Welcome to

4DN4
Advanced Internet Communications

Mon. Jan. 5 – Wed. April 8, 2015
Prof. Ted Szymanski
Department of ECE
McMaster University

Web-Page: Please go to Avenue-to-Learn.
Email: teds@mcmaster.ca

Lecture Videos: : Please go to Avenue-to-Learn
(when the ATL web-site is ready, by Fri. Jan. 9, 2015)

Tentative 4DN4 Topics We'll Cover:

Topic	Weight (roughly)
1. TCP/IP Socket Programming – Part 1 (C, PYTHON, JAVA)	3-4 weeks
2. Network Modelling and Delay (Matlab)	1-2 weeks
1. TCP/IP Flow Control	brief review
3. TCP/IP Socket Programming – Part 2 (C, PYTHON, JAVA)	1 week
4. Weighted Fair Queueing Traffic Regulators	2 weeks
5. Connection-Oriented Networks (ATM/MPLS)	1 week
6. Leaky-Bucket / Token Bucket Traffic Regulators	1 weeks
7. Class Presentations	2 weeks
8. EXAMPLES (for remaining time)	1 weeks

Textbooks We'll Use:

Required Textbook: **Communication Networks: Fundamental Concepts and Key Architectures**, A. Leon-Garcia & I. Widjaja, latest edition, ISBN 0-07-246-352-X, McGraw Hill,

Reference Textbook: **TCP/IP Sockets in C; Practical Guide for Programmers**, M. Donahoo & K. Calvert, Morgan Kaufmann Publishers, 2008, ISBN 1-55860-826-5

Reference Textbook: **TCP/IP Sockets in JAVA; Practical Guide for Programmers**, M. Donahoo & K. Calvert, Morgan Kaufmann Publishers, 2008, ISBN 9780123768551

Reference Textbook: **JAVA; Practical Guide for Programmers**, Elsevier/ Morgan Kaufmann Publishers, 2003

Reference Textbook: **Learn Python the Hard Way**, http:// learnpythonthehwardway,org

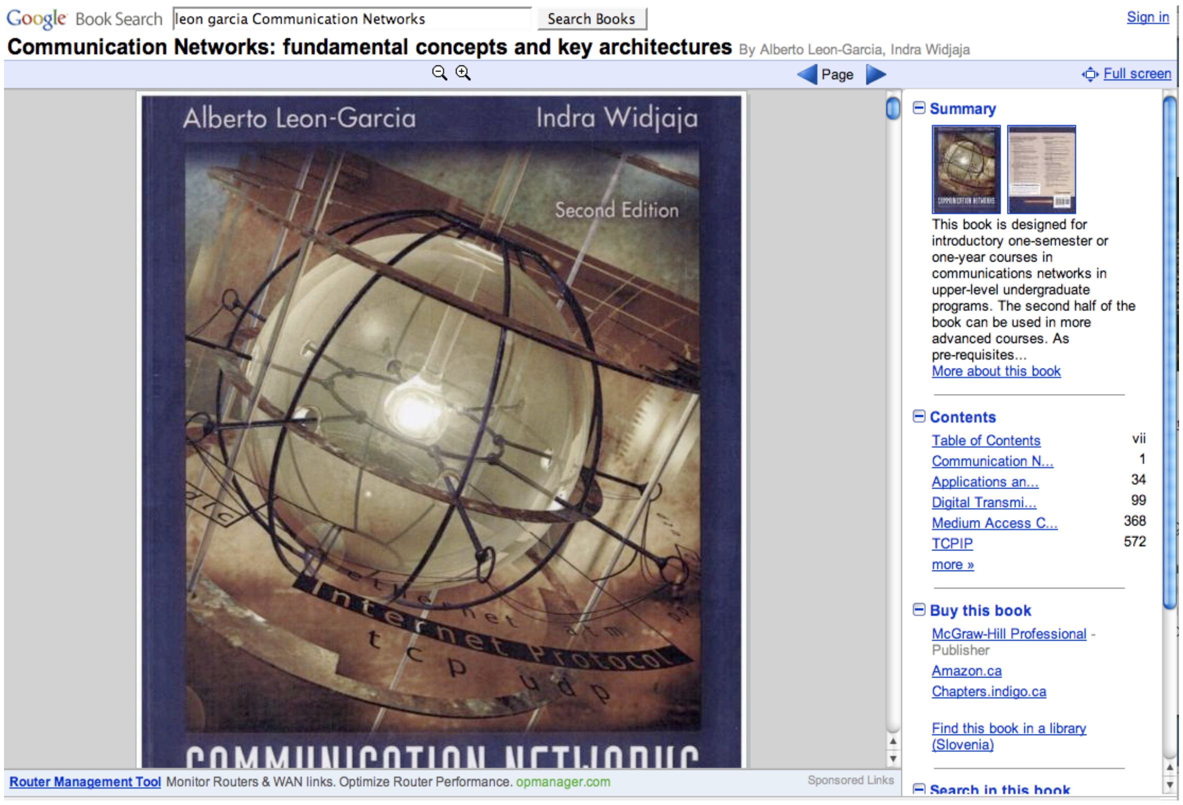
Topics in More Depth

Tentative Course Contents (in more depth):
1) Advanced Internet Protocols: Sockets, Internet Protocol v6, TCP, UDP, sockets, Linux, SSL, C and Java programming.

In the class lectures, we will demonstrate socket programming in the C, Python or JAVA languages, running on the Mac OS X. For C/JAVA code, we will use the the NETBEANS **Integrated Development Environment** (IDE). Students with Linux or Windows machines can use any IDE that they have access to (i.e.,KDE IDE, the Microsoft Visual C++ IDE). For PYTHON, the prof. will use a unix terminal window and a command-line interface (until an easy-to-use IDE is found). Students can gain useful experience by bringing their laptops to class and programming web-based applications like Napster along with the prof. in real time. Therefore, you can prepare by (1) installing the Netbeans IDE (or any other IDE), or (2) installing a PYTHON interpreter on your laptop, in the first 1 week of class. By week 2 of classes, we will be programming socket applications in class.

2) Internet Quality of Service (QoS), RTP, RSVP, Integrated and Differentiated Services (IntServ, DiffServ); Congestion control in IP networks. Quick start, Random Early Detection.

3) QoS in ATM & MPLS networks: constant and variable bit rate services. Digital switching, switch scheduling, traffic grooming and shaping, traffic models, traffic queueing models, Leaky Bucket, Token Bucket traffic grooming, Weighted Fair Queuing


Google Book Search leon garcia Communication Networks Search Books Sign in

Communication Networks: fundamental concepts and key architectures By Alberto Leon-Garcia, Indra Widjaja

Page Full screen

Summary

This book is designed for introductory one-semester or one-year courses in communications networks in upper-level undergraduate programs. The second half of the book can be used in more advanced courses. As pre-requisites...

More about this book

Contents

Table of Contents	vii
Communication N...	1
Applications an...	34
Digital Transmi...	99
Medium Access C...	368
TCPIP	572
more »	

Buy this book

McGraw-Hill Professional - Publisher

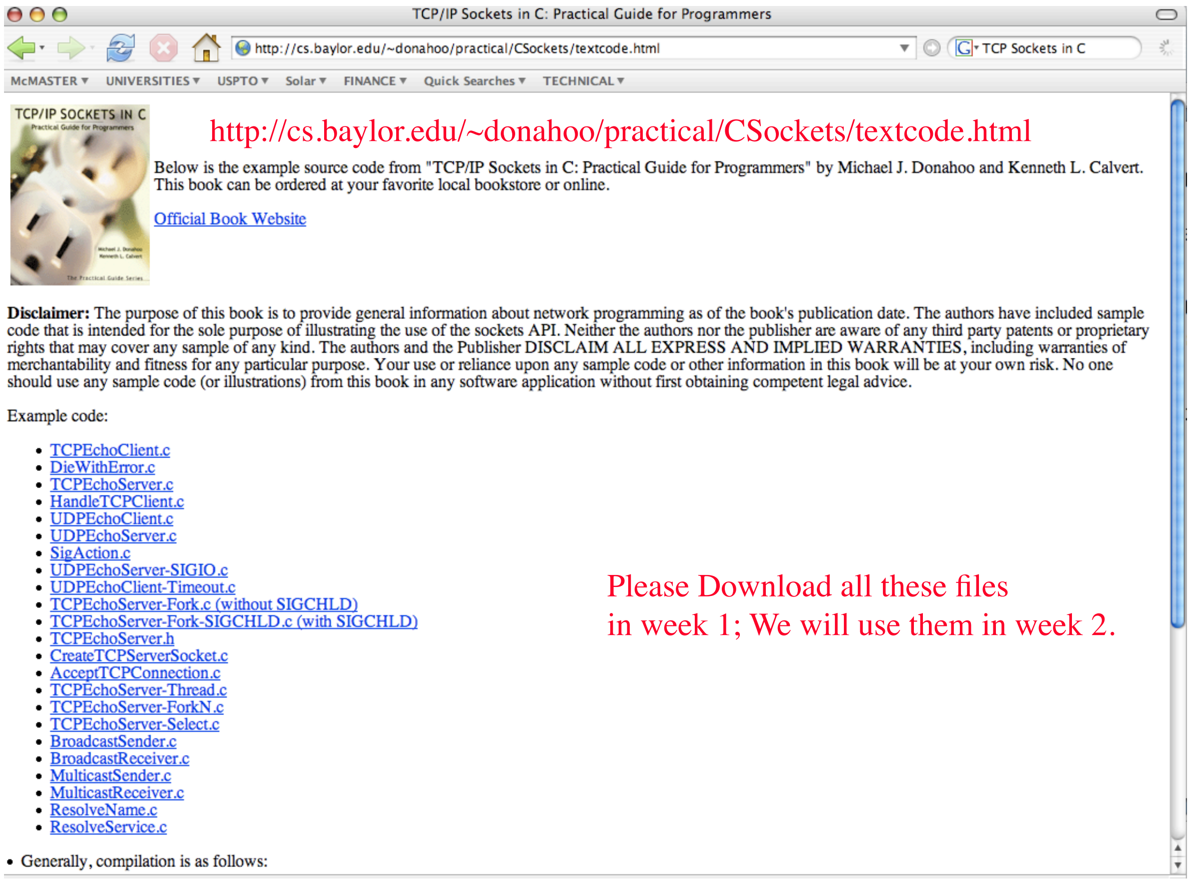
Amazon.ca

Chapters.indigo.ca

Find this book in a library (Slovenia)

Search in this book

Router Management Tool Monitor Routers & WAN links. Optimize Router Performance. opmanager.com Sponsored Links


TCP/IP Sockets in C: Practical Guide for Programmers

http://cs.baylor.edu/~donahoo/practical/CSockets/textcode.html

http://cs.baylor.edu/~donahoo/practical/CSockets/textcode.html

Below is the example source code from "TCP/IP Sockets in C: Practical Guide for Programmers" by Michael J. Donahoo and Kenneth L. Calvert. This book can be ordered at your favorite local bookstore or online.

Official Book Website

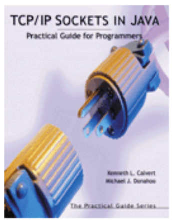
Disclaimer: The purpose of this book is to provide general information about network programming as of the book's publication date. The authors have included sample code that is intended for the sole purpose of illustrating the use of the sockets API. Neither the authors nor the publisher are aware of any third party patents or proprietary rights that may cover any sample of any kind. The authors and the Publisher DISCLAIM ALL EXPRESS AND IMPLIED WARRANTIES, including warranties of merchantability and fitness for any particular purpose. Your use or reliance upon any sample code or other information in this book will be at your own risk. No one should use any sample code (or illustrations) from this book in any software application without first obtaining competent legal advice.

Example code:

- TCPEchoClient.c
- DieWithError.c
- TCPEchoServer.c
- HandleTCPClient.c
- UDPEchoClient.c
- UDPEchoServer.c
- SigAction.c
- UDPEchoServer-SIGIO.c
- UDPEchoClient-Timeout.c
- TCPEchoServer-Fork.c (without SIGCHLD)
- TCPEchoServer-Fork-SIGCHLD.c (with SIGCHLD)
- TCPEchoServer.h
- CreateTCPServerSocket.c
- AcceptTCPConnection.c
- TCPEchoServer-Thread.c
- TCPEchoServer-ForkN.c
- TCPEchoServer-Select.c
- BroadcastSender.c
- BroadcastReceiver.c
- MulticastSender.c
- MulticastReceiver.c
- ResolveName.c
- ResolveService.c

- Generally, compilation is as follows:

Please Download all these files
in week 1; We will use them in week 2.



http://cs.ecs.baylor.edu/~donahoo/practical/JavaSockets/textcode.html

Below is the example source code from "TCP/IP Sockets in Java^{TM}: Practical Guide for Programmers" by Kenneth L. Calvert and Michael J. Donahoo. This book can be ordered at your favorite local bookstore or online.

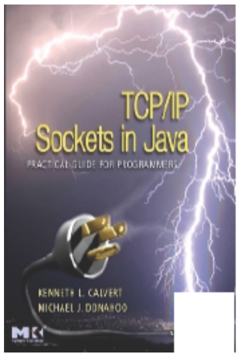
Official Book Website

Disclaimer: The purpose of this book is to provide general information about network programming as of the book's publication date. The authors have included sample code that is intended for the sole purpose of illustrating the use of the sockets API. Neither the authors nor the publisher are aware of any third party patents or proprietary rights that may cover any sample of any kind. The authors and the Publisher DISCLAIM ALL EXPRESS AND IMPLIED WARRANTIES, including warranties of merchantability and fitness for any particular purpose. Your use or reliance upon any sample code or other information in this book will be at your own risk. No one should use any sample code (or illustrations) from this book in any software application without first obtaining competent legal advice.

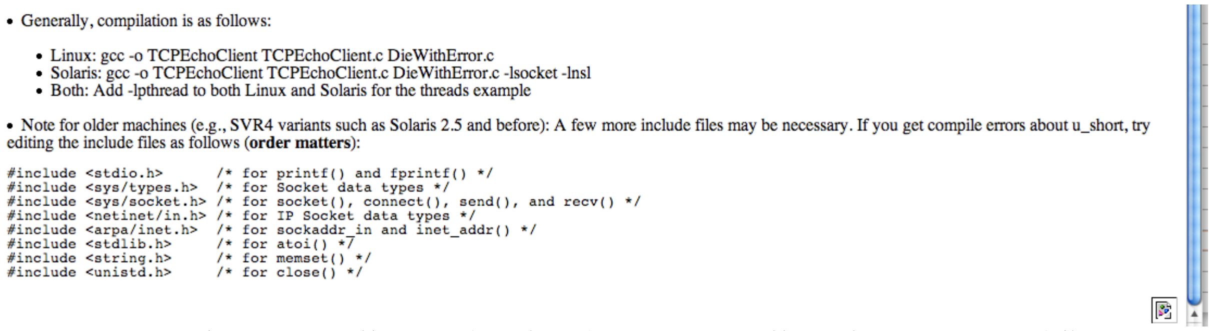
Example code:

- Chapter 2
  - InetAddressExample.java
  - TCPEchoClient.java
  - TCPEchoClientGUI.java
  - TCPEchoServer.java
  - UDPEchoClientTimeout.java
  - UDPEchoServer.java
- Chapter 3
  - ItemQuote.java
  - Framer.java
  - ItemQuoteEncoder.java
  - ItemQuoteDecoder.java
  - DelimitedInputStream.java
  - ItemQuoteTextConst.java
  - ItemQuoteEncoderText.java
  - ItemQuoteDecoderText.java

Please download these files in week 1 of class.



Option 1: using Unix/Linux Command-Line Interface (least attractive way to program, unless you already know C very well)



- Generally, compilation is as follows:
  - Linux: gcc -o TCPEchoClient TCPEchoClient.c DieWithError.c
  - Solaris: gcc -o TCPEchoClient TCPEchoClient.c DieWithError.c -lsocket -lnsl
  - Both: Add -lpthread to both Linux and Solaris for the threads example
- Note for older machines (e.g., SVR4 variants such as Solaris 2.5 and before): A few more include files may be necessary. If you get compile errors about u_short, try editing the include files as follows (**order matters**):

```
#include <stdio.h>      /* for printf() and fprintf() */
#include <sys/types.h>  /* for Socket data types */
#include <sys/socket.h> /* for socket(), connect(), send(), and recv() */
#include <netinet/in.h> /* for IP Socket data types */
#include <arpa/inet.h>  /* for sockaddr_in and inet_addr() */
#include <stdlib.h>     /* for atoi() */
#include <string.h>     /* for memset() */
#include <unistd.h>     /* for close() */
```

- You can always compile C code using the GCC compiler using a command-line-interface, on most OSs. However, using an **INTEGRATED DEVELOPMENT ENVIRONMENT** (IDE) with an easy user-interface is usually easier.

- The point of above: Please spend an hour in the first week of class to get a C compiler and IDE running on your laptop. It will make programming easy and fun.

4DN4 Software and Laboratories

- Design several TCP/IP Socket applications
- Sample TCP Socket code available at website associated with reference textbook:

  http://cs.baylor.edu/~donahoo/practical/CSockets/textcode.html

  http://cs.ecs.baylor.edu/~donahoo/practical/JavaSockets/textcode.html

- Please download these sample files in the first week of class, as you will need them in the 2nd week of class.

4DN4 Software and Laboratories

There will be several laboratories, covering TCP/IP socket programming.

In 2006, we tried the new programming language C#, which was a relatively new object-oriented programming language developed by Microsoft. Using C# turned out to be difficult, so we returned to using the C programming language.

In 2007-2011, we used the 'C' programming language, and the KDE C integrated development environment running on the LINUX or Windows operating system. (We used the Xcode IDE for the Macs).

In 2013 and 2014, we used the Netbeans IDE, using 2 languages, C and JAVA, and it worked quite well. Netbeans was easier than any other IDE I have used.

In 2015 the prof, will also experiment with the PYTHON programming language. It resembles MATLAB and JAVA, and tends to be easier than JAVA to get started with. The prof. will use a unix terminal window and a command-line interface, until an easy-to-use and well-documented PYTHON IDE is found.

Some previous labs are listed here (these are Tentative):

4DN4 Software and Laboratories

• Lab 1 involves establishing a Client-Server system between two machines, for the 'Internet-of-Everything' concept.

• Lab 2 involves establishing a FTP-based Client-Server file-sharing application called "McNapster". Users can find other machine(s) on the web and search them for files, typically .jpg or .mpg files, and then download the files or upload new files for sharing.

• Lab 3 involves establishing a "Peer-to-Peer" (P2P) application, called "Mactella", similar to the Gnutella, Kazza and Bit-Torrent P2P systems. Each peer discovers and maintains a list of other known peers found on the Internet. Each peer is able to request and download files from the pool of known Mactella peers. Peers are able to share their lists of music / pictures / video of other known Mactella peers, bypassing any centralized server. (Gnutella, Kazza and Bit-Torrent were organized in this manner, after the centralized server in Napster as shut down by the US government in 2000.)

A team may propose to use another language. Other languages we will consider include C#, Perl, or Objective-C. However, we (the Prof and TAs) will not be able to provide any technical support for these other languages.

4DN4 Submitted Assignments

Assignments will involve programming in Matlab. Matlab is well suited for mathematical programming, which our assignments will require. (PYTHON can also be used).

Tentative Assignments:
1) Progamming Dijkstra's Shortest Path algorithm, and routing in networks.
2) Programming a Weighted Fair Queueing scheduler.
3) Programming a Leaky Bucket Traffic Shaper.

We will ask for electronic submission of assignments, using Avenue-to-Learn. Students should keep backup copies of all submitted work. In the event that a submitted work can't be found, the student will need to submit another copy of the original submission. Students can work in groups of 2 students, and one student can submit one assignment report in Avenue-to-Learn with both student's names and numbers. (However, the retroactive addition of student names to a team assignment is not allowed.)

4DN4 Laboratory Access – JHE 234

• In 2015, the Labs will be run in the general engineering computer labs in JHE 234. We hope to have TAs present in these rooms during our scheduled lab times.

• Supervised labs run Mon, Tues, Wed, Thurs., every other week.

• LAB: Mon, Tues, Wed, Thurs, 2:30-5:20 pm, JHE 234

• LAB ACCESS: You may be able to access the lab after hours, but you may not have any TA support.

Marking Scheme

• Assignments (approx. 3)	approx. 10 %
• Laboratories (approx. 3)	approx. 15 %
• Presentation	approx. 5 %
• 1 Scheduled Midterm Test	approx. 20 %
• 1 Final Exam	approx. 50 %.

• All grade weights are +10 or –10 %, subject to prof’s discretion

• **Participation:** Each year the prof. may award a few percentage points to selected students who participate in the class lectures by getting socket code running, to interact with the profs machine, by asking questions or contributing interesting perspectives, or just making the class fun.

• Missed Quizzes/Finals : Marks are taken from the final exam, or potentially an oral quiz / exam.

Software Development Tools

• The Prof will spend time developing, compiling and executing socket programs in C, PYTHON or JAVA, during class lectures in January/February. Our laptops should be able communicate directly over the 802.11 Wifi network (if available in ABB-136).

• It will be more fun if you can also edit, compile and run these same applications, at the same time. To do this, please install a C, PYTHON or JAVA **"Integrated Development Environment"** (IDE), on your laptop within the first week of classes.

• If you use an Apple Mac, you can also use the XCODE IDE, which supports C and Java. This software is on the install disks of the OS, and is available on the www by joining the ADC (Apple Developers Connection). You can install it.

• If you use Windows, you can use the Microsoft Visual C++ development environment.

• I recommend the NetBeans IDE for C or JAVA. It is easy to install on most Oss including MAC OS X, Windows and Linux machines (less than 20 minutes).

• Please go to the class web-site on Avenue-to-Learn: We may have tutorials on how to install the programming environments on your laptops.

Class Times - 2015

• Current Schedule:

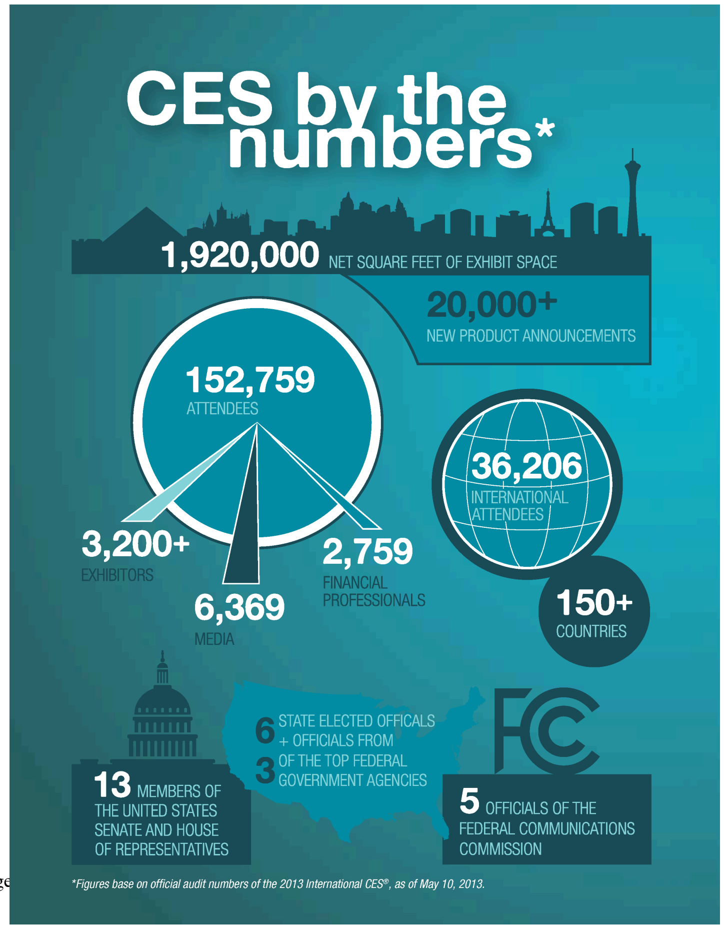
• Class Times: Mon, Wed, Thurs, 1:30-2:20 pm, ABB-136

• Tutorial: Wednesday, 9:30-10:20 am, T13/125

• The prof. and TAs will video recording the lectures and tutorials, so that you can review them afterwards.

• The profs Office Hours will after the Mon/Wed classes, 2:30, in the ABB lobby. If you email me in advance, we can also talk via a skype video link at a pre-arranged time : (my skype ID: prof_ted_szymanski). This gives everyone flexibility.

• We hope that you will have a lot of fun in this class, developing your own Internet applications for laptops and even the Apple iPhone or iPAD. Maybe one of you will create the next big web tool, such as YouTube, Twitter, Bit-Torrent, or the next big web application.


CES by the numbers*
1,920,000 NET SQUARE FEET OF EXHIBIT SPACE
20,000+ NEW PRODUCT ANNOUNCEMENTS
152,759 ATTENDEES
36,206 INTERNATIONAL ATTENDEES
3,200+ EXHIBITORS
2,759 FINANCIAL PROFESSIONALS
6,369 MEDIA
150+ COUNTRIES
6 STATE ELECTED OFFICALS + OFFICIALS FROM 3 OF THE TOP FEDERAL GOVERNMENT AGENCIES
FCC
13 MEMBERS OF THE UNITED STATES SENATE AND HOUSE OF REPRESENTATIVES
5 OFFICIALS OF THE FEDERAL COMMUNICATIONS COMMISSION
*Figures base on official audit numbers of the 2013 International CES®, as of May 10, 2013.

Questions ?