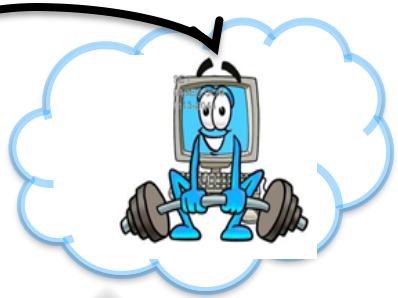
Cloud is not a silver bullet: A Case Study of Cloud-based Mobile Browsing

Ashiwan Sivakumar[1], Vijay Gopalakrishnan[2], Seungjoon Lee[2],
Sanjay Rao[1], Subhabrata Sen[2] and Oliver Spatscheck[2]
[1] School of ECE, Purdue University
[2] AT&T Labs – Research

PURDUE UNIVERSITY

at&t

Cloud Augmented Mobile Web Browsing



Can I do this job?

But I'm out of fuel

Offloading heavy-lifting to the cloud

- Reducing web latencies
- Reducing power consumption
- Saving data transfer $$$

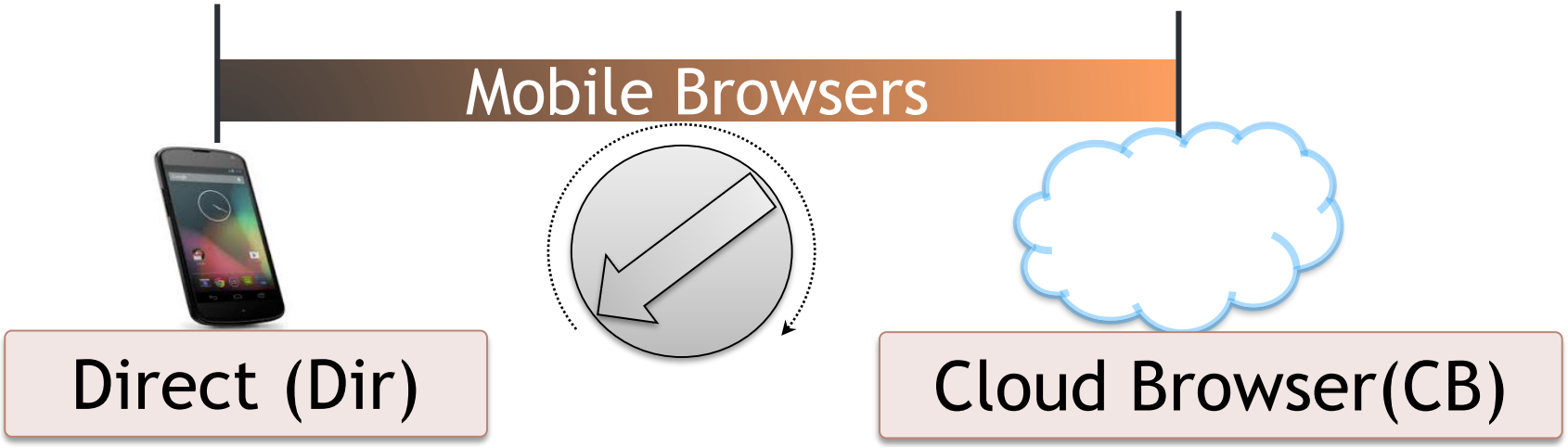
Cloud central theme in mobile app development !!

Re-evaluating Cloud-based Mobile Browsing

- Increase in processing power
  - CPU speed : 4x in 6 years [ARM]
- Cellular networks becoming ubiquitous
  - Mobile traffic growth : 7x in 4 years [CISCO]
- Battery continues to be a resource limitation
  - Cellular radio interface significant component

Time to revisit assumptions given new trade-offs

Design Space of Mobile Browsers



Fully device-based

Traditional browsers Android

Partly uses cloud

Amazon Silk, Chrome Beta, Wang et al. MCC'13 etc.

Cloud-heavy thin client

SkyFire, Opera Mini, B. Zhao et al. ICDCS'11 etc.

- Proprietary nature of solutions
  - Need for a systematic understanding

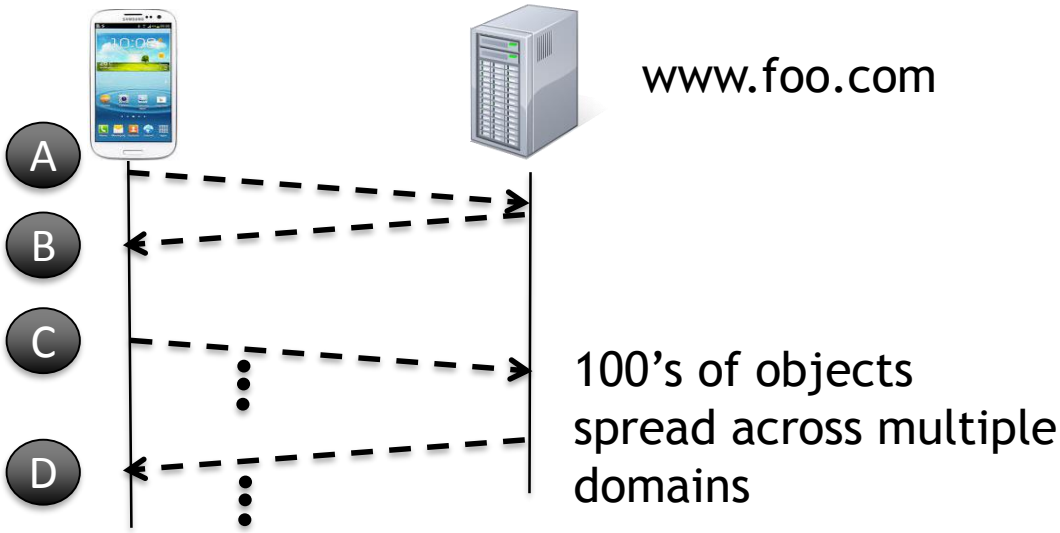
Our Contributions

- First step towards understanding trade-offs in architecting mobile browsers
- Study the functionality of an operational mobile cloud browser (CB)
  - A popular browser with a user base of over 300 M
- Key findings
  - Offloading JavaScript (JS) can hurt
    - Increases network energy for 60% of pages (~10J worst case)
    - Made worse in an interactive session (~60.9J)
  - Data compaction != Network energy savings
    - Increases network energy for 80% of pages (~10J worst case)

Outline

- Page Download Process – Direct Vs. CB
- Setup, Methodology and Metrics of Interest
- Evaluating CB – Performance and Energy
- Conclusions and Future Work

Mobile Web Browsing 101



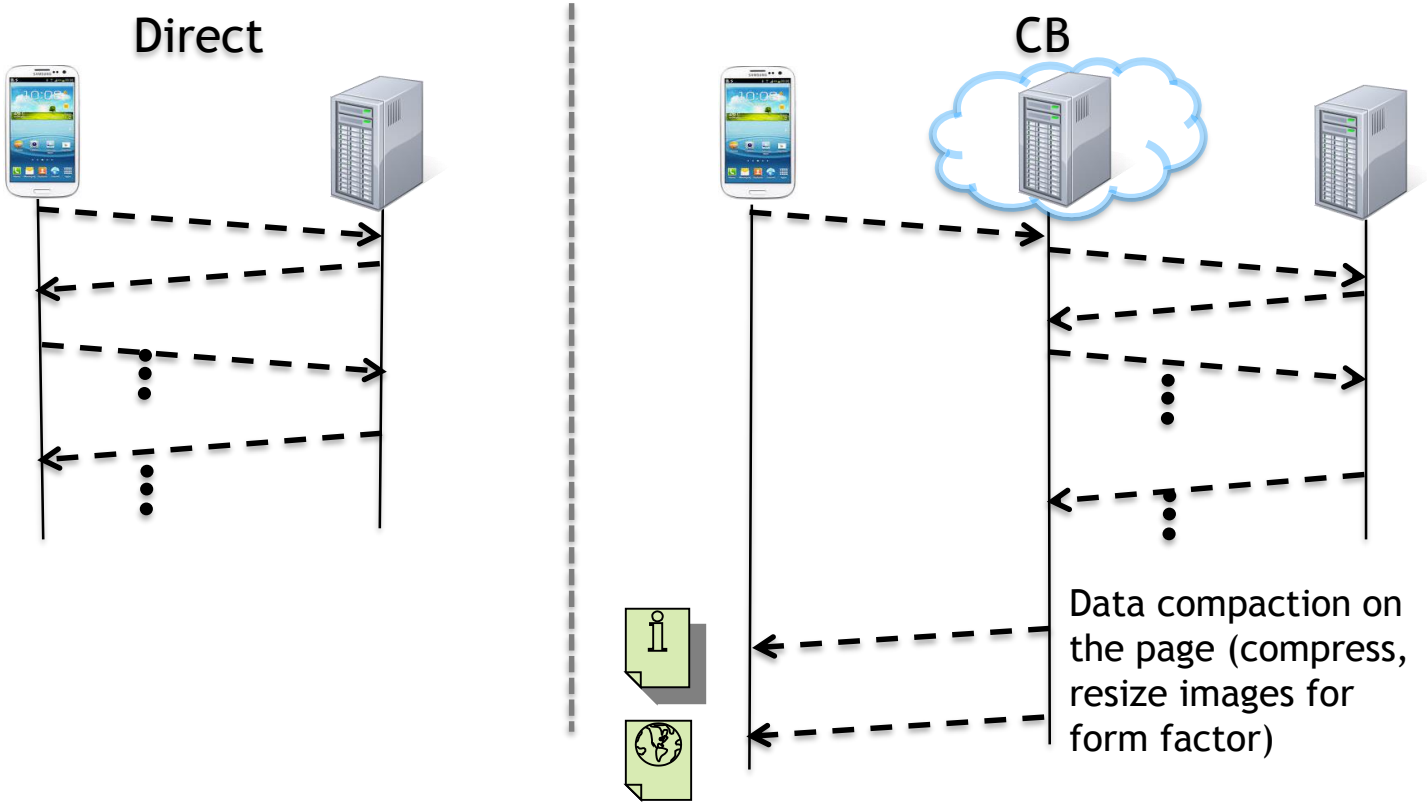
A – HTTP GET index.html

B – Parse index.html tag-by-tag

C – Fetch required objects spread across many domains

D – Evaluate JavaScript (JS), Cascading Style Sheets (CSS)

Direct Vs. CB



- CB runs JS in the cloud
- Sends compact page in proprietary format

Outline

- Page Download Process – Direct Vs. CB
- Setup, Methodology and Metrics of Interest
- Evaluating CB – Performance and Energy
- Conclusions and Future Work

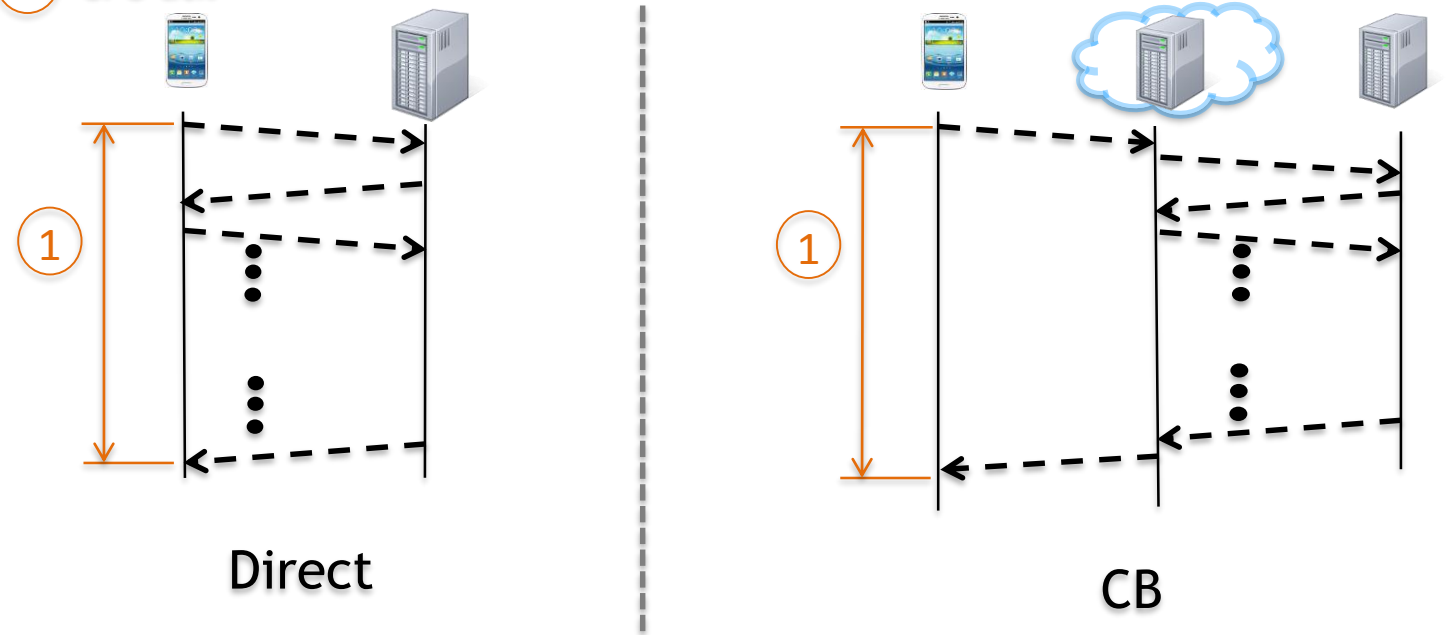
Setup and Methodology

- Setup :
  - Samsung galaxy S3 phone
  - 4G LTE network
  - 40 from top 100 pages in Alexa
- Methodology :
  - Conduct active measurements
  - First-time download disable local caching
  - Direct, CB – multiple back-to-back runs
  - In the night time
  - Each run 60 sec long

Metrics of Interest

2 tcpdump

3 CPU util



1 Page Download Time

2 Network Energy (ARO – Mobisys'12)

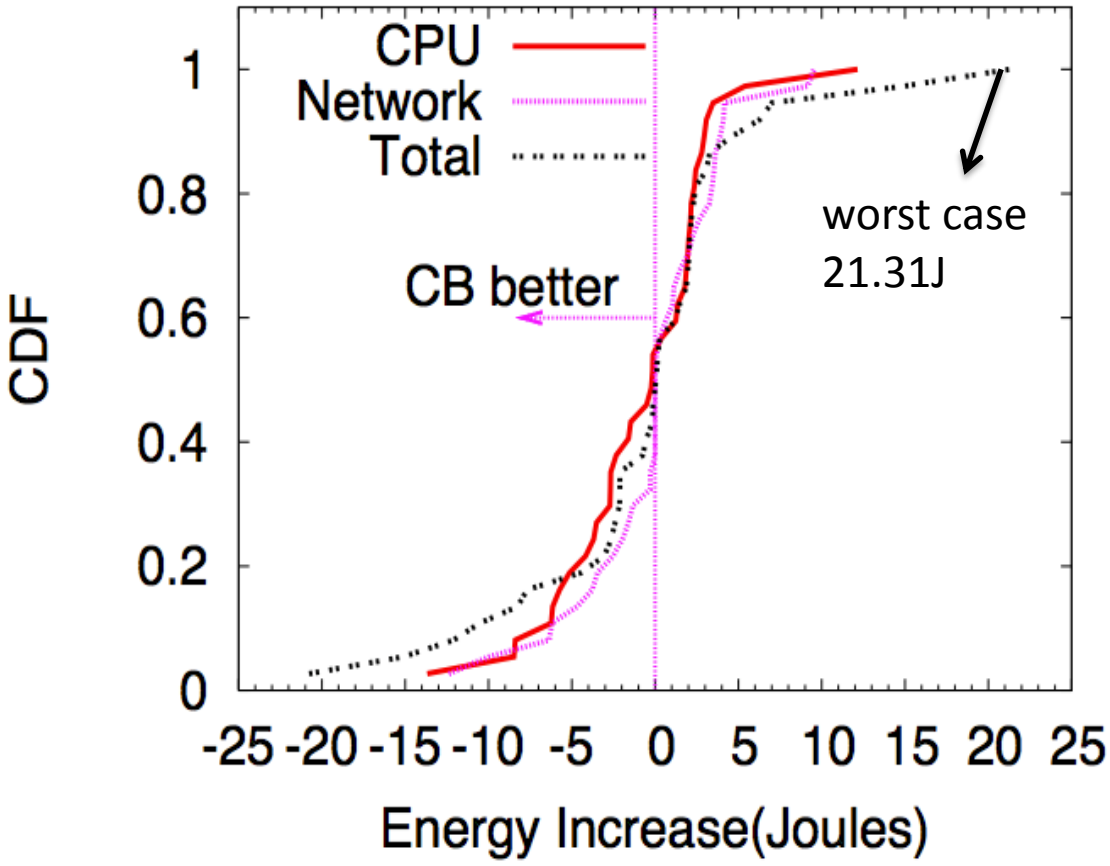
3 CPU Energy (PowerTutor - CODES+ISSS'10)

Total energy = 2 + 3

Outline

- Page Download Process – Direct Vs. CB
- Setup, Methodology and Metrics of Interest
- Evaluating CB – Performance and Energy
- Conclusions and Future Work

CB Evaluation Results



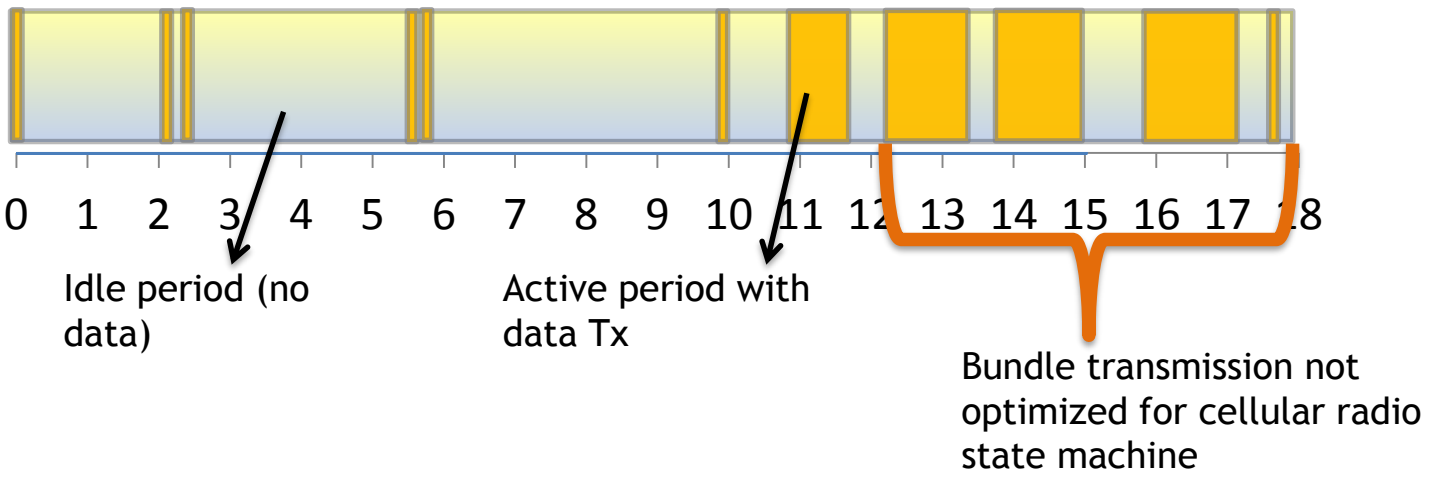
- Multiple back-to-back runs with Direct, CB
- Compute energy(download time) increase with CB
- Negative value => CB better

Not a win in both performance and energy

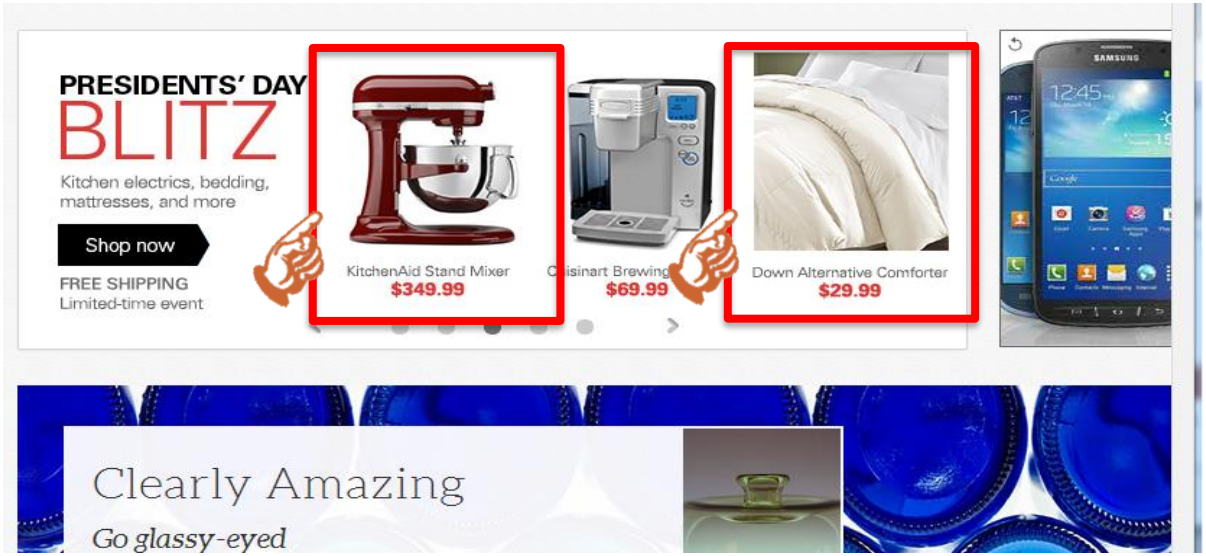
When does CB lose?

- Pages light on JS processing (intuitive)

  (e.g. 40% pages CB increases total energy by 21.31J)
- Pages with long-running JS. Why?
  - Periodic data transfer when pages change

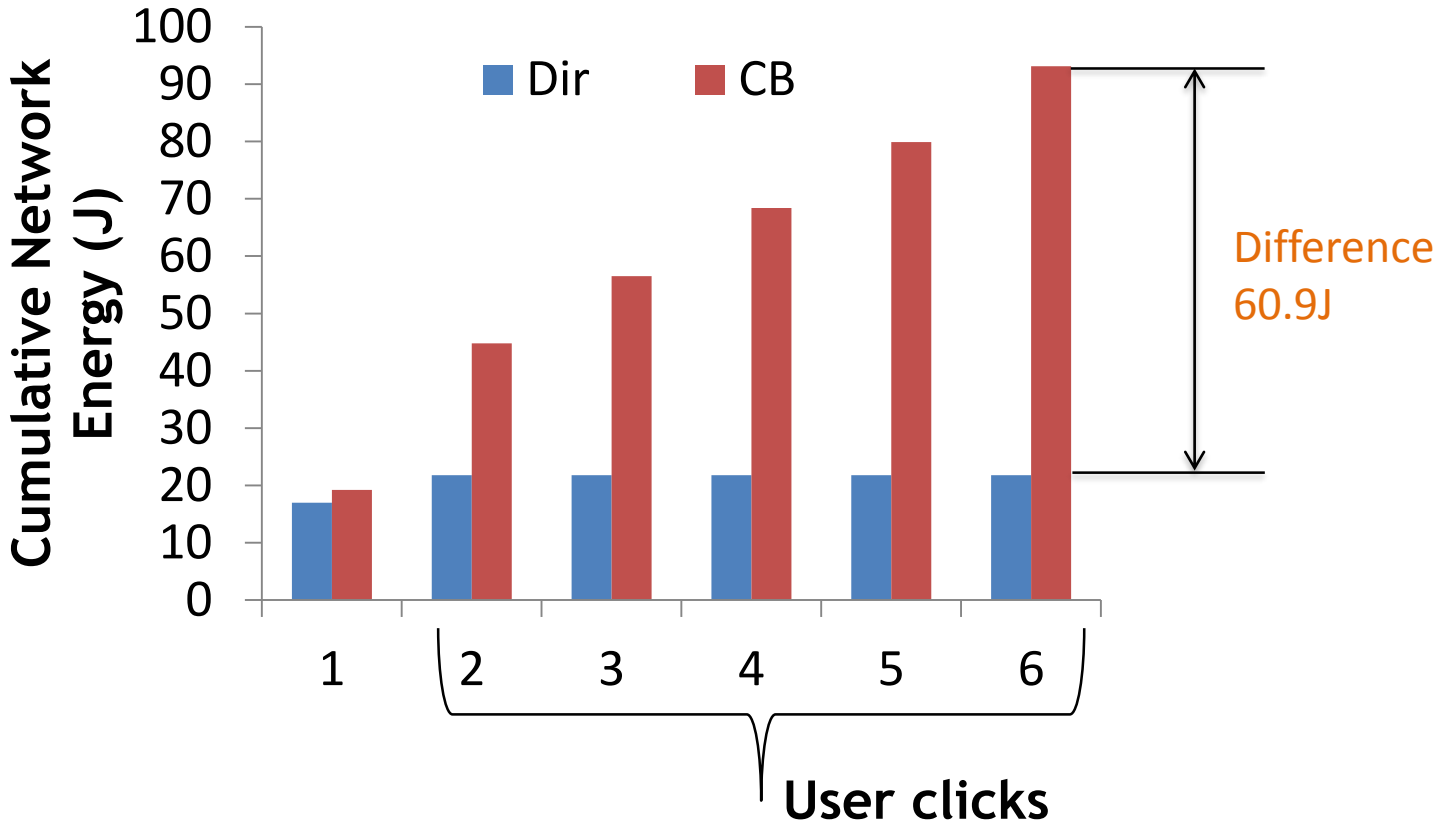


CB Overhead to Support Interactivity



- Interactive user session
- With local caching in Direct and CB

Cumulative Network Energy Increases



CB hurts more in an interactive session

Data Compaction in CB

- Achieves less compaction with no JS
- Loses in total energy for 80% of pages with no JS
  - E.g. despite 90% compression, network energy increases by 10J
- Why does CB lose despite compression?
  - Longer compression time == longer radio wait time

Data compaction != network energy savings

Conclusions

- Devices getting powerful and cellular networks becoming ubiquitous
  - Need to revisit trade-offs
- First step towards understanding trade-offs in architecting mobile browsers
- Key findings
  - ❖Offloading JS can hurt
  - ❖Data compaction != Network energy savings

Thank You!!

Questions