Mobile Computing #MC06 Broadcast

CS60002: Distributed Systems
Winter 2006-2007

Today

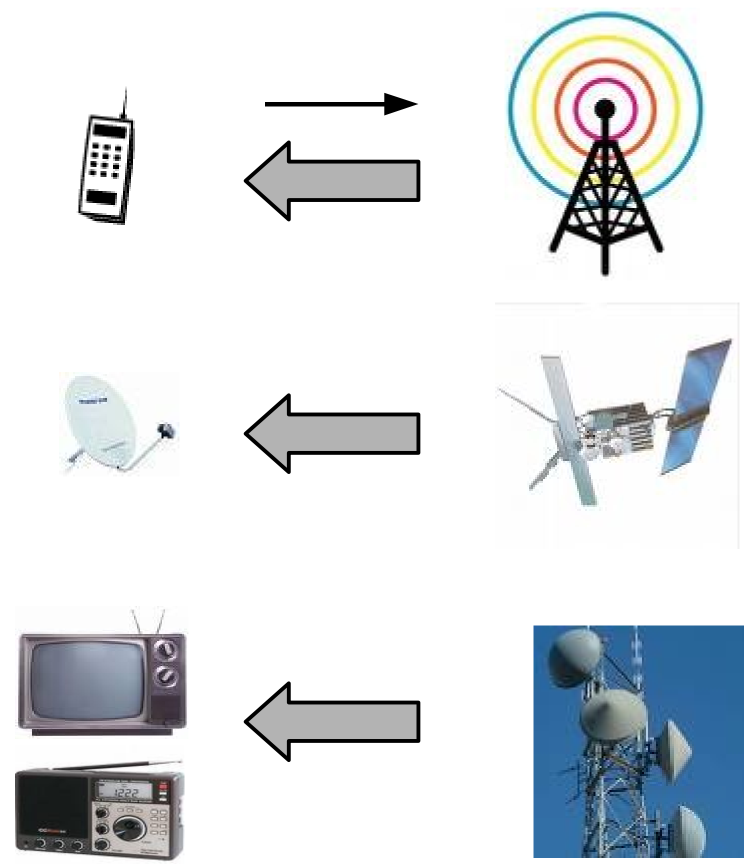
- ✔ Device databases
  - Flash, OR/direct
- ✔ Synchronization
  - Algorithms
- ✔ Push/notifications
  - Scale to MM
- ✗ Handheld design
  - CPU, RTOS, battery
- ✗ Core Mobile Apps
  - Email/IM, PDA,browse
- ✔ IP Protocols
  - IMS
- ➔ Broadcast
  - Algorithms
- • Device Management
  - Software & Config

Agenda

- Asymmetric Communication
  - Definition, reasons, examples
- Broadcast
  - Protocols
  - Commerical products
- Algorithms for Broadcast Communication
- Reading Assignment

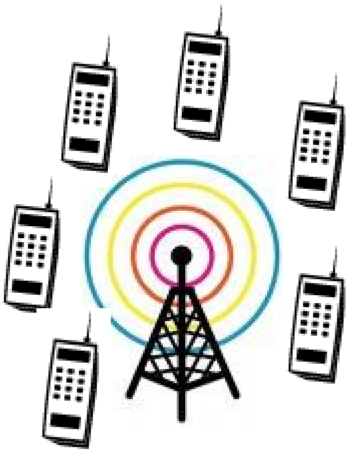
Asymmetric Communication

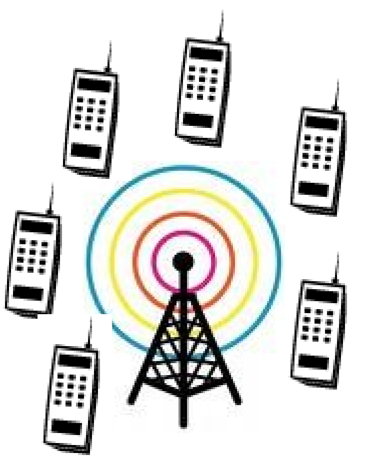
- What?
  - Server ==> Mobile
  - Mobile --> Server
- Why?
  - Shared medium
  - Power + secrecy
  - Regional
  - Application
    - Shared interests
    - Urgency



Broadcast

- Asymmetric
  - (S ==> C) only
  - No (C --> S) at all!
  - Except for stats
- Errors
  - Block erasures
- Timing
  - Clients are not synchronized





Products and Protocols

- Radio
  - AM, FM
  - HD Radio
- TV
  - NTSC/PAL, dNTSC
  - DVB-T/DVB-H
  - DMB/MediaFlo
- DVR/PVR
  - Tivo, MythTV
- Cell Broadcast
  - aka CBS, SMS-CB
- Datacasting
  - IPDC in DVB-H
  - UDcast from INRIA

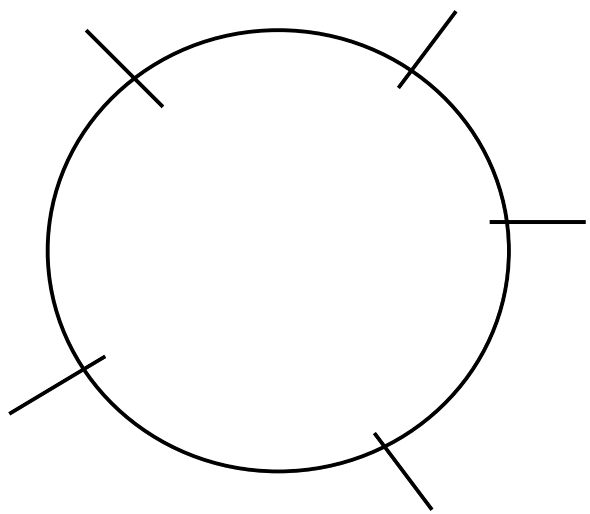
Live vs. Off-line

- Live Broadcast
  - Radio, TV, CB Radio,
    - News, weather, traffic, stocks,
  - Typically used only for streaming media
- Broadcast of on-demand content
  - MovieBeam, Top Up TV
  - DVR/PVR (e.g., TiVo)
  - Idle screen on mobiles (e.g., CellTick.com)

Rest of this lecture ...

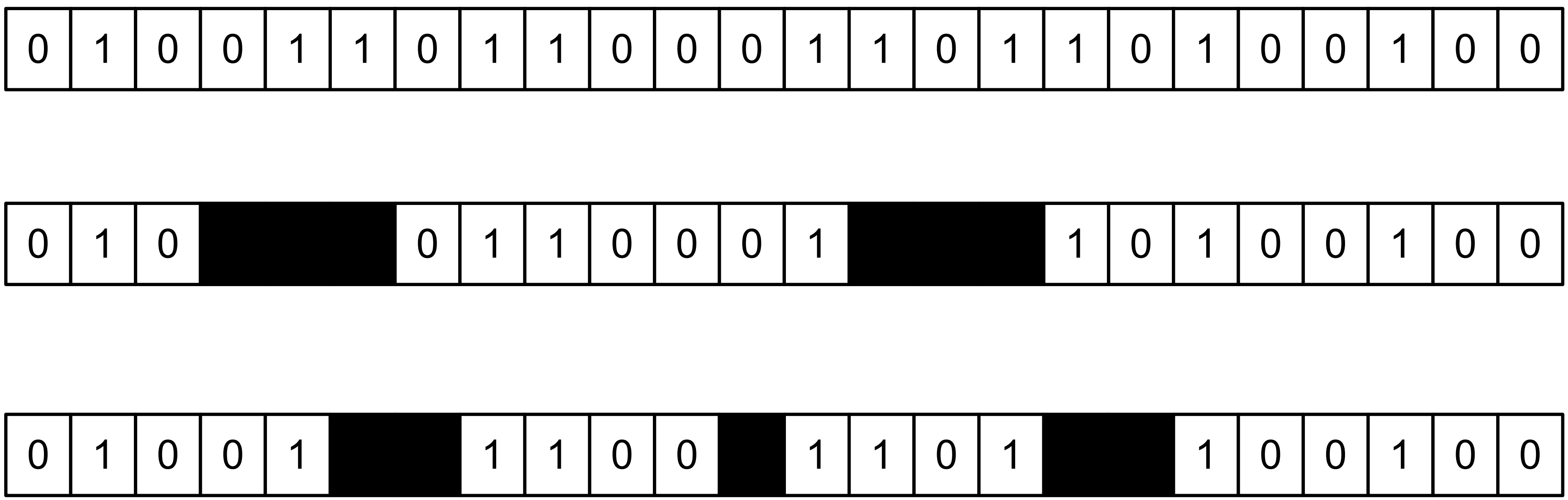
- Focus of one problem
  - Efficient, error-correcting broadcast
- Restrictions
  - Server broadcasts BLOBs to MM clients
  - Zero feedback from clients to server
  - Each client starts listening at time of its choosing
  - Each client drops an arbitrary subset of packets
- Parameters
  - Communication overhead
  - Encoding and decoding efficiency

Clients listen at different times



- Different start times
  - e.g., User choice
- Different interruptions
  - e.g., Coverage
- Option: Loop
  - Wastes bandwidth
  - Wait for last piece
- Better solution?

Clients drop packets



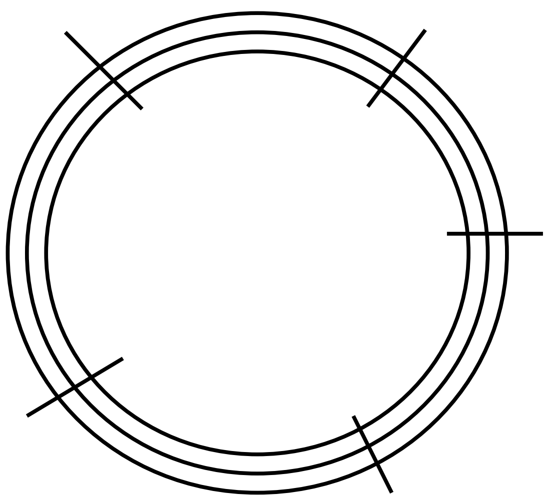
- Different clients drop different packets
  - (e.g., dead spots in coverage, lifts)

(BTW, typically, packets will be bigger than single bit)

Requirements

- Bandwidth efficiency
  - But, should support large stretch factors
- Time independent
  - Independent start times, resilient to interrupts
- Time efficient
  - Avoid long waits for the last packet
- Computationally efficient
  - Especially for decoding on mobiles
- Scalable

Data Carousel/Broadcast Disk



- Broadcast packets in loop
  - Inefficient use of bandwidth
  - But, often app requires it
- Clients listen till last packet
  - Schedule saves power
  - But not time
- Multiple tracks+redundancy
  - Saves time

Erasure Codes

- Type of forward error correcting (FEC) codes
  - Server pads data with redundancy
  - Client reconstructs data from partial reception
- Encode message
  - From k blocks into $k+l$ blocks
  - Original recovered from any k' blocks
- Rate is $(k+l)/k$
  - Measure of bandwidth efficiency

Erasure Codes (contd.)

- Rateless erasure codes
  - Rate $(k+l)/k$ not fixed a priori
  - Keep adding more packets as needed
- Optimal erasure codes
  - $k' = k$
- Near optimal erasure codes
  - $k' = (1+\epsilon)k$

Reed-Solomon Codes

- Optimal erasure code
  - But not (yet) computationally efficient
- How does it work?
  - Data = coefficients of a polynomial of degree k-1
    - $a_0 + a_1x + a_2x^2 + ... + a_{(k-1)}x^{(k-1)}$
  - Encode data as values of polynomial at *(k+l)* points
  - Client receives any k values
  - Solves system of equations to recover a_0 thru $a_{(k-1)}$

Reed-Solomon Codes

- Optimal
  - Any k blocks can be used to recover original k
  - But, fixed rate (not rateless)
- Computationally challenging
  - Practical considerations enforce small coefficients
  - Decoding solves large system of equations
    - And these systems are not sparse

Reed Solomon Codes

- Reading assignment
  - Required
    - “Encoding/decoding Reed Solomon codes”, Matache
      http://www.ee.ucla.edu/~matache/rsc/slide.html
      (The slides on decoding are optional)
  - Optional
    - “Reed-Solomon Codes”, Bernard Sklar
      PDF posted at course website.
      (This is good reference material as you read Matache.)

Tornado Codes

- Near Optimal
  - Need (1+\epsilon)k blocks to construct original k
  - Encoding complexity proportional to -log(\epsilon)
  - But, requires fixed rate (not rateless)
- Brief sketch
  - (Please take notes from blackboard)
- Reading assignment (Optional)
  - Byers, et al 1998 (Omit Sections 6 & 7)

LT Code

- Near Optimal and rateless
  - Need (1+\epsilon)k blocks to recover k
  - Need not fix rate ahead of time
- Brief sketch
  - (Please take notes from blackboard)
- Reading assignment (Optional)
  - Byers et al, 2002 (Sections I thru VI only)

Recap

- Asymmetric Communication
  - S ==> C, C --> S
- Broadcast
  - Rate: Ratio of broadcast size to content size
  - Optimal: Does it suffice to receive only content size
- Encodings for Broadcast Communication
  - Reed-Solomon, Tornado, LT
- Reading Assignment
  - Reed-Solomon (required), Tornado & LT (optional)