электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

**УДК 004.514**

# Принципы построения веб-интерфейса маршрутизатора

**Целовальникова О.А.[1], Паус А.С.[2]**

*[1,2]Студенты, кафедра «Компьютерные системы и сети»*
*МГТУ им. Н.Э. Баумана, г. Москва, Россия*

*Научный руководитель: Самарев Р.С., к.т.н.,*
*доцент кафедры «Компьютерные системы и сети» МГТУ им. Баумана, г. Москва, Россия*

МГТУ им. Н.Э. Баумана
samarev@acm.org
presidentoksana@gmail.com

## Введение

В настоящее время маршрутизаторы класса SOHO можно все с большей уверенностью относить к классу бытовых приборов, доступных каждому, у кого есть компьютер, чем к экзотическим, сложным в использовании и настройке сетевым устройствам. Управлять маршрутизатором можно различными способами. Для тех, кто хочет осуществлять удаленное подключение и использовать все возможности роутера, в ПО устройства включена опция SSH, что позволяет получить доступ к маршрутизатору в консольном режиме. Однако рядовой пользователь вынужден прибегать к помощи специально адаптированного и понятного web-интерфейса, которым снабжена каждая из существующих современных прошивок. В связи с тем, что возможности, доступные к настройке через web-интерфейс, в ПО, штатно поставляемом производителями маршрутизаторов, сильно урезаны, а потребности пользователей постоянно растут, все большее количество людей отдает предпочтение альтернативным прошивкам.

Существует целый ряд прошивок, конкурирующих между собой. Данная статья посвящена процессу модернизации одной из них. Программное обеспечение, о котором пойдет речь, предназначено для устройств для дома и малого офиса и не используется при работе с устройствами более высоких классов.

**Обзор существующих технологий**

Маршрутизатор — сетевое устройство, пересылающее пакеты данных на сетевом уровне модели OSI между сегментами сети или между различными локальными сетями. Все маршрутизаторы, поставляемые на рынок, можно подразделить на три класса, каждый из которых занимает свою нишу в организации домашних, корпоративных и глобальных сетей. Выделяют магистральные маршрутизаторы (ISP/WAN), маршрутизаторы филиалов (Branch/SMB) и устройства для малого бизнеса и домашнего использования (SOHO).

Именно устройства последнего из перечисленных классов представляют интерес в рамках данной работы. Зачастую они представляют собой не маршрутизаторы в чистом виде, многофункциональные устройства, включающие в себя беспроводную точку доступа, коммутатор, маршрутизатор и прочие системы, часто представленные в виде отдельных модулей.

Подобные устройства поставляются с уже установленной на них прошивкой от производителя. Однако ее можно легко заменить продуктом сторонних разработчиков.

Альтернативных прошивок на сегодняшний день существует множество. Рассмотрим преимущества и недостатки самых известных из них, а именно OpenWRT, DD-WRT, Tomato, Vampik, Gargoyle и «прошивки от энтузиастов», модернизации которой посвящена данная статья.

OpenWRT представляет из себя одну из первых появившихся альтернатив стандартным прошивкам. Основное преимущество OpenWRT доступно «продвинутым» пользователям, которые могут собрать прошивку под свои нужды. Существует множество пакетов, поддерживаемых в данном ПО. Однако веб-интерфейс не отличается особым удобством и в зачастую пользователи предпочитают работать с OpenWRT через Telnet или SSH, что, конечно, не является плюсом с точки зрения обычного человека.

DD-WRT — крайне популярная прошивка, имеющая на вооружении широкие возможности, доступные к реализации через веб-интерфейс. Она отличается стабильностью работы, однако довольно сильно нагружает роутер, что не оптимально для владельцев слабых устройств.

Tomato завоевала свою аудиторию в первую очередь очень удачной реализацией веб-интерфейса администрирования. Большое достоинство данного ПО — очень гибкая настройка QoS и расширенная система мониторинга сети и состояния маршрутизатора.

Vampik — модифицированная версия прошивки от энтузиастов, которая, в свою очередь, является продолжением «проекта Олега».

Gargoyle — одна из готовых сборок OpenWrt с приятным интерфейсом. Отличается бесперебойностью работы, однако отсутствует интернационализация. Данная прошивка — не лучший вариант для пользователя, не владеющего английским языком.

Одним из ключевых показателей целесообразности использования того или иного программного обеспечения является скорость загрузки и выгрузки данных, полученная при беспроводном подключении к маршрутизатору, работающему под управлением данной прошивки.

Согласно тестированию, проведенному пользователем Iliapan и опубликованному на сайте http://www.overclockers.ru (http://www.overclockers.ru/lab/40205_4/Testirovanie_alternativnyh_proshivok_sovremennyh_routerov.html), «прошивка от энтузиастов» — проект wl500g.googlecode.com показывает один из лучших результатов по скорости скачивания и загрузки файлов на торрент. Конкуренцию составляет прошивка Tomato.

На данный момент основным преимуществом «прошивки от энтузиастов» является расширенный по сравнению с другими ПО функционал. Однако интерфейс на сегодняшний день устарел, а эргономика расположения элементов оставляет желать лучшего. Поэтому было принято решение о разработке принципиально новой концепции построения интерфейсной части. Цель работ по модернизации — избежать недостатков, существующих в аналогичных решениях, и воплотить преимущества различных прошивок в одной.

**Концепция развития веб-интерфейса**

В текущей версии «прошивки энтузиастов» каждая страница была описана в виде отдельного шаблона ASP, генерирующего HTML код, а взаимодействие с сервером было синхронным. У данного способа генерации интерфейса есть много минусов:

1. Синхронные запросы, отправляемые на сервер, блокируют окно браузера до прихода ответа, что неудобно для пользователя и сильно замедляет рабочий процесс.
2. Для загрузки очередной страницы интерфейса требовалось загрузить и обработать целый шаблон, формирующий большой объем данных. Соответственно, время загрузки шаблона и нагрузка на маршрутизатор при такой реализации достаточно велики.
3. Чтобы изменить внешний вид страницы интерфейса или переместить какое-либо поле в другое место, необходимо было целиком переписывать ASP-шаблон данной страницы. В случае если требуется поменять стиль кнопки на всех страницах, возникала необходимость переписывания ASP-шаблона каждой из страниц.
4. Интернационализация интерфейса из-за шаблонной структуры также была затруднительна в связи с тем, что шаблон содержал фиксированный текст, не поддающийся изменению без изменения шаблона целиком.

В связи с вышеперечисленными недостатками, а также с моральным устареванием интерфейса, было принято решение о его модернизации. Первоочередная цель изменений в интерфейсе — привлечение новых пользователей за счет обновленного более дружелюбного внешнего вида и сохранение прежней аудитории за счет полного сохранения функционала. Таким образом, были поставлены следующие задачи, отражающие принципы построения нового интерфейса:

1. Интерфейс должен быть разделен на независимые модули, позволяющие вести разработку каждого структурного элемента интерфейса (будь то визуализация, алгоритмы проверки введенных значений и т.д.) автономно от других, а также облегчить последующую замену данных модулей последующими разработчиками.
2. Интерфейс создается полностью динамическим.
3. Взаимодействие клиентской части с сервером должно осуществляться асинхронно.
4. Необходимо предусмотреть возможность простой смены визуального представления страниц.
5. Процесс интернационализации интерфейса необходимо максимально упростить.
6. Система разработки интерфейса должна быть простой и понятной последующим разработчикам, при этом ее структура должна быть построена таким образом, чтобы максимально оградить разработчиков от совершения различного рода ошибок.

С учетом требований была предложена новая система описания страниц интерфейса.

В новом интерфейсе была исключена шаблонная структура, вместо неё каждая страница представлена в виде многоуровневой структуры JavaScript описания. HTML-разметка гибко генерируется по этому описанию автоматически при помощи специальных функций. Например, нижеприведенный фрагмент кода описывает содержание одной из страниц интерфейса:

```
var page={
'head':tr('Quick Setup'),
'subhead':'ZVMODELVZ',
'content':[
{'id':'tab1','head':'System','sections':[
{'head':'Time','content':[
{'type':'select','id':'time_zone','label':'Time Zone:','options':[
{'value':'+3','text':'+3:00 Moscow'},
{'value':'+0','text':'+0:00 London'},
{'value':'+5','text':'+6:00 Ekaterinburg'},
```

```
]},
]},
{'head':'Router Login','content':[
{'type':'text','id':'name_t','label':'User name:','value':'admin'},
{'type':'password','id':'pass_t','label':'Password','value':'admin'}
]},
{'head':'Operation Mode','content':[
{'type':'select','id':'operatio_mode','label':'Current mode:',
'options':[
{'value':'gateway','text':'Home Gateway'},
{'value':'router','text':'Router'},
{'value':'access','text':'Accsess Point'},
]},
{'type':'text','value':'','label':'Home Gateway:','label':'Home Gateway:'},
{'type':'text','value':help[1],'label':'Router:'},
{'type':'text','value':help[2],'label':'Accsess Point:'}
]}
]},
```

Далее приведен фрагмент кода, генерирующего HTML-разметку по данному описанию:

```
function getPageHTML()  {
  var cont = page.content;
  if( !cont ) return tr('Blank Page');
  var str='<div class="cont_inner"><div class="pageHeader">\
   <span>'+page.head+'</span>';
  for(var i=0; i<cont.length; i++) {
  if( cont[i].head )
   str += '<div class="pageHeader_anchor" id="'+cont[i].id+'">'+
         cont[i].head+'</div>';
}
str+='</div> \ ';
for(var i=0; i<cont.length; i++) {
  if( (! expertMode) && cont[i].expert ) continue;
  str+='<div id="'+cont[i].id+'">';
  var sects = cont[i].sections;
  if( !sects ) continue;
  for(var j=0; j<sects.length; j++) {
   if( (! expertMode) && sects[j].expert ) continue;
```

```
if ( sects[j].head ) {
  str+='<div class="page_inner_header">'+sects[j].head+'</div>';
};
var sec=sects[j].content;
if( !sec ) continue;
str+='<table> \ ';
for(var k=0; k<sec.length; k++) {
  var item=sec[k];
  if( !item ) continue;
  switch( item.type ) {
   case 'password':
    str += '<tr><td><label for="'+
          item.id+'">'+item.label+
          '</label></td><td><input id="'+item.id+
          '" type="password" value="'+item.value+'"/></td></tr>';
   break;
   case 'text':
    str += '<tr><td><span>'+item.label+
          '</span></td><td><span>'+
           ((item.text)?item.text:item.value)+'</span></td></tr>';
   break;
   case 'select':
     str += '<tr><td><label for="'+item.id+'">'+item.label+
           '</label></td><td><select id="'+item.id+'">';
     var opts=item.options;
     if( !opts ) break;
     for (var l=0; l<opts.length; l++){
       var opt=opts[l];
       if( !opt ) continue;
       str+='<option value="'+opt.value+'">'+opt.text+'</option>';
     }
     str+='</select></td></tr>';
```

Таким образом, полная смена визуального представления сводится к редактированию нескольких функций, а не целых страниц.

Шаблоны страниц в старой версии интерфейса загружались с сервера целиком. При изменении параметров страницы перерисовывались полностью. JavaScript-описание

позволяет динамически изменять структуру HTML-кода. Другими словами основной код скачивается с сервера один раз, а изменение значения полей осуществляется посредством их выборочной перезаписи JavaScript-функциями, что исключает необходимость повторного вывода остальных элементов HTML-разметки, не претерпевающих изменений. Благодаря подобным преобразованиям снижается нагрузка как на сервер, так и на машину клиента.

При выборе способа обмена информацией между клиентом и сервером предпочтение было отдано асинхронному взаимодействию посредством json-запросов. В отличие от решения с синхронным взаимодействием, пользователю не нужно дожидаться перезагрузки страницы и он может продолжать работу во время передачи информации на сервер.

Также в новой реализации веб-интерфейса прошивки, благодаря использованию асинхронного взаимодействия, уменьшилось общее количество http-запросов. Учитывая то, что сервер нагружается больше при увеличении количества http-запросов, нежели при росте их объема, уменьшение количества запросов от интерфейса существенно снижает нагрузку на сервер. Стоит отметить, что в новом интерфейсе большее количество алгоритмов выполняется на стороне клиента, что также способствует разгрузке маломощной аппаратной базы маршрутизатора.

При разработке нового интерфейса, необходимо было учесть, что данный проект поддерживает не постоянное сообщество, и необходимо было предусмотреть возможность максимально простой смены разработчиков. Это была одна из причин выбора описания страниц на языке javascript. Используя данное описание, можно легко контролировать возможные ошибки сторонних разработчиков и исправлять их в кратчайшие сроки.

**Заключение**

Таким образом, мы разрабатываем современный, удобный веб-интерфейс, имеющий преимущества перед нынешим. Часть работ была уже сделана.

Пользователи неохотно переходят на что-то новое, поэтому существенный переход на новую прошивку будет осуществлен через полгода – через год. После модернизации интерфейс, предположительно, будет актуален ближайшие пару лет.

**Список литературы**

1. Мержевич В. - Для тех, кто делает сайты: [Электронный ресурс]. 2002. Режим доступа: http://www.htmlbook.ru/. (Дата обращения: 9.11.2012).
2. Луканидин Д. Макет динамического интерфейса: [Электронный ресурс]. 2011. Режим доступа: http://sovetic.ru/other. (Дата обращения: 15.11.2012).

3. The jQuery Project: [Электронный ресурс]. 2010. Режим доступа: http://docs.jquery.com/ (Дата обращения: 17.12.2012).