Синхронная Символьная Иерархия

Автор : Балыбердин А.Л.

Новосибирск, 2022 г.

Оглавление.

Синхронная Символьная Иерархия
Принципы формирования синхронного канала
(устранение эффекта проскальзывания)

Автор : Балыбердин А.Л.

Новосибирск, 2022 г.

Передача голосового трафика практически сразу перешла на синхронные каналы (с постоянной скоростью и задержкой передачи), а компьютерные коммуникации до настоящего времени базируются на «телеграфном способе общения» (Асинхронная пакетная коммутация). Для конечного пользователя идеальной сетью является та, которая предоставляет точно запрашиваемую скорость передачи данных с заранее известной и неизменной задержкой передачи данных (мое личное мнение). У синхронных (плезиохронных) сетей есть много недостатков, не позволивших им выиграть соревнование у пакетной коммутации. Предлагаемая сеть, основанная на принципах Синхронной Символьной Коммутации или Иерархии (ССИ), представляет собой способ прозрачного перехода компьютерных сетей к синхронному способу связи и лишена практически всех недостатков присущих таким технологиям как PDH. В серии статей будут постепенно и достаточно подробно для того, чтобы являться технически заданием для инженера электроника, описываться принципы функционирования ССИ.

Устранение эффекта проскальзывания

У синхронного способа передачи данных есть неприятный эффект потери или дублирования единичных битов передаваемых данных. Появляется из-за отсутствия жесткой синхронизации генераторов всех передатчиков сети (имеется достаточно много материалов, описывающих эту проблему и методы борьбы с ней). В сетях PDH приняты сложные и дорогие методы борьбы с этим явлением, что в значимой степени и погубило данную технологию вместе с использующей ее телефонной индустрией (Мое мнение: Ее погубил монополизм и пренебрежительное отношение к потребителю).

В практически во всех современных сетях связи канальный уровень является синхронным. Передатчик синхронизируется собственным генератором, а приемник восстанавливает тактовую частоту из принимаемых данных. Участок сети между такими передатчиком и приемником можно считать синхронным каналом передачи данных. Скорости передачи в современных каналах связи достигают десятков и сотен гигабит, при частотах тактирования коммутаторов в единицы гигагерц. Поэтому единицей передачи являются не отдельные биты, а слова или символы (десятки и сотни бит). Канальный уровень полностью соответствует требованиям ССИ и может быть заимствован без изменений и дополнительных доработок.

Уровень потока символов

Эффект проскальзывания реализуется в двух вариантах как пропуск и как добавление лишнего символа. Вариант пропуска символа является самым «неудобным» поскольку требует повторной передачи утерянных данных, а это может занять много времени и потребует применения буферов большого размера. Буфер большого размера добавляет большую и переменную задержку,

что является неприемлемым для конечного пользователя (Требование к каналу в ССИ: Минимальная и заранее известная задержка). Коммутатор должен быть устроен так что бы время коммутации стремилась к четырем тактам синхронизации (прием символа, запись в буфер коммутатора, чтение буфера коммутатора, передача символа). В пределе буфер коммутатора (коммутационная матрица) должен выродится в регистр, хранящий один символ.

Необходимо создать механизм гарантирующий, что будет реализовываться вариант удвоения символа (скорость на входе коммутатора меньше скорости на выходе) и сделать это с минимальными аппаратными и финансовыми затратами. Современные (бюджетные) кварцевые генераторы частоты имеют точность на уровне 50 ppm (расхождение от указанной частоты не превышает 50 миллионных частей) считаем, что взаимная ошибка гарантированно не превышает 500 ppm (для примера).

Самый простой способ реализовать такой механизм — это заменить в потоке пользовательских данных часть полезных символов символами «Компенсирующий символ». Полезная скорость потока сократится на 0.02%, но это не является значимой величиной.

Для того что бы отличать пользовательские символы от компенсирующих необходима какая-то метка или одна из битовых комбинаций должна быть зарезервирована. Теоретически можно использовать служебные комбинации из линейных кодов, но ранее декларировано отсутствие требований к канальному уровню. Кроме того, данный символ не является единственным, далее может быть развит в целое дерево различных служебных символов в том числе и пользовательских (структурирование передаваемых данных).
Бит ноль (младший бит передаваемого символа) должен указывать на тип символа. Если равен нолю пользовательский тип, если единица служебный тип (интерпретируемый коммутатором).
Сразу добавлю, что следующий за младшим бит равный нулю указывает что символ содержит данные, равный единице содержит «сигнал». Эффективность (КПД) использования первичного потока не пострадает (для 100 битового символа 98%).

Механизм вставки компенсирующих символов должен обеспечивать минимальный размер буфера. Нельзя производить единовременную вставку, например 500 компенсирующих символов один раз в секунду. Наиболее просто и эффективно равномерно (через одинаковые промежутки времени) добавлять их в поток пользовательских данных. При таком подходе в буфере приемника (в идеале) всегда будет ровно один символ, очень редко буфер будет пуст или содержать 2 символа. При чтении из пустого буфера, должен читаться компенсирующий символ.
Положительный эффект: Средняя задержка из-за размера буфера коммутатора

стремится к половине времени передачи одного символа в потоке.

Пример реализации:
Передатчик генерирует синхронизацию и сигнал запроса считывания следующего символа из FIFO коммутатора. Между ним и коммутатором находится блок, который один раз за установленное число тактов блокирует выборку символа, а вместо него вставляет компенсирующий символ (счетчик + мультиплексор).
На приемной стороне, при приеме компенсирующего символа блокируется уже запись в FIFO коммутатора.

Алгоритм устранения проскальзывания достаточно прозрачен и не нуждается в отдельном НИР. Можно только сравнить эффективность (КПД) использования первичного потока с уже существующими вариантами сетей.

Мое мнение: будет лучше даже чем у пакетной коммутации, в ней этот механизм заменен на необходимость передачи преамбулы.

Синхронная Символьная Иерархия
Принципы формирования виртуальных каналов
(разделение единого физического канала на множество виртуальных)

Автор : Балыбердин А.Л.

Новосибирск, 2022 г.

Обязательным атрибутом современной сети, является разделение физических линий связи на множество отдельных виртуальных каналов. Каналы могут быть и совсем виртуальными, как в случае с пакетной коммутацией, так и прописаны в формате передаваемых конструкций (PDH, SHD и т.д.). Поскольку плезиохронные каналы появились во времена голосовой телефонии и других потребителей не было, виртуальные каналы имели одинаковую скорость 64КБит\сек. Для современных сетей требуются каналы с различной скоростью и различными свойствами.

В статье «ССИ_Синхронный поток_Эффект проскальзывания (1)» показан принцип формирования базового потока символов, теперь будет описан принцип его разделения на отдельные плезиохронные потоки с произвольной скоростью передачи символов.

Ограничения: Сумма скоростей виртуальных каналов не должна быть больше скорости физического канала. Время коммутации не должно быть больше единиц периодов передачи символов в виртуальном канале и стремиться к половине периода передачи в конкретном виртуальном канале (Чем медленнее виртуальный канал, тем больше времени его данные находятся в FIFO коммутатора)

Ограничение на время коммутации существенно ограничивает варианты построения коммутатора. Самый простой вариант — это массив модулей много-портового FIFO (3-5 символов), число портов чтения (записи) равно числу внешних интерфейсов коммутатора. Каждое FIFO соединяет вход коммутатора с выходом, какой вход с каким выходом определяется в момент создания виртуального канала. Число модулей FIFO ограничивает (равно), число одновременных виртуальных каналов, создаваемых коммутатором.

Модули приемника (передатчика) должны выдавать на адресные входы коммутатора номер FIFO к которому происходит обращение. Для примера, зададим точность установки скорости передачи виртуального канала в один символ в секунду при скорости физического канала 100G и размере символа в 100 бит потребуется 1E9 тайм-слотов. Максимальное число отдельных виртуальных каналов ограничивается числом FIFO модулей. Формирование последовательности опроса FIFO производится с учетом скорости передачи в каждом виртуальном канале. Необходимо гарантировать что FIFO размером 3-5 символов никогда не переполнится. При этом не снижать КПД использования физического канала больше чем на единицы процентов (допускается использование для пользовательских данных до 95 % пропускной способности физического канала на постоянной основе).
Думаю алгоритмов раскладки достаточно много, лично я нашел два. Первый больше подходит для пояснения принципа раскладки, но сильно неудобен для реализации. Второй более подходит для практической реализации.

Мультиплексирование (разделение) канала связи (1)
(*Алгоритм симметричен, одинаков для приемника и передатчика*)

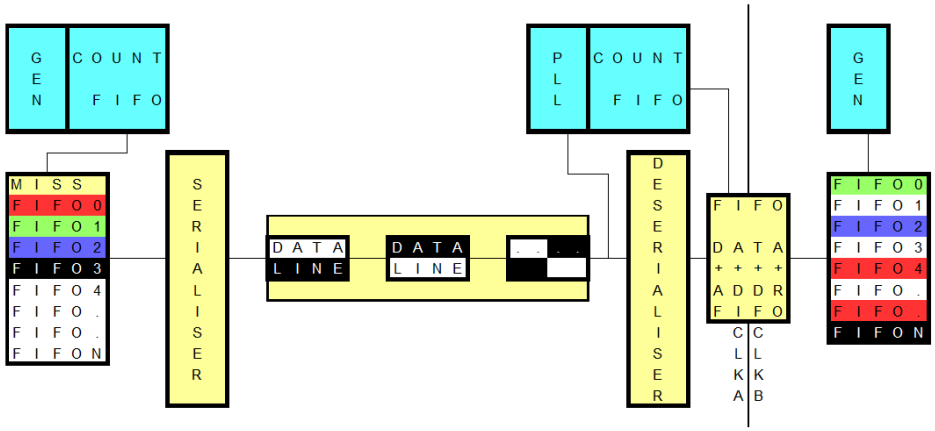
1. Сортируем создаваемые каналы по убыванию скорости передачи.
2. Для каждого канала создаем счетчик (регулирует скорость передачи) и назначаем FIFO, где хранятся передаваемые символы.
3. Каждый такт синхронизации в каждый счетчик прибавляем константу пропорциональную скорости передачи, константа = V (требуемая) / V (физического потока).
4. Каждый такт проверяем счетчики, в очередности убывания скоростей привязанных к ним каналов, на переполнение (значение больше единицы), пока не найдем первый счетчик, содержащий значение больше единицы.
5. Из найденного счетчика вычитаем единицу, а в суммарный поток добавляем один символ из буфера передаваемых символов (создан в связке со счетчиком).

Данный алгоритм проверен на программной модели, максимальное расстояние между соседними символами в пределах виртуального канала составляет 2.3 периода передачи символа в пределах данного виртуального канала.

Мультиплексирование (разделение) канала связи (2)

(*Алгоритм симметричен, одинаков для приемника и передатчика*)

1. Создаем по одному счетчику для каждого приемника и передатчика физического канала (примерно, как в «ССИ_Синхронный поток_Эффект проскальзывания (1)»), с помощью служебных символов синхронизируем их относительно потока символов в данном физическом канале. Для выбранного примера счетчик от 0 до 10E9 - 1.
2. Получается, что каждый символ будет иметь уникальный номер (в пределах одной секунды)
3. При обращении к коммутатору преобразовываем содержимое счетчика в адрес (номер) FIFO.

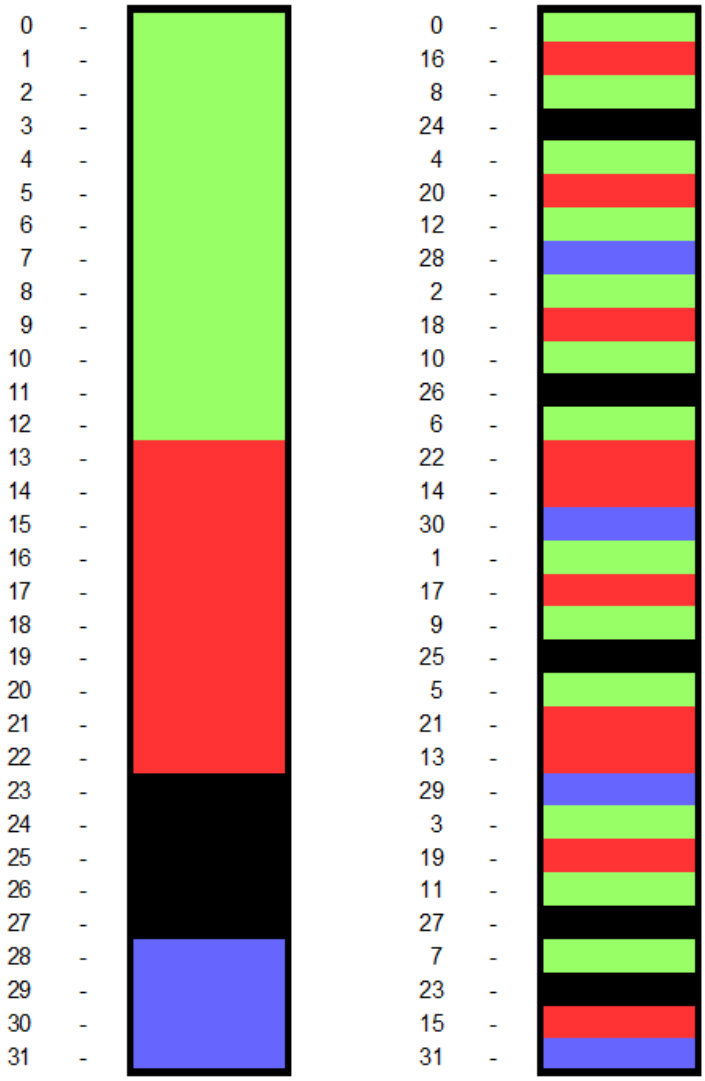


Алгоритм преобразования — Счетчик <:> Адрес FIFO:

1. Диапазон значений в 10E9, по сути является номерами тайм-слотов в передаваемой периодической структуре. Остается только распределить эти тайм-слоты максимально равномерно и пропорционально скорости каждого виртуального канала.
2. Каждому адресу FIFO соответствует несколько значений счетчика. Ноль если канал не создан, единица если канал имеет скорость один символ в секунду и так далее.
3. Представим весь диапазон значений счетчика как некоторое адресное пространство и присвоим каждому каналу непрерывный диапазон значений равный символьной скорости виртуального канала. Выглядит как выделение памяти в современных вычислительных системах.
4. Каждый модуль FIFO откликается на любое значение в пределах этого диапазона, выдавая или записывая символ.
5. Просто подавать счетчик на адресные входы FIFO нельзя, данные будут передаваться непрерывными последовательностями раз в секунду. Для равномерного распределения опросов FIFO нужно «перевернуть» значение счетчика (поменять местами старшие разряды с младшими).
6. FIFO также должно сравнивать этот «перевернутый» счетчик с

диапазоном значений, занятый данным виртуальным каналом.

7. Пример работы алгоритма преобразования «Счетчик- адрес FIFO»



Второй алгоритм распределения тайм-слотов гораздо эффективнее в реализации чем первый и соответствует всем требованиям ССИ, но не был проверен на программной или аппаратной модели.

Темы для НИР

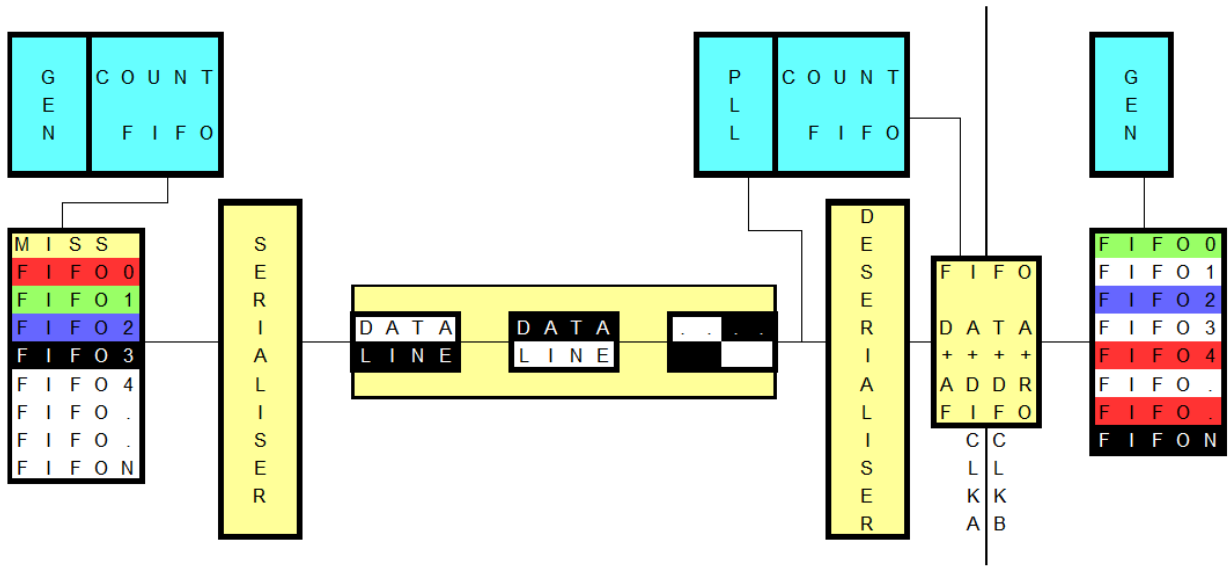
1. Проверить равномерность получаемых виртуальных потоков.
2. Практически измерить максимальное число символов в FIFO, для разного числа виртуальных каналов и различных вариантов скоростей.
3. Проверить зависимость неравномерности символьной скорости от последовательности (места расположения) виртуальных каналов в пространстве комбинаций счетчика.
4. Проверить работу алгоритма, когда число комбинаций счетчика не равно степени двойки.

Синхронная Символьная Иерархия
Принципы работы коммутатора синхронных потоков

Автор : Балыбердин А.Л.

Новосибирск, 2022 г.

Выбор структуры коммутатора.
Синхронные коммутаторы достаточно просты по своей конструкции и представляю собой различным образом соединяемые небольшие FIFO (сети Клоза и т.д.), весь вопрос только в размерах коммутационного поля.
ССИ изначально предназначается для достижения предельно возможных скоростей передачи при малых задержках, значит наиболее оптимальным будет «однослойный» коммутатор. В настоящее время скорости передачи выросли до такого уровня что появилась потребность в физическом сокращении пути между источником и приемником данных.
Если для первых IP сетей передачи данных размер пакета в кабеле, путь проделываемый электромагнитной волной за время передачи пакета данных, достигал 150 км и было все равно в какой части города поставить коммутатор (кроме экономии кабеля), можно было укладывать «красивые» пучки проводов, создавать специальные зоны для размещения коммутаторов.
В настоящее время эти расстояния варьируются от полутора метров до размера кристалла (единицы миллиметров) и дополнительный путь по кабелю в несколько метров вносит задержку большую чем непосредственно передача пакета в канале. Тоже самое можно сказать и о времени коммутации.



Вывод: для высокопроизводительных систем более оптимальной является сеть, соединяющая физически соседние элементы и для такой сети необходим коммутатор с минимальным временем коммутации. Можно даже ввести метрику отношение задержки передачи при прямом соединении (кабель проложенный по прямой между приемником и передатчиком) или через сеть с последовательностью коммутаторов. Необходимо решение оптимальное для коммутатора и однослойным коммутационным полем (мультиплексор, FIFO, демультиплексор).

Структура коммутатора
Приемная часть состоит из стандартного канального уровня с модулем фрагментации потока (ССИ_Синхронный поток_Фрагментация потока (2)) и FIFO буфера физического канала для переноса в другой Clock Domain. Со стороны приемной части тактовая частота восстанавливается из принимаемых данных, со стороны коммутатора это частота тактирования ядра коммутатора. Именно на входе этого FIFO и происходит удаление «Компенсирующих символов» (ССИ_Синхронный поток_Эффект проскальзывания (1)) из принимаемого потока. Это гарантирует меньшую среднюю скорость на приеме символов чем скорость обработки ядром коммутатора.

Доступ пользователя к сети происходит через модифицированные приемники (передатчики), где канальный уровень заменен на пользовательский физический интерфейс.

Ядро коммутатора представляет собой набор небольших модулей, по одному на каждый виртуальный канал. Число физических каналов относительно невелико, поэтому можно каждый модуль (или группу модулей) сделать коммутируемым на любой из имеющихся каналов. Коммутация такого модуля (группы) происходит в момент создания виртуального канала и не изменяется до его удаления. Модуль после создания (назначения) его конкретному виртуальному каналу (конфигурирования) начинает откликаться только на сигналы от конкретного физического интерфейса и на конкретный диапазон адресов (тоже самое и на выходе). Следует отметить, что на один и тот же диапазон адресов (на прием) может откликнуться несколько модулей, что дает возможность копировать поток и направлять его в несколько физических каналов одновременно. Копирование позволяет строить не просто линейные маршруты, а широковещательные для нескольких получателей (деревья), один источник много получателей. При этом каждый получатель увидит одинаковую последовательность символов (последовательная консистентность).
Принимаемые данные могут быть служебными (обрабатываются УУ коммутатора) и данными пользователя (передаются в рамках созданных виртуальных каналов), определяются младшим битом символа.
Примерно таким же образом происходит формирование суммарного канала. Из циклического счетчика символов формируется адрес буфера виртуального канала. Если выбранный канал содержит данные, то они считываются и отправляются в передатчик. Если их нет или адрес не принадлежит ни одному буферу, то запрашивается наличие данных в модуле УУ коммутатора. Если их нет, то передается символ нет данных. На приемной стороне по типу символа определяется обрабатывающий блок (буфер виртуального канала, УУ коммутатора).
В случае приема ошибочного символа (приемник не смог восстановить символ из принятого сигнала) происходит замена на символ «ошибка», а получатель определяется по адресу FIFO. Далее этот символ так и передается в виртуальном канале. При приеме данных из виртуального канала, для

восстановления данных можно запрашивать повторную передачу только этого символа.

Закрытие (удаление) виртуального канала происходит при вычитывании из FIFO пользовательского символа - «сигнал закрыть канал», после его чтения (в сторону передатчика) модуль виртуального канала сбрасывается в неактивное состояние (пока его повторно не инициализирует УУ коммутатора). Принципы организации асинхронной передачи, функционирования устройства управления, алгоритм создания виртуального канала будут описаны в следующих статьях.

Темы для НИР:

1.Устройство синхронного коммутатора достаточно прозрачно и на этом этапе нужно проработать именно структуру, без реализации УУ и блока передачи асинхронных данных. Интерфейс для инициализации модулей виртуальных каналов должен обеспечивать полную инициализацию за один такт.

Синхронная Символьная Иерархия
Принципы управления виртуальными каналами
(создание, удаление, изменение скорости)

Автор : Балыбердин А.Л.

Новосибирск, 2022 г.

Современные телекоммуникации разделены на два основных типа синхронные (SDR и прочее) и асинхронные (IP и другие пакетные сети).

Синхронные сети требуют достаточно большого времени на создание нового соединения, обеспечивают постоянство скорости и задержек передачи. Есть возможность задействовать практически всю полосу физического канала. Отсутствует взаимное влияние виртуальных каналов и нет потерь данных из-за переполнения буферов. Синхронные сети отличает достаточно простая схемотехника.

Асинхронные сети в противоположность не требуют никаких действий по установлении соединения (отправил пакет и все), но такой подход не позволяет использовать больше половины физической пропускной способности сети и присутствует сильное взаимное влияние различных абонентов, порой приводящее к существенной потере скорости и данных (после загрузки канала в 75% КПД использования канала падает практически до нуля). Требуют наличия в коммутаторе буферов достаточно большого объема, что становится проблемой при больших скоростях передачи.

ССИ в своей основе является синхронной сетью, при этом она избавлена от основных недостатков присущих синхронным сетям. Основным плюсом является механизм быстрого создания виртуального канала без предварительного запроса приемника. Это порождает некоторую асинхронность сети на уровне доступа. В момент прохождения одного из промежуточных коммутаторов может выясниться, что дальнейшее создание виртуального канала невозможно из-за отсутствия пропускной способности. В современных сетях загрузка изменяется быстрее чем распространяется информация о ее изменениях. В целом ССИ можно охарактеризовать как систему с асинхронной передачей короткого запроса на создания виртуального канала и синхронной передачей основных данных.
Для создания нового виртуального канала необходимо получить информацию о возможных маршрутах, последовательности промежуточных коммутаторов и их физических каналов. Данная информация может задаваться как константа, так и получаться от какой-либо сетевой службы. Эту информацию необходимо получить заранее.
Этап непосредственного создания виртуального канала не требует дополнительного времени.

Примерная визуализация ССИ сети, где каждый перекресток — это коммутатор, а дорога — это кабель связи.

Последовательность создания нового виртуального канала.
Физический канал в сетях ССИ разделяется на отдельные виртуальные каналы с помощью алгоритма описываемого в «ССИ_Коммутатор синхронных потоков (3)». Символы разделены на две основных группы служебные (интерпретируются коммутатором) и пользовательские (интерпретируются пользователями и промежуточными модулями виртуальных каналов).
Физический канал разбит на два потока служебный и пользовательский. При создании нового канала, необходимая пропускная способность отбирается у служебного потока. Служебный канал не разбивается на отдельные виртуальные каналы и является единым потоком, имеющим синхронную (еще не распределенную между виртуальными каналами) и асинхронную составляющую (не используемую в данный конкретный момент распределенную пропускную способность).
Возможность создания виртуального канала есть только тогда если есть не распределенные тайм-слоты (синхронная составляющая) и есть свободный модуль виртуального канала. Модуль виртуального канала, достаточно простое устройство и микросхемы, построенные по современным технологически нормам, могут содержать сотни тысяч таких модулей, но число их конечно.
Для создания нового виртуального канала необходимо поместить в служебный поток символы с запросом, содержащим требуемую скорость и маршрут.
В момент приема коммутатором символов с этим запросом, УУ коммутатора

выделяет запрошенную часть физического канала (на прием и на передачу) и один из свободных модулей виртуального канала, а также транслирует символы запроса в служебный канал следующего затребованного коммутатора. Если предоставить затребованную производительность невозможно, то запрос отбрасывается. Механизмов удаления таких обрезков может быть много, передать служебное сообщение предыдущему коммутатору является наиболее оптимальным. При обработке запроса, от маршрута «отрезается» использованная часть и в момент исчерпания данных маршрута считается, что цель достигнута, канал построен и запрос удаляется. Относительная маршрутизация не требует слишком сложного УУ коммутатора. Отмечу, что полоса служебного потока (при удачном выделении полосы) всегда больше, чем скорость виртуального потока, соответственно запрос «пробежит» маршрут виртуального канала быстрее чем данные для этого канала, значит данные можно передавать сразу за запросом (без паузы). Удаление канала инициируется пользовательским символом закрыть канал, при его отправки передатчиком все использованные ресурсы освобождаются. Для противодействия накапливания «мусора» следует ограничить время «жизни» канала.

Изменение скорости уже созданного виртуального канала.
Пропускная способность виртуального канала определяется числом последовательных комбинаций циклического счетчика символов (ССИ_Синхронный поток_Фрагментация потока (2)). Каждый новый канал создается добавлением в конец выделенной области нового резерва.
В процессе удаления виртуального канала все модули виртуального канала с большим чем у удаляемого значением счетчика, производят вычитание числа тайм-слотов (скорости удаляемого потока) из своего диапазона комбинаций счетчика. Был диапазон N до N+скорость канала, стало N - скорость удаляемого до N+ скорость канала - скорость удаляемого канала. В результате выделенная область становится монолитной (противодействие фрагментации). Алгоритм изменение скорости выглядит примерно также, но без удаления канала. Простое изменение границы начиная с некоторого значения комбинаций счетчиков.
Все эти операции возможно проделать за один такт синхронизации, они не содержат логических цепочек неконтролируемого размера.

Конкретные алгоритмы создания, удаления, переноса и других действий над виртуальными каналами должны быть проработаны в момент создания стандарта ССИ.

Задачи для НИР:
1. Как влияют процессы удаления и изменения скорости виртуальных каналов на синхронность потоков в соседних каналах (где редактирование не происходит). Иными словами: как влияет перемещение виртуального канала в диапазоне значений циклического счетчика символов?

Синхронная Символьная Иерархия
Асинхронная передача данных

Автор : Балыбердин А.Л.

Новосибирск, 2022 г.

ССИ в своей основе, синхронная сеть и одной из основных претензий к таким системам является неэффективное использование каналов для компьютерных применений. Возможность быстро создавать и удалять каналы с произвольной скоростью часть этих претензий снимает, но остается мгновенное недоиспользование зарезервированной пропускной способности. Кроме того, для вычислительных систем очень часто требуется быстро передавать короткие сообщения, с гарантией доставки в определенном диапазоне времени. Для решения этих задач в ССИ предусмотрен механизм асинхронной передачи данных.

Формирование асинхронного потока символов.

Если в момент формирования суммарного потока нет ни пользовательских ни служебных данных (ССИ_Коммутатор синхронных потоков (3)), то есть возможность передавать данные блоку асинхронной передачи данных. Происходит формирование асинхронного потока символов, он также как и служебный поток состоит из служебных символов и не разбивается на виртуальные каналы. Можно даже сказать, что блок асинхронной передачи — это часть УУ коммутатора.

Создание асинхронных виртуальных каналов.

Каких-то новых виртуальных каналов не создается, просто в описании канала добавляется метка о возможности внеочередной (не синхронной) передачи. Все остальное, выделение модуля виртуального канала, необходимого числа тайм-слотов идентично обычному синхронному виртуальному каналу. Решение о асинхронной передаче для конкретного виртуального канала принимает УУ коммутатора.

Алгоритм работы блока асинхронной передачи данных.

(Самый *простой вариант*)

- В момент приема символа, модуль виртуального канала сигнализирует о возможности внеочередной передачи (асинхронности).
- Если канал асинхронен, то номер модуля виртуального канала запоминается в специальном буфере УУ коммутатора.
- Когда есть возможность асинхронной передачи, происходит формирование последовательности (пакета) служебных символов в теле которого закодирован список адресов модулей виртуальных каналов, для которых есть данные для асинхронной передачи.
- Далее в моменты, когда нет синхронных данных производится запрос модулей виртуальных из списка и передача данных (только тип данных изменяется на служебные).
- Если в момент присутствия адреса в очереди происходит выбор модуля с адресом присутствующем в буфере УУ и больше нет данных для передачи, то соответствующий адрес удаляется из очереди на асинхронную передачу. Таким образом, виртуальный канал приобретает синхронную и асинхронную составляющую скорости передачи. *Передать если есть возможность, но с задержкой не больше, чем заранее задано и синхронной скоростью не менее заданной.*

При таком способе передачи данных будет теряться в районе 20% процентов от пропускной способности асинхронного потока (издержки на формирование пакета символов), но это и так не используемые тайм-слоты и альтернативой такого использования, которых является заполнение их символами «нет данных» (по факту полное неиспользование). Если каждому виртуальному потоку, допускающему асинхронную передачу приписывать некоторую синхронную производительность или просто не выделять пропускную способность сверх некоторого уровня заполнения суммарного физического потока, то формируется некоторый резерв производительности, который в сумме с недоиспользованием синхронного трафика, будет использоваться для формирования асинхронной составляющей трафика. Получается достаточно быстрый механизм передачи асинхронных сообщений, да еще и утилизирующий имеющуюся пропускную способность физического канала практически на 100%. Можно добавить механизм позволяющий изменять очередность, например, чем больше номер модуля виртуальных каналов, тем больше приоритет виртуального канала в получении асинхронной составляющей трафика.

У данного механизма есть и недостатки, нет контроля за уровнем заполнения соответствующего буфера в модуле виртуального канала следующего коммутатора и может происходить переполнение. В самом простом виде передача асинхронных сообщений выглядит как отправка некоторого числа символов в асинхронный виртуальный канал, число символов определяется как минимальный размер буфера в промежуточных коммутаторах. И ожидание уведомления о приеме символов. Поскольку время доставки определяется физическим значением расстояния по кабелю, то для увеличения производительности асинхронного канала можно создать несколько параллельных асинхронных каналов и отправлять данные последовательно в каждый из них. Получаем полностью прозрачную замену пакетной коммутации. Данный механизм напоминает процесс отправки пакетов протоколом TCP IP. Для частой передачи коротких сообщений (обновление значения переменной и т.д.) данный механизм достаточно оптимален. При проектировании сети, размер символа и размер буфера следует выбирать еще и с учетом размеров коротких сообщений.

Синхронная Символьная Иерархия
Дезагрегация оперативной памяти

Автор : Балыбердин А.Л.

Новосибирск, 2022 г.

В настоящее время есть несколько направлений в развитии интерфейсов оперативной памяти вычислительных систем. Для оптимизации низкоуровневых интерфейсов это OMI (Open Memory Interface) и прочее, которые позволяют решать проблему нехватки электрических контактов для подключения внешней оперативной памяти. Появились первые полностью оптические варианты меж-чипового интерфейса.
https://ayarlabs.com/supernova/
https://3dnews.ru/assets/external/downloads/2020/12/04/1026993/demophot.mp4
Есть множество технологий позволяющих в той или иной степени дезагрегировать оперативную память (RDMA). Появились предпосылки к появлению процессоров, не нуждающихся в традиционной материнской плате для своей работы. Для таких процессоров есть возможность реализовать крайне плотную упаковку чипов и соответственно минимизировать задержки передачи данных между взаимодействующими элементами.
Но пока нет самого главного: Сети, универсальной и лишенной недостатков современных пакетных сетей связи.
Сеть ССИ является оптимальным решением для таких систем: крайне большие и гарантированные скорости передачи, малые стабильные задержки, отсутствие взаимного влияния виртуальных каналов, не требуется больших, да и вообще отдельных коммутаторов.

Как это может выглядеть реализация будущего суперкомпьютера:

1. Пластиковая подложка с большим числом впаянных в пластик оптических волокон, которые реализуют сеть соединений между компонентами вычислительной системы.
2. В подложке есть «вырезы» с оптическим интерфейсом для установки модулей процессора и любых других (память, внешние интерфейсы и прочее). Расстояния связи совсем небольшие и потому особых требований к точности реализации нет.
3. Процессор оформлен в виде модуля небольшой толщины (2-5мм) из двух пластин, хорошо проводящих тепло и электрический ток, между которыми находится кремниевая подложка с микросхемой процессора и прочими чиплетами.
4. На торцах такого модуля располагаются выводы оптического интерфейса, которые через оптический интерфейс соединяются с оптическим волокном подложки.
5. Верхняя и нижняя пластина представляют собой электроды для подачи питания на процессор. Расстояние между процессорами рассчитываются из требований по отводу выделяемого при работе тепла к радиаторам, одновременно являющимися и шинами питания. Сечение таких шин достаточно большое и позволяет передавать питающие токи в сотни кило ампер.
6. Размеры подложки могут быть практически любыми и эту подложку можно располагать самыми разными способами между последовательностями пластин радиатора (питания). Полученную

подложку можно банально резать «ножницами» для получения требуемых размеров и формы, те подложка может стать универсальным продуктом с высокими тиражами.
7. При стандартизации оптического интерфейса, появится стандарт на оптико-электрический модуль, с возможностью универсальной установки в данную подложку. Если модуль расположен рядом с краем подложки, то есть возможность устанавливать любые внешние интерфейсы, разъем модуля выступает за пределы радиаторов.
8. Часть оптических интерфейсов можно использовать для связи «слоев» такого пирога напрямую, что позволит создавать сети с многомерной архитектурой.
9. Можно сделать вырезы без оптического интерфейса и использовать их передаче тепла от радиатора охлаждающей жидкости.

Такой DataFlow суперкомпьютер кубической формы со стороной в 10 метров, теоретически может иметь производительность 10E20 Flops и максимальной задержкой передачи данных менее 1 мкс (если удастся отвести от него один гигаватт тепла). Для соседних элементов задержки передачи составят первые наносекунды, что сопоставимо с задержками передачи данных в пределах кристалла. Это позволит решить проблему «стягивания» всей электроники процессора на один кристалл, которая из-за проблем с теплоотводом зашла в тупик и породила такое явление как «темный кремний».

Алгоритмы совместного доступа к разделяемому ресурсу на примере распределенной общей памяти (DSM)

Для понимания дальнейшего текста необходимо прочитать:

- *ССИ_Синхронный поток_Эффект проскальзывания (1)*
- *ССИ_Синхронный поток_Фрагментация потока (2)*
- *ССИ_Коммутатор синхронных потоков (3)*
- *ССИ_Управление списком виртуальных потоков (4)*
- *ССИ_Асинхронная передача данных (5)*
- *https://parallel.ru/krukov/lec6.html*

Учитывая все вышесказанное можно представить, как будет выглядеть модуль оперативной памяти. Такие модули (кремниевая подложка с чиплетами) должны иметь оптический интерфейс, электрический интерфейс не позволит передать весь трафик (100Т и более) за пределы подложки. В состав модуля входит контроллер виртуальной памяти и множество многослойных микросхем памяти типа HBM3. Еще раз повторюсь никакого электрического интерфейса за пределы подложки.

Модуль процессора может вообще не содержать значимых объемов оперативной памяти и оптимальной конфигурацией будет использование современной КЭШ памяти третьего уровня для организации системы

виртуальной памяти, хранения страниц виртуальной памяти (примерно, как в архитектуре X86). Для заполнения и вытеснения страниц виртуальной памяти использовать внешние модули оперативной памяти.

Наиболее оптимальным типом виртуальной распределенной памяти (мое мнение) является память с последовательной консистентностью. Такой тип памяти не гарантирует мгновенную одинаковость данных для всех копий разделяемого ресурса, но гарантирует что последовательность изменений будет строго одинаковой для всех клиентов. С учетом достаточно больших скоростей сети ССИ, состояние при котором происходит обновление будет очень небольшим и практически никак не зависит от числа копий. Время обновления зависит только от выделенной для конкретного разделяемого ресурса скорости и физического расстояния от исходных данных до копии.
Поскольку требуется соблюдение строгой последовательности обновлений, то должен быть модуль, который решает кто и в какой последовательности будет получать доступ к разделяемому ресурсу. Наиболее оптимальным будет назначить таковым, контроллер виртуальной памяти к которому присоединена разделяемая физическая память.
Кроме предоставления доступа, арбитража, рассылки обновлений, контроллер виртуальной памяти может предоставлять еще много различных «сервисов». Такими «сервисами» могут быть предоставление данных в формате и последовательности наиболее удобной для последующей обработки, возможно даже одновременно в разных вариантах для разных потребителей. Различных вариантов блокировок, приоритетов, предварительных подготовительных работ и предоставления данных к нужному времени и многое другое. Не будет преувеличением называть контроллер виртуальной памяти универсальным процессором управления и подготовки потоков обрабатываемых данных.

Число копий данных может быть очень большим и необходим механизм позволяющий осуществлять гарантированный доступ без излишнего резервирования ресурсов сети, но и с гарантией заранее заданной пропускной способности (гарантировать определенный минимум производительности). ССИ для этого предоставляет два уникальных механизма: возможность создавать канал передачи данных в виде дерева и асинхронный тип канала. Запросы на изменения «оригинала» данных от владельцев копий могут поступать достаточно асинхронно и потому их лучше передавать с использованием большого числа асинхронных каналов с некоторой гарантированной синхронной скоростью. Запросы на обновление данных в копиях, происходят достаточно часто (каждый запрос на изменение оригинала порождает запрос на изменение копии для всех владельцев копий данных), соответственно оптимальным будет один синхронный канал с достаточной производительностью. Для обеспечения сбалансированности системы, скорость синхронного канала должна быть пропорциональна сумме гарантированных скоростей асинхронных каналов. В таком случае контроллеру виртуальной памяти не потребуется буфер для хранения запросов из-за занятости канала

обновления данных.

Примерный алгоритма работы (опускаем подготовительные операции):

- Если процессор уже содержит локальную копию данных и ему требуется изменить конкретное значение. Будет создан запрос на изменение данных, установлена метка в локальной памяти о неактуальности содержимого данной ячейки для локального процессора, далее процессор продолжит свою работу, не дожидаясь завершения работы по актуализации данных (алгоритм КЭШ с прямой записью).
- Через некоторое время запрос будет получен модулем виртуальной памяти, хранящем «оригинал», обработан и в синхронный виртуальный канал будет отправлена информация для обновления данных во всех локальных копиях.
- Виртуальный канал рассылки обновлений строится как дерево, корнем которого является владелец «оригинала» данных. При поступлении данных в промежуточный коммутатор, они могут быть отправлены сразу в несколько выходных физических каналов нескольким коммутаторам одновременно (нет необходимости строить канал, последовательно обходящий всех владельцев копий или обращаться к ним последовательно). Такой подход к отправке уведомлений минимизирует задержку передачи, которая никак не зависит от числа копий.
- После получения от контроллера виртуальной памяти обновленных данных, происходит изменение содержимого локальной памяти и сброс сигнала о неактуальности конкретного слова (для владельца, который породил это изменение)
- Процессор, владеющий локальной копией (если он ее перед этим не изменял) свободно читает содержимое ячеек памяти, никак не синхронизируя с процессом их обновления. Успел прочитать не обновленные данные считаем, что чтение случилось раньше, чем другой процессор их изменил. Даже если по абсолютному времени данные уже были обновлены, но не доставлены конкретному владельцу копии.
- Процессор, владеющий копией (если он недавно обновил данные и имеется метка о неактуальности) может вести себя по-разному. Он может проверить значение метки и выполнить какое-то действие в соответствии со значением метки. Может просто попытаться прочитать значение ячейки памяти и быть «остановленным» до момента пока не будет сброшена метка о неактуальности (не получено именно его обновление от контроллера виртуальной памяти — своеобразная синхронизация вычислительного процесса относительно последовательности изменений).
- Можно и вообще принимать во внимание только локальную последовательность изменений данных.
- Если запрос на изменение потерян (ошибка передачи), то данное событие может быть детектировано в течении первых микросекунд по отсутствию

данных обновления на свой запрос изменения (таймаут времени неактуальности ячейки памяти)

Данный тип алгоритма достаточно прост в реализации и будет хорошо работать для высокоскоростной ССИ. Скорость в 100Т почти на три порядка превышают производительность локальной динамической памяти, время «открытия» новой страницы динамической памяти сопоставимо с временем распространения электромагнитной волны на расстояние в 10 метров, что делает распределенную виртуальную память сопоставимой по скорости с локальной динамической памятью и по факту лишает смысла нахождение динамической памяти рядом с процессором. Конкретные протоколы взаимодействия процессора и модуля оперативной памяти могут быть сильно разными, но все их объединяют и в значительной степени определяют сервисы сети ССИ.

Синхронная Символьная Иерархия
Создание единого физического потока из нескольких параллельных

Автор : Балыбердин А.Л.

Новосибирск, 2022 г.

Особенности построения высокоскоростного коммутатора.
Высокоскоростным коммутатором будем считать такой коммутатор, для которого за один такт синхронизации принимается больше одного символа на физический канал. Тактовая частота ядра коммутатора 4ГГц, символ 100 бит: пропускная способность физического канала более 400G будет требовать обработки более одного символа за такт.

Появляется вопрос: что делать если потребуются большие скорости физического канала, например 10T. Логика диктует увеличение числа одновременно обрабатываемых символов, для скорости 10T и частоты 4ГГц необходимо за один такт коммутировать по 2500 бит (25 символов). С точки зрения схемотехники шина данных шириной в 2500 бит не является чем-то запредельным (для HBM ширина шины данных может достигать 4096 бит). Физический канал передачи данных с такими скоростями состоит из большого числа параллельных линий, каждая линия передает один символ за такт работы коммутатора. С помощью программируемых задержек можно выравнивать моменты приема символов на всех линиях и добиться одновременного приема одновременно отправленных символов. Остается объединить их в единый физический канал (25 линий для скорости 10T), так что бы счетчик символов был один для всех линий.
Для ядра коммутатора такой подход выливается в необходимость увеличения ширины данных в число раз равное числу линий, составляющих отдельный физический канал с одновременным увеличением размера буфера в такое же число раз и пересчетом периода опроса буфера виртуального канала.
Если к буферу виртуального канала обращается низкоскоростной канал, то должна происходить операция с одним символом. Если обращается высокоскоростной канал, то операция с несколькими. Все остальные правила и алгоритмы остаются неизменными.

Синхронная Символьная Иерархия
Иерархия (объединение нескольких виртуальных каналов в один)

Автор : Балыбердин А.Л.

Новосибирск, 2022 г.

Под иерархией в традиционных синхронных коммуникациях, в большинстве случаев понимается объединение нескольких менее скоростных каналов в один высокоскоростной. В технологии пакетной коммутации понятие скорость физического канала размыта, наиболее близкой к понятию «иерархия» технологией будет инкапсуляция протоколов низкого уровня в протоколы более высокого уровня.
Для понимания необходимо прочитать : ССИ_Синхронный поток_Фрагментация потока (2).

Понятие иерархии в ССИ

Структура каналов в ССИ «инвертирована», относительно традиционных каналов передачи данных. В среднем будет наблюдаться ситуация, когда скорость соединения будет уменьшаться по мере удаления от источников потока данных (процессор, накопители, память и прочее). Если сказать по другому, то чем больше расстояние тем меньше скорость среднего виртуального (физического канала) канала.
Для ССИ под иерархией следует понимать объединение нескольких виртуальных каналов в один. Из статьи про фрагментацию физического потока можно сделать вывод о том, что основными ограничениями сети являются число модулей виртуальных каналов и пропускная способность физических каналов. Из принципа разделения физического канала можно сделать вывод: чем больше скорость виртуального канала, тем меньше задержка коммутации (соответственно задержка передачи). Значит есть необходимость объединять «дальние» низкоскоростные виртуальные каналы в один более скоростной, для экономии модулей виртуальных каналов и уменьшения времени коммутации. Кроме того, данный механизм будет служить базой для алгоритмов распределения трафика в больших сетях (можно назвать аналогом MPLS из Ethernet). Чем больше уровень иерархии (степень вложенности), тем больше размер сети, в которой такой канал применяется и тем выше уровень системы управления трафиком.

Алгоритм создания и декомпозиции суммарного канала.

1. Для объединения нескольких каналов в один, достаточно расположить их единым массивом (в пространстве состояний счетчика). В первом коммутаторе они будут считаться отдельными каналами.
2. В следующем коммутаторе сопоставить им один модуль виртуального канала, отзывающегося на весь диапазон состояний счетчика.
3. Для первого коммутатора в цепочке, там, где еще отдельные виртуальные каналы, есть небольшое отличие по вставке компенсирующих символов. При отсутствии данных в любом их буферов, необходимо выдавать служебный символ «буфер пуст» который не будет удаляться в процессе дальнейшей передачи, вплоть до момента декомпозиции суммарного канала. Применение такого символа необходимо для корректного обратного распределения данных по отдельным модулям виртуального канала на приемной стороне.

4. По причине превращения в не удаляемые, исходных компенсирующих символов, второй коммутатор (при чтении суммарного канала) должен увеличить скорость передачи суммарного канала на 500ppm и «штатно» вставлять уже обычные компенсирующие символы
5. В коммутаторе приемника необходимо снова создать отдельные модули виртуальных каналов для обратного распределения составляющих суммарного канала по отдельным модулям виртуальных каналов.
6. При проектировании модуля виртуального канала необходимо учесть, что на приемной стороне диапазон адресов может быть расположен не в том же месте, где он находится на стороне передатчика и необходимость удалять компенсирующие символы, вставленные предыдущим коммутатором. Соответственно последовательность записи в отдельные буфера будет не совпадать с последовательностью приемника, что потребует доработки базового алгоритма модуля виртуального канала.
7. Для получения возможности многократно объединять (многоуровневая иерархия) виртуальные каналы необходимо в тело не удаляемого символа «буфер пуст» добавить счетчик уровня иерархии. Данный счетчик будет увеличиваться на единицу при создании суммарного канала и уменьшаться на единицу при разборке канала. При достижении значения ноль, такой символ необходимо заменить на обычный компенсирующий символ. Такой же счетчик иерархии добавить и для остальных управляющих символов, влияющих на состояние модуля виртуального канала (например символ «удалить канал»)
8. Все остальные алгоритмы увеличения или уменьшения скорости и числа каналов должны действовать аналогично каналу первого уровня иерархии.

Данное описание нельзя считать полным алгоритмом, скорее это пожелания или границы, за которые нельзя выходить. Формирование полного алгоритма функционирования суммарных каналов возможно только после описания такого же алгоритма (последовательность работы и сигнальные символы) для иерархии первого уровня. Эти работы и необходимо выполнить **в рамках НИР**.

Синхронная Символьная Иерархия
Ethernet поверх синхронной сети

Автор : Балыбердин А.Л.

Новосибирск, 2022 г.

Предположим, что ССИ превосходит все существующие на данный момент сети передачи данных.

Как внедрять новую сеть?

Если попытаться «силовым» методом внедрять новую сеть на практике, то ничего не выйдет. Пользователи и производители скажут: *«Да, новая сеть хороша, но и уже существующие худо-бедно выполняют свои функции и менять ничего не будем».* Прецеденты такого поведения есть даже при внедрении минимальных изменений. Сеть IP V6 очень похожа на IP V4, но за 26 лет от момента своего создания доля ее трафика только превысила 30% (в среднем по миру согласно исследованиям компании GOOGLE).

Средний пользователь меняет оборудование только после выхода его из строя. Многим компаниям приходится использовать мошеннические приемы намеренного устаревания для того, чтобы обеспечить приемлемый период смены поколений. Такой подход требует очень много финансовых ресурсов и вызывает обоснованное недовольство пользователей.

Что делать?

Считаю, что необходимо внедрять новые технологии посредством метода «Троянского коня». Новое устройство проектировать так, чтобы в окружении «старого» оборудования оно полностью соответствовало старым протоколам. В случае если два и более новых устройства напрямую соединены кабелем, они превращаются в часть сети ССИ. При этом соединения со старыми протоколами остаются неизменными.

В чем плюсы такого подхода для пользователя: нет необходимости сразу заменять работающее оборудование на новое и при этом постепенно растет качество предоставляемых услуг (скорость, вероятность потери данных, задержки передачи и т.д.). Конечный пользователь не почувствует никаких отрицательных последствий перехода с одной технологии связи на другую, просто однажды «проснется» и обнаружит что старой сети нет.

В чем плюсы такого подхода для производителя оборудования:

- Нет необходимости в резком переходе на новые сети передачи данных.
- Нет необходимости одновременно поддерживать старую и новую линейку оборудования, замена модельного ряда происходит постепенно.
- Нет риска потерять покупателя из-за ошибок в проектировании нового устройства (всегда есть возможность продать старое устройство).
- Поскольку старые протоколы реализуются посредством виртуализации (эмуляции средствами ССИ и DataFlow), то это не вызывает значимого удорожания проектируемого устройства.
- За счет большей эффективности сети ССИ возможно произойдет существенное удешевление ЭКБ, используемой при проектировании, это «почистит» рынок от излишне «тяжелых» решений и ускорит переход на новую технологию связи.

- Самое главное у производителей, которые ранее не могли создавать свои микроэлектронные решения (отставание в технологиях, патентные препоны и т.д.) появится возможность догнать лидеров рынка.
- С большой вероятностью ССИ станет основой нового интерфейса процессоров, уйдет разнородный медный и заменится на оптический сетевой. В настоящее время многие вопросы будущего интерфейса уже освещены (описано в статье: ССИ_Дезагрегация оперативной памяти (6)).

Примерная реализации такого оборудования.

Преобразование асинхронного трафика в синхронный.
Ethernet является асинхронной пакетной сетью и для того что бы совместить ее с синхронными виртуальными каналами ССИ необходимо решить в какой момент (в каком месте) убирать асинхронность трафика. Наиболее оптимальным место является именно момент входа трафика в синхронную сеть, в этом месте можно посредством различных механизмов ограничения скорости потока (обратное давление для полудуплекса и т.д.) и относительно небольшого буфера регулировать скорость потока. На выходе в асинхронную сеть преобразование из синхронного потока в асинхронные особые проблемы с переполнением физического потока не предвидится (если следующее устройство принимает весь трафик), скорости суммируемых потоков регулируются на входе в ССИ. На входе в сеть ССИ буферизации подвергается относительно небольшой объем трафика. В таком случае размер буфера будет относительно небольшим и распределенным по всей сети, что позволит разместить его непосредственно на кристалле устройства. В настоящее время каждый коммутатор вынужден выравнивать скорости потоков, должен иметь достаточно большую по размеру буферную память, которая может вносить большую случайную задержку передачи.

Алгоритм обработки (преобразования) заголовков пакетов.
В отличии от синхронного канала, привязанного к конкретным приемнику и передатчику, асинхронный пакетный канал нуждается в маршрутизации каждого пакета индивидуально. Другими словами, необходимо анализировать заголовок каждого канала и направлять его в «свой» виртуальный канал. Физический уровень канала является общим (одинаковым) для старой и новой сети, до момента интерфейса потока символов. Новая аппаратура должна реализовывать, в дополнение к стандартной для ССИ обработке потока, протокол обслуживания старой сети с интерфейсом к коммутатору ССИ.
В момент выделения (приема) заголовка, необходимо активизировать механизм его обработки. Для простоты будем предполагать, что он реализован в виде логической схемы. Базовый для ССИ алгоритм очень простой и практически не увеличивает стоимость микросхемы при работе только в старой сети.
Поскольку заголовок пакета принимается относительно редко и при условии быстрой обработки (за один такт или конвейер), такое устройство может быть в единственном экземпляре для всех каналов IP V4. Результатом обработки будут

изменения в заголовке пакета (если необходимо) и перенаправление его в нужный виртуальный канал, связывающий данный порт с адресатом.

Эффект от применения сети ССИ.
Пока сеть ССИ не вышла за пределы кристалла особого внешнего эффекта нет и ее можно рассматривать как некоторую внутреннюю структуру или набор правил проектирования. Если появилось два или более устройств напрямую связанных между собой физическим каналом, то начинают появляться плюсы использования новой сети. В сети ССИ нет прямого взаимного влияния пользовательских потоков друг на друга, нельзя резким возрастанием скорости одного «задавить» трафик другого пользователя (переполнить буфера коммутатора). Время доставки и скорость становятся стабильными, а значит такие приложений как голосовое общение, видео будут работать стабильнее. По мере нарастания процента нового оборудования, эти эффекты будут проявлять во все большей степени, причем наибольшее влияние будет оказывать именно операторская его часть.

Регулирование пропускной способности всей совокупности виртуальных каналов
В больших сетях время сбора данных о загруженности отдельных физических линий, выработки решений и рассылки команд сильно отстает от развития текущей ситуации. К моменту реакции на перегрузку, она может закончиться по причине исчерпания быстрого и краткосрочного трафика. Перераспределение пропускной способности в большой сети происходит с неизбежной задержкой, обусловленной временем передачи информации по физическим линиям связи. В обычных асинхронных (Ethernet) коммутаторах пульсации трафика могут привести к переполнению буферов, потерям пакетов и существенному увеличению времени передачи пакетов. Для высоких скоростей изменения ситуации делать какой-то выделенный контроллер (дирижёр) смысла нет — пока он примет решение все уже успеет поменяться.

Сеть ССИ гарантирует, что созданный виртуальный канал будет работать вне зависимости от загрузки линии связи. Правда при этом не гарантирует, что можно создать новый канал (по причине отсутствия пропускной способности).

Асинхронность перенесена с каждого акта передачи (для пакетных сетей) на акт создания виртуального канала (сеть ССИ).

Если рассматривать пакетную коммутацию, то для каждого пакета есть вероятность потери, пакет (требование его передачи) асинхронен и в целом не предсказуем. Для управления такой системой в пределе требуется контроллер, который присоединен ко всем устройствам сети и мгновенно пересчитывает ее функционирование, а это невозможно (скорость света и т.д.). Поэтому идея программно-определяемых сетей (SDN), мертворожденная.

Для ССИ асинхронным событием является только запрос на создание виртуального канала. Передача данных (синхронная составляющая) для уже созданного канала гарантирована и не зависит от степени загруженности физического канала. Далее все просто — запрос на создание канала может быть отклонен по причине отсутствующей пропускной способности в одном из промежуточных узлов. Сам запрос отбрасывается, но статистика запросов накапливается и передается соседним узлам и учитывается в более высокоуровневом сервисе, формирующем маршруты каналов («карте» сети). Происходит достаточно быстрое (пусть и не мгновенное) перестроение карты и маршрутов, построенных с ее учетом. Получаем постоянно изменяющуюся распределенную систему управлению сетью, каждое устройство управляет собой, но при этом через «карту» учитывает состояние соседей. На такой «карте» кроме физического соединение указывается еще и загруженность линий связи. Более высокоуровневые системы анализа сети могут вмешиваться в подконтрольные им области и редактировать данные карты и правила доступа и т.д.

Локальная перегрузка сети приведет к небольшому (пропорционально времени перестроения карты сети) увеличению времени удачного создания виртуального канала, но никак не повлияет на работу уже установленных (если хозяин оборудования своим решением не разорвет соединение, но это осознанное решение). Через небольшое время создаваемые каналы начнут обходить «проблемное место». Скорее всего проблема и не возникнет, еще задолго до полного исчерпания пропускной способности данный механизм ее увидит и парирует.

Примерный алгоритм регулирования скорости конкретного канала.

В каждый канал ССИ (помимо полезной информации) можно добавлять служебные символы, в теле которых кодировать «пожелания» приемника и передатчика о параметрах и типе данного конкретного виртуального канала (Пример: хочу больше скорости, хочу: Постоянную скорость на длительный срок и т.д.). Каждый промежуточный коммутатор самостоятельно воспринимает поток таких символов и вносит предварительные изменения, резервирует на некоторое время ресурс. При этом совсем не обязательно именно столько сколько запросили (возможно даже уменьшит). Далее ожидает от источника трафика сигнала на фиксацию изменений, если такого сигнала не поступило резерв снимается.
Пример:
Пробежали такие символы от источника к приемнику и обратно. Источник увидел, что промежуточные коммутаторы готовы изменить параметры

виртуального канала. Источник помещает в передаваемые данные сигнал фиксации изменений и начинает передавать данные уже в соответствии с новыми параметрами виртуального канала. Данный запрос может быть исполнен поскольку уже предварительно согласованы со всеми промежуточными коммутаторами. Инициатором запроса на изменение параметров виртуального канала не обязательно должен быть именно источник данных, им может быть и промежуточный коммутатор испытывает потребность в выделении дополнительной пропускной способности более приоритетному виртуальному каналу или приемник данных, испытывающий трудности с обработкой потока принимаемых данных. Он добавляет или корректирует существующие запросы, требуя от уже существующих виртуальных каналов «отдать» часть ресурса согласно механизму приоритетов.
В результате работы всех алгоритмов получаем постоянно действующий и достаточно быстрый механизм перераспределения пропускной способности сети в целом.
Верхний уровень занимается созданием новых, группированием в суммарные или удалением не используемых виртуальных каналов. Оставить эту задачу на нижнем уровне нельзя, поскольку для ее решения требуется знание топологии сети и многие другие «административные» полномочия.

Итог:
Пока устройство работает в окружении сетей, использующих старые протоколы, оно ничем не отличается от своих «собратьев». Если появляется прямое соединение с устройством, поддерживающим сеть ССИ, происходит их объединение в кластер сети ССИ. По мере роста размера такого кластера (объединения в другими) происходит улучшение качества обслуживания и в момент появления прямого соединения через сеть ССИ пользователь получает весь объем сервиса новой сети. Для аппаратуры это приводит к отключению модулей, поддерживающих старые протоколы (до полного отключения питания). Если реализация старых протоколов производилась программным способом, то высвобождается еще и ресурс вычислителя.

Синхронная Символьная Иерархия системы промышленной автоматизации

Автор : Балыбердин А.Л.

Новосибирск, 2023 г.

Введение.
В настоящее время для систем промышленной автоматизации применяются операционные системы реального времени, позволяющие реагировать на получаемые от датчиков сигналы в пределах заданного интервала времени. Для построения сложных АСУ ТП необходимо объединять несколько управляющих систем в единую сеть. Сеть АСУ ТП тоже должна соответствовать некоторому наборы требований, таких как гарантия времени доставки, гарантия скорости, возможность построения резервных каналов и прочее.

В настоящее время абсолютное большинство сетей являются сетями, построенными на принципах пакетной коммутации, а пакетная коммутация не может предоставлять никаких гарантий по скорости, потерям и доставке. Можно сказать, что системы реального времени существуют, а вот сетей реального времени нет.
Сеть ССИ построена на принципах коммутации плезиохронных потоков данных и полностью соответствует самым жестким требованиям систем реального времени.

Какие плюсы можно получить при внедрении сети ССИ в современные системы промышленной автоматизации?
Используя в качестве коммутаторов устройства с сетевыми интерфейсами ССИ, можно создавать полностью синхронные распределенные системы передачи данных с гарантированной пропускной способность и латентностью, не зависящими от загруженности сети данными от других устройств. Иметь практически мгновенное оповещение о выходе из строя линий связи или управляющих контроллеров.

Можно добавить небольшое число «лишних» линий связи, так что была возможность построить несколько независимых соединений между взаимодействующими устройствами. Что позволит не терять данные в локальных сбоях в процессе передачи и строить механизмы реакции на такие события.

Если вспомнить историю развития электронных систем, то раньше электроника была существенно менее надежной и для получения требуемого качества работы применяли полное многократное дублирование с мажоритарными элементами для выявления ошибок. Современные контроллеры стали более надежными, компактными, быстрыми и из-за дороговизны от использования этой практики в большинстве систем отказались. Возможность новой сети позволяют ее возобновить без излишних финансовых затрат, получив существенное повышение надежности управляющих систем.
Большинство контроллеров обладают многократно большей, чем это необходимо для решения конкретной задачи производительностью и ее можно использовать для дублирования управляющих программ. Сеть ССИ позволяет одновременно передавать данные нескольким адресатам. Эти два свойства,

позволяют запустить копию одной и той же функции на нескольких физических контроллерах и контролировать не только одинаковость данных пришедших различными маршрутами, но и идентичность результатов работы ПО более нижнего уровня. В реальном масштабе времени можно производить дублирование критичных частей управляющего ПО и при выявлении сбоев в работе аппаратных средств, реализовывать различные виды корректирующих реакций. Процесс дублирования можно производить на всех уровнях системы управления. Реализовывать данный механизм нужно в виде стандартной библиотеки автоматизирующей большую часть работы разработчика системы. Кроме того, появляются достаточно простые механизмы для отладки и сохранения параметров работы системы в целом с привязкой к шкале реального времени, а не для отдельных контроллеров как сейчас.

Использование данных механизмов позволит эффективно создавать управляющие системы крайне высокой надежности без использования слишком большого объема дополнительных аппаратных средств.

Пути внедрения ССИ в современные системы промышленной автоматизации.
Идеальным решением для систем реального времени был бы прямой интерфейс к сети ССИ (сетевая карта или модуль с данным интерфейсом), но пока таких промышленных контроллеров нет. Для того что бы данная проблема не тормозила внедрение сети ССИ, необходимо найти такое решение когда интерфейс ССИ имеет только коммутатор, а вся остальное оборудование только стандартные для микроконтроллеров и промышленных компьютеров интерфейсы (Ethernet, RS-485, CAN и т.д.).

Для примера опишем подключение микроконтроллера через интерфейс RS485 (half-duplex).
Поскольку свойством гарантии доставки в реальном времени обладает только устройство с реализованным интерфейсом в сеть ССИ, то выбираем тип связи Master-Slave. Мастером является коммутатор, вернее модули RS-485 реализованные в структуре коммутатора. Подчиненными, все устройства, подключаемые к этим каналам. Алгоритм доступа к каналу получается не сильно отличным от канонического, подчиненное устройство получает доступ к каналу только после разрешения от мастера. Виртуальные каналы ССИ имеют свойство передавать символы даже если нет передачи реальных данных, что гарантирует периодическую отправку символов в сторону подчиненных устройств. Для единообразия добавляем обязательный ответ от подчиненного устройства мастеру, содержащий данные для передачи. Если в течении установленного срока ответного пакета нет, то считается что данное конкретное устройство отсутствует или вышло из строя. Остальные правила обмена будут сформулированы в процессе создания протокола.

Примерный алгоритм обмена данными.

1. Модуль RS-485, в составе коммутатора ССИ, постоянно мониторит состояние FIFO виртуальных каналов данных, предназначенных для передачи подчиненным устройствам.
2. В момент появления данных происходит формирование информационного пакета с адресом конкретного устройства на шине RS485 и принятыми из сети ССИ данными, с дальнейшей его передачей через интерфейс RS485 подчиненному устройству.
3. Блок RS485 на подчиненном устройстве, принимает некоторое число данных (заголовок пакета) и инициирует прерывание работы микроконтроллера для декодирования заголовка пакета.
4. Обработчик прерывания проверяет принадлежность пакета данному устройству и если адрес соответствует, то начинает готовить пакет с ответными данными (данные будут отправлены в сеть ССИ).
5. После завершения приема входящего пакета, подчиненное устройство отправляет ответ.
6. В свою очередь коммутатор ССИ, после передачи пакета ожидает ответ. Если в течении некоторого времени его нет, то в соответствующий виртуальный канал (закрепленный за данным адресом RS-485) помещается символ, сигнализирующий об ошибке связи.
7. Если пакет принимается, то данные из этого пакета извлекаются и помещаются в буфер виртуального канала.

Синхронная Символьная Иерархия
Символьное представление данных

Автор : Балыбердин А.Л.

Новосибирск, 2023 г.

Требования к кодированию передаваемых данных и их реализацию:

- Данные цифровые: используется цифровой канал передачи данных.
- Данные могут быть синхронными и асинхронными: к каждой единице данных добавляем идентификатор (0 — асинхронные, 1-синхронные). *В современной вычислительной технике такого понятия нет.*
- Данные представляются значением или типом: добавляем идентификатор (0 значение, 1- тип данных). *В современной вычислительной технике все данные являются просто числами и интерпретируются только обрабатывающей их программной, что приводит к множеству ошибок.*
- Для экономии «транзисторов», считаем, что данные всегда передаются младшим битом вперед. Последовательность данных от первого до последнего, по порядку.
- Логические данные необходимо «разложить» по «физическим» символам, которые могут быть разными для разных типов аппаратных систем.
- Локальные данные могут иметь битовую емкость больше «физического» символа, занимать больше одного «физического» символа. Добавляем идентификатор, указывающий на завершение текущего символа (0 — будет продолжение в следующем «физическом» символе, 1 — завершение текущего символа символа).
- Битовая емкость логического символа может быть как меньше «физического» символа, так и не кратными размеру «физического» символа. Добавляем идентификатор, указывающий что данные являются значащими (0 - данные пропускаются, 1 - данные используются в обработке). Если есть хоть один ноль в четвертом бите атрибутов, то передаваемые данные целиком поместились в «физический» символ. Если нулей нет, то данные будут продолжены в следующем символе (даже если там будет ноль передаваемых битов).

Получаем, что каждому биту передаваемых данных нужно добавить четыре бита атрибутов, что приведет к крайне неэффективному расходованию пропускной способности канала передачи данных. К счастью, атрибуты будут очень часто повторяться, битовый размер данных в большинстве случаев существенно больше одного.

Предварительно получается, что можно добавлять биты атрибутов только в момент их изменения, но и тут есть проблема: Символ, который определяется как последовательность бит атрибутов плюс битовое значения данных постоянно будет разного размера и нужно как-то определять где начинается новый символ.

С другой стороны, аппаратура передачи данных, микросхемы памяти и прочее работают с постоянным, пусть и не всегда одинаковым размером символа.

Наиболее оптимальным решением будет каждый «аппаратный символ»

символ начинать с битов атрибута, такая аппаратная привязка местоположения атрибутов будет максимально эффективной. Для выравнивания битового размера символа использовать размножение последнего атрибута, помечая лишние биты как не используемые и сразу за единичным битом до конца «физического» символа будут размещены действительные данные. Если символ перемещается через границу между аппаратурой с различным размером «физического» символа, то необходимо производить преобразование размера символа на лету. Разницу в битовом размере, а значит и накладные размеры, необходимо учитывать при вычислении скорости передачи.

Есть два условия, которые лимитируют размер символа снизу и сверху. Слишком маленькие символы будут иметь большой процент битов атрибутов в своем составе (предел 1 значащий бит на 4 бита атрибутов). Слишком большие символы будут иметь много битов пропуска данных, не используемых для кодирования значений данных.

Итог: Размер символа, соответственно битовый размер данных, может быть любым (оптимальным для конкретной выполняемой задачи). Да есть некоторые потери, не используемые биты, но они контролируемы через изменение размера аппаратных символов.

Следует пояснить понятие «тип данных»
В современной вычислительной технике структуры данных никак не выделяются в памяти или при передаче. Результат интерпретации полностью зависит от программы, которая будет обрабатывать эти данные. Для того что бы снизить вероятность ошибки приходится вводить различные контрольные суммы, указатели размеров, синхронизирующие и кодовые последовательности. Такой подход не решает проблему в целом, только уменьшает вероятность возникновения ошибки.

Понятие «тип данных» можно кратко интерпретировать как способ разбить единый поток символов на составляющие его «предложения».

С другой стороны, можно считать, что список типов данных является своеобразным API распределенной вычислительной системы. Функции API подразумевают передачу параметров, их можно передавать как отдельными символами данных (следуют за символами типа данных), так и в виде «интегрированных» в тело символа самого типа данных (размер символа не ограничен и данные добавляются со стороны младшего значащего бита). В таком случае тип данных может быть различного размера (передается младшими битами вперед). Интерпретация начинается в момент окончания приема именно логического символа, в таком случае все необходимые данные уже находятся в буфере приемника. Необходимость размешать данные такого составного типа в буфере приемника ограничивают длину такого символа.

Изначально никаких требований к «стандартизации» списка такого API нет. Считаем, что требования к функциональной полноте и возможности обратного преобразования объектов позволит выполнить автоматическое преобразование из одного локального API в другое.