Announcements


Homework 7 due today (Hw 8 available soon)

HW 5 + 6 will be returned in lab
Today’s Plan

- Ethernet Analysis
- Vocal exhaustion or Transport protocols
Efficiency = \frac{P/R}{W \times T + P/R}

W = \frac{1 - A}{A}

A = \left(1 - \frac{1}{Q}\right)(Q - 1)

P = \text{expected/average packet size}
R = \text{transmission rate (M & B call it C)}
W = \text{expected # of slots between transmissions}
T = \text{expected length of a contention slot}
A = \text{Probability exactly one computer sends in a slot}
Q = \text{number of computers trying to sent}
\[ A = \text{Probability that exactly 1 computer attempts a transmission in a given slot} \]

\[ W = \text{expected \# of slots between transmissions} \]

\[ W = \frac{1 - A}{A} = \frac{\text{Prob}(0 \text{ or } > 1 \text{ transmits})}{\text{Prob( exactly 1 transmits) }} \]
Great Expectations

Principle 2: Computing Expected Values:

Given $F$: outcome $\rightarrow$ value

$$\text{Expected}(F) = \sum \text{Prob}(\text{outcome}) \times F(\text{outcome})$$

all outcomes
Expected Contention Slot

Outcomes: \{ X \text{ any } Y \text{ collide then } X \text{ delays } 0 \text{ and transmits, } X \text{ and } Y \text{ collide then collide again then } Y \text{ delays } 0 \text{ and transmits, } \ldots \ \}

\[ F( \text{collide, } X \text{ delays } 0 \text{ & transmits } ) = 1 \text{ (slots wasted)} \]

\[ \text{Expected}(F) = \sum \text{ Prob}(\text{outcome}) \times F(\text{outcome}) \]

all outcomes
Expected Contention Slot

Outcomes: \{ X \text{ any } Y \text{ collide then } X \text{ delays 0 and transmits, } X \text{ and } Y \text{ collide then collide again then } Y \text{ delays 0 and transmits, } ... \}

\[ F( \text{ collide, } X \text{ delays 0 & transmits } ) = 1 \ (\text{slots wasted}) \]

\[ W = \sum \text{Prob( 1st success in slot } i \text{ ) } \times i \]
\[
\text{Prob( 1st success in slot } i \text{ ) } = ???
\]

\[
A = \text{Prob( exactly 1 transmission in a slot )}
\]
Probability Principles

\[ \text{Prob( 1st success in slot } i \text{ )} = ??? \]

Principle 3: The probability of two independent events both happening is the product of their separate probabilities if the events happen (or don’t happen) independently.

\[ A = \text{Prob( exactly 1 transmission in a slot )} \]
\[ \text{Prob( 1st success in slot } i \text{ )} \]

\[ = A \times (1 - A)^i \]

\[ A = \text{Probability that exactly 1 computer attempts a transmission in a given slot} \]
Expected Contention Slots

\[
W = \text{Expected \# of slots wasted}
\]

\[
= \sum_{\text{all } i} \text{Prob( 1st success in } i \text{ ) } \times i
\]

\[
= \sum_{i = 0}^{\infty} A \times (1 - A)^i \times i
\]
"I think you should be more explicit here in step two."
Expected Contention Slots

\[ W = \text{Expected \# of slots wasted} \]

\[ = \sum_{\text{all } i} \text{Prob( 1st success in } i \text{ ) } x \ i \]

\[ = \sum_{i = 0}^{\infty} A x (1 - A)^i x i \]

\[ = \frac{(1 - A)}{A} \]
A = \frac{1}{Q} (1 - \frac{1}{Q})^{(Q - 1)}

\begin{align*}
A &= \text{Probability that exactly 1 computer attempts a transmission in a given slot}
\end{align*}
Slippery Slots

Probability a particular computer tries to send $\approx \frac{1}{S}$

$S = \text{The number of backoff slots the computer is currently using}$
S = The number of backoff slots all of the computers are currently using.

Q = The total number of computers that are trying to send

Probability a particular computer sends alone $\approx \frac{S}{Q}$.
Synchronized Slots

Probability a particular computer sends alone

\[ P \approx \frac{1}{S} \left( 1 - \frac{1}{S} \right)^{(Q - 1)} \]

S = The number of backoff slots all of the computer are currently using!!!

Q = The total number of computers that are trying to send
Probability of Success

\[ \text{Probability some lucky computer sends alone} \approx \frac{Q}{S} \left( 1 - \frac{1}{S} \right)^{(Q - 1)} \]

\( S \) = The number of backoff slots all of the computer all currently using

\( Q \) = The total number of computers that are trying to send
S = The number of backoff slots all of the computer all currently using

Q = The total number of computers that are trying to send

\[ A \approx \frac{Q}{S} \left( 1 - \frac{1}{S} \right)(Q - 1) \]
But Metcalfe and Boggs Say...

\[ A = (1 - \frac{1}{Q})(Q - 1) \]

\[ S = \text{The number of backoff slots all of the computer all currently using} \]
\[ Q = \text{The total number of computers that are trying to send} \]
We assume that a queued station attempts to transmit in the current slot with probability \(1/Q\), or delays with probability \(1 - (1/Q)\); this is known to be the optimum statistical decision rule, approximated in Ethernet stations by means of our load-estimating retransmission control algorithms [20, 21].

6.1 Acquisition Probability

We now compute \(A\), the probability that exactly one station attempts a transmission in a slot and therefore acquires the Ether. \(A\) is \(Q \times (1/Q) \times ((1 - (1/Q))^2 \times (Q - 1))\); there are \(Q\) ways in which one station can choose to transmit (with probability \((1/Q)\)) while \(Q - 1\) stations choose to wait (with probability \(1 - (1/Q)\)). Simplifying,
We assume that a queued station attempts to transmit in the current slot with probability \( 1/Q \), or delays with probability \( 1 - (1/Q) \); this is known to be the optimum statistical decision rule, approximated in Ethernet stations by means of our load-estimating retransmission control algorithms [20, 21].

6.1 Acquisition Probability

We now compute \( A \), the probability that exactly one station attempts a transmission in a slot and therefore acquires the Ether. \( A \) is \( Q*(1/Q)*((1 - (1/Q))**(Q - 1)) \); there are \( Q \) ways in which one station can choose to transmit (with probability \( (1/Q) \)) while \( Q - 1 \) stations choose to wait (with probability \( 1 - (1/Q) \)). Simplifying,
Thinking Optimistically

\[ A \approx \frac{Q}{S} \left( 1 - \frac{1}{S}\right)^{(Q-1)} \]

\[ \frac{dA}{dS} \approx \frac{Q}{S^3} (Q - S)\left( 1 - \frac{1}{S}\right)^{(Q-2)} \]

\[ S = \text{The number of backoff slots all of the computer all currently using} \]

\[ Q = \text{The total number of computers that are trying to send} \]
Probability of Success?

\[ A \approx \frac{Q}{S} \left( 1 - \frac{1}{S} \right)(Q - 1) \]

\[ < \left( 1 - \frac{1}{Q} \right)(Q - 1) \]

\( S = \text{The number of backoff slots all of the computer are currently using} \)

\( Q = \text{The total number of computers that are trying to send} \)
Efficiency = \frac{P/R}{W \times T + P/R}

= \frac{1}{W \times T + 1} \div \frac{P/R}{P/R}

P = \text{expected/average packet size}
R = \text{transmission rate (M & B call it } C\text{)}
W = \text{expected # of slots between transmissions}
T = \text{expected length of a contention slot}
# The Bottom Line

<table>
<thead>
<tr>
<th>$N = Q$</th>
<th>$P = 4K$</th>
<th>$P = 1K$</th>
<th>$P = 512$</th>
<th>$P = 48$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0.99</td>
<td>0.94</td>
<td>0.89</td>
<td>0.44</td>
</tr>
<tr>
<td>5</td>
<td>0.98</td>
<td>0.94</td>
<td>0.88</td>
<td>0.41</td>
</tr>
<tr>
<td>10</td>
<td>0.98</td>
<td>0.93</td>
<td>0.87</td>
<td>0.39</td>
</tr>
<tr>
<td>128</td>
<td>0.98</td>
<td>0.93</td>
<td>0.86</td>
<td>0.37</td>
</tr>
</tbody>
</table>
MEASURED CAPACITY OF AN ETHERNET
SIGCOMM '88 Symposium proceedings on Communications architectures and protocols

David R. Boggs
Jeffrey C. Mogul
Christopher A. Kent

Ethernet Utilization in MBits/sec

Number of Hosts

0 5 10 15 20 25 30

64

128

256

512

768

1536

3072

4000

1024
Idle Sense: An Optimal Access Method for High Throughput and Fairness in Rate Diverse Wireless LANs

Martin Heusse, Franck Rousseau, Romaric Guillier, Andrzej Duda

LSR-IMAG Laboratory
Grenoble, France

{Martin.Heusse, Franck.Rousseau, Romaric.Guillier, Andrzej.Duda} @imag.fr

ABSTRACT
We consider wireless LANs such as IEEE 802.11 operating in the unlicensed radio spectrum. While their nominal bit rates have increased considerably, the MAC layer remains practically unchanged despite much research effort spent on improving its performance. We observe that most proposals for tuning the access method focus on a single aspect and disregard others. Our objective is to define an access method optimized for throughput and fairness, able to dynamically adapt to physical channel conditions, to operate near optimum for a wide range of error rates, and to provide equal time shares when hosts use different bit rates.

We propose a novel access method derived from 802.11 DCF [2] (Distributed Coordination Function) in which all hosts use similar values of the contention window CW to benefit from good short-term access fairness. We call our method Idle Sense, because each host observes the mean number of idle slots between transmission attempts to dynamically control its contention window. Unlike other proposals, Idle Sense enables each host to estimate its frame error rate, which can be used for switching to the right bit rate. We present simulations showing how the method leads to high throughput, low collision overhead, and low delay. The method also features fast reactivity and time-fair channel allocation.

General Terms
Algorithms, Performance

Keywords
Wireless LANs, 802.11, Access Methods, Fairness

1. INTRODUCTION
Since the advent of the first 802.11 wireless LANs, much research effort has been spent on improving their performance. Successive variants have increased the nominal bit rate at the physical layer. However, the MAC layer remains practically unchanged despite many proposals for tuning its performance. Most of this work focuses on a single aspect while disregarding others. For example, many researchers try to improve throughput, but they neglect other performance aspects such as short-term fairness, adaptation to channel conditions, or handling multiple bit rates. Unlike these proposals, we follow a global approach to the design of an access method for wireless LANs by taking into account all aspects and trying to find the best tradeoff between antagonistic objectives.

We elaborate the principle of our method by considering a modification to the basic CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) access method: contending hosts do not perform the exponential backoff algorithm after collisions, but rather dynamically converge in a fully distributed way to similar values of their contention windows.
Idle Sense: An Optimal Access Method for High Throughput and Fairness in Rate Diverse Wireless LANs

Martin Heusse, Franck Rousseau, Romaric Guillier, Andrzej Duda

LSR-IMAG Laboratory*
Grenoble, France

{Martin.Heusse, Franck.Rousseau, Romaric.Guillier, Andrzej.Duda} @imag.fr

ABSTRACT
We consider wireless LANs such as IEEE 802.11 operating in the unlicensed radio spectrum. While their nominal bit rates have increased considerably, the MAC layer remains practically unchanged despite much research effort spent on improving its performance. We observe that most proposals for tuning the access method focus on a single aspect and disregard others. Our objective is to define an access method optimized for throughput and fairness, able to dynamically adapt to physical channel conditions, to operate near capacity, and be fair to all users. Hence we propose the distributed method Idle Sense, because each host observes the mean number of idle slots between transmission attempts to dynamically control its contention window. Unlike other proposals, we follow a global approach to the design of an access method for wireless LANs by taking into account all objectives, trying to find the best tradeoff between them.

SIGCOMM’05, August 21–26, 2005, Philadelphia, Pennsylvania, USA.
Copyright 2005 ACM 1-59593-009-4/05/0008 ...$5.00.

1. INTRODUCTION
Since the advent of the first 802.11 wireless LANs, much
try to improve throughput, but they neglect other performance aspects such as short-term fairness, adaptation to channel conditions, or handling multiple bit rates. Unlike these proposals, we follow a global approach to the design of an access method: contending hosts do not perform the exponential backoff algorithm after collisions, but rather dynamically converge in a fully distributed way to similar values of their contention windows.

General Terms
Algorithms, Performance

Keywords
Wireless LANs, 802.11, Access Methods, Fairness
W = expected # of slots between transmissions

\[ W = \frac{1 - A}{A} \]

\[ A = (1 - \frac{1}{Q})(Q - 1) \]

W = expected # of slots between transmissions
A = Probability exactly one computer sends in a slot
Thinking Optimistically

Optimal value of $W$

Number of computers trying to send
Idle Thoughts

\[ W = \frac{1 - A}{A} \]

\[ A = \frac{1}{Q} \left( 1 - \frac{1}{Q} \right)(Q - 1) \]

\[ I = ??? \]

\[ W = \text{expected \# of slots between transmissions} \]
\[ A = \text{Probability exactly one computer sends in a slot} \]
\[ I = \text{expected \# of consecutive idle slots} \]
My Idle?

Number of computers trying to send

Optimal consecutive idle slots
Famous?

Lycos, 1994
Alta Vista, 1995
Yahoo, 1995
Backrub, 1998
Famous?

Williamstown 27°

Enter search term...

Simplify Your Digital Life.

Lycos has recently launched a suite of wearable devices

Find Out More
Famous?

Lycos, 1994

Alta Vista, 1995

Yahoo, 1995

Backrub, 1998

Ads related to professor tom murtagh

Murtagh at Amazon.com - Buy Murtagh at Amazon!

Buy Murtagh at Amazon! Free Shipping on Qualified Orders.

Professor Tom | when.com

Explore Professor Tom Discover More on When.com!

Murtagh | wow.com

Search for Murtagh Look Up Quick Results Now!

Tom Murtagh at Williams College - RateMyProfessors.com

Rating and reviews for Professor Tom Murtagh from Williams College Williamstown, MA United States.

Tom Murtagh - Home

tom.murtagh.net/index.html

Tom Murtagh has been applying his mental energy towards various creative endeavors for over 25 years. These include screenwriting, fiction, marketing, copy writing...

Thomas Murtagh | InformIT

Kim B. Bruce, Reuben C. and Eleanor Winslow Professor of Computer Science at Pomona College, holds a Ph.D. from University of Wisconsin at Madison, and formerly...
Famous?

- Lycos, 1994
- Alta Vista, 1995
- Yahoo, 1995
- Backrub, 1998
A Eulogy For AltaVista, The Google Of Its Time

Danny Sullivan on June 28, 2013 at 6:53 pm

Goodbye AltaVista. You deserved better than this. Better than the one-sentence send-off Yahoo gave you today, when announcing your July 8 closure date. But then again, you always were the bright child neglected by your parents.

The Amazing AltaVista

You appeared on the search engine scene in December 1995. You made us go “woah” when you arrived. You did that by indexing around 20 million web pages, at a time when indexing 2 million web pages was considered to be big.

Today, of course, pages get indexed in the billions, the tens of billions or more. But in 1995, 20 million was huge. Existing search engines like Lycos, Excite & InfoSeek (to name only a few) didn’t quite know what hit them. With so many pages, you seemed to find stuff they and others didn’t.
Famous?

- Lycos, 1994
- Alta Vista, 1995
- Yahoo, 1995
- Backrub, 1998
BackRub is a "web crawler" which is designed to traverse the web.

Currently we are developing techniques to improve web search engines. We will make various services available as soon as possible.

Sorry, many services are unavailable due to a local network failure beyond our control. We are working to fix the problem.
The Anatomy of a Large-Scale Hypertextual Web Search Engine

Sergey Brin and Lawrence Page
{sergey, page}@cs.stanford.edu
Computer Science Department, Stanford University, Stanford, CA 94305

Abstract

In this paper, we present Google, a prototype of a large-scale search engine which makes heavy use of the structure present in hypertext. Google is designed to crawl and index the Web efficiently and produce much more satisfying search results than existing systems. The prototype with a full text and hyperlink database of at least 24 million pages is available at http://google.stanford.edu/

To engineer a search engine is a challenging task. Search engines index tens to hundreds of millions of web pages involving a comparable number of distinct terms. They answer tens of millions of queries every day. Despite the importance of large-scale search engines on the web, very little academic research has been done on them. Furthermore, due to rapid advance in computer hardware and software, there is a great need for more scalable and efficient algorithms.
GOOGLE - 1998
Crawling the Web

Recall Lab 6 (our web crawling lab):

- For each URL (http://...) you can find:
  - Retrieve and scan the page’s HTML for `<a href="URL">` tags to find other pages.
  - For each page found count the number of links from it and to it.
Indexing the Web

Create an “inverted index” (think recursive list) that associates words with sets of web pages in which each word was found.

For each URL (http://...) you can find:

- Retrieve and scan the page’s HTML for <a href="URL"> tags to find other pages.
- For each page found count the number of links from it and to it.
- For each page found add it to the sets in the inverted index for each word it contains.
About 1,180,000,000 results (0.99 seconds)

Dell PowerEdge R930 Rack Server (Intel®) w/ Intel Xeon
$9,549.01
Dell
Free shipping

Dell PowerEdge T330 Tower Server (Intel®) w/ Intel Xeon
$1,249.00
Dell
Free shipping

"Refurbished Poweredge R210, Configured to Order"
$179.00
ServerWorlds
Free shipping

Server (computing) - Wikipedia

In computing, a server is a computer program or a device that provides functionality for other programs or devices, called "clients". This architecture is called the client–server model, and a single overall computation is distributed across multiple processes or devices.

Database server · Application server · File server · Client (computing)
Gopher (protocol) - Wikipedia
https://en.wikipedia.org/wiki/Gopher_(protocol) ▼
The Gopher protocol /gofer/ is a TCP/IP application layer protocol designed for distributing, .... Here, the client has established a TCP connection with the server on port 70, the standard gopher port. The client then sends a string followed by ...

HTTPS - Wikipedia
https://en.wikipedia.org/wiki/HTTPS ▼
HTTPS is a communications protocol for secure communication over a computer network which ... In practice this means that even on a correctly configured web server, eavesdroppers can infer the IP address and port number of the web ...

Converting Server published projects to open in... |Tableau Community
https://community.tableau.com/message/610163 ▼
20 hours ago - 3 posts - 2 authors
I have several projects that were published onto our server. These were originally viewed with the reader but when we purchased server ...

Lead Server - Craigslist Chicago
https://chicago.craigslist.org/cht/fsh/6107350592.html
5 hours ago - Lead Servers, Servers (Chicago Yacht Club, IL) We are looking for friendly and motivated employees with flexible schedules to fill all shifts ...
Too Many Pages

For any small set of search words, the index will contain thousands of pages that contain all (or at least most) of the words.

How can a search engine pick the pages of greatest interest to the person doing the search?

- Number of occurrences?
- Location of occurrences (titles, section headers, links, ...)?
- Popularity of each of the pages found?
Citation Indices

Are all citations of equal value?
The Random Surfer
Random Walks on the Web

Let $Pr_i(\text{URL}) =$ probability of being at page URL after clicking on $i$ randomly selected links in web pages assume one starts at a page selected at random.
Welcome to Page 1

You can read all about:

- Page 2
- Page 3
- Page 4

Welcome to Page 2

You can read all about:

- Page 3
- Page 4

Welcome to Page 3

You can read all about:

- Page 1

Welcome to Page 4

You can read all about:

- Page 1
- Page 3
Neighbors

\[
\text{out}(1) = \{2, 3, 4\} \quad \text{out}(2) = \{3, 4\} \quad \ldots
\]
Neighbors

\[ \text{in}(1) = \{3, 4\} \quad \text{in}(2) = \{1\} \quad \ldots \]
Out Degree

1
\[ C(1) = 3 \]

2
\[ C(2) = 2 \]

3
\[ C(3) = 1 \]

4
\[ C(4) = 2 \]
\text{In Degree}

\begin{align*}
\text{In}(1) &= 2 \\
\text{In}(2) &= 1 \\
\text{In}(3) &= 3 \\
\text{In}(4) &= 2
\end{align*}
Random Surfer

\[ \text{Pr}_i(j) = \text{prob. at page j after i clicks} \]
Random Surfer

\[ P_{i+1}(2) = \frac{P_r(i)(1)}{3} \]
$\text{Random Surfer}$

$Pr_{i+1}(1) = ?$
Random Surfer

\[ \text{Pr}_{i+1}(1) = \text{Pr}_i(3) + \frac{\text{Pr}_i(4)}{2} \]
Random Surfer

\[ P_{r_{i+1}}(j) = \sum_{k \in \text{in}(j)} \frac{P_{r_i}(k)}{C(k)} \]
<HTML>
  <HEAD>
    <TITLE>something short but sweet</TITLE>
  </HEAD>
  <BODY>
    something not so short but sweet
  </BODY>
</HTML>
A BRIEF EXAMPLE

<html>
<head>
<TITLE>Tom Murtagh's Home</TITLE>
</head>

<body>
<H1>Tom Murtagh</H1>

<P>OK!!! I've given in and made a home page. Well, it's a start...</P>

</body>

</html>
Clients vs. Servers
Clients vs. Servers

Role of clients - retrieve data from servers
1. Contact server
2. Send request(s)
3. Retrieve response(s)
4. Disconnect

Role of servers - “serve” data to clients
1. Patiently wait for client connections
2. Accept valid connection
3. Receive request(s)
4. Send data to clients
5. Close client connection
Clients vs. Servers

Role of clients - retrieve data from servers
1. Contact server (... = new NetConnection("cortland", 110); )
2. Send request(s) ( toServer.out.println("RETR ..."); )
3. Retrieve response(s) ( response = toServer.in.nextLine(); )
4. Disconnect ( toServer.close(); )

Role of servers - “serve” data to clients
1. Patiently wait for client connections
2. Accept valid connection
3. Receive request(s)
4. Send data to clients
5. Close client connection;
Clients vs. Servers

Role of clients - retrieve data from servers
1. Contact server (...
   = new NetConnection("cortland", 110);)
2. Send request(s) (toServer.out.println("RETR ...");)
3. Retrieve response(s) (response = toServer.in.nextLine();)
4. Disconnect (toServer.close();)

Role of servers - “serve” data to clients
1. Patiently wait for client connections
2. Accept valid connection
3. Receive request(s) (request = fromClient.in.nextLine();)
4. Send data to clients (fromClient.out.println(response))
5. Close client connection (fromClient.close());
Clients vs. Servers

Role of clients – retrieve data from servers
1. Contact server ( ... = new NetConnection(“cortland”, 110); )
2. Send request(s) ( toServer.out.println(“RETR ...”); )
3. Retrieve response(s) ( response = toServer.in.nextLine(); )
4. Disconnect ( toServer.close(); )

Role of servers – “serve” data to clients
1. Patiently wait for client connections ( ??? )
2. Accept valid connection ( fromClient = ??? )
3. Receive request(s) ( request = fromClient.in.nextLine(); )
4. Send data to clients ( fromClient.out.println( response )
5. Close client connection ( fromClient.close() );
Port Numbers

- Servers “listen” for client connection requests on specific ports
- Recall that POP email uses port 110
- Ports are like phone number extensions
- In Squint, we indicate that a server wants to activate a port by saying:
  
  ```java
  TCPPort connectPort = new TCPPort( 110 );
  ```
Waiting for a Call

A server can

- wait for a client to create a NetConnection, and
- get access to the NetConnection

by saying:

```
NetConnection fromClient = connectPort.acceptNetConnection();
```
In our client code, we added a MessageListener to be notified when the server sent us a new message:

```java
toServer.addMessageListener(this);

... 

public void dataAvailable() {
    response = toServer.in.nextLine();
    ...
}
```
Network Server Events

We can ask Java to execute a method when our server receives a connection request:

```java
connectPort.addConnectionListener( this );

public void connectionEstablished( TCPPort p ) {
    NetConnection fromClient = connectPort.acceptNetConnection();
    ...
}
```
Accessing Files

Programs can access data in files using `println` and `nextLine` (just as programs access data sent through `NetConnections`):

```java
Scanner inFile =
    new Scanner( new File ( "file's name") );

while ( inFile.hasNextLine() ) {
    String currentLine = inFile.nextLine() ;
    ... code to process current line ... 
}
```
We can ask Java to execute code that may fail while providing a backup plan:

```java
try {
    . . . risky code . . .
} catch ( SomeTypeOfException ex ) {
    . . . code to address problem . . .
}
```