Система контроля и управления транспортными потоками

В системах контроля и управления транспортными потоками, используемых для эффективного менеджмента систем общественного транспорта, индикации пробок, управления светофорами, автострадами и стоянками, огромное количество видео и статистических данных должно быть быстро и надежно обработано в режиме реального времени.

Подготовлено по материалам компании Advantech, *info@sea.com.ua*

Для создания эффективной системы, работающей в жестких условиях, требуются физические среды передачи данных, обладающие высокой пропускной способностью и производительностью, такие как оптоволокно и кабель UTP.

При построении систем контроля и управления транспортными потоками огромное количество видео и статистических данных должно быть обработано быстро и надежно в режиме реального времени. Поэтому еще на стадии проектирования необходимо рассмотреть возможность расширения системы при сохранении ее стабильной работы. Необходимо также использовать резервирование путей для передачи данных в случае отключения узла, при этом обеспечивать достаточную пропускную способность каналов для передачи видео потоков.

Система видеонаблюдения на автостраде А9 в Германии

Автобан А9 в Германии, длиной 529 км, проходит от Берлина до Мюнхена через города Лейпциг и Нюрнберг. Для управления всем автобаном требовалась новая и усовершенствованная система видеонаблюдения. В решении, предоставленном компанией Advantech, было предложено внедрить высокоскоростную Ethernet-магистраль в систему управления и мониторинга транспортными системами. Так, коммутаторы компании Advantech серии EKI-93xx поддерживают гигабитные скорости передачи данных, имеют возможность кольцевого резервирования и гарантируют надежную и стабильную работу сетей для больших систем видеонаблюдения.

Обеспечивая надежное подключение между IP-камерами, контроллерами и центром управления, а также удовлетворяя самым последним требованиям к сетевым технологиям, коммутаторы EKI-9316P/EKI-9312P компании Advantech являются идеальным решением для обновления существующих сетей до гигабитных скоростей или создания новых систем на основе технологии Gigabit.

Основные характеристики мультипортовых гигабитных Ethernet-коммутаторов EKI-9316P/9312P:

- 12/16 гигабитных портов Ethernet для увеличения скорости видеопотока;
- 4 оптических порта подходят для сетей большой протяженности;
- 8/12 PoE-портов, совместимых со стандартами IEEE 802.3 af и at, для подключения 12 IP-камер или поворотных камер видеонаблюдения для наилучшего качества наблюдения.
- энергопотребление 294,22 Вт при полной нагрузке 6 PoE/PoE+ (EKI-9316P) и 203,42 Вт при 4 PoE/PoE+ (EKI-9312P), при этом собственное потребление ~21,82 Вт (при рабочей температуре от -40 до 75°C).
- двойной образ прошивки: в случае возникновения ошибок при выполнении работы активного образа, коммутатор EKI-9316P/9312P переключается на резервный образ для большей надежности системы.
- создание и конфигурация резервного образа, восстановление и обновление через USB2.0

Особенности ПО EKI-9316P/9312P

1. Использование резервирования: объединение колец. Функция объединения колец (рис. 2) позволяет добавить возможность резервирования для подключенных групп устройств. Резервированная передача данных между двумя группами позволяет поддерживать связь в случае возникновения ошибки связи.

2. IGMP-приложения. IGMP (протокол управления группами интернета) может быть использован в сетевых приложениях типа «один-ко-многим», таких как онлайн потоковое видео и онлайн-игры, и обеспечивает более эффективное использование ресурсов при поддержке данных типов приложений. Пример на рис. 4 показывает применение IGMP в системах видеонаблюдения.

3. Управление PoE. В приложениях реального времени каждое устройство имеет различные требования к питанию. Пользователи могут управлять и контролировать PoE через специальный интерфейс, например, следить за количеством потребляемой электроэнергии.

В процессе настройки или работы коммутаторов, при возникновении каких-либо неполадок обязательно проверьте следующие настройки:

1. Проверьте системную информацию (system information, рис. 8) для

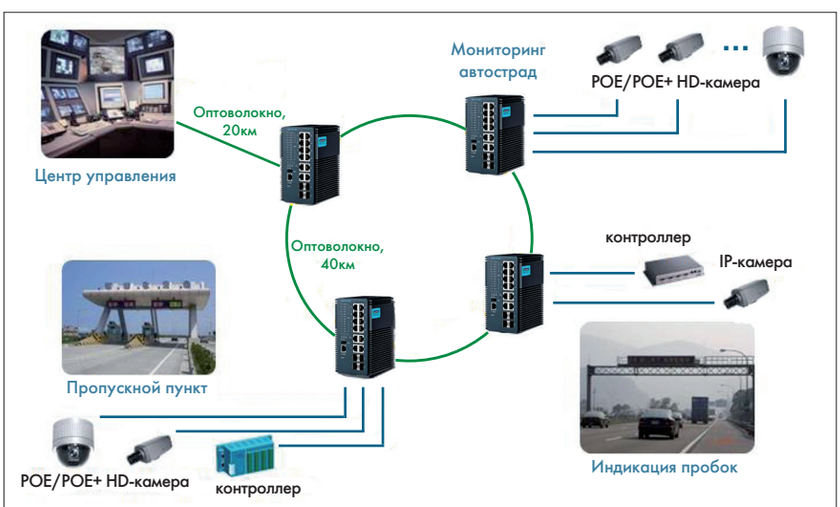

Рис. 1. Система видеонаблюдения на автостраде А9 в Германии.

предотвращения базовых ошибок, таких как конфликт IP-адресов.
2. Проверьте статистику подключений к портам (port statistics) для отслеживания ошибок подключения.
3. Проверьте программные настройки (software function) для поиска ошибок в конфигурации оборудования.

Потоковая безопасность в системах сбора данных

Потоковая безопасность – это концепция программирования, применимая к многопоточным программам. Когда программа разработана с учетом принципов потоковой безопасности, она может работать сразу с несколькими процессами, вызываемыми одновременно. Такой подход гарантирует одновременное исполнение программой нескольких задач без возникновения ошибок, таких как перезапись данных, конфликт памяти или проблема распределения ресурсов. Важность потоковой безопасности приобретает все большее значение с увеличением количества многопоточных программных сред и многоядерных процессоров.

Без учета принципов потоковой безопасности в многопоточной системе могут возникнуть условия конкуренции при чтении и записи данных. К примеру, если два потока одновременно обращаются к одному адресу памяти, может возникнуть два варианта записи значения в переменную (рис. 11). В результате нельзя предсказать точно, будет ли переменная *integer value* равна 1 или 2. Это явление называется состоянием гонки.

Программисты всегда стремятся избежать состояния гонки. Данную ошибку очень сложно исправить, так как конечный результат является непредсказуемым и зависит от соотношения времени обращения к памяти различными потоками. Поэтому важно избежать возникновения этого состояния на начальном этапе программирования. Если потоковая безопасность при создании мультипотоковых программ не была прописана в драйверах должным образом, возможно возникновение серьезных ошибок, таких как системные сбои или повреждения общей памяти.

В промышленных и измерительных приложениях поставщики аппаратных решений для систем сбора данных, такие как Advantech, предоставляют инженерам драйверы и комплекты средств разработки для создания собственных программ. Многопоточное программирование становится все более популярным, так как это позволяет использовать всю мощность новейших процессоров,

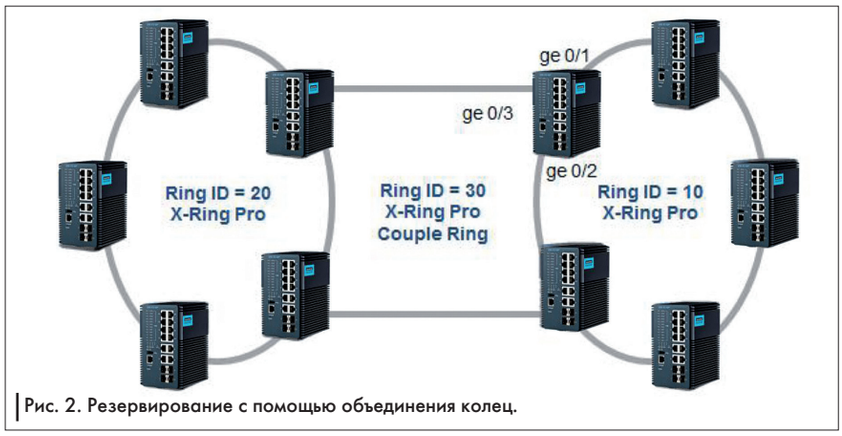

Рис. 2. Резервирование с помощью объединения колец.

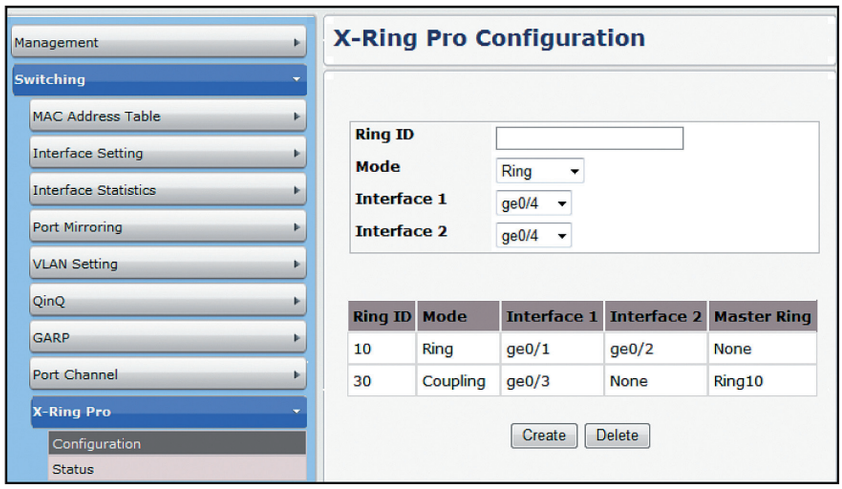

Рис. 3. Конфигурация кольцевого резервирования через Web-интерфейс.

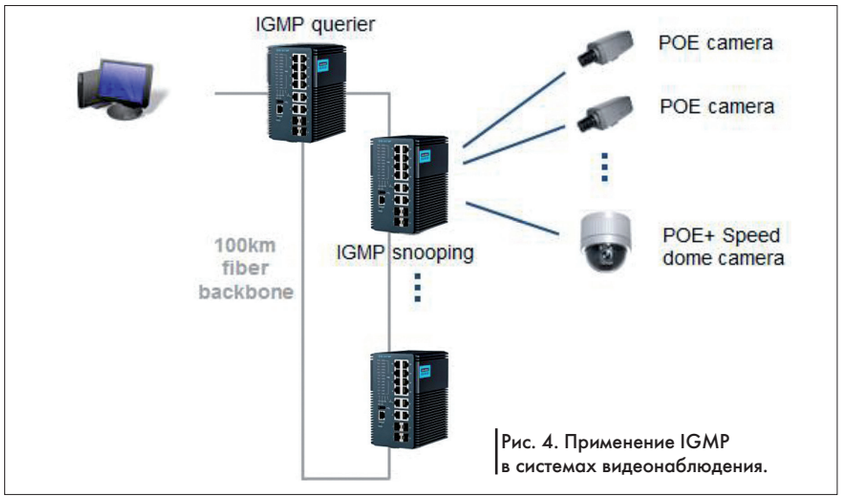

Рис. 4. Применение IGMP в системах видеонаблюдения.

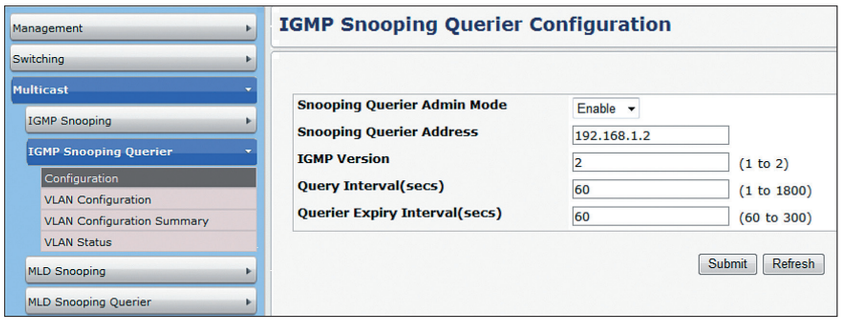

Рис. 5. Веб-конфигурация IGMP-запроса.

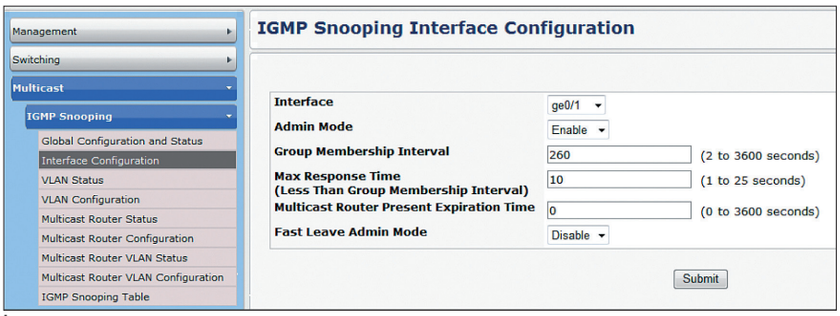

Рис. 6. Отслеживание трафика IGMP: веб-конфигурация.

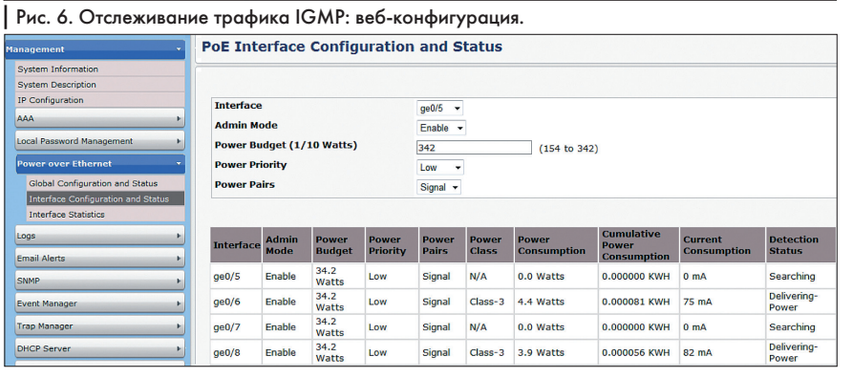

Рис. 7. Web-интерфейс для конфигурации PoE.

увеличивая производительность измерительных приложений и систем управления. При этом разработчики ПО должны использовать механизм потоковой безопасности с тем, чтобы поддерживать надежность системы на высоком уровне и иметь возможность увеличивать многопоточность системы.

Если драйверы устройств не используют технологию потоковой безопасности, пользователи – инженеры, создающие приложения для данного оборудования – должны самостоятельно внедрить потоковую безопасность в свою программу. Это называется «создавать потоковую безопасность в пользовательском режиме» (или на уровне приложений), в то время как встроенная безопасность на уровне драйверов называется «режимом ядра» (уровень ядра).

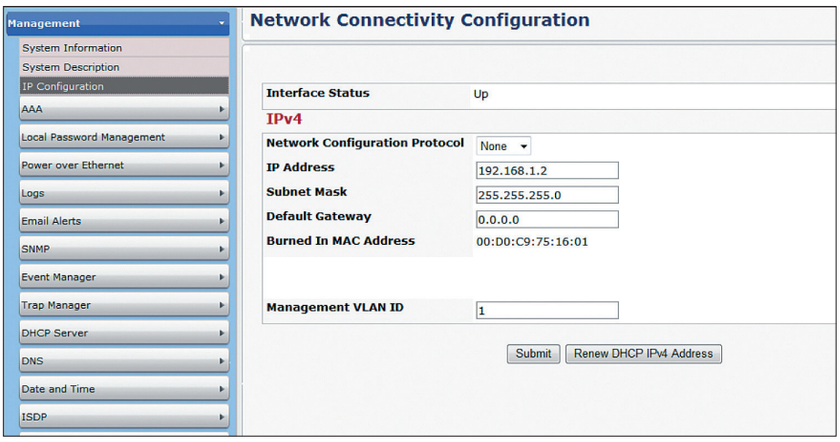

Рис. 8. Проверьте системную информацию для предотвращения базовых ошибок, таких как конфликт IP-адресов.

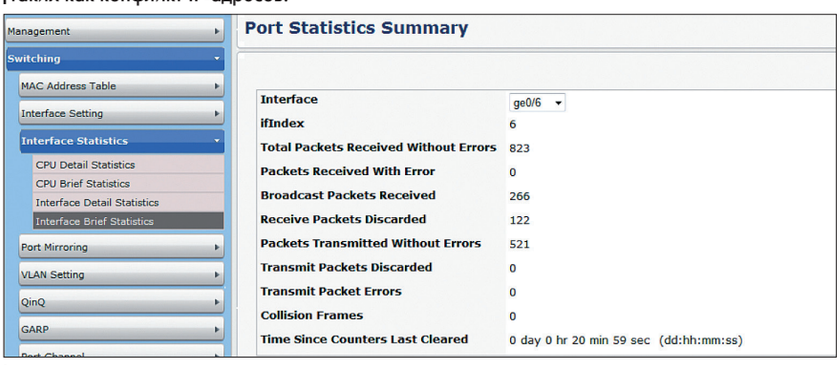

Рис. 9. Проверьте статистику на портах (port statistics) для отслеживания ошибок подключения.

Существует несколько способов создания потоковой безопасности на уровне пользователя.

1. Локальная память потока. Переменные в программе локализованы, поэтому каждый программный поток работает с собственными копиями данных. Эти переменные сохраняют свои значения в рамках подпрограмм внутри кода. Такое разделение позволяет программе обеспечить потоковую безопасность, так как переменные хранятся и обрабатываются независимо друг от друга, даже если код, работающий с ними, вызывается другим потоком. Таким образом, локальная память защищает общие данные или переменные от конфликта записи.

2. Взаимное исключение. Взаимное исключение обеспечивает последовательный доступ к общим ресурсам (данным или устройствам). Специальный программный механизм гарантирует, что только один поток читает или пишет значения в общие данные или устройства в каждый момент времени. Для реализации данного механизма используется два распространенных метода – семафор и мьютекс. Оба этих метода работают подобно светофору, управляющему доступом к общим ресурсам для нескольких процессов одновременно. При использовании данного метода необходимо иметь в виду, что неправильная реализация может создать негативные последствия, такие как взаимная блокировка.

3. Атомарные операции. Все общие данные или переменные должны поддерживать атомарные операций, которые не могут быть прерваны другими потоками. Эти операции обращаются к критическим секциям, запрашивающим общие ресурсы (данные или устройства), которые не должны быть одновременно доступны более чем одному потоку.

Потоковая безопасность на пользовательском уровне и в режиме ядра

Разработчикам приложений систем сбора данных очень сложно на пользовательском уровне создавать потокобезопасные программы. Гораздо проще, если поставщик оборудования заранее предусмотрел возможность потоковой безопасности в режиме ядра. Во-первых, разработчики оборудования лучше знают собственный продукт и гораздо быстрее смогут выявить

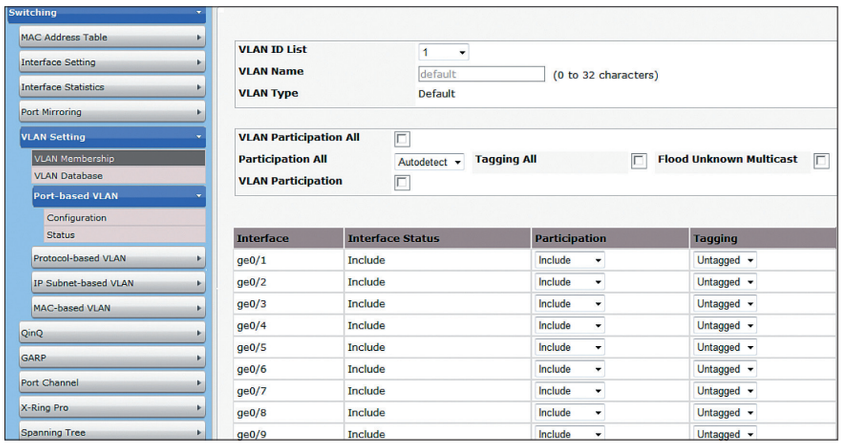

Рис. 10. Проверьте программные настройки (software function) для поиска ошибок в конфигурации оборудования.

Thread 1	Thread 2		Integer value
			0
read value		←	0
increase value			0
write back		→	1
	read value	←	1
	increase value		1
	write back	→	2

Thread 1	Thread 2		Integer value
			0
read value		←	0
	read value	←	0
increase value			0
	increase value		0
write back		→	1
	write back	→	1

Рис. 11. Состояние гонки для многопотокового программирования.

часть кода, отвечающую за потоковую безопасность. Кроме того, после реализации данного механизма необходимо провести множество тестов программы. Например, программа должна быть протестирована на различных платформах в течение долгого времени, чтобы убедиться в стабильности работы приложения. Такие тестирования гораздо легче организовать разработчику систем сбора данных.

Реализация данного механизма на уровне ядра может включать создание задач с различными уровнями приоритетов, которые увеличивают безопасность потоков. Таким образом, осуществить потоковую безопасность в режиме ядра легче, чем на пользовательском уровне, что позволит разработчикам приложений упростить написание программ и сэкономить время. Кроме того, потоковая безопасность, реализованная на уровне ядра, имеет более высокую производительность, в то время как ее реализация в пользовательском режиме обычно уменьшает эффективность при выполнении программы.

Компания Advantech использует последние тенденции отрасли и учитывает требования большинства пользователей. Для того, чтобы помочь разработчикам сократить время программирования систем, драйвер DAQNavi компании Advantech обеспечивает потоковую безопасность на уровне ядра, что позволяет программистам не задумываться о способах реализации потоковой безопасности в пользовательском режиме.