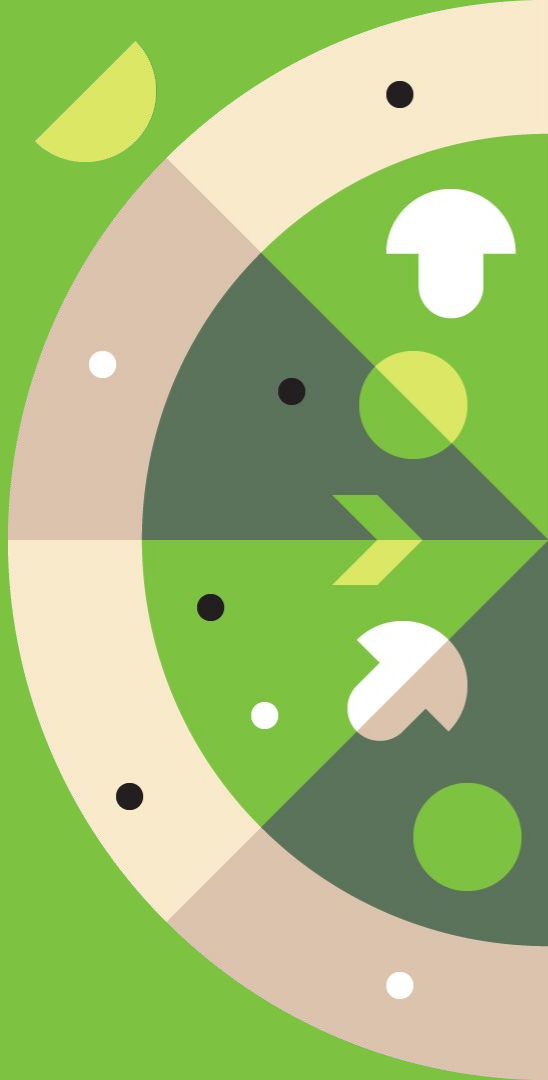
UBER eats

The Evolution of the Uber Eats Architecture

Jing Fu
Uber Eats Platform
Dec 08, 2018



UBER eats

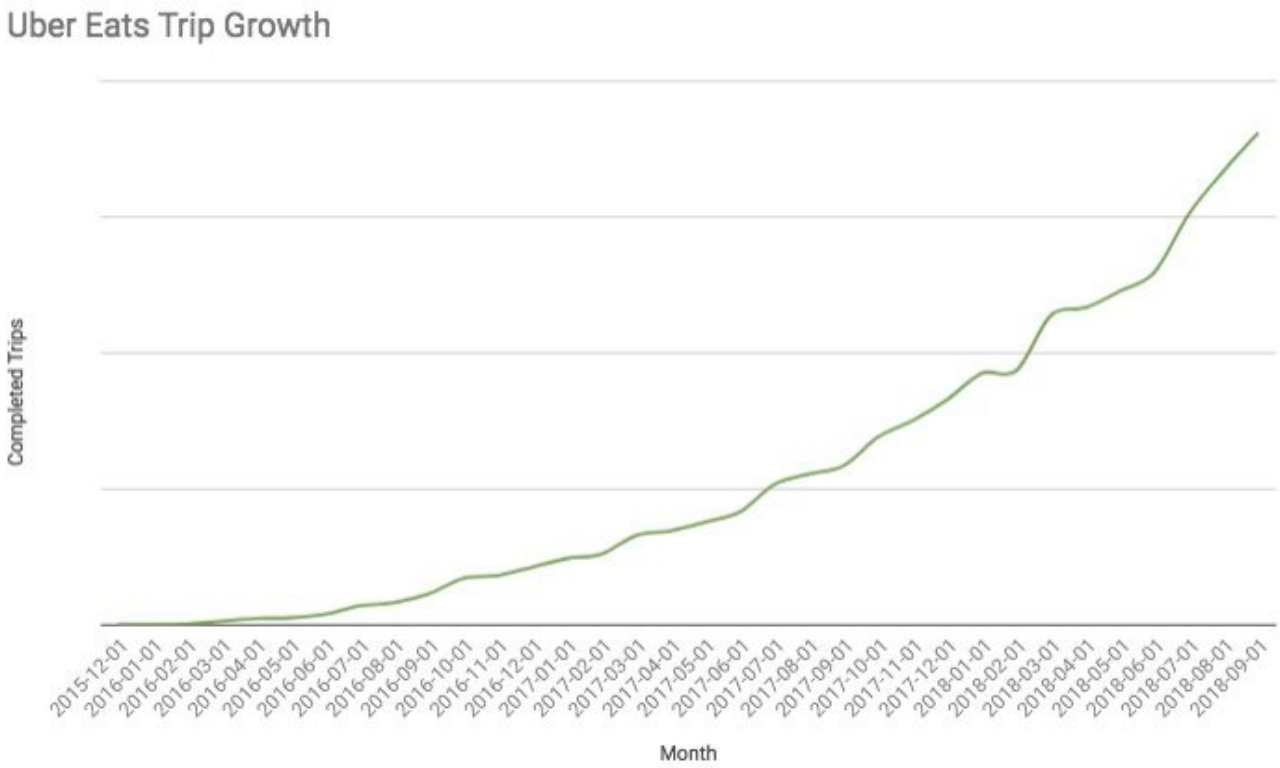
Agenda

1. **Business Overview & Challenges**
2. Architecture Overview & Evolution
3. Leveraging Ridesharing Platforms
4. Tackling i18n Challenges
5. Q&A

Our Scale

> 350
Cities

> $6B
Gross Bookings



Then

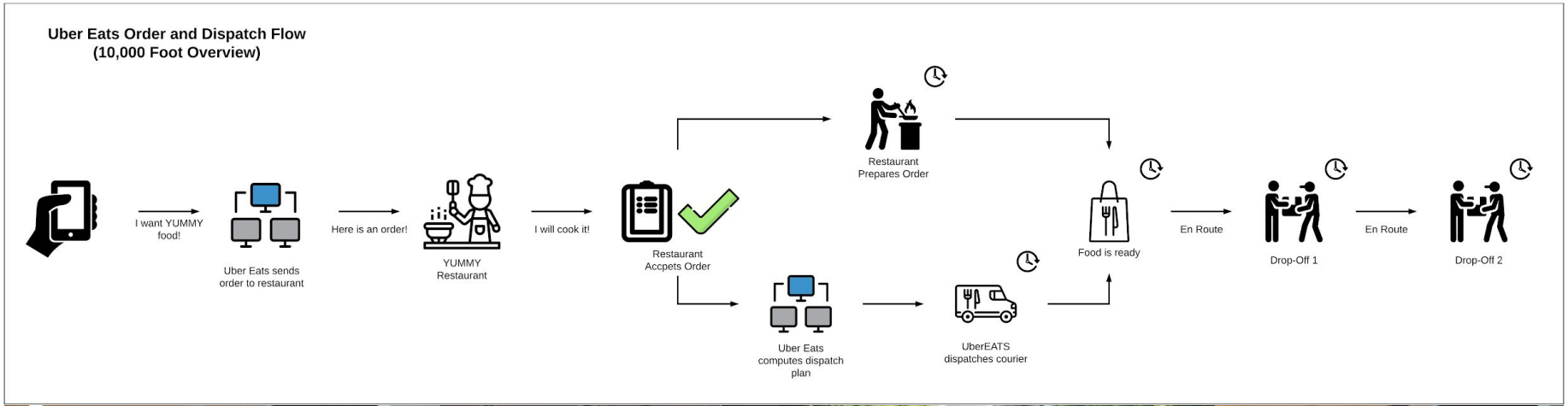

uberFRESH


UBER RUSH

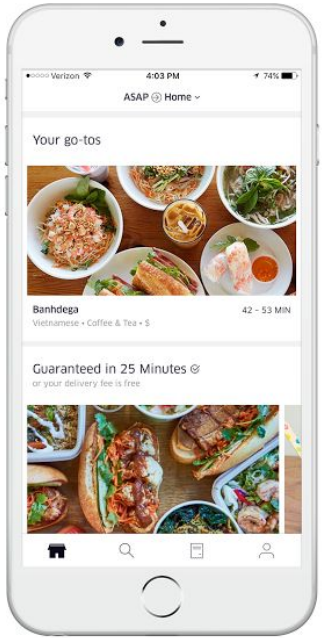
Now


9:41
Uber
Uber
Uber Eats
Uber Eats
Weather

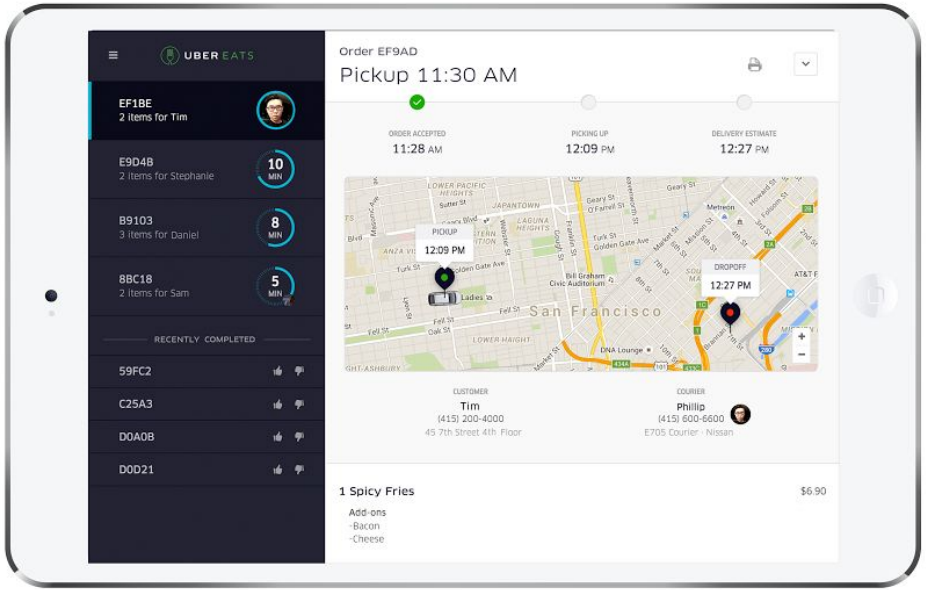
How does Uber Eats work today?



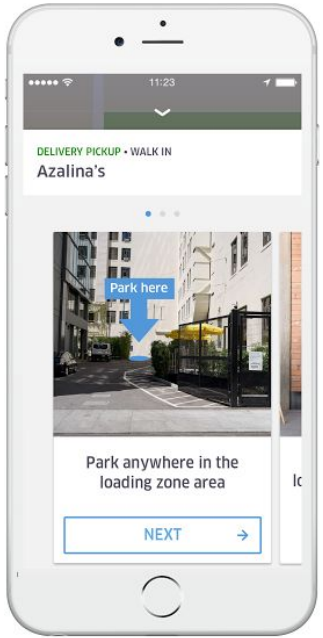
On-demand Uber Eats



UberEATS App



UberEATS restaurant app



Uber partner app: delivery pickup

Challenges

- Marketplace complexity vs resource constraint
- Internationalization (i18n)
  - Operation (reliability)
  - Performance (app, network)
  - Extensibility (dev)

UBER eats

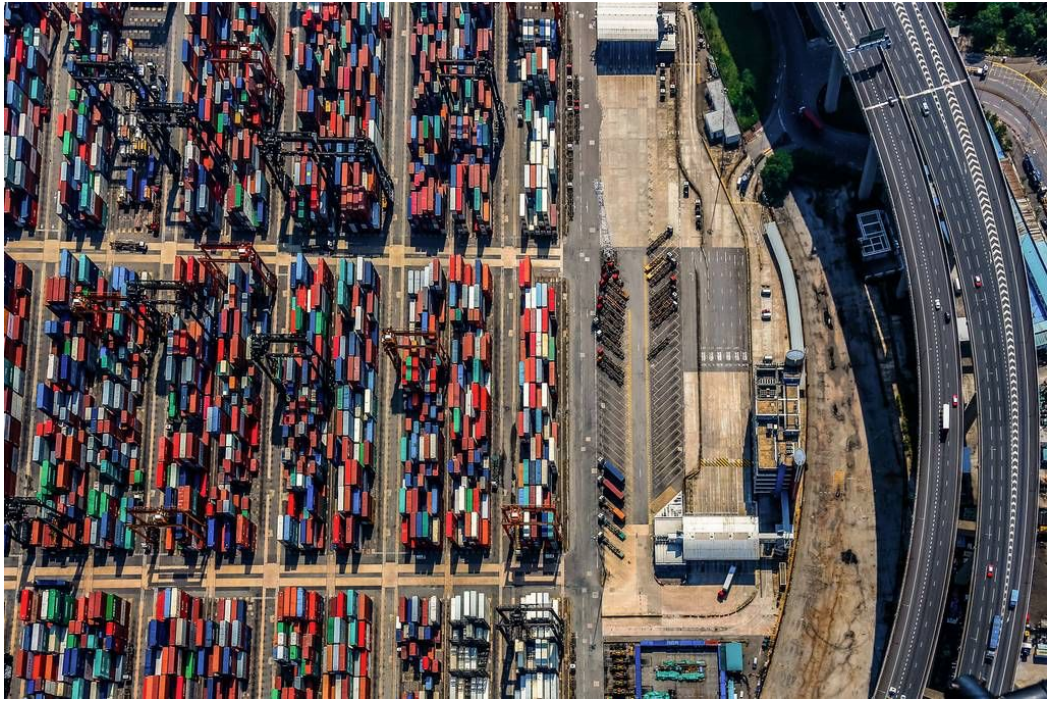
UBER eats

Agenda

1. Business Overview & Challenges
2. **Architecture Overview & Evolution**
3. Leveraging Ridesharing Platforms
4. Tackling i18n Challenges
5. Q&A

Background

- Uber:
  - Monolith (from 2009) => lots of microservices
  - Py/JS => Golang/Java
  - MySQL => Cassandra
- Uber Eats (2015):
  - Microservices* + Golang + Cassandra* at the onset



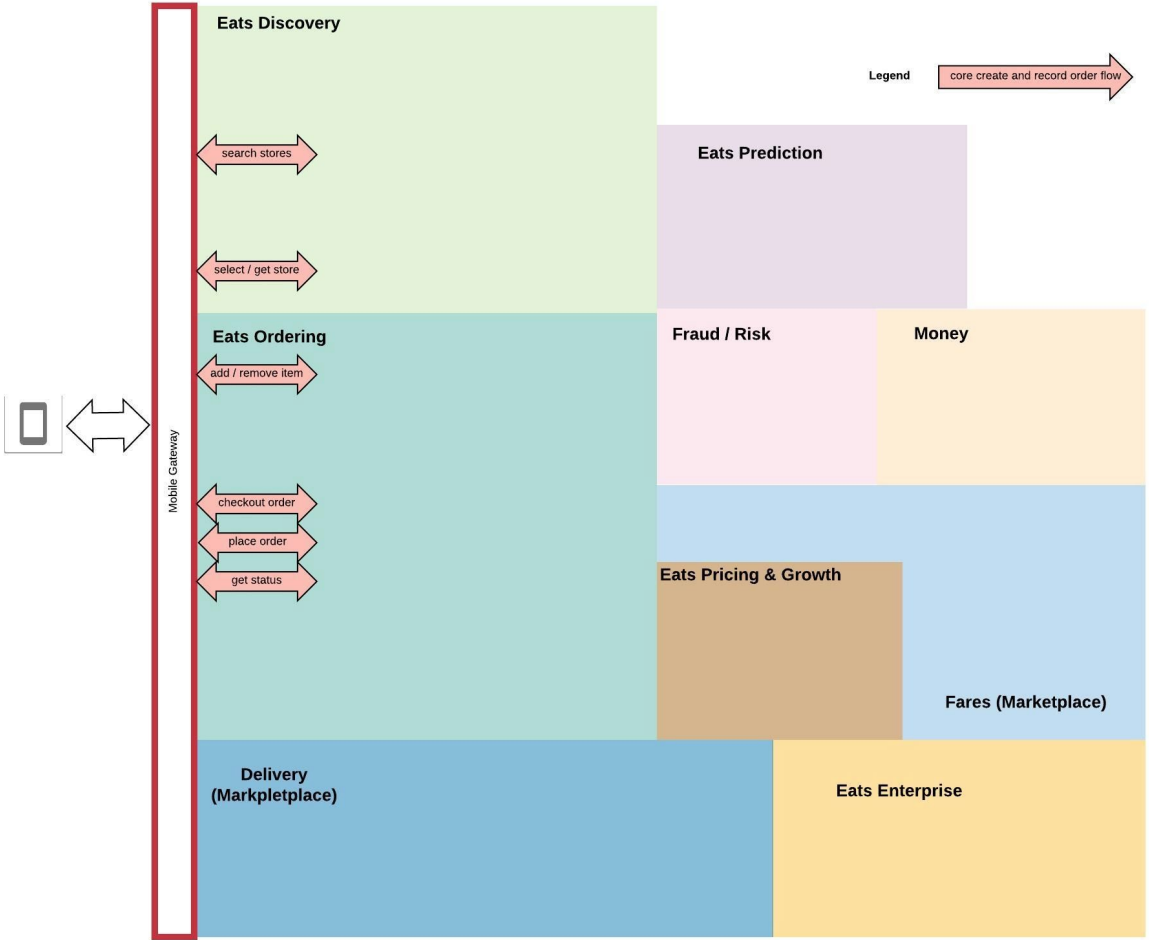
Pain points

- 0 => 1 => N cities
- Microservices (70+)
  - Long e2e chain
  - Messy dep graph*
  - Hairy migrations*
  - Any service can bring down the biz*

Identify core flows

- Revisit product phases
- => *Core Flows*
- => Tier 1 services
- => Extra rigor for T1
- => Tech convergence*
- => Fewer services

Simplified architecture (flows)



UBER eats

Agenda

1. Business Overview & Challenges
2. Architecture Overview & Evolution
3. **Leveraging Ridesharing Platforms**
4. Tackling i18n Challenges
5. Q&A

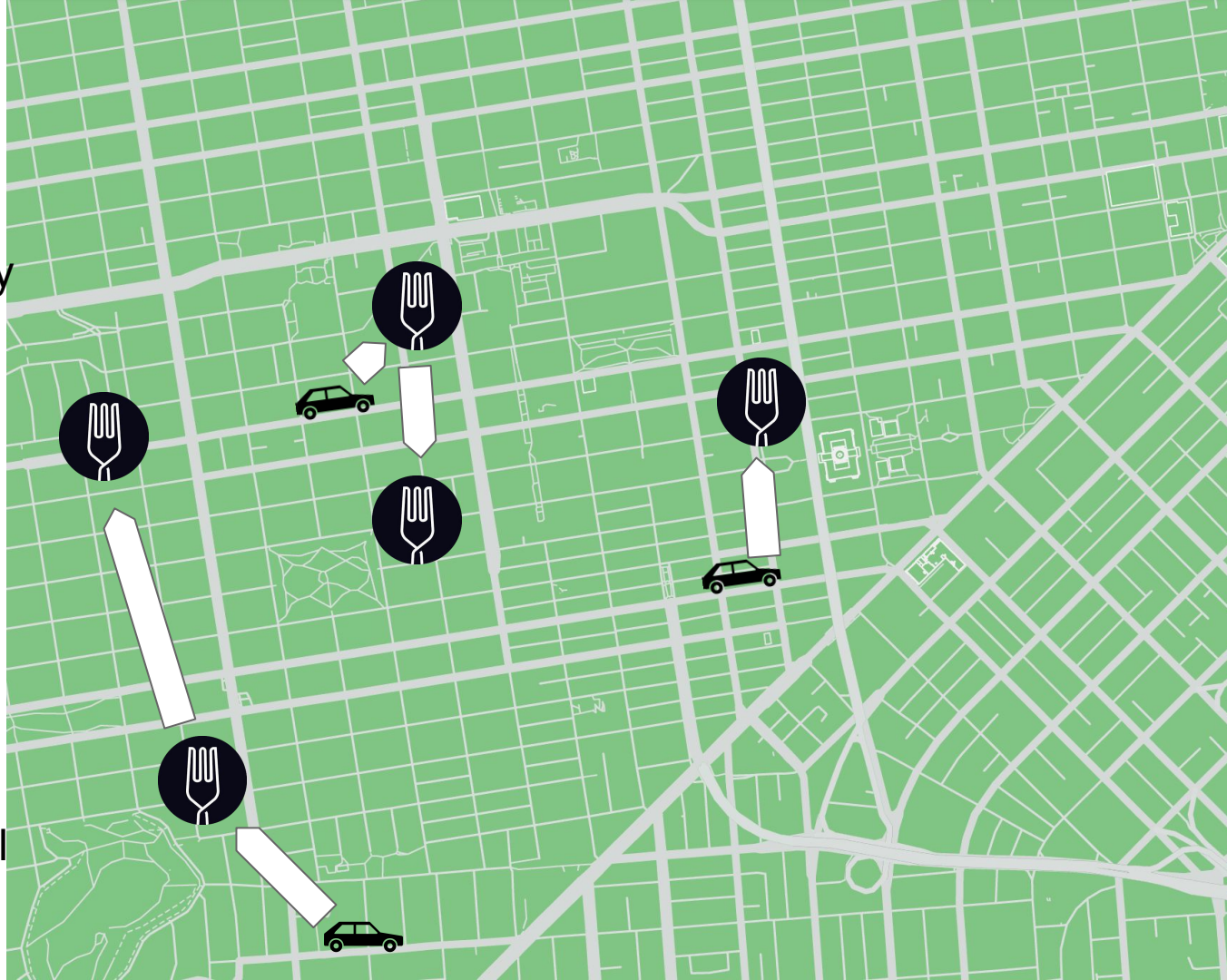
Batching: before

- Greedy matching
- 1 order 1 delivery
- “Nearest” wins

UBER eats

Batching: after

- Clustering
- >1 orders per delivery
  - Efficiency ↑
  - Win win
- Constraints
  - Eater ETA
  - Route overlap
- System
  - Scan local/global



UBER eats

Case study: disaster recovery

- Active-active (2 DC)
- 3 levels of mitigation
  - DNS (L1)
  - Data center (L2)
  - Service (L3)
- Tiered operation power
  - DNS: SRE
  - DC: Ring0
  - Service: owners
- Recent example

Case study: storage

- MySQL => C*
- Gocql can be too much
- 2 different kinds of entities
  - State machine vs SOT
- Write-optimal: K-V + dual-write
  - State machine, e.g. order/cart
- Read-optimal: K-V + Redis
  - SOT entities, e.g. menu/store

Many more examples..

- Machine Learning Platform (eng blog)
- Experimentation Platform (eng blog)
- Forecasting Platform (eng blog)
- Dynamic Configuration Platform
- Translation Platform
- Deployment Platform
- ...

UBER eats

Agenda

1. Overview & Challenges
2. Architecture Overview & Evolution
3. Leveraging Ridesharing Platforms
4. **Tackling i18n Challenges**
5. Q&A

Challenge #1: Operation at global scale

- Things go wrong all the time
  - Nature (weather)
  - Ops (promo eyeball fanout)
  - Eng (dev)
- Can lead to cascading failure
- Reliability key to customer trust

Solution: Graceful degradation

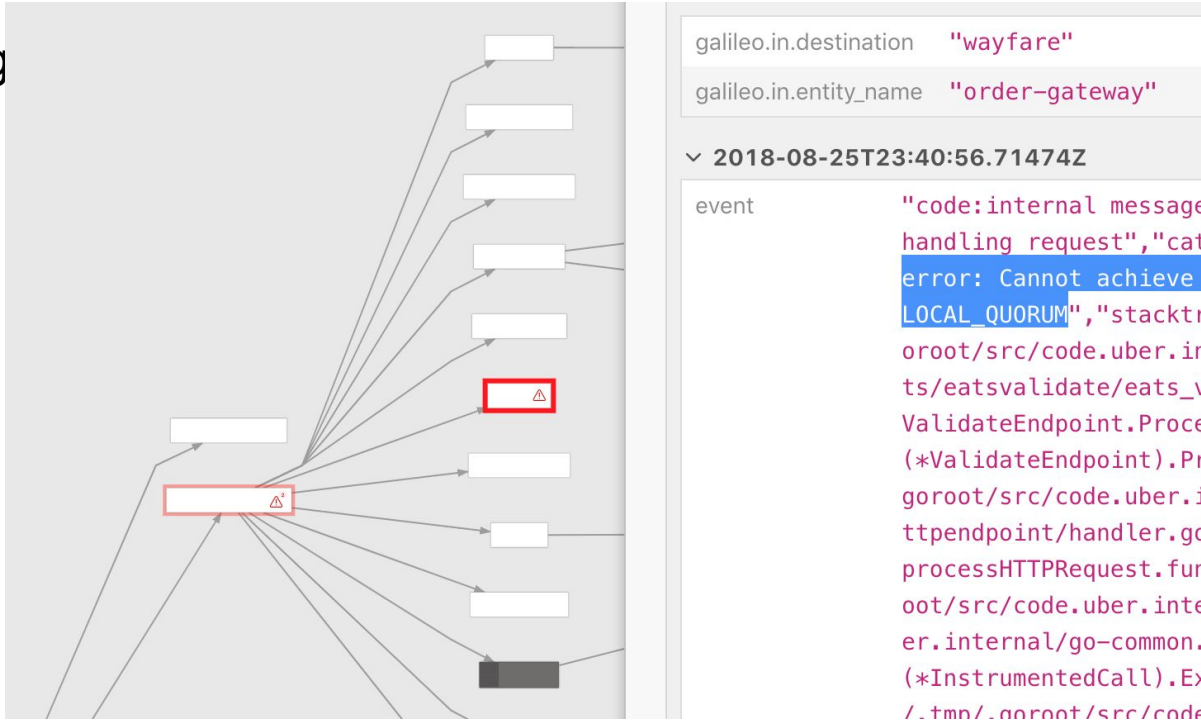
- Circuit breaking
  - Client rejects outgoing req highly likely to fail
- Load shedder
  - Server rejects incoming req when exceeding X delay
- City & user rate limiting
  - City counter via centralized city routing in RTAPI

Solution: External probing

- Simulate core flow globally 24x7
- Alert when M concurrent failures in N minutes
- Highly effective (time, SNR)
- => Auto rollbacks (deploy/config), or manual intervention

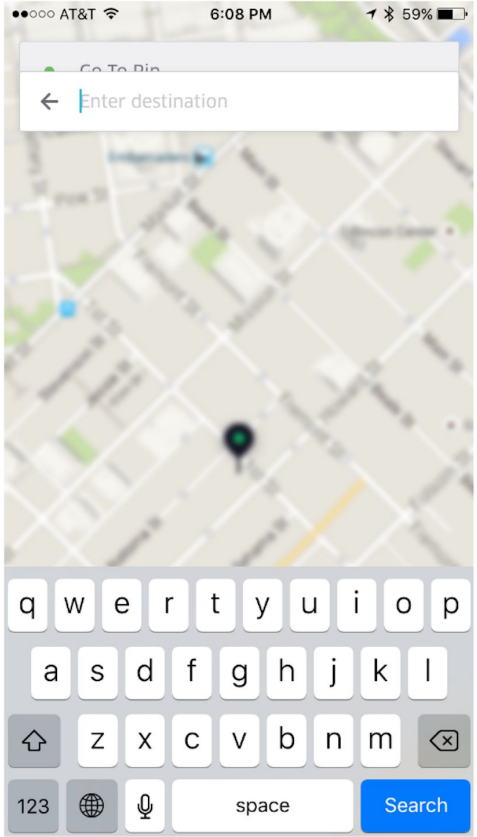
Solution: Instant root causing

- Integrated w/ monitoring
- UI w/ problematic stack & error message
- Via tracing injection throughout the stack
- => fast mitigation

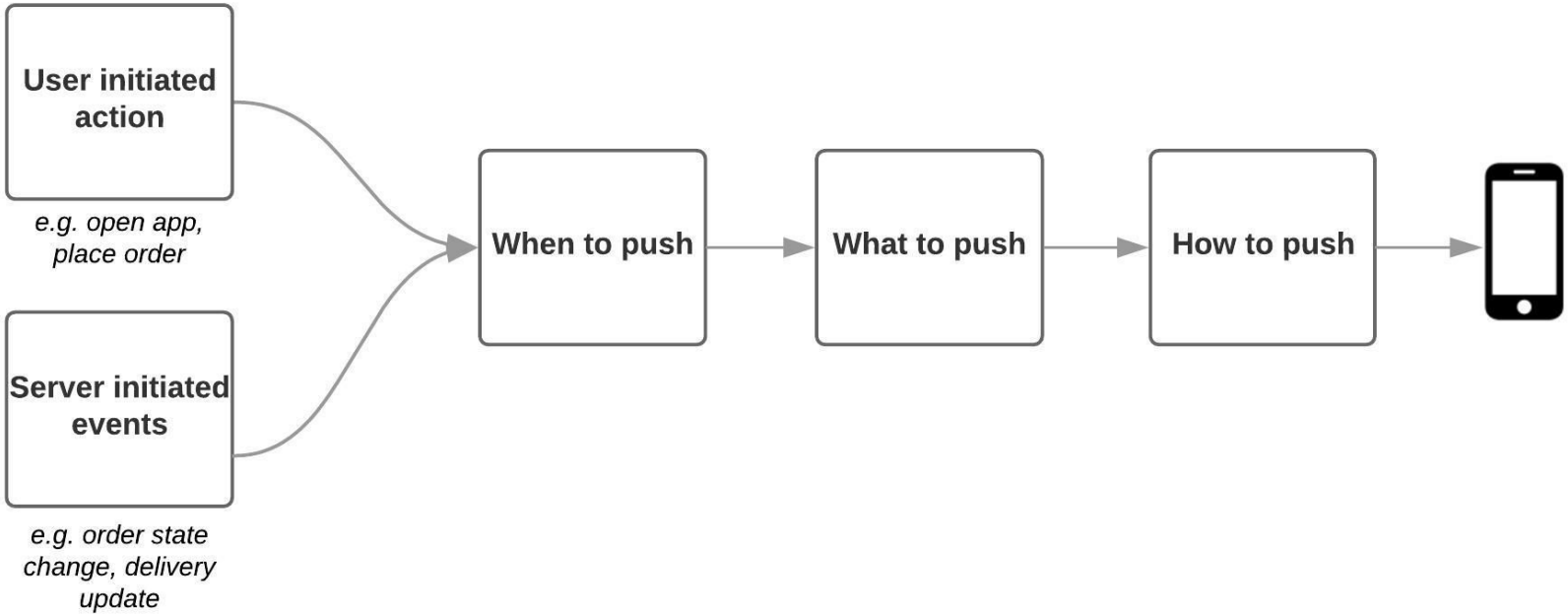


Challenge #2: Performance around the globe

- Slow & unreliable networks (512Kbps=broadband in India)
- App assumes developed markets
  - Polling for updates
  - Parallel net calls
  - Large payload
- => Subpar experience

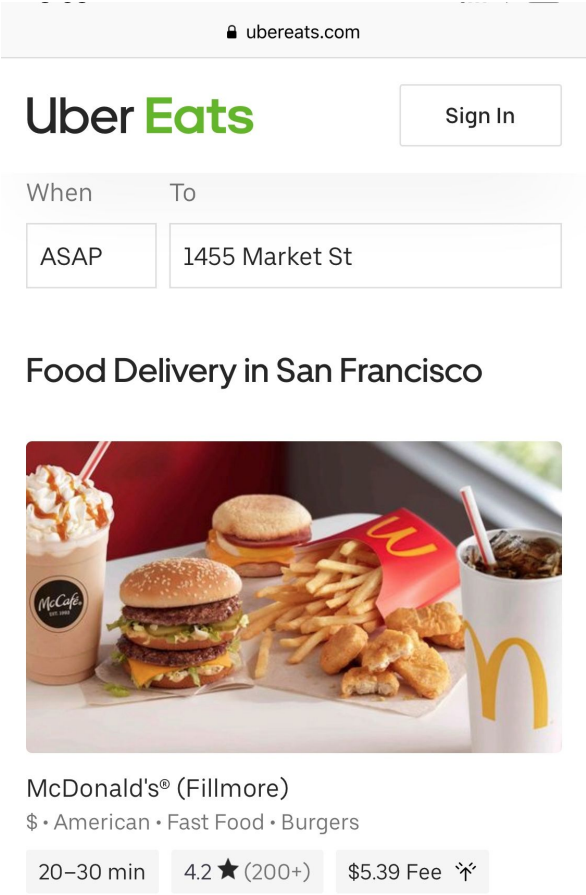


Solution: Push Framework



Solution: Many more..

- Pagination (fewer stores)
- Lazy loading
- Web Eats (UberLite)
- Cash
- ...



UBER eats

Agenda

1. Overview & Challenges
2. Architecture Overview & Evolution
3. Leveraging Ridesharing Platforms
4. Tackling i18n Challenges
5. Q&A

Uber