1. Сервер отправляет запрос на камеру и проверяет доступна ли она в данный момент.
2. Как только камера говорит, что она работает, сервер отправляет сообщение о готовности клиенту.
3. Клиент отправляет своих ICE candidates (ip address, port, protocol, type, priority).
4. Сервер в ответ отправляет своих ICE candidates (ip address, port, priority, protocol, type).
5. Клиент и сервер выставляют их по приоритетам и протоколам.
6. После это каждый из них формирует список пар кандидатов по семействам IP address.
7. Сформированный список пар выставляется по приоритетам по формуле.
8. Отправка
9. Пользователь заходит в браузере на определенный URL. На срабатывает скрипт, который запускает клиента и он в свою очередь подключается к сигнальному серверу. Клиенту на выбор предоставляется N-ое количество камер. После выбора одной из камер, клиент отправляет на сервер IP address выбранной камеры.
10. При подключении клиента к серверу, сервер создает новый отдельный поток для этого клиента. С помощью этого потока, клиент и сервер смогут взаимодействовать друг с другом передавая разную информацию в будущем.
11. Пока клиент ожидает, сервер при получении адреса отправляет запрос на камеру и ждет от нее ответа. Запрос выглядит примерно так: "OPTIONS rtsp://"+ ip address камеры +"/axis-media/media.amp RTSP/1.0\r\nCSeq: 1\r\nUser-Agent: RTSPClient\r\n\r\n". В случае успешной отправки, в ответ на сообщение, камера присылает описание своих аргументов и команд которые она может отправлять. И в случае положительного результата, сервер отправляет клиенту уведомление о готовности к подключению.
12. Клиент получив положительный ответ на запрос начинает строить и отправлять ice candidates. Каждый из таких кандидатов состоит из ip address, port, priority, type. Свой ip клиент может получить несколькими путями. Первый — получить адрес и порт в локальной сети, что сделать можно вызвав одну функцию. Второй – сформировать и отправить STUN запрос на публичный STUN сервер, состоящий из нескольких параметров, header stun request, magic cookies and id, которые помогают отличить этот запрос от других запросов. В ответ STUN server отправляет ответ похожий на запрос, но с дополнительным полем. Это поле является закодированный ip address и порт, передаваемые в параметрах, которые нужно получить воспользовавшись magic cookies. IPv4 имеет всегда один и тот же размер. Его размер состоит из 4 байт. Чтобы получить адрес нужно, каждый байт IP xor(^) с каждым байтом magic cookies. Этот аргумент всегда имеет одно и то же значение 0x2112A442. И адрес делает логическое исключающее «или» с каждым байтом этого аргумента. Для получение значение порта нужно сделать такую же лог. Операцию только с двумя первыми байтами. Не маловажная часть ice candidate это приоритет. Поскольку благодаря ему алгоритм подбирает самого наилучшего candidate для создания пары. Приоритет вычисляется по определенной формуле. Формула выглядит примерно так:

priority = (2^24)\*(type preference) + (2^8)\*(local preference) + (2^0)\*(256 - component ID)

Type preference это число от 0 до 126 включительно и выставляются относительно типов ice candidates. Рекомендовано использовать для host 126, 110 для peer-reflexive, 100 для server-reflexive. Можно увидеть дополнительные определения.

Самым лучшим типом для кандидата это host, потому что host означает, что этот кандидат является простым кандидатом в сети и не имеет никаких посредников. Host показывает что обращаться к кандидату можно напрямую через сигнальный сервер без других участников в этой схеме. Server-Reflexive это тип, который показывает, что кандидат является отдельным участником, отправляющий данные на еще один сервер, выступающий в роли промежуточного кандидата. Peer-reflexive отличаешься от предыдущего типа, он выступает в роле не сервера-кандидата, а промежуточного равноправного кандидата.

Local preference так же моет иметь разные значения. Поскольку клиенты бывают двух типов, полный и не полный, то и количество кандидатов в каждом может отличаться. Если клиент не полный, то у него может быть всего один кандидат, он имеет всего один ip address, и в этом случае значение local preference составляет 65535. Если клиент имеет несколько разных кандидатов, то каждый из них имеет значение, но они уникальны для каждого. Что касается полной и не полной реализации объекта, то лучше всего называть их Full (F) и Lite (L). И существует всего 3 варианта реализации для объединения клиента. Full-Full, Full-Lite, Lite-Lite. Самой сложной является F-F, а самой легкой L-L.

Lite-Lite. Как было сказано выше, lite клиент имеет всего одного кандидата с одним ip адресом. И установление подключения между такими клиентами не составляет больших трудностей, но это не все. Каждый клиент из такой реализации не выполняет проверок на подключения в виде STUN запросов от контролирующего на контролируемого клиента. При подключении двух клиентов, каждому из них отводиться своя роль и в реализации L-L не определяется контролирующий и контролируемый клиент, что и является минусом этой реализации.

Full-Full. Самая трудная реализация. Поскольку каждый из клиентов имеет более одного кандидата и разные IP address, что усложняет задачу. Один из клиентов должен взять на себя роль контролирующего, а другой контролируемого. Для установления соединения необходимо составить пары кандидатов и выбрать из неё самую успешную, так вот каждый из клиентов занимается именно этим. И не маловажная часть это отправка проверок на каждого из удалённых кандидатов для точного составление контрольной пары кандидатов. Но такие проверки не только при подключении, но при дальнейшей работе программы.

Full-Lite. Эта реализация имеет сходство с предыдущими реализациями. На этот раз роль контролирующего клиента займет клиент с Full реализацией, а роль контролируемого клиента получит клиент с реализацией Lite. Так вот контролирующий клиент будет создавать пары из кандидатов и выстраивать их по приоритетам. Получаемый список после сортировки называется конечный список пар. После проведения всех проверок в виде STUN запросов Full клиент получает контрольную пару. И настраивает соединение.

Так вот, что под собой подразумевается построение контрольного списка. Контрольный список – это список с уже выбранными парами кандидатов, общающиеся друг с другом, но для начала необходимо построить пары кандидатов, которые смогут между собой общаться. Контролирующий клиент создает пары кандидатов и объединяет их в контрольный список. Этот же клиент связывает своих и удаленных кандидатов для одного и того же потока данных с одним и тем же семейством IP address. Если семейства адресов совпадают, то клиент создает пару кандидатов и вычисляет для этой пары приоритет. Формула очень проста так как для нее необходимо только иметь приоритеты одного и другого кандидата. Выглядит она вот таким образом.

Pair priority = 2^32\*MIN(G,D) + 2\*MAX(G,D) + (G>D?1:0)

G – приоритет от контролирующего клиента

D – приоритет от контролируемого клиент

Находиться минимальное и максимальное числа между двух приоритетов. Далее контролирующий клиент составляет список и сортирует по этому приоритету. Чем выше приоритет тем лучше пара.

Так вот, когда пары расставлены по приоритетам то можно запускать проверки на подключение для каждой пары. Проверки выглядят в форме STUN запрос и ответ. Каждый из клиентов отправляет запрос и ждет на него ответа, но запросы их немного отличаются. Каждый запрос в аргументах сообщения несет большую информацию. Самыми важными из них являются вот такие вот аргументы: priority, use-candidate, ice-controlled и ice-controlling. И каждый из них имеет свой не мало важный смысл.

Priority – это аргумент отправляется обоими клиентами и содержит в себе приоритет, который имеет локальный кандидат клиента отправителя, а именно вычисляемый по формуле для кандидатов, но используя те же параметры необходимо вычислять приоритет не для данного типа что в кандидате, а для рефлексивного кандидата, а именно изменить переменную type preference на 110.