

ECE 465 Semester Project: Internet Odd-and-Even

Prof. Paris
Spring 2005

February 14, 2005

1 Overview

You may be more familiar with “rock, paper, scissors” but “odd-and-even” is a similar game. You may remember the commercial before the Olympics in which the later gold-medal winning beach volleyballers used a game of “odd-and-even” to determine who has to retrieve an errant ball from the icy ocean waters.

The essence of the game is that each of two players places a bet on either odd or even. Then, both players simultaneously show either one or two fingers and the sum of the fingers shown by both players is taken. Whether this sum is even or odd determines the winner. Note, that the choice to show one or two fingers is unrelated to the players bet.

For this project, you will design and implement a network application that allows two or more hosts to play (a generalized form of) “odd-and-even” against each other. We will generalize the game to allow more than two players as follows. Assume that N players are participating. Each player bets on a different number between 0 and $N - 1$. Then, each player selects a second number, unrelated to his bet, between 0 and $N - 1$. The players reveal this second number to the each other and compute the sum of all numbers and find the remainder when the sum is divided by N . Clearly, the remainder must be between 0 and $N - 1$ and the player with the corresponding bet wins the round.

To illustrate, assume that there are 3 players P_i , $i = 0, 1, 2$ betting on 0, 1, and 2, respectively. Let the second numbers chosen by the three players be 1, 2, and 2. Then, the sum is 5 and the remainder after division by 3 is 2. Hence, player P_2 wins this round.

A few additional challenges must be overcome to translate the game into cyberspace. Among them, there must be a mechanism to determine the hosts that want to participate in a game and, thus, the number of players. This is to be done automatically through an appropriately designed protocol. Also, the concept of simultaneously revealing the chosen number (by flashing the corresponding number of fingers) doesn’t translate very well at all. Instead, we must rely on cryptographic methods to achieve the same effect and prevent cheating. Both of these aspects will be discussed again below.

2 Principles of Operation

As described above, each round unfolds in three stages:

Discovery: Any hosts interested in playing are automatically identified and communicate their interest to play. At the end of this stage the number N and identity of the players for the round is known.

Betting: Each participating host, selects a number between 0 and $N - 1$ and communicates this number to the other players. If two or more hosts pick the same number, then it is possible that there are either multiple winners or no winner at all. At the end of this stage, each player will have selected a number between 0 and $N - 1$ and this choice is known to all other players. selecting the number for the bet should be either automatic (i.e., done by the application) or by a human.

Play: Each participating host, selects a second number (either automatically or via human input) between 0 and $N - 1$, encrypts this number, and sends it to all other players. Once all players have transmitted their encrypted numbers, they reveal the cryptographic key used for encryption so that the selected numbers can be revealed. With the numbers in the clear now, the sum and remainder can be computed by every player and a winner is determined.

Notice how each step, involves action on the part of each host and communication between hosts. Consequently, a suitable set of protocols is required to allow independently designed applications to play together. The work products for this project fall into two categories: each group (see below) will have to design and implement a network application (i.e., a program) to play Internet Odds-and-Evens; all groups together are to design a set of protocols describing the format, contents, and sequence of messages exchanged between participating hosts. Ultimately, the applications developed by different groups are to play against each other.

3 Requirements

The three stages described above build on each other; discovery must occur before the betting stage, and the betting stage must occur before the play stage. This dependency dictates the sequence in which the application should be standardized and implemented.

The following provides a closer look at the specific events that must occur in each stage. It also contains some thoughts to get you started designing the protocol and application.

3.1 Stage I: Discovery

All hosts collaborate to support discovery. Clearly, it is not possible to rely on uni-cast message (point-to-point messages) at this stage because hosts don't

know about each other (let alone their IP addresses). Hence, it will be necessary to use either broad-casting or multi-casting for discovery. Both broad-casting and multi-casting are straightforward to use. Broad-casting generally limits communications to hosts on the same subnet while multi-casting can facilitate communications across routers.

Either way, hosts can send invitations to play and solicit responses from other hosts. Fundamentally, one host must take the initiative to send an invitation and other hosts must listen for such an invitation. All hosts should be able to send an invitation for a new game and should do so periodically before until other hosts are found and again after a round has been completed. Once an invitation has been received no further invitations should be sent. The response to an invitation should reveal the identity of the responding host.

The protocol to be designed for this stage should specify the format of messages used for invitation and responses. It should specify when either of these messages should be sent, including by whom and how often.

The products for the discovery phase must be a protocol (agreed upon among all groups) implementations demonstrating discovery functionality. Specifically, the invitation and response mechanism for discovery must be standardized and implemented. It is *strongly* recommended that you undertake thorough testing within and between groups.

You should start thinking about a fundamental design decision you will have to make. The standards body will have to decide if communications after discovery should continue to use multi-casting or if uni-casting should be employed. The former has the advantage that it naturally supports disseminating information to all participants but it is limited to UDP at the transport layer; this may make it necessary to include some form of error control in the application (e.g., to recover from a packet that did not reach one host but all others). With uni-casting, TCP can be employed to provide a reliable transport layer. However, disseminating information to all participants is not well supported. An option is to “elect” a master for each round (e.g., the sender of the invitation that started the round), send all information to the master and let the master distribute information. This design decision has a strong impact on the design of the subsequent stages and should be carefully considered.

3.2 Phase II: Betting

The second phase is probably the most straightforward of the three phases. Using the mechanism that you have selected to distribute information, each participant selects a number between 0 and $N - 1$ and distributes that information to all other players. Your program should allow the selected number to be either automatically generated or entered by a human user.

The protocol to be designed must support the transmission of the selected number to other participants. Specify the format and contents of corresponding protocol messages and how they are distributed. Add betting functionality to your application and conduct interoperability testing.

At the end of this stage, all hosts should be able to send their bet and receive the bet that other hosts have placed.

3.3 Phase III: Play

The final phase will likely require two rounds of messages to be exchanged. The first set of messages, contain the numbers selected by the individual hosts in encrypted form. Encryption is necessary to prevent that a host cheats by first observing all the other numbers and then selecting the number that will win the game for him. Think of these messages as sealed envelopes distributed to all players.

A slightly less obvious way to cheat, is for a host to have multiple encryption keys that lead to different decryptions of his message. When keys are revealed (see below) the cheating host could wait again until he knows all the other numbers and then select the key that decrypts his message so that he wins the game. To prevent this form of cheating, each host must submit some proof that he did not change his encryption key. A cryptographic hash function may be the appropriate mechanism for this purpose.

As indicated already, upon disseminating all sealed (encrypted) numbers the hosts reveal the keys they used to encrypt the number. This implies that a different key must be used in each round. Revelation of the keys allows decryption of the numbers so that they can be added, the remainder can be computed, and a winner (or winners) is found.

The main difficulty in this phase is likely the design of an appropriate cryptographic protocol to ensure integrity of the game. Once the cryptographic procedures are identified the dissemination of the information should be straightforward.

As before, the products of this phase should be a set of standards describing the exchange of information, including cryptographic protection. Extend the functionality of your application to implement these standards and arrive at a fully working application. Conduct thorough testing within and between groups. The final applications are demonstrated in class.

4 Organization

4.1 Groups

You will work on this project in groups; the group assignments are given below (see section 5). Each group is responsible for organizing its work. It is recommended that responsibilities are assigned directly to group members; such responsibilities should include at least:

- Group leader
- Software developers
- Testers

- Standards/Protocol Designers
- Documentation Writers and Manager

Clearly, each group member will likely have multiple responsibilities.

4.2 Standards and Protocols Design

As explained above, you will be responsible for designing most of the protocols to be used. These will be relatively simple. Nevertheless, the groups will have to work together to develop these standards. Otherwise, it is highly unlikely that the peers from different groups will work together. For this purpose you should form standards committees to which all groups contribute. The work products of these standards groups are documents that specify the protocols to be used. These committees should also coordinate interoperability testing.

You should begin with the design of the protocols for the applications needed in phase I and later proceed to the phase II and phase III protocols.

4.3 Schedule

The project begins immediately and will run through the end of the semester. Groups will present their projects and conduct interoperability demonstrations during the last class of the semester. You should plan to complete the discovery phase the week of March 14, the betting phase the week of April 4, and the play phase the week of April 25.

I will schedule group progress reports and reports from the standards committee (alternating) every week during class. The first progress report from groups will be on February 22. The first report from the standards group will be on March 1.

4.4 Deliverables

Each group must maintain a project web site that documents the group's progress. At the very minimum that web site should document the group's plan, including assigned responsibilities, milestones, and a list of current problems. Group "proprietary" information should not go on the web site unless it is password protected. Each group will have to submit a final report that documents the design and implementation.

4.5 Programming Languages and Computers

You should be able to use any computer for this project as long as it is able to connect to the Internet. You may have problems if your peer is behind a firewall. Each group is to field at least one peer.

The programming language to be used is your choice. Use the programming language that you are most comfortable with. Both C/C++ and Java are good choices. If there is expertise in your group, Perl or Python may work even

better. It may be easier to find a good compiler to work on Unix, but with the availability of CygWin you have access to the same tools on Windows. Perl and Python also work both on Unix and Windows.

4.6 Resources

For the discovery process, you may want to refer to the universal plug-and-play (UPnP) specification. These are good starting points for UPnP technology:

- The UPnP Forum: <http://www.upnp.org>
- Intel UPnP Technology page: <http://www.intel.com/technology/UPnP/>

4.7 Grading

In addition to the final report, each group member will turn in a confidential evaluation of all the other team members. This peer evaluation will be combined with my overall evaluation of the group's work to arrive at a final grade for the each student. Hence, both individual efforts as well as the group's performance will factor into the final grade for the project.

5 Group Assignments

5.1 Group I

- Goldberg, Jonathan S.
- Malla, Arvind
- Mei, Frederick J.
- Singh, Bikramjit
- Vinjamuri, Ramadivya

5.2 Group II

- Ahmad, Osman
- Choi, Keith J.
- Flores, Diana L.
- Hutchison, Richard C.
- Norris, Joel T.

5.3 Group III

- Andorful, Benjamin
- Bae, Hyun J.
- Nguyen, Thien Tam L.
- Ogundiyun, Mayowa
- Sylla, Aliou

5.4 Group IV

- Guevara, Renato
- Hussein, Abdurahman B.
- Ramia, Fouad J.
- Subaitani, Daniel F.
- Walker, Marcus A.

5.5 Group V

- Ashman, Zachary C.
- Jung, Seungjun
- Kumar, Jatinder
- Singh, Jaskaran
- Vaidya, Suchit A.

5.6 Group VI

- Fatehi, Ehsan
- Gable, Brad J.
- Murray, Daniel C.
- Nakai, Keith P.
- Mohsseni Behbahani, Maytham

5.7 Group VII

- Bajracharya, Sujan M.
- Do, Khang
- Duong, Lam H.
- Tran, Thinh D.
- Joshi, Sudeep