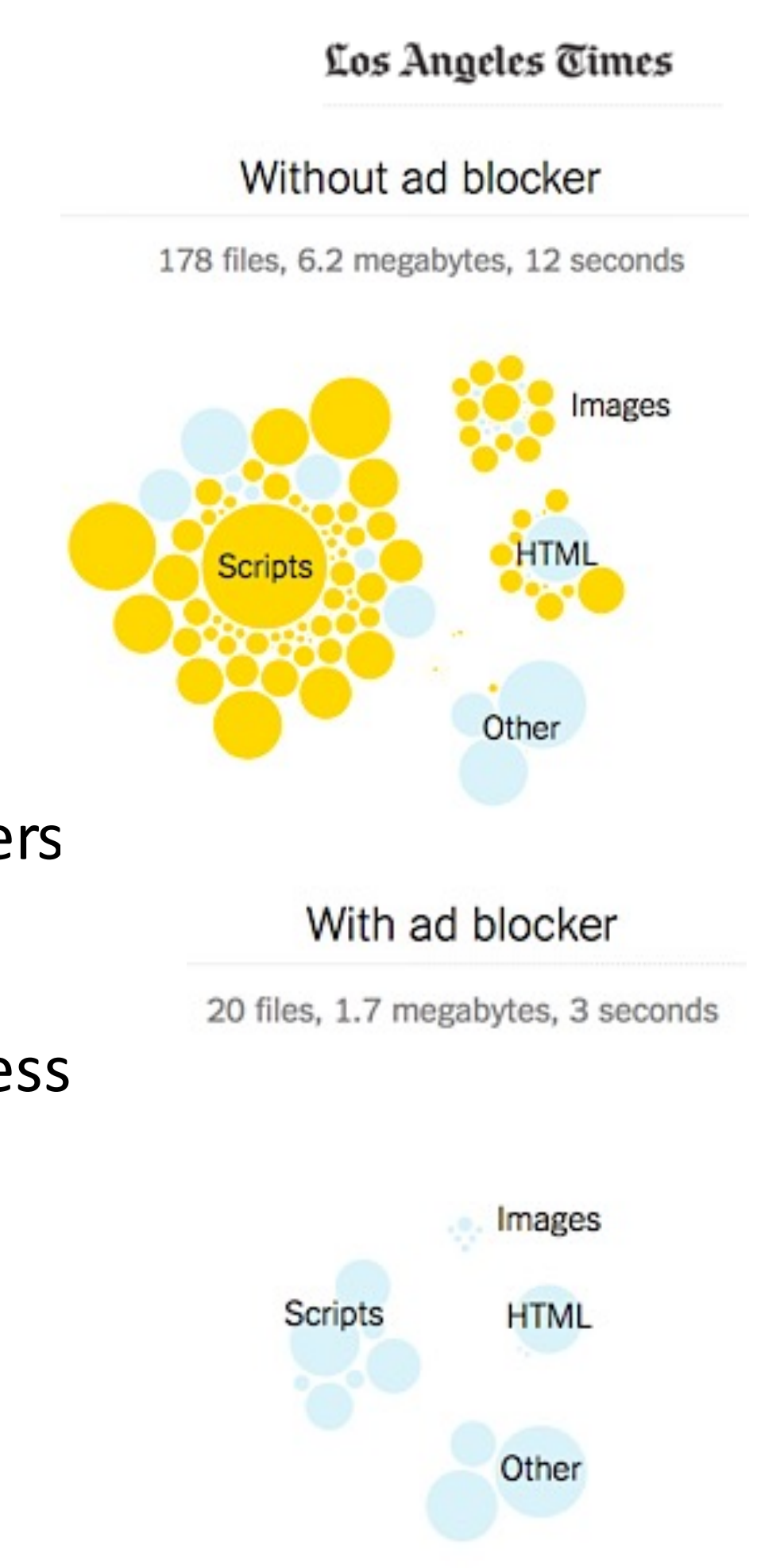
Monetization on the Modern Web: Automated Micropayments From Bitcoin Enabled Browsers

Samvit Jain
Advisor: Arvind Narayanan
Princeton University

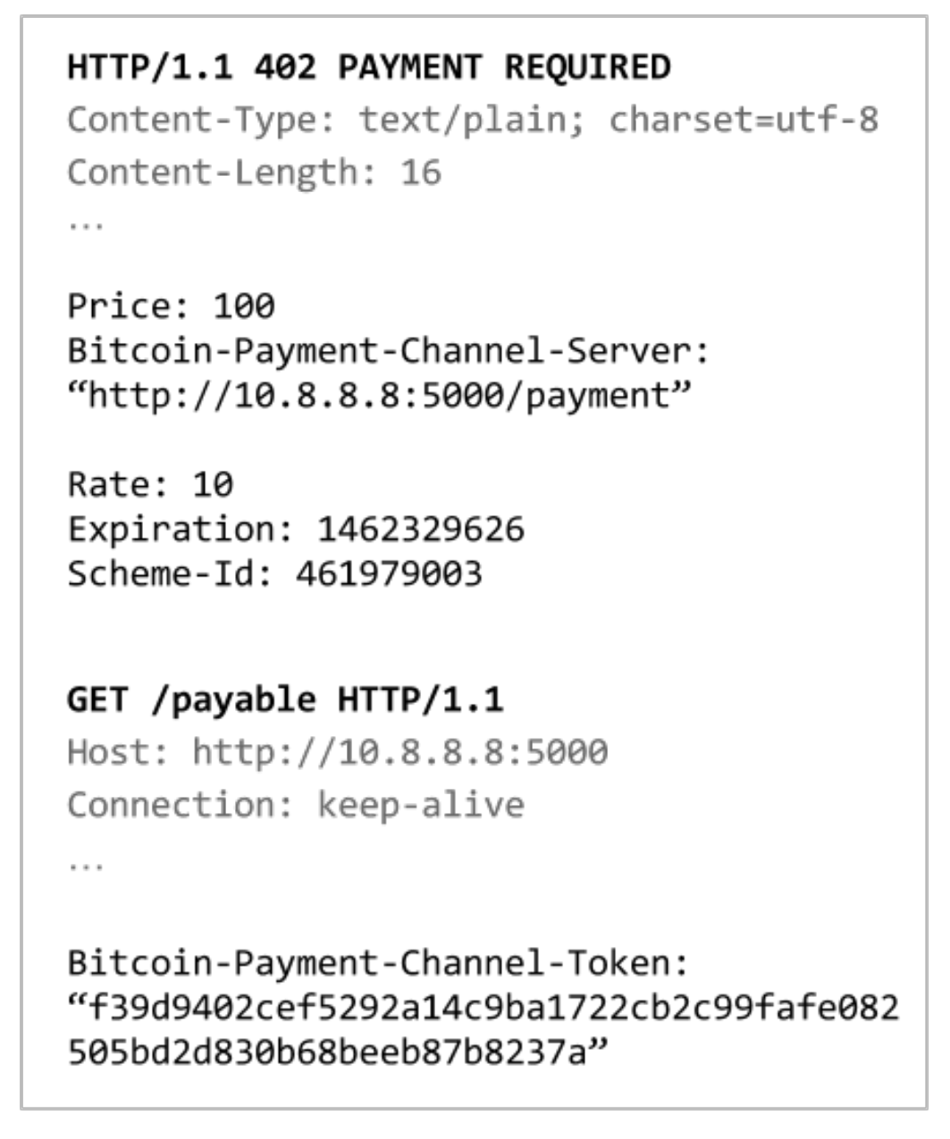
Motivation

- Loading 2000 word CNN article on iPhone 6+ takes
  - 200 HTTP requests to 25 domains
  - 2 MB of mobile data ($0.04)
  - 13 seconds to load
- Advertising
  - Webpage performance bottleneck
  - Third-party tracking, user data aggregation
  - Content quality and metrics don't line up
  - Blocking – Chrome AdBlock, iOS 9 content blockers
- Subscriptions
  - Personal data req. – credit card info, billing address
  - Time cost of registration, renewals
  - Paywalls prevent impulse or infrequent use
- Bitcoin micropayments as an alternative
  - Pay-per-use allows granular billing
  - Non-contractual, pseudonymous payments
  - Bitcoin as programmable money – potential for automation



Approach and Design

- Key goal: solve cognitive load problem
  - Associated with "click to pay" micropayments implementations
  - User must continuously make decisions about whether to pay for content, deterring usage
- Proposed solution: browser extension that funds user's web activity
  - Takes user payment preferences (i.e. thresholds) on install
  - Makes payment to site if resource price under user threshold
- Desired system properties
  - *Measurability* – can a web service accurately track/bill a client for usage?
  - *Granularity* – can service be provided/paid for in an incremental way?
  - *Setup* – how many components must be installed to use the system?
  - *Reproducibility* – can web servers easily import payments-related code?
  - *Security* – can sensitive data be protected in all cases?
  - *Scalability* – can servers accept many payments concurrently?
  - *Error handling* – can both parties recover from a broken connection?
- Metered Payments
  - Construct that allows user to pay for fluid service (e.g. social content feed, audio/video) at regular time intervals
  - To support, extended HTTP 402 Protocol by adding 1) billing scheme (e.g. rate) to instructional headers, 2) series of follow-up GET requests
- Sessions
  - Introduced browser cookie to track payment sessions
  - Allows user to cease metered payments by simply changing or closing browser tab

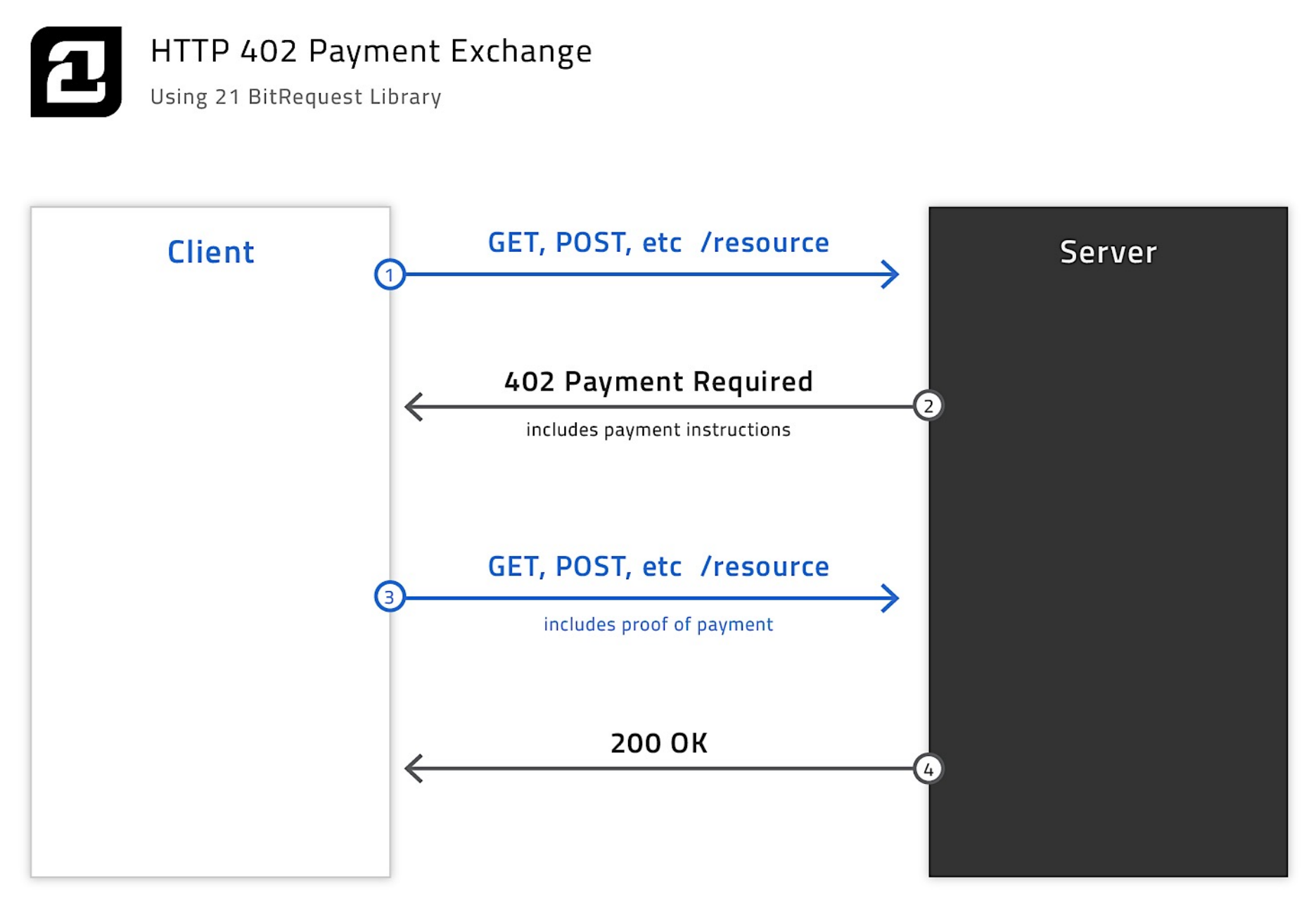


```
HTTP/1.1 402 PAYMENT REQUIRED
Content-Type: text/plain; charset=utf-8
Content-Length: 16
...

Price: 100
Bitcoin-Payment-Channel-Server:
"http://10.8.8.8:5000/payment"

Rate: 10
Expiration: 1462329626
Scheme-Id: 461979003


GET /payable HTTP/1.1
Host: http://10.8.8.8:5000
Connection: keep-alive
...

Bitcoin-Payment-Channel-Token:
"f39d9402cef5292a14c9ba1722cb2c99fafe082
505bd2d830b68beeb87b8237a"
```

Instructional (Response) and Proof-of-Payment (Request) Headers

Background

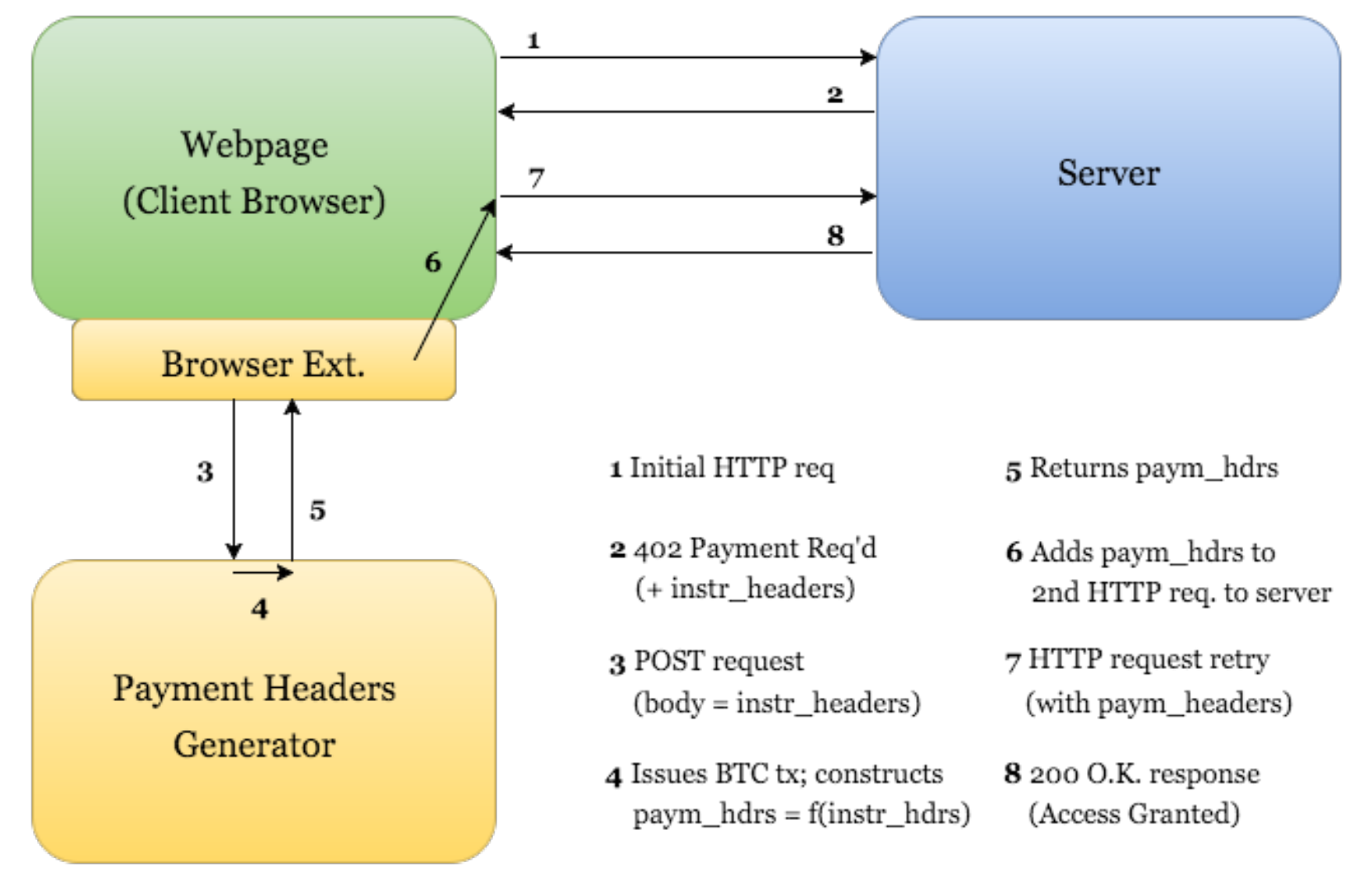
- Current online payment mechanisms
  - PayPal – hypersensitive to fraud, high percentage of transactions blocked; fraud mitigation costs dominate small transactions ($0.30 minimum fee)
  - Credit card model – requires giving online merchant one's "private key"; assumes that damage to vendor's reputation will outweigh gains from strategic default
  - Bitcoin – standard Bitcoin transactions also involve high transaction fees


HTTP 402 Payment Exchange
Using 21 BitRequest Library
Client
Server
1
GET, POST, etc /resource
2
402 Payment Required
includes payment instructions
3
GET, POST, etc /resource
includes proof of payment
4
200 OK

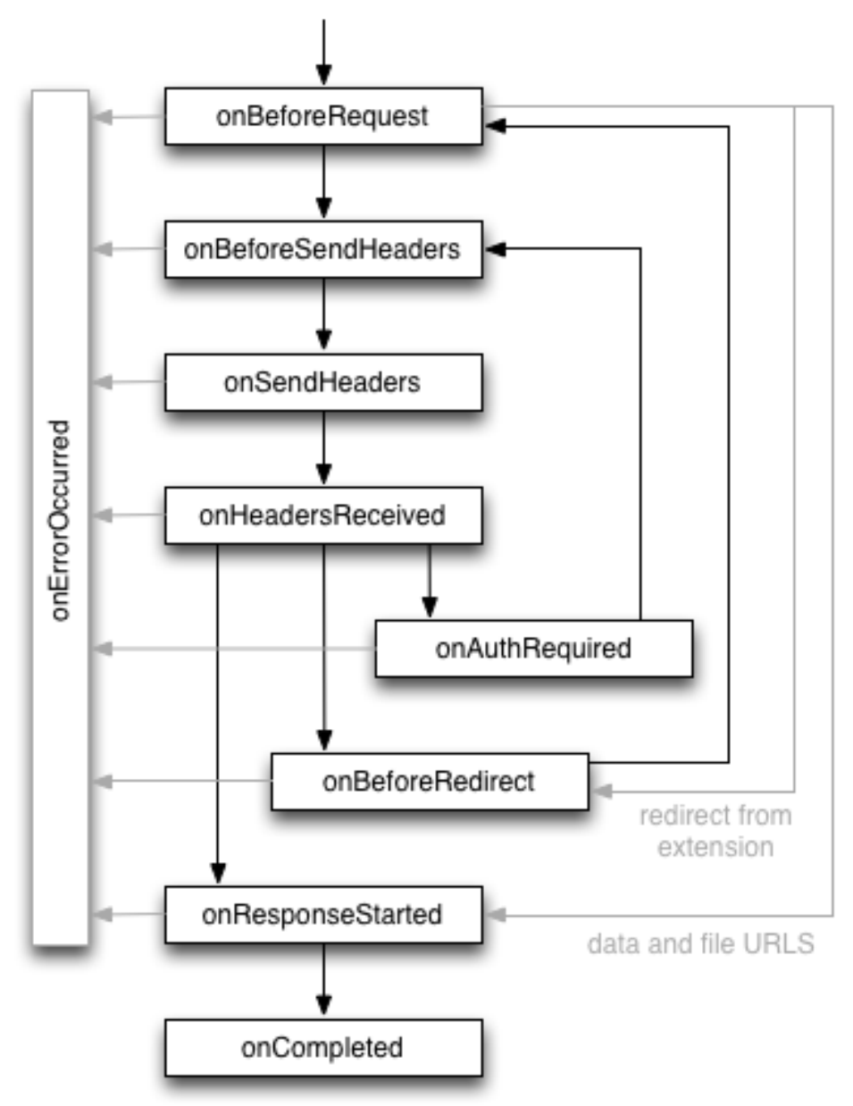
- Micropayment channels
  - Client-server contract that enables series of very small payments for a fluid service (e.g. video)
  - Uses Bitcoin tx feature called nLockTime to hold client deposit in escrow
  - Only two transactions are published to blockchain and incur fees
- HTTP 402 Protocol
  - Allows client to pay for access to a web resource via a Bitcoin transaction
  - Server returns 402 Payment Required status and instructions on first client request
  - Client retries request with proof-of-payment header fields
  - Server checks headers and grants access to resource (e.g. webpage) if valid

Implementation

- Components
  - *Webpage* – page or resource monetized via Bitcoin payment requirement
  - *Client* – visitor to monetized webpage
  - *Server* – backend process that handles billing and payments for service
  - *Client headers generator* – software module with access to client's Bitcoin wallet that issues payment transactions, generates proof-of-payment headers
  - *Client browser extension* – app installed by user that funds usage of payable web services by adding payment headers to HTTP requests made by webpage
- Browser extension
  - Forked Chrome extension Requestly which allows user to modify HTTP headers
  - Implemented listeners for events onHeadersReceived and onBeforeSendHeaders
  - Stored payments state in two global dictionaries, which map payable URL to instructional and payment headers, respectively


Webpage (Client Browser)
Server
Browser Ext.
Payment Headers Generator
1
2
3
4
5
6
7
8
1 Initial HTTP req
2 402 Payment Req'd (+ instr_headers)
3 POST request (body = instr_headers)
4 Issues BTC tx; constructs paym_hdrs = f(instr_hdrs)
5 Returns paym_hdrs
6 Adds paym_hdrs to 2nd HTTP req. to server
7 HTTP request retry (with paym_headers)
8 200 O.K. response (Access Granted)

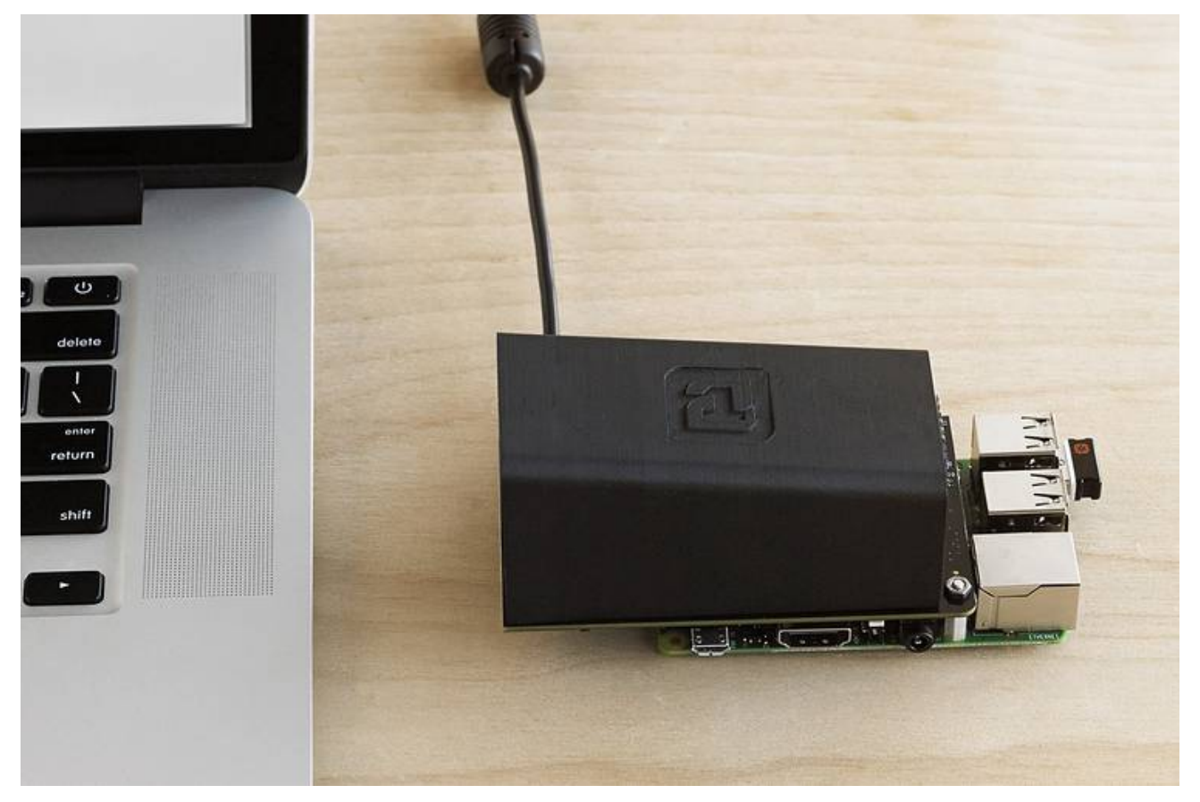
HTTP 402 Protocol Implementation



Chrome Web Request Life Cycle

Technology and Related Work

- 21 Bitcoin Computer
  - Linux machine and personal mining device that generates a small, continuous stream of Bitcoin for programmatic use
- 21 Bitcoin Library
  - Software library that allows developers to write client and server applications which make and accept payments in Bitcoin
  - Features used: APIs for HTTP 402 Protocol, wallet integration



- Related work
  - Streamium – stream live video to audience, accept payments in Bitcoin
  - Blendle – ad-free online journalism portal, with pay-per-article model
  - Brave – new web browser that replaces ads, trackers with "clean ads"

Evaluation

Extension capabilities	User responsibilities
Payments logistics Preferences-based decisions Anomaly detection and handling	Decide whether to follow link Decide how long to stay on site Wallet, generator maintenance

- Other evaluation criteria
  - *Measurability* – AJAX request loop built into webpage, but vulnerable to user tampering; production implementation will require server tracking
  - *Granularity* – channels protocol enables fine-grained payments
  - *Setup* – 21 Bitcoin Library and generator; browser extension; TLS certs
  - *Reproducibility* – payment.required Python decorator from 21 Bitcoin Library allows endpoints to support 402 Protocol with no configuration
  - *Security* – use of HTTPS on client-server and browser-extension channels protects payment headers from misuse
  - *Scalability* – possible bottlenecks include payments state and load balancing, distributed wallet design, server security
  - *Error handling* – browser cookie enables webpage to record whether user has paid initial fee; new concerns will arise when server no longer stateless

Further Work

- User preferences and machine learning
  - Browser extension can adapt its behavior based on user actions
- More complex billing schemes
  - Other metrics – pages viewed (news articles), data usage, page actions
- Economics of micropayments
  - To maximize revenue, services will set prices at each user's maximum willingness to pay
  - Ability to price discriminate requires tracking user behavior on other sites