Моделирование, анализ и оптимизация протокола TCP

Нури Абдель Жабар
ДонНТУ ВТ97в
abdel_nouri@hotmail.com

Abstract

Nouri Abdel jabar. TCP Modeling, analysis and optimization. The transport control protocol (TCP) is intended for use as the reliable protocol of dialogue between hosts - computers in communication computer networks with packet switching, and also in systems uniting such networks.
As an overall objective of the protocol TCP it to ensure reliable service for the communications between processes in multi-network system and to be the common protocol for the communications between hosts-computers in set of networks, that‘s why it is necessary to analyze optimize some areas of this protocol, for example self-synchronization of the congestion avoidance process. For this purpose it is possible to use MATLAB as a means of imitating modeling.

Цель

Использование программного обеспечения MATLAB (модули SIMULINK и STATEFLOW) для разработки модели протокола TCP на основы сетей Петри. Анализировать некоторые алгоритмы управления перегрузки в сети, разработанных в новых версиях протокола TCP и с помощью разработанной модели проверять эффективность этих алгоритмов.

Реализация сетей Петри с помощью Stateflow

Stateflow это интерактивный инструмент проектирования для моделирования и имитации сложных систем. Тесно интегрированный с Simulink® и MATLAB®, Stateflow дает возможность элегантно проектировать встроенные системы, эффективно объединяя сложную логику управления и наблюдения внутри Simulink моделей[9]. С помощью Stateflow можно быстро разрабатывать графические модели управляемых событиями систем, используя теорию конечного состояния машин, формализм диаграмм состояния и нотацию потоковых диаграмм. Комбинация Stateflow, Simulink и MATLAB создает уникальную интегрированную среду моделирования, в которой можно моделировать, имитировать и анализировать сложные динамические системы.
Основой сети Петри является понятие условно-событийной системы. Компоненты системы и их действия представляются абстрактными **событиями**. Событие может произойти (реализоваться) один раз, повториться многократно или не произойти ни разу. Совокупность действий, возникающих как реализация событий при функционировании системы, образует **процесс**, порождаемый этой системой. В общем случае одна и та же система может функционировать в одних и тех же условиях по-разному, порождая некоторое множество процессов[11][12]. Для того, чтобы событие произошло, необходимо появление ситуации, в которой это событие может быть реализовано. При этом ситуация определяется как совокупность некоторых **условий** возникновения события. То есть событие реализуется, если выполнены условия его реализации. Условие может иметь емкость: условие не выполнено (емкость равна нулю), условие выполнено (емкость равна единице), условие выполнено с n-кратным запасом (емкость равна n, где n - целое положительное число).В результате совершения события происходит изменение условий и наступление пост условий реализации события.
В результате развития аппарата сетей Петри был разработан ряд расширений. Од из наиболее мощных — E-сети (evaluation — «вычисления», «оценка») — «оценочные сети». С помощью

Stateflow можно легко реализовать E-сети на основе блоков состояния и условных/безусловных переходов[10].

T/F/J-переход

T-переход («исполнение», «простой переход»). Его графическое представление аналогично представлению вершины-перехода сети Петри .
Переход срабатывает при наличии метки во входной позиции и отсутствии ее в выходной позиции. Формально это можно записать так: (1;0) |— (0;1).
F-переход позволяет отразить в модели занятость некоторого устройства (подсистемы) в течение некоторого времени, определяемого параметром t .
F-переход («разветвление») срабатывает при тех же условиях, что и T-переход:
с содержательной точки зрения, F-переход отображает разветвление потока информации (транзактов) в системе.
J- *переход* («объединение»), переход срабатывает при наличии меток в обеих входных позициях и отсутствии летки в выходной позиции:
(1.1:0) |— (0.0:1)
Он моделирует объединение потоков или наличие нескольких условий, определяющих некоторое событие. Пример реализации T/F/J-переход в среде STATEFLOW на рисунке 1

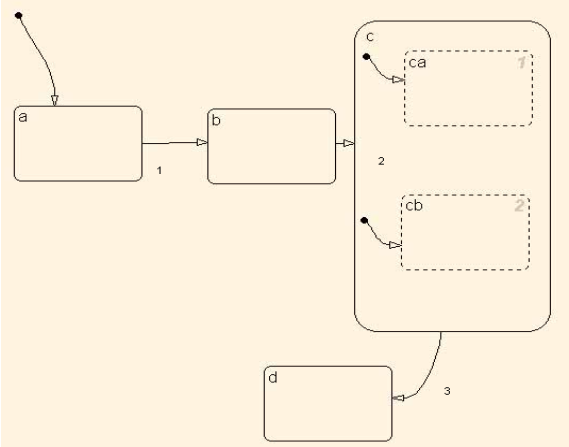

Рисунок 1 : 1-2-3 Пример реализации T-перехода/F-перехода/J-перехода в среде STATEFLOW.

X-переход

X-переход («переключатель»). По сравнению с тремя предыдущими типами переходов, он содержит дополнительную управляющую («разрешающую») позицию.
Логика его срабатывания задается следующими соотношениями:
X-переход изменяет направление потока информации (транзактов). В общем случай разрешающая процедура может быть сколь угодно сложной, но сущность ее работы заключается в проверке выполнения условий разветвления потока (с точки зрения программиста, разрешающая позиция аналогична условной инструкции типа if.
Пример реализации X-переход в среде STATEFLOW на рисунке 2

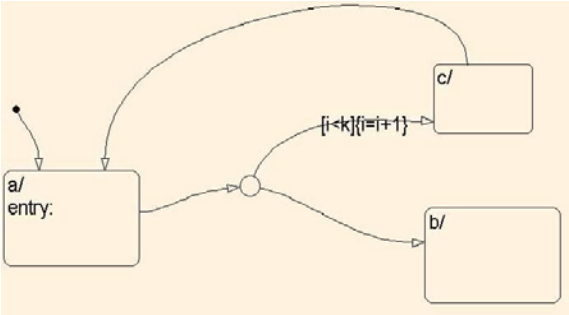

Рисунок 2: Пример реализации X-перехода в среде STATEFLOW.

Y-переход

Y-переход отражает приоритетность одних потоков информации (транзактов) по сравнению с другими. При этом разрешающая процедура может быть определена различным образом: как операция сравнения фиксированных приоритетов меток как функция от атрибутов меток (например, чем меньше время обслуживания, выше приоритет). В некотором смысле он работает аналогично инструкции выбора типа case. В таблице1 используется следующая логика (приоритет перехода,состояние1, состояние2,состояние3).

Таблица 1:Логика срабатывания X-перехода:

(0,1,1,0)	(0,0,1,1)
(0,1,0,0)	(0,0,0,1)
(0,0,1,0)	(0,0,0,1)
(1.1,1,0)	(0,1,0,1)
(1,1,0,0)	(0,0,0,1)
(1,0,1,0)	(0,0,0,1)

Пример реализации Y-переход в среде STATEFLOW на рисунке 3

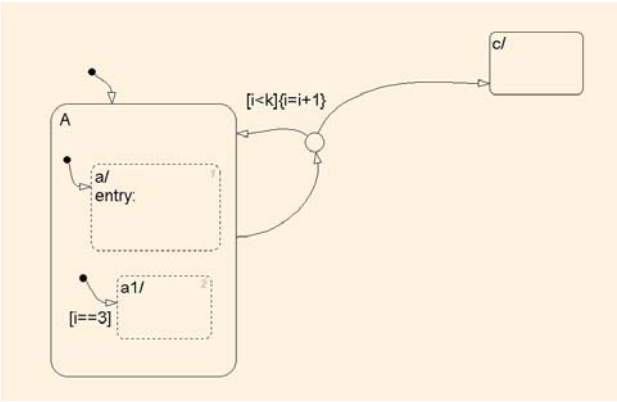

Рисунок 3: Пример реализации У-перехода в среде STATEFLOW.

Для реализации модели протокола в предыдущем простом случае, достаточно с одной стороны моделировать поведение протокола, поскольку требуется от станции B только прием пришедших сегментов и отправка их подтверждения то есть следующий номер в очереди в зависимости от длины принимаемого сегмента.
Для этого, весь процесс обработки сегментов достаточно реализовать в среде SIMULINK :

Это будет подсистема станции B (Sub_system_station_B), она имеет следующие параметры:

Входные параметры[11][12] :

- RSEG_CTL : Тип принимаемого сегмента (ACK,FIN,SYN,…).

- RSEG_LEN : Длина принимаемого сегмента .
- RSEG_WND : Размер плавающего окна.
- RSEG_SEQ : Номер последовательности принимаемого сегмента .

Выходные параметры :

- SEG_CTL : Тип отправляемого сегмента (ACK,FIN,SYN,…).
- SEG_ACK : Номер следующего сегмента в последовательности .
- SEG_WND : Размер плавающего окна.

Для простоты подсистема станции B не изменяет тип сегмента и размер плавающего окна, а только вычисляет номер сегмента в последовательности который она будет ожидать.
Схема подсистемы показана на рисунке 4.

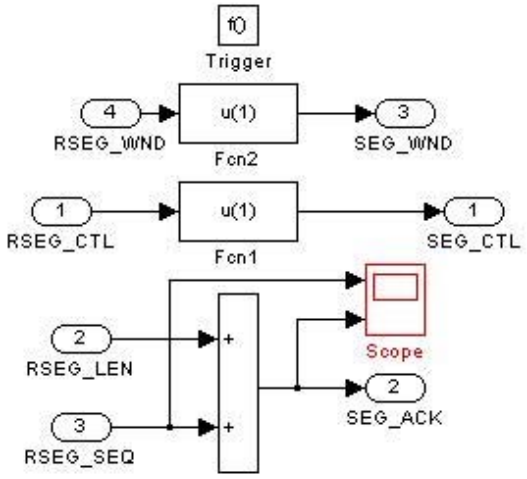

Рисунок 4: Подсистема станции B.

Что касается станции A, она моделирует поведение протокола в стадии синхронизации, установленного соединения также закрытия соединения.

Модель станции A реализуется с помощью компонента SIMULINK'a STATEFLOW :
Элемент Chart представляет собой событийное поведение работу протокола на станции A, чтобы работа модели приближалась к реальным условиям, используются элементы памяти (Memory) и элементы формирования задержек (Variable integer delay) что приводит к тому что ответы от станции B приходят на станцию A с контролируемой задержкой пользователем модели.
Схема модели приведена на рисунке 5.

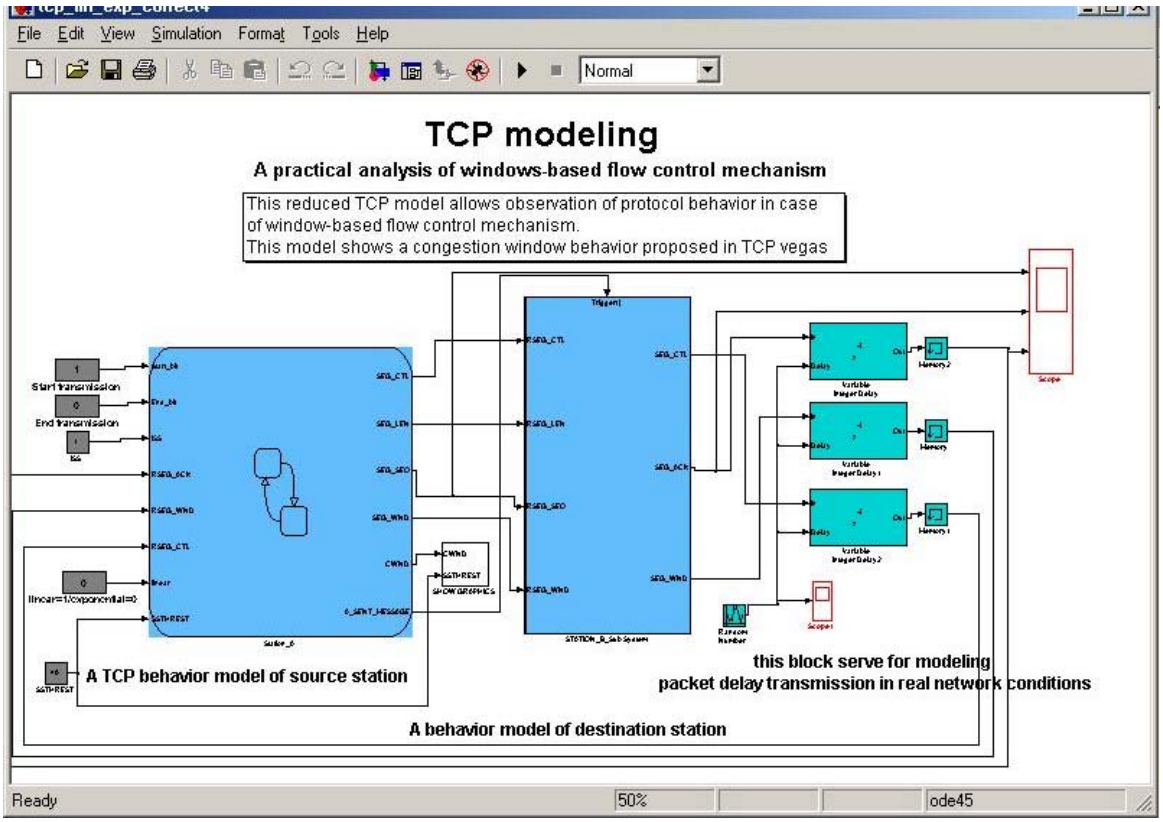

Рисунок 5: Схема модели.

Реализованные алгоритмы управления перегрузки

В первоначальной модель реализуется два простых алгоритмов управления перегрузки :

- Линейный алгоритм роста окна перегрузки. Рисунок.
- Экспоненциальный алгоритм роста окна перегрузки.

В первом алгоритме CWND инкрементируется каждый раз когда станция A принимает подтверждение какого-нибудь сегмента до достижения порогового значения перегрузки которое предлагает сеть в зависимости от количества станций отправителей подключенных к станции принимателя или к промежуточному маршрутизатору.
Второй алгоритм имеет экспоненциальный характер поскольку CWND инкрементируется или по экспоненциальному закону или в два раза т.е CWND(k+1) = 2 * CWND(k).

Обзор развития алгоритмов управления перегрузки

Основная проблема состоит в том, что протокол TCP при своей работы не взаимодействует с сетью для определения оптимальных параметров передачи или по крайне мере, для определения оптимальной скорости своего трафика.
Для реализации нового алгоритма добавляется входной параметр SSTHREST в блок Chart станции A, т.е пороговое значение окна перегрузки поскольку в данной реализации протокола, TCP выполняет алгоритм управления перегрузки после того CWND достигает заранее установленного значения, и это есть механизм обратной петли (feed-back mechanism).[2][5][6][7][8]Хотя это не является реальным условия в котором работает протокол но зато на этом этапе можно проверить и анализировать другие аспекты протокола, также
выходной параметр CWND из блока Chart станции A который поступает в подсистему Show_Graphics для просмотра этого параметра.

Также изменена система формирования задержки кадра в процессе трансляции т.е задержка кадра постоянно изменяется с помощью генератора случайных чисел который поступает в компоненты формирования задержки, после того мы получаем разные случайные задержки для каждого кадра.

Подсистема Show_Graphics состоит из следующих компонентов SIMULINK'а :

- Два входных порта :
- SSTHREST : пороговое значение перегрузки.
- CWND : размер окна перегрузки.
- Цифровой генератор временных импульсов.
- Компонент формирования производной функции.
- Компоненты визуализации дискретных и непрерывных сигналов.

И так для визуализации рост окна перегрузки, подключены порты CWND и цифровой генератор временных импульсов к компоненту XYGraph, CWND подключен к $\frac{du}{dt}$ которого подключен к SCOPE вместе с CWND для визуализации скорость изменения роста окна перегрузки.
Также при необходимости подключается CWND и SSTHREST через элемент (&) к компоненту STOP чтобы установить моделирование .(рисунок 6)

-

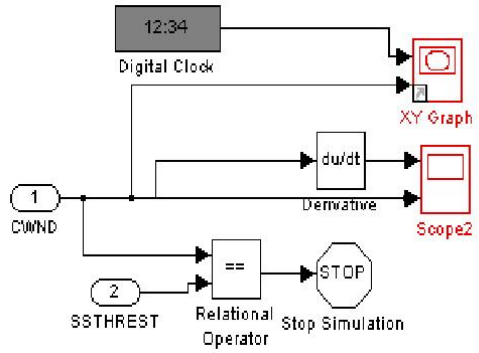

Рисунок 6: Подсистема Show_Graphics.

А внутри блока Chart Station A реализуется собственно алгоритм поведения протокола TCP (общий рисунок 8):

В этом блоке можно определить целую иерархическую структуру блоков состояния Поскольку каждое состояние может включить в себя другие, на рисунке 7 приведена иерархическую структуру состояний блока Chart Station_A, и так можно заметить что станция A включает в себя необходимые обработаемые состояния станции A. SYN_SEND, FIN_WAIT, CLOSED, LISTEN, ESTABLISHED и ряд других промежуточных и вспомогательных состояний которые были внедрены для упрощения работы протокола. На рисунке 7 также видно все объявленные переменные необходимые для работы протокола TCP

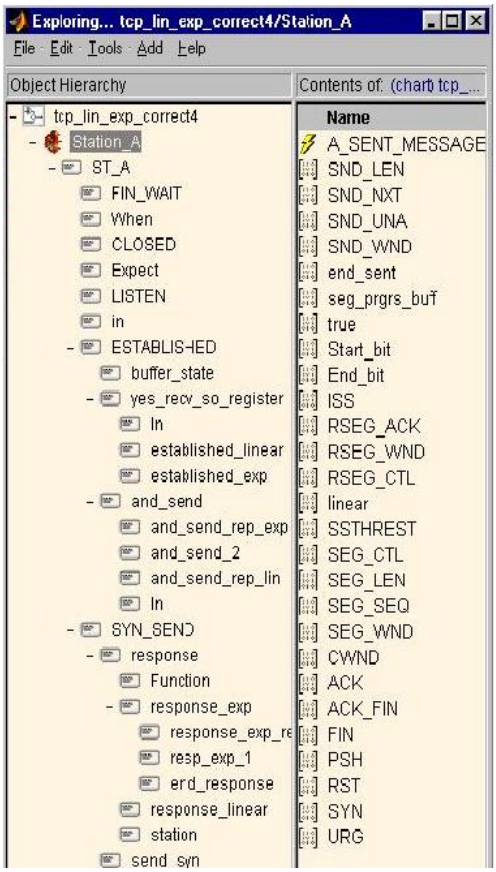

Рисунок 7: Иерархическая структура состояний.

Работа протокола начинается когда пользователь разрешает станции A начать передачу, на рисунке 8 вход в состояние LISTEN позволяет переменная Start_bit, тогда станция A инициализирует синхронизацию :
SND_NXT = ISS ;(первоначальный номер сегмента в буфере ПО станции A)
SND_UNA = SND_NXT;(сегмент ожидающий подтверждение)
SND_WND = SND_WND;(инициализация размер плавающего окна отправки)
Затем переходит в состояние SYN_SEND где устанавливаются переменные поля заголовки TCP .
После выхода из состояния вызывается процедура A_SENT_SYN.

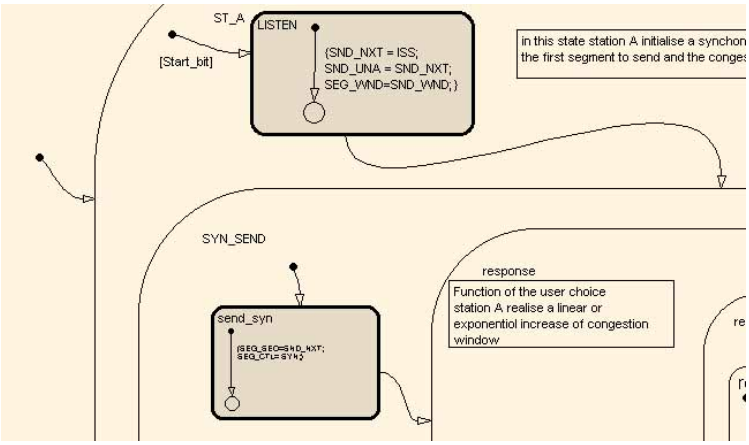

Рисунок 8: Вход в состояние LISTEN.

Следующий шаг определяется значением окна перегрузки или переход в response_exp или в response_linear в зависимости от уравнения $CWND \leq \frac{SSTHREST}{2}$ смотрите [3] для более уточнения по алгоритму управления перегрузки. Рисунок 12/ Рисунок 9-1.

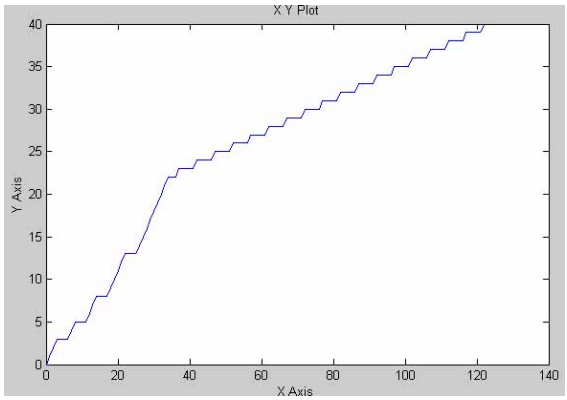

Рисунок 9-1: Алгоритм медленный старт.

На выходе из состояния SYN_SEND и точнее из response_exp или из response_linear идет вызов процедуры отправки установленного сегмента (рисунок 10).

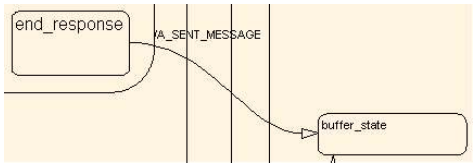

Рисунок 10: Отправка сегмента (/A_SENT_MESSAGE).

После выхода из состояния SYN_SEND и после приема ответа от станции B разрешается вход состояние ESTABLISHED в промежуточное состояние buffer_state после выхода из которого протокол решает заканчивать соединение если приходящий сегмент имеет бит контроля = FIN, если нет то продолжает отправку сегментов определенным методом в зависимости от значения CWND относительно порогового значения SSTHREST.
По определению алгоритма медленного старта при приращении окна перегрузки(линейно или экспонециально), протокол должен формировать все сегменты входящие в данном цикле отправки и послать их вместе[1][2].
Этот процесс выполняется при наступлении состояния established_linear или established_exp. Приращение окна перегрузки в каждом типом роста происходит по разному :

- При established_exp : окно перегрузки приращается при приеме подтверждении всего набора отправленных сегментов.
- При established_linear : окно перегрузки приращается при приеме подтверждении на каждый отправленный сегмент.

Этот процесс можно наблюдать в рисунке 12.

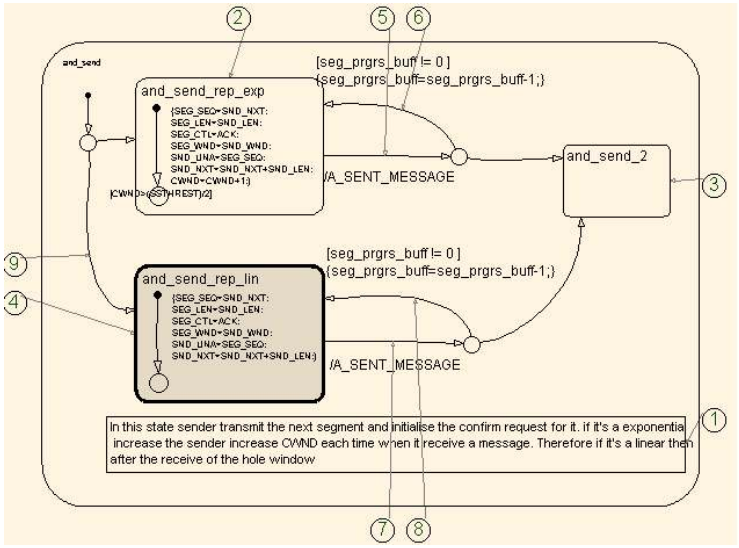

Рисунок 12: Приращение окна перегрузки.

Выводы и перспективы развития и оптимизации

По результатам исследований модели можно заметить что , легко подбирать формулу роста окна перегрузки, следовательно поведение протокола TCP приближается к оптимальному, но эффективность протокола резко снижается, поскольку он определяет главный пороговый параметр (SSTHREST) только после наступления перегрузки. Остался определить эти оптимальные значения исходя из математической модели, путем решения дифференциального уравнения алгоритма управления перегрузки.
В дальнейших работах можно заниматься теоретическим и практическим анализом поведения протокола TCP в более реальных условиях как совместное взаимодействия с другими протоколами стека TCP/IP как UDP.Также можно косвенно решать эту проблему путем анализа и оптимизации работы промежуточных устройства в сети (в основном маршрутизаторы) через которые проходит весь трафик TCP т.е оптимизировать алгоритмы приема трафика на входных портах или заняться протоколами маршрутизаций.

Список литературы

[1] М. Кульгин, “Технологии корпоротивных сетей,” СПБ; Издательство «Питер», 704 с, 1999
[2] М. Кульгин, “Коммутация и маршрутизация IP/IPX - трафика,” СПБ; Издательство «Компютер Пресс», 1998.
[3] V. Jacobson and M. Karels, “Congestion Avoidance and Control,” Proc. ACM SIGCOMM’88, pp. 314-239, Aug. 1988.
[4] G. Hasegawa, M. Murata, and H. Miyahara, “Fairness and Stability of Congestion Control Mechnism of TCP,” in Proceeding of 11th ITC Special seminar, pp. 255-262, October, 1998.
[5] S. Keshav, “A control-theoretic approch to flow control,” in Proceeding of ACM SIGCOMM’91, pp. 3-15, Septembre, 1991.
[6] H. Ohsaki, M. Murata, T. Ushio, H. Miyahara, “A Control Teoretical Analysis of a Window-Based Flow Control Mechanism in TCP/IP Networks” in Proceeding of IEEE/ACM SIGCOMM’99, pp. 1-13, 1999.
[7] H. Ohsaki, M. Murata, T. Ushio, H. Miyahara, “Stability Analysis of Window-Based Flow Control Mechanism in TCP/IP Networks,” in proceedingsof the 1999 IEEE/International Conferenceon Control Applications, Kohala Coast-Island of Hawai, Hawai, USA l August, 22-27, 1999.
[8] R. Morris, “Scalable TCP Congestion Control,” Thesis, Harvard University Cambridge Massachussets, January 1999.
[9] http://www.softline.ru/Matlab/Stateflow_full.asp.html “Stateflow 3.0.2 .”
[10] А.Гультяев, “Визуальное моделирование в среде MATLAB,” учебный курс – СПБ: Питер, 2000,-432с,: ил.
[11] RFC 793, “DARPA INTERNET Specification program”
[12] RFC 991, “IP protocol”