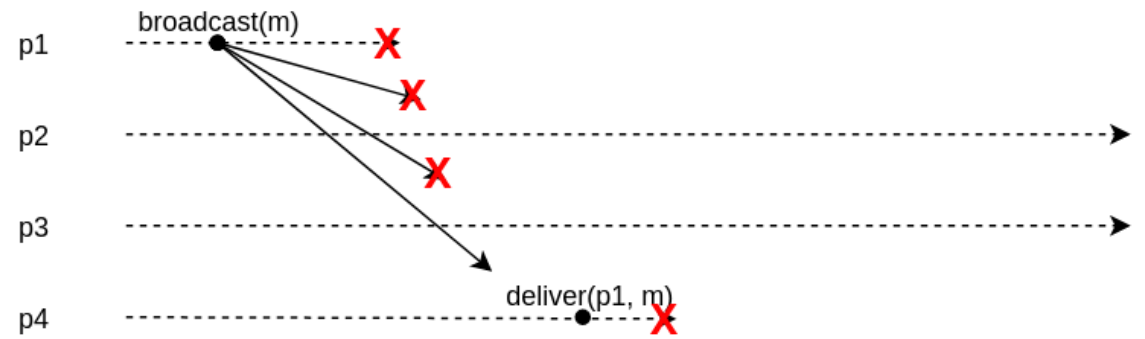
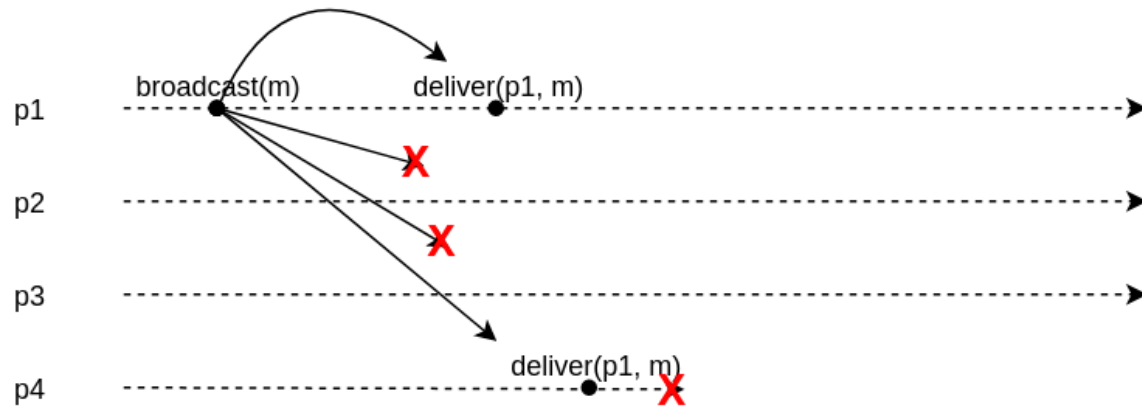
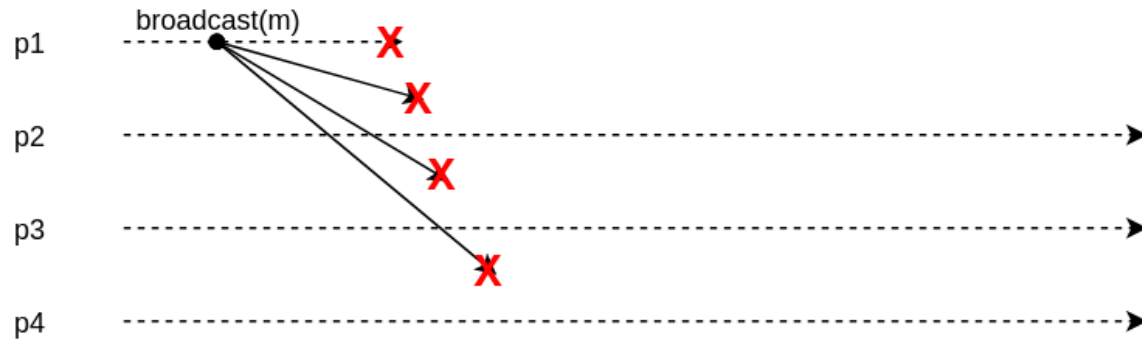


- BEB гарантирует доставку только если отправитель корректный
- При отказе отправителя часть процессов в группе может получить и доставить сообщение, а часть нет
- Отсутствует **согласие** между процессами относительно доставки сообщения

Reliable Broadcast (RB): Свойства

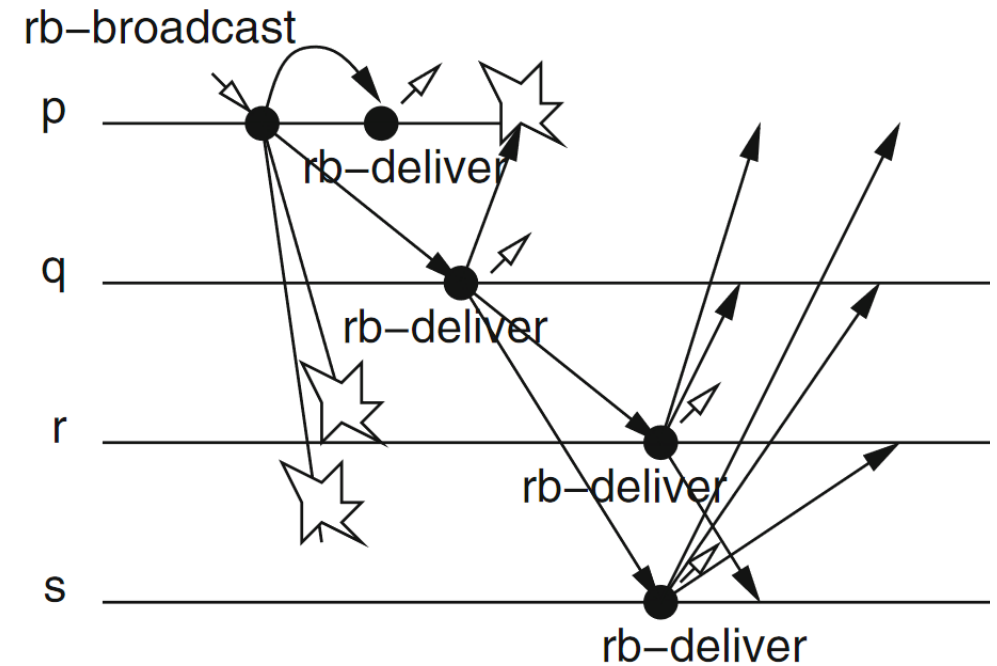
- **No Duplication:** корректный процесс p доставляет сообщение m не более одного раза
- **No Creation:** если корректный процесс p доставил сообщение m с отправителем s , то m было ранее разослано s
- **Validity:** если корректный процесс разослал сообщение m , то он в конце концов доставит m
- **Agreement:**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

Какие истории удовлетворяют RB?



Eager Reliable Broadcast

```
On init:  
    delivered = {}  
  
On broadcast(msg):  
    BEB.broadcast(msg)  
  
On BEB.deliver(msg) :  
    if msg not in delivered:  
        delivered.add(msg)  
        if self != msg.sender:  
            BEB.broadcast(msg)  
            deliver(msg)
```



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

Другие реализации Reliable Broadcast

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

Reliable Broadcast 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 сообщений процесса в группе 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 рассыляется через IP multicast
 - значение S_p^g увеличивается на 1

Reliable Broadcast 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 NAK

Reliable Broadcast поверх IP Multicast (3)

- Особенности
 - требуется постоянная (бесконечная) рассылка сообщений
 - необходимо (вечное) хранение доставленных сообщений на всех процессах
 - попутно получили сохранение порядка сообщений

Uniform Agreement

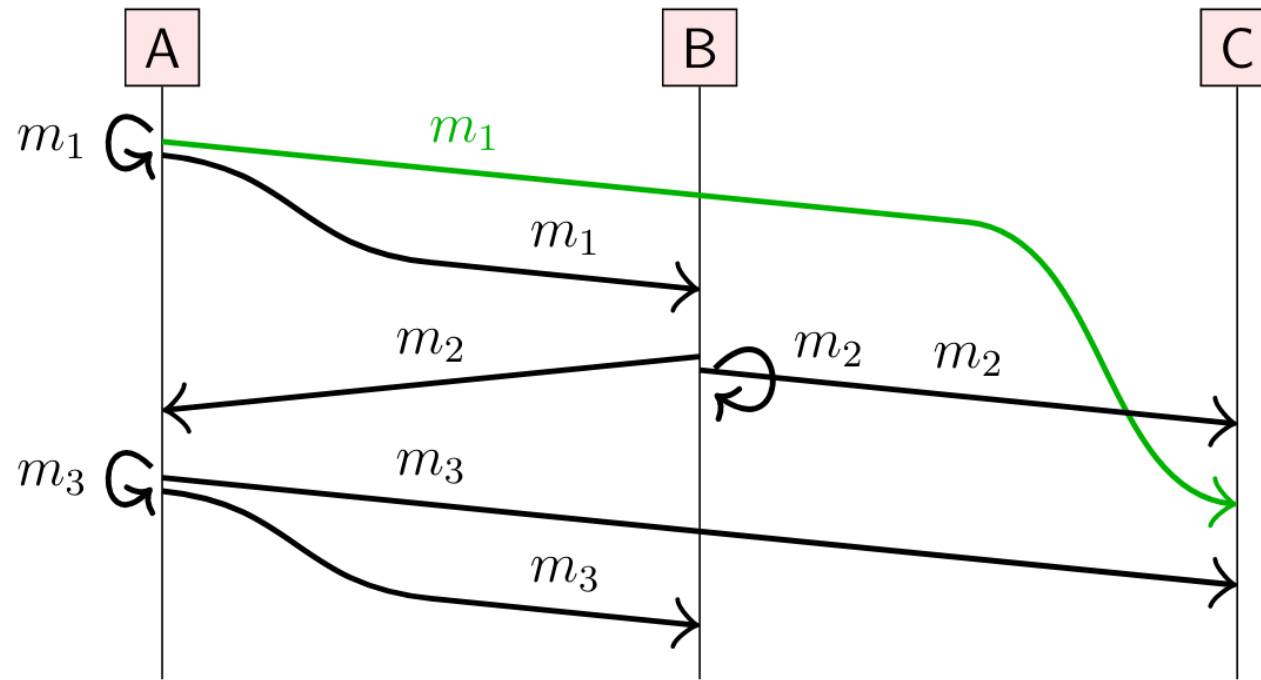
- Расширение свойства Agreement с корректных до всех процессов
 - Если некоторый процесс доставил сообщение m , то все корректные процессы в группе в конце концов доставят m
- Даже если процесс отказал после доставки сообщения, корректные процессы также должны доставить это сообщение
- Удовлетворяют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

Порядок доставки сообщений

- Рассмотренные варианты надежной рассылки не дают гарантий относительно порядка доставки сообщений
- Гарантии порядка учитываются отдельно
- Возможные варианты
 - Произвольный порядок
 - Порядок отправки (FIFO Order)
 - Причинный порядок (Causal Order)
 - Полный порядок (Total Order)

FIFO Order

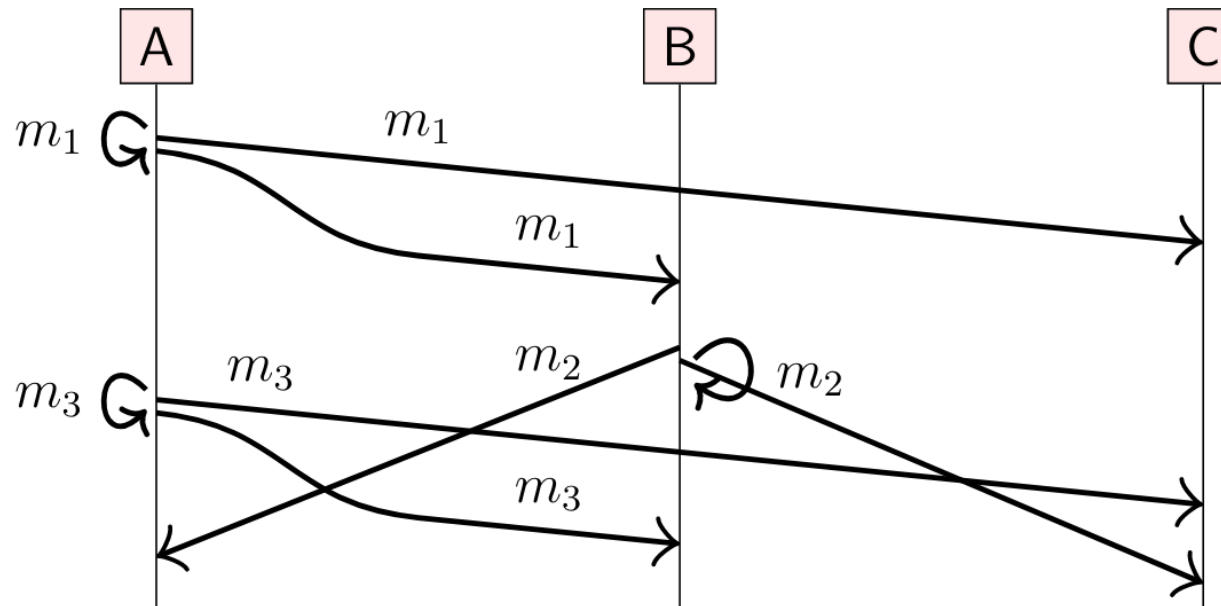
Если корректный процесс сначала разослал m а потом m' , то любой корректный процесс, который доставил m' , доставит m до m'



Causal Order

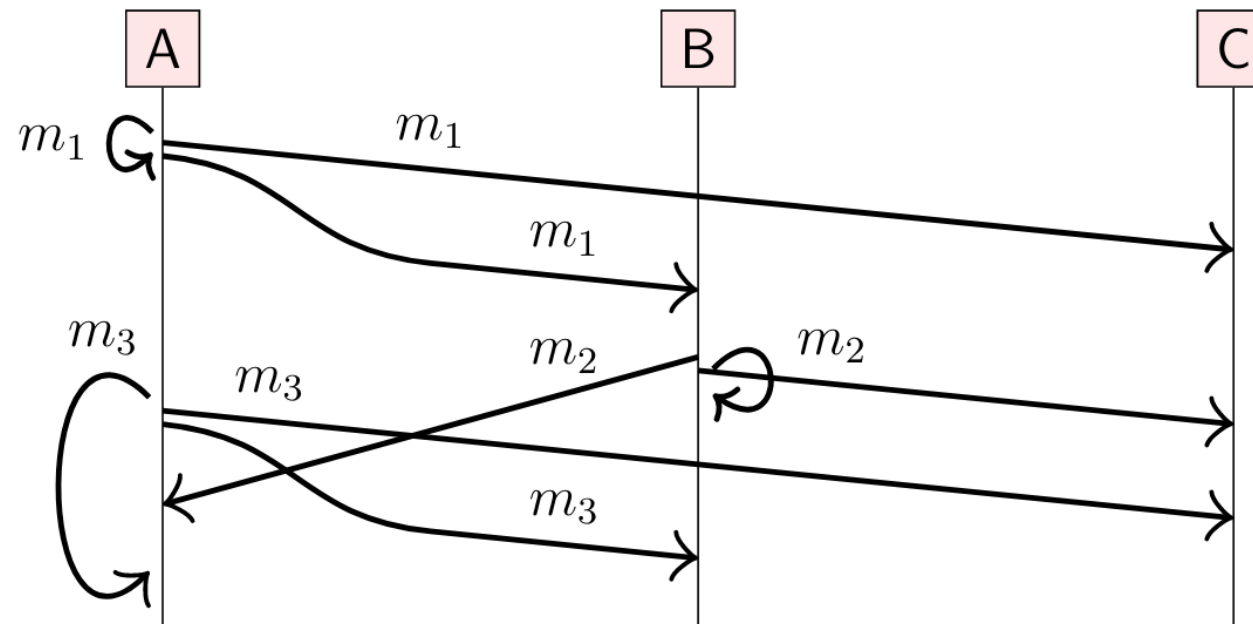
Сообщения доставляются с сохранением причинно-следственных зависимостей

- Сообщение m могло повлиять на сообщение m' от процесса p , если m было отправлено или доставлено процессом p до отправки m'
- Если сообщение m могло повлиять на сообщение m' , то любой корректный процесс, который доставил m' , доставит m до m'



Total Order

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



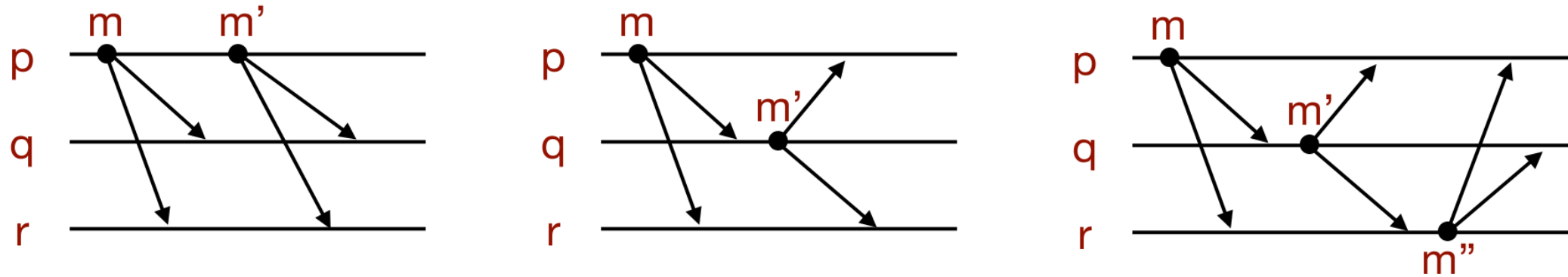
Порядок доставки сообщений

- FIFO Order
 - частичный порядок
- Causal Order
 - 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

Реализация FIFO Broadcast

- Основана на использовании sequence numbers
- Каждый процесс p хранит локально
 - S_p^g - сколько сообщений p отправил в группу g
 - 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 Best-effort Broadcast
 - если использовать Reliable Broadcast, то получим Reliable FIFO Broadcast

Реализация Causal Broadcast



- 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
- Разновидность векторных часов, рассматриваемых далее в курсе
- Для рассылки можно использовать Best-effort или Reliable Broadcast

Реализация Causal Broadcast

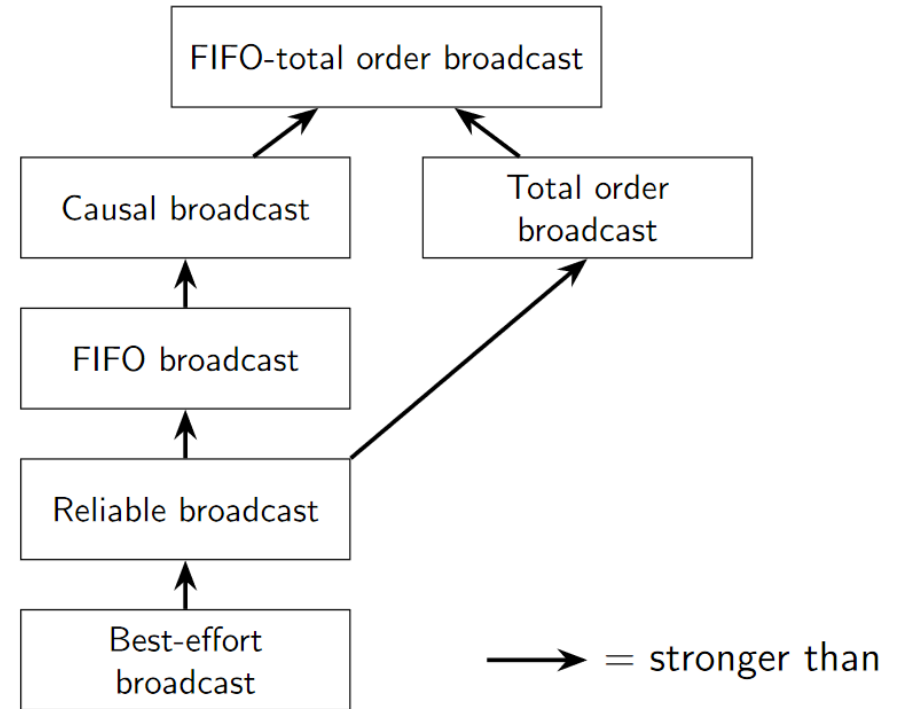
```
On init:  
     $V[j] = 0 \ (j = 1..N)$   
  
On broadcast(msg):  
     $V[self] += 1$   
    BEB.broadcast(<msg, V>)  
  
On BEB.deliver(<msg, V*>) from j:  
    place <V*,msg> in hold-back queue  
    wait until  $V^*[j] = V[j] + 1$  and  $V^*[k] \leq V[k] \ (k \neq j)$   
    remove <V*,msg> from hold-back queue  
    deliver(msg)  
     $V[j] += 1$ 
```

Реализация Total Order Broadcast

- 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 к лекции

Разновидности рассылки

- Best-effort Broadcast
- Reliable Broadcast
- (Reliable) FIFO Broadcast
- (Reliable) Causal Broadcast
- (Reliable) Total Order Broadcast



Reliable Total Order Broadcast (a.k.a. Atomic Broadcast) эквивалентен задаче консенсуса, рассматриваемой позже в курсе

Масштабируемость?

- Ранее акцент был на гарантиях надежности и порядка
- Мы использовали упрощающие предположения
 - отказы только типа полной остановки
 - все процессы знают друг друга
 - состав групп зафиксирован
- Как обеспечить масштабируемость, если
 - участников много
 - они могут находиться в разных частях Интернета и не знать друг о друге
 - классические подходы не работают или создают большую нагрузку на сеть

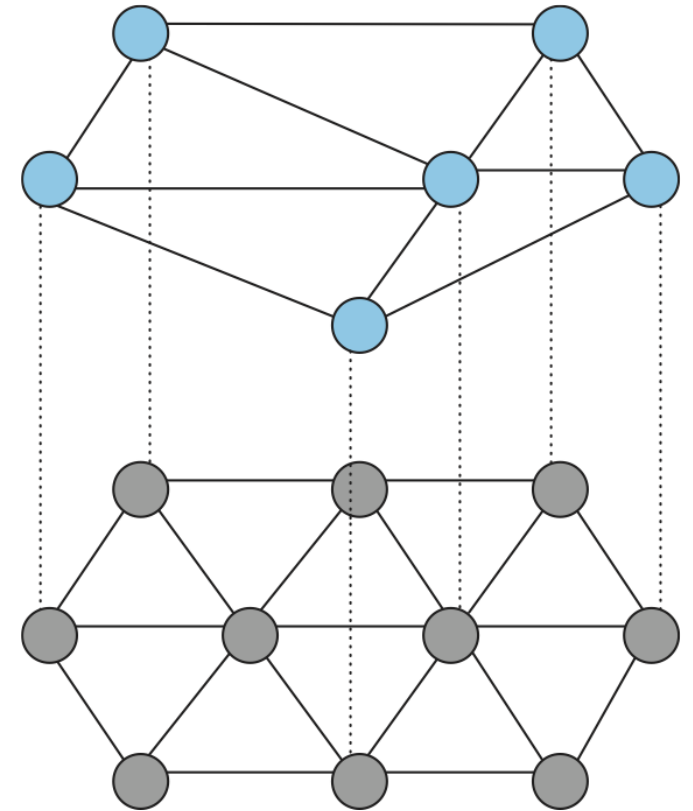
Масштабируемые подходы

- Топологии на базе оверлейных сетей
 - рассылки по дереву или mesh-сети
- Распространение информации с вероятностными гарантиями
 - probabilistic broadcast, epidemic protocols, gossiping...

Ослабленные гарантии надежности и порядка взамен на лучшую масштабируемость и устойчивость

Оверлейная сеть (overlay network)

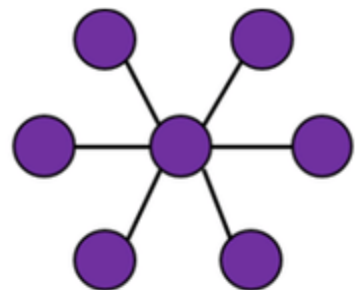
- "Виртуальная" сеть поверх физической сети
- Реализует набор сервисов
 - специфичных для приложения
 - более эффективных, чем доступные в обычной сети
 - недоступных в обычной сети
- Основные элементы
 - Топология
 - Адресация узлов
 - Протоколы
 - Алгоритмы маршрутизации



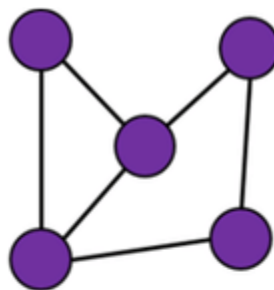
Применение оверлейных сетей

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

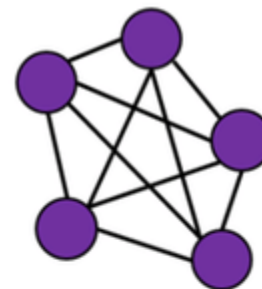
Возможные топологии



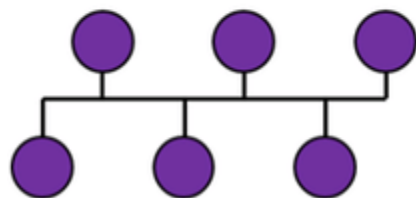
Star



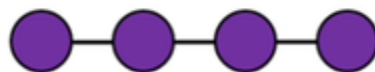
Mesh
(partially connected)



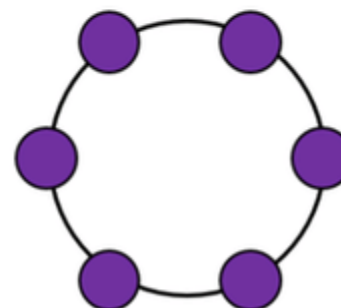
Mesh
(fully connected)



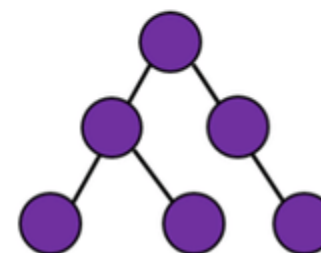
Bus



Linear

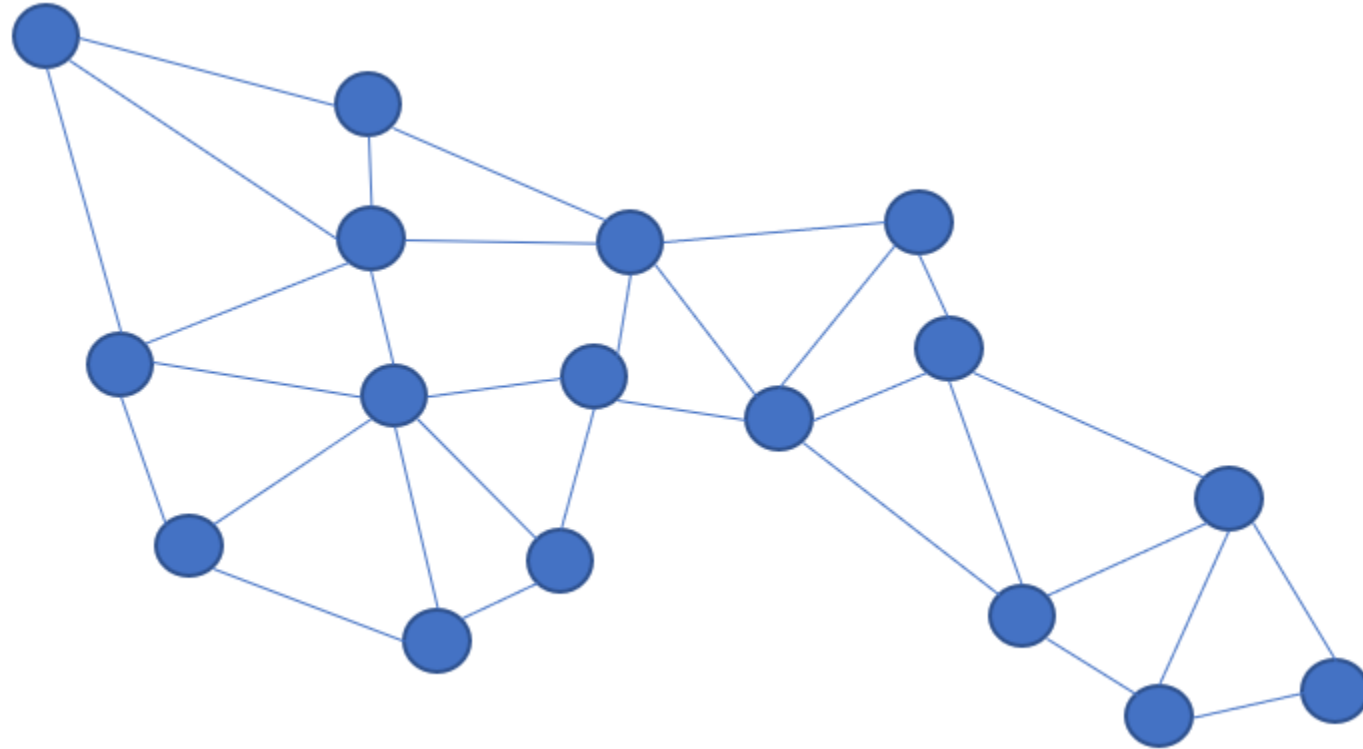


Ring



Tree

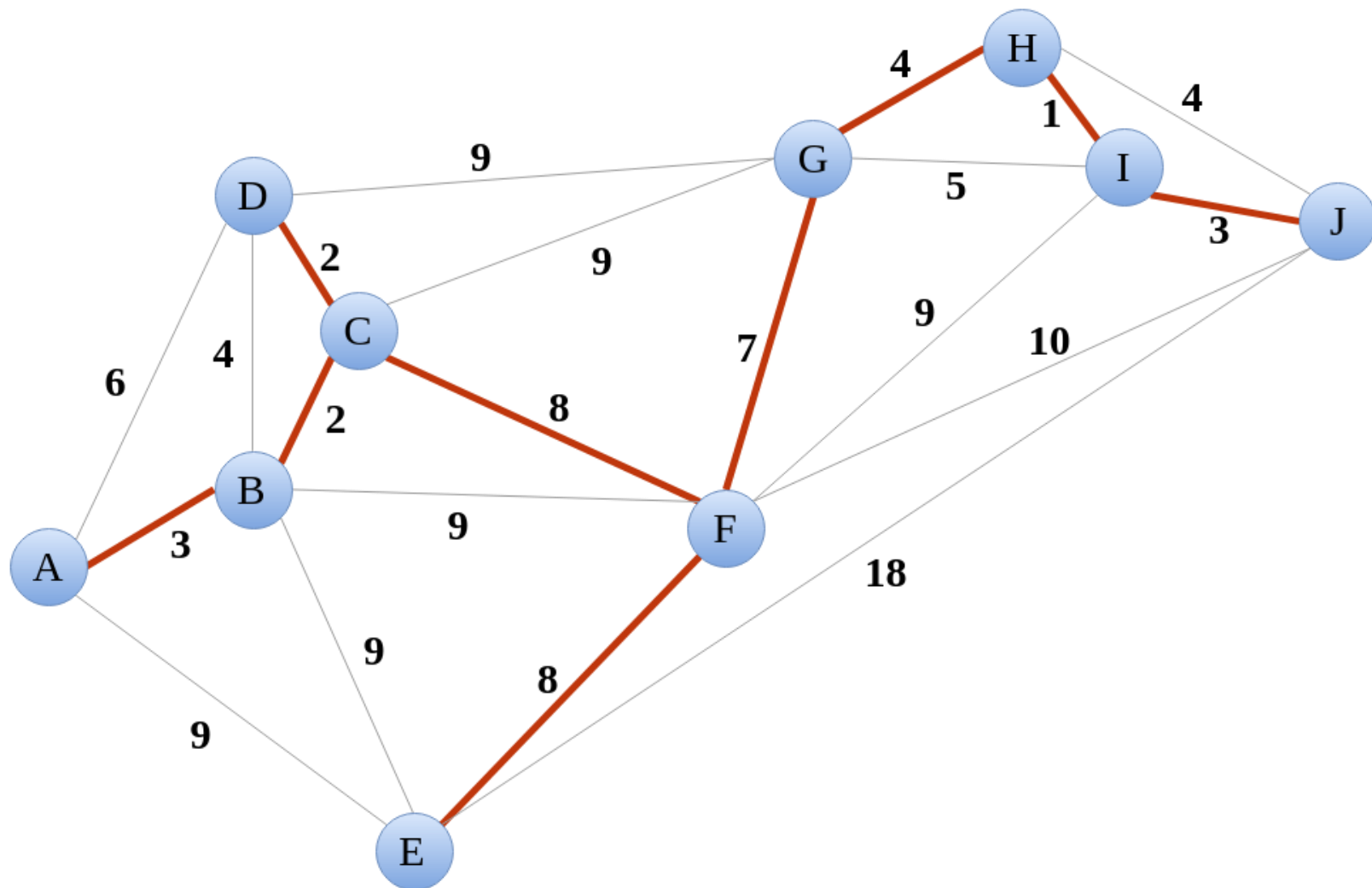
Mesh-сеть



Mesh-сеть: Особенности

- Высокая устойчивость к отказам за счёт избыточных связей
- Хорошо приспособлена к динамическому составу участников
- Сложнее организовать эффективную рассылку
- Может требоваться буферизация полученных данных (pull)

Дерево



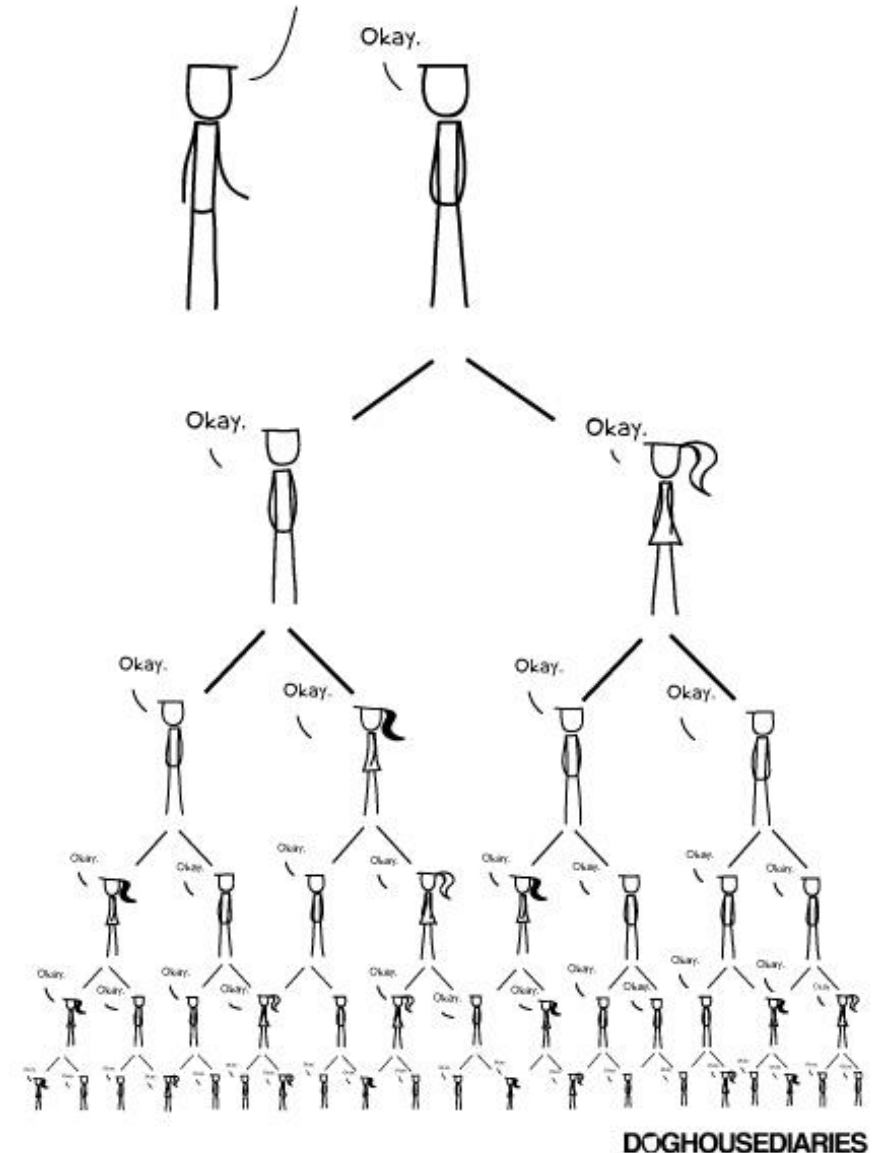
Дерево: Особенности

- 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 реконфигурация дерева
- Починка дерева в случае отказа
- Примеры: [Plumtree](#) (НИС), switch-trees (литература)

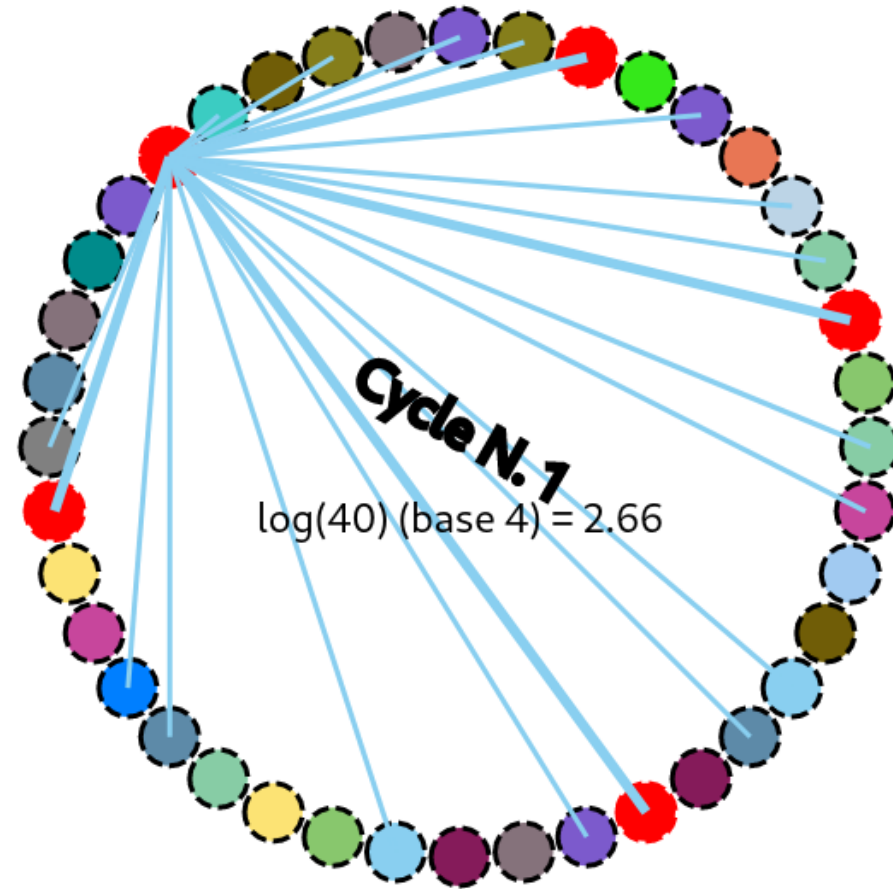
Gossip

- 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
- Для распространения данных на все узлы требуется $O(\log N)$ раундов

Just PROMISE not to tell anyone else.



Симулятор

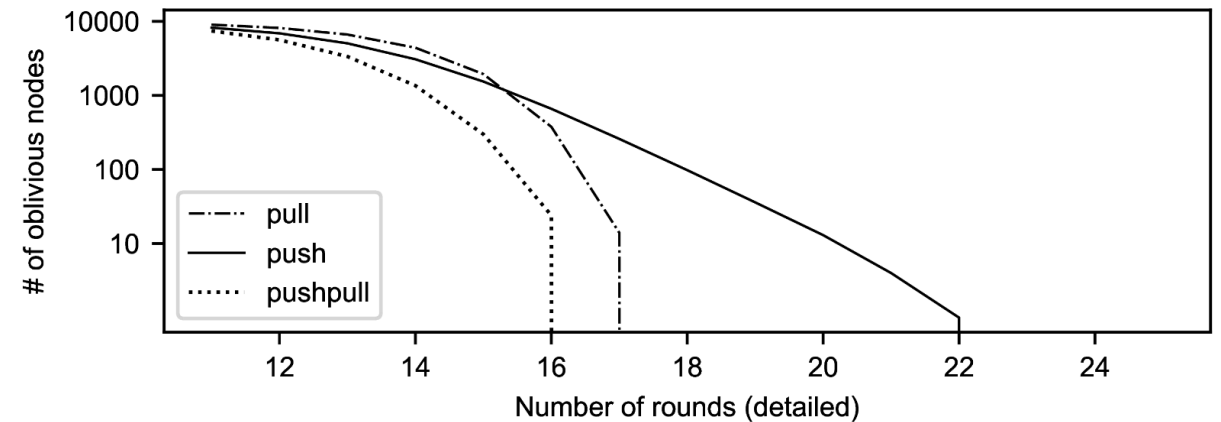
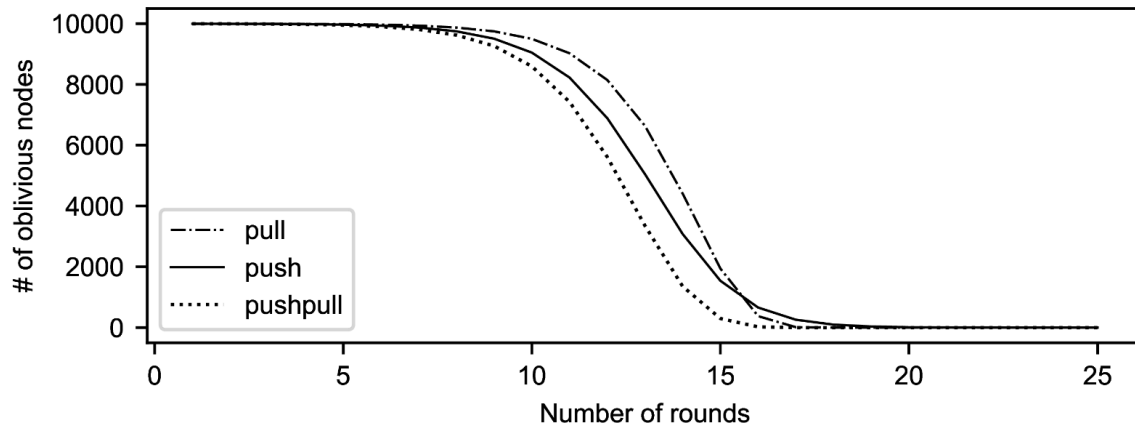


<https://flopezluis.github.io/gossip-simulator/>

Анти-энтропия

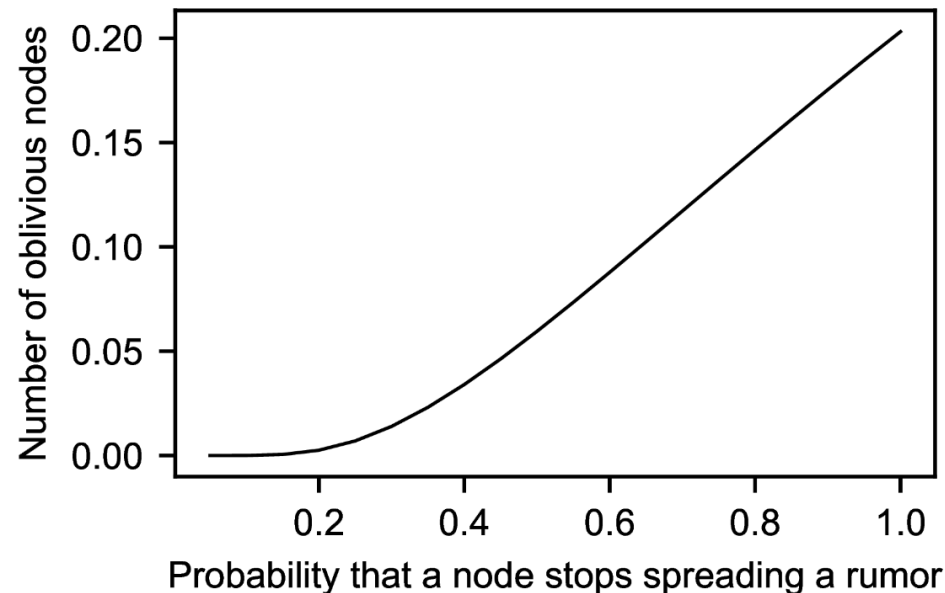
Узел p выбирает случайным образом другой узел q

- **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



Материалы

- [Distributed Systems: Concepts and Design](#) (разделы 4.4, 4.5, 6.2, 15.4)
- [Сети для самых маленьких](#) (часть 9)
- [Computer Networks: A Systems Approach](#) (разделы 4.3, 9.4)
- [Distributed Systems Course](#) (разделы 4.2-4.3)
- [Distributed Systems: Principles and Paradigms](#) (раздел 4.4)
- [Epidemic Broadcast Trees](#)