Распределённые системы. Семинары.

Бабичев С. Л., Коньков К. А.

Москва, 2008-2022

Алгоритмы выборов

Алгоритм забияки: требования

- Каждый процесс имеет свой авторитет `N`.
- Каждый процесс знает авторитеты других процессов
- Авторитеты всех процессов различны
- Используется синхронная надёжная связь
- При исполнения алгоритма процессы могут исчезать и появляться

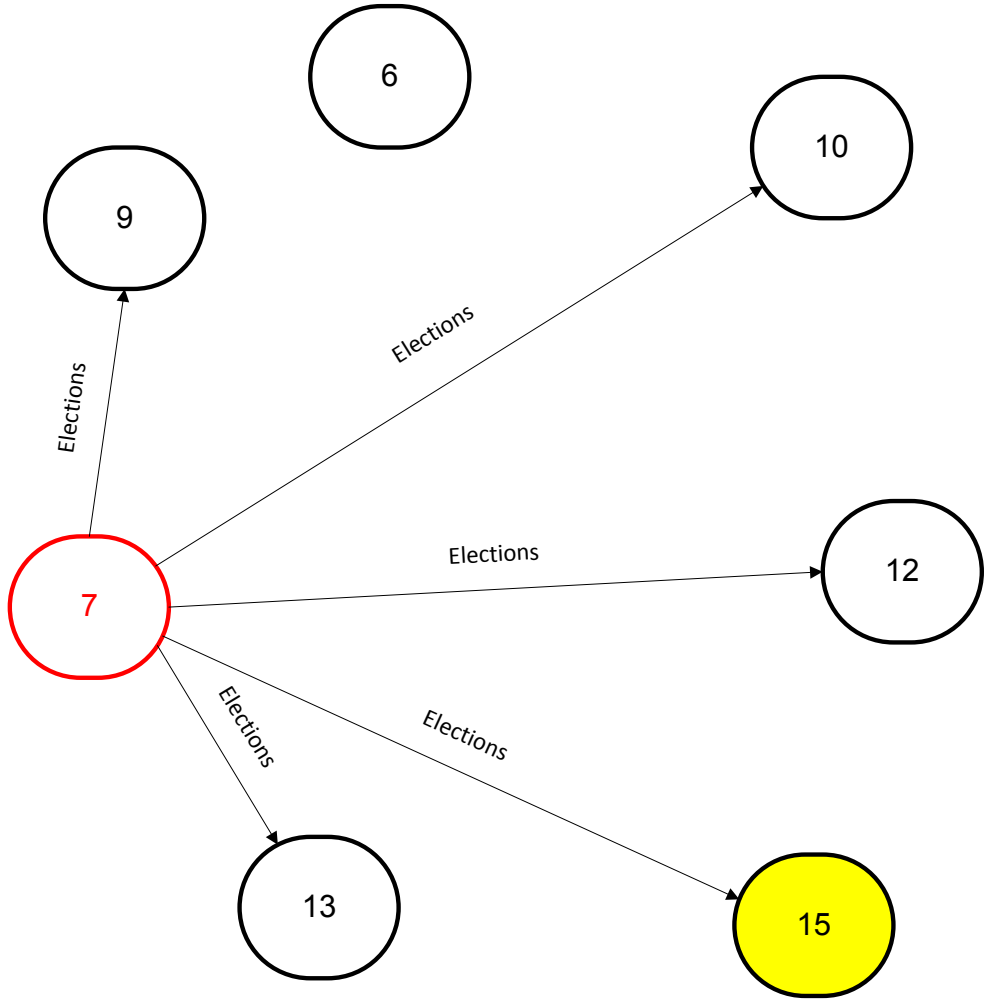
Алгоритмы выборов

Алгоритм забияки

1. P рассылает всем Q таким, что N(Q) > N(P) сообщение Elections.
2. Если все молчат, то P выигрывает.
3. Если P получает ответ от Q, что он жив, то ожидает определённое время получения сообщения о победе кого-либо в выборах. Если не дожидается, перезапускает процесс выборов.
4. Если P получает сообщение о выборах от процесса с меньшим N, то P отвечает сообщением Alive и сам запускает выборы.
5. Тот, кто считает себя победителем, рассылает сообщение Coordinator
6. Если процесс P получает сообщение Coordinator от процесса с меньшим N, процесс P инициирует выборы

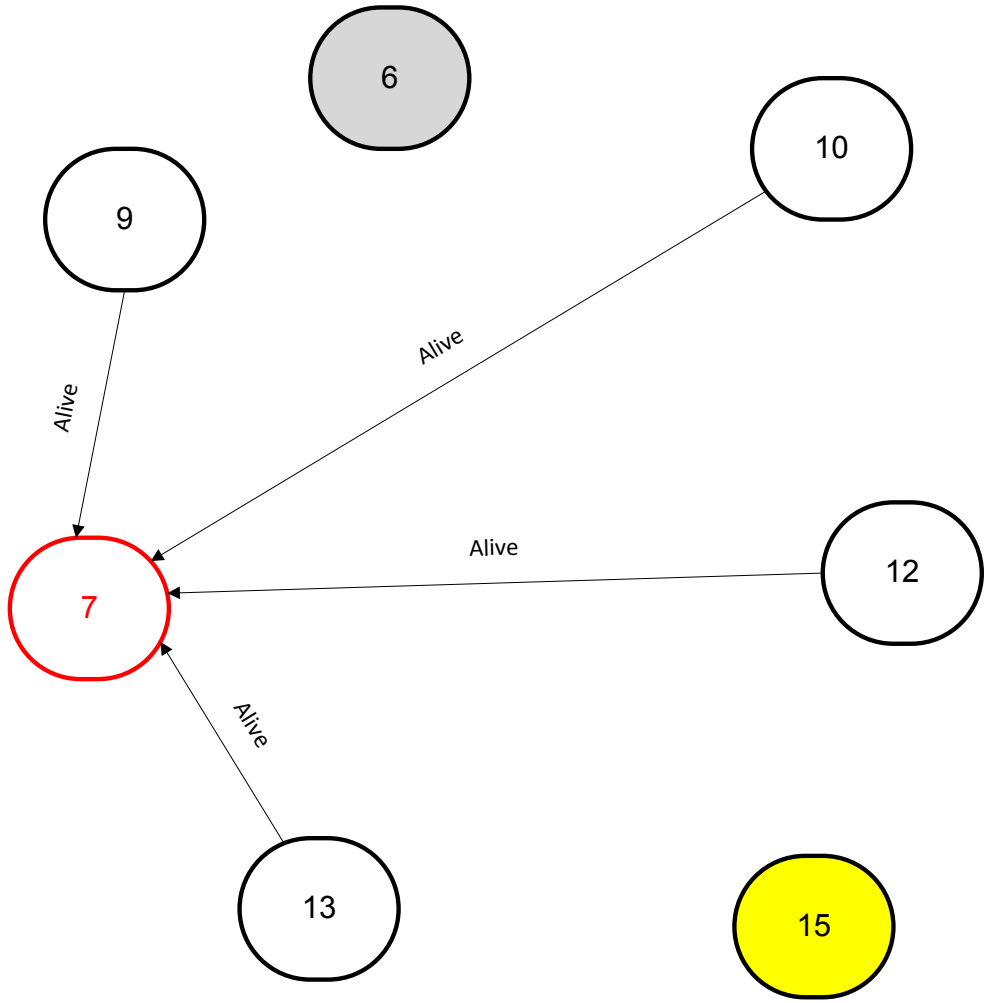
Алгоритм забияки

Процесс 7 инициирует выборы. Процесс 15 недоступен. Процессу 6 посылать ничего не надо.



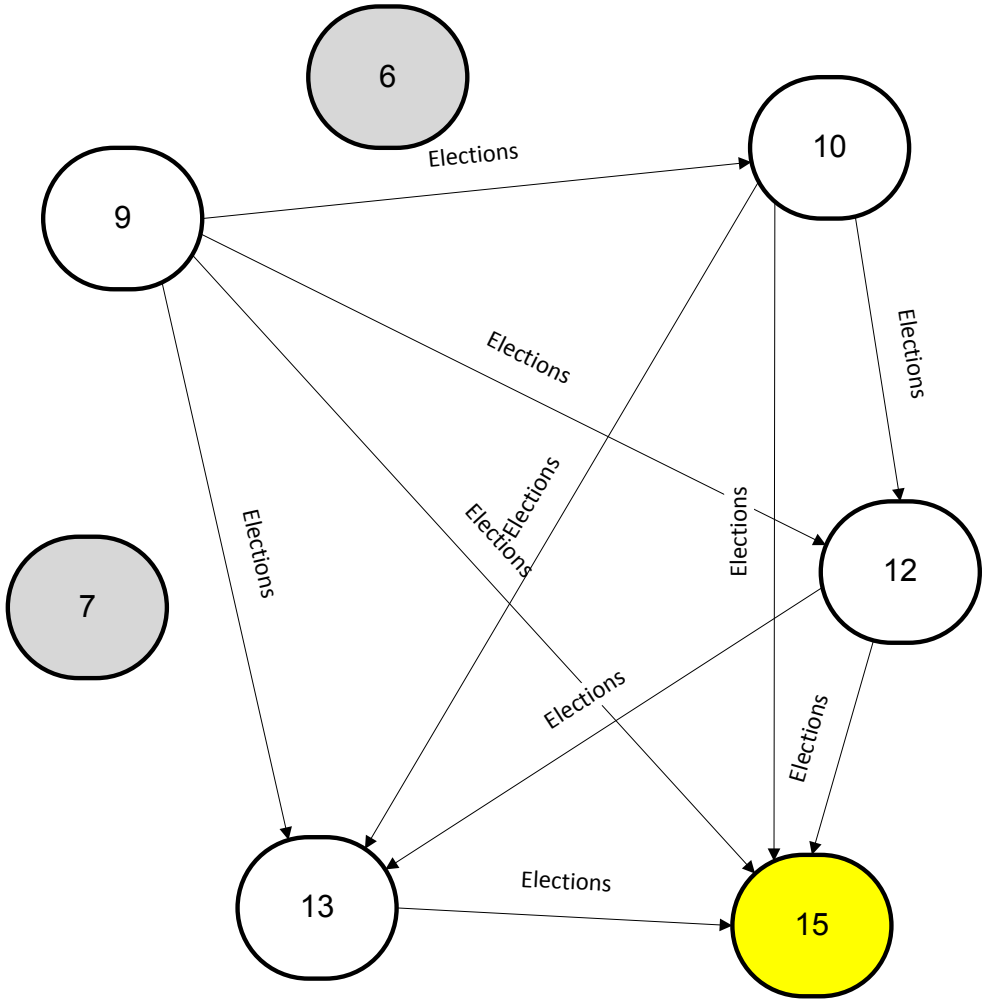
Алгоритм забияки

От процессов 9,10,12,13 приходят ответы. От 15 — молчание.



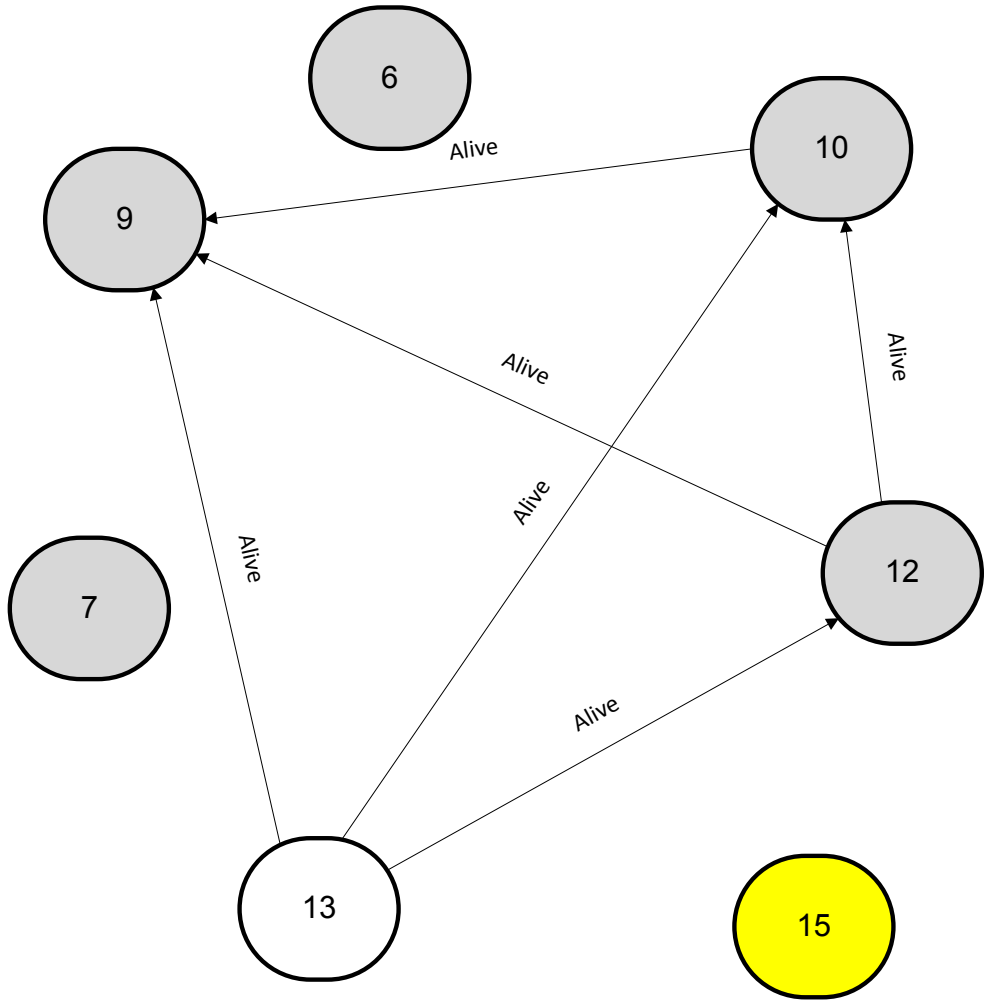
Алгоритм забияки

Процессы 9,10,12,13 инициируют выборы. 6 и 7 самоустранились.



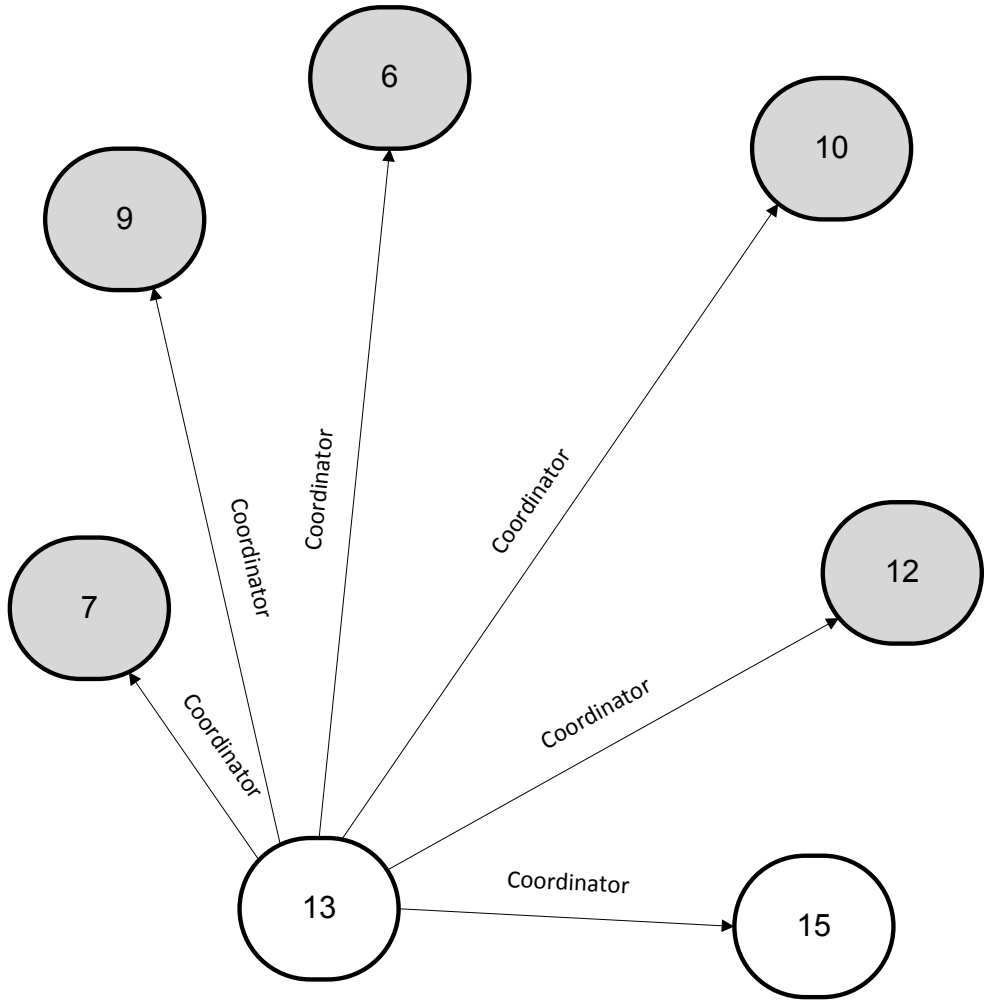
Алгоритм забияки

Процессы 10, 12 и 13 посылают ответы. 9, 10 и 12 самоустраняются.



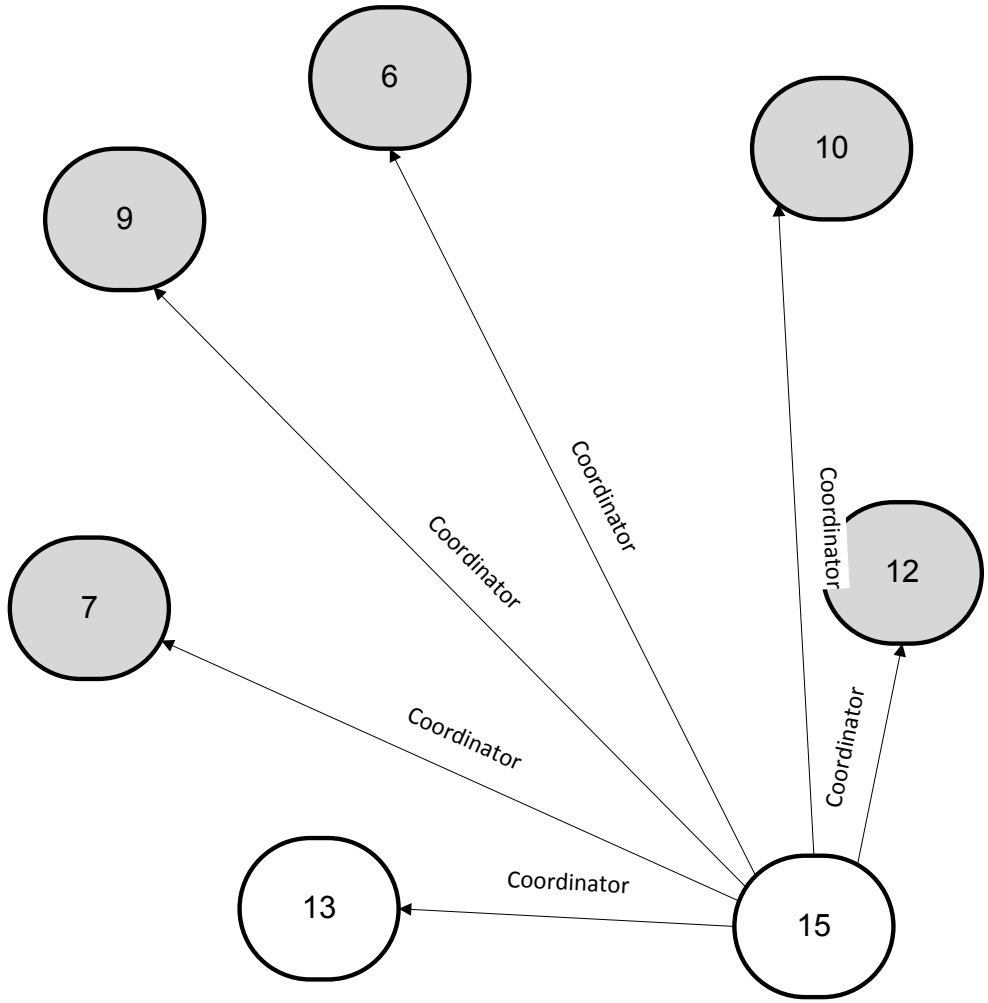
Алгоритм забияки

Процесс 13 рассылает сообщение `Coordinator`. Процесс 15 стал доступен.



Алгоритм забияки

Процессу 15 некому слать `Elections`. Он рассылает `Coordinator`. Победа в выборах за ним.



Кольцевой алгоритм

- Кольцевой алгоритм (Чанга-Робертса) требует, чтобы процессы были связаны в однонаправленное кольцо коммуникационными каналами.
- У каждого процесса, как и в алгоритме забияки, имеется уникальная характеристика $\mathtt{N}$, для которой определена операция сравнения.
- Каждый процесс в алгоритме может считать или не считать себя избирателем.
- В начале алгоритма все процессы считают себя не избирателями ($P_i.Elector = false$).

Кольцевой алгоритм — первый этап

Первый этап:

1. Процесс `P` начинает выборы. Он передаёт сообщение (`M`), содержащее его идентификатор (`M.ID`), авторитет (`M.N`) и флаг отсутствия координатора (`M.Coordinator = null`) соседу
2. Процесс, который передал сообщение считает себя избирателем. `P.Elector = true`
3. Как только процесс `Q` получает сообщение, он сравнивает авторитет процесса, содержащийся в сообщении (`M.N`) со своим.
4. Если `M.N > Q.N`, то `Q` отправляет сообщение дальше в том же виде, как оно пришло.
5. Если `M.N < Q.N` и `Q.Election = false`, то `M.N = Q.N` и `M.ID = Q`. Сообщение отправляется далее.
6. Если `M.N < Q.N` и `Q.Election = true`, то сообщение уничтожается.
7. Если `M.N = Q.N`, то `Q.Coordinator = true`

Кольцевой алгоритм — второй этап

Второй этап:

1. Процесс-координатор (`Q.Coordinator = true`) помечает себя как не избирателя (`Q.Elector = false`), устанавливает `M.Coordinator = Q` и посылает сообщение `M` соседу.
2. Процесс `P`, принимающий сообщение с `M.Coordinator != null`, сохраняет идентификатор координатора (`P.Coordinator = Q.Coordinator`) и посылает сообщение далее.
3. Если сообщение принимает процесс `Q = M.Coordinator`, то сообщение уничтожается и выборы завершены.

Взаимное исключение

- Централизованный алгоритм
- Распределённый алгоритм
- Алгоритм маркерного кольца

Взаимное исключение: централизованный алгоритм

- В централизованном алгоритме имеется некий арбитр, который принимает решение о предоставлении ресурса.
- Арбитр может быть избран голосованием. Процессы выбирают локального лидера группы.
- Арбитр избран. Он должен завести локальный примитив синхронизации, который ограждает требуемый ресурс от одновременного обращения.

Взаимное исключение: распределённый алгоритм

Распределённый алгоритм не требует выбора лидера.

1. P собирается войти в критическую секцию.
2. P создаёт сообщение с именем секции и рассылает его всем заинтересованным (группе процессов).
3. Q получает сообщение.
4. Если Q не находится в KO и не собирается туда заходить, он посылает OK.
5. Если Q находится в KO, то он не отвечает, а помещает запрос в очередь.
6. Если Q собирается войти в KO, но ещё не вошёл, он сравнивает метки логического времени. Минимальная побеждает. Если принятая $<$ номер, send OK.
7. Если собственная $<$ номер, то получатель ставит запрос в очередь, ничего не посылая.
8. Послав сообщение, P останавливается и ждёт разрешения. После получения всех OK он входит в область. После покидания области P посылает OK всем процессам в их очереди и удаляет такие сообщения из своей очереди.