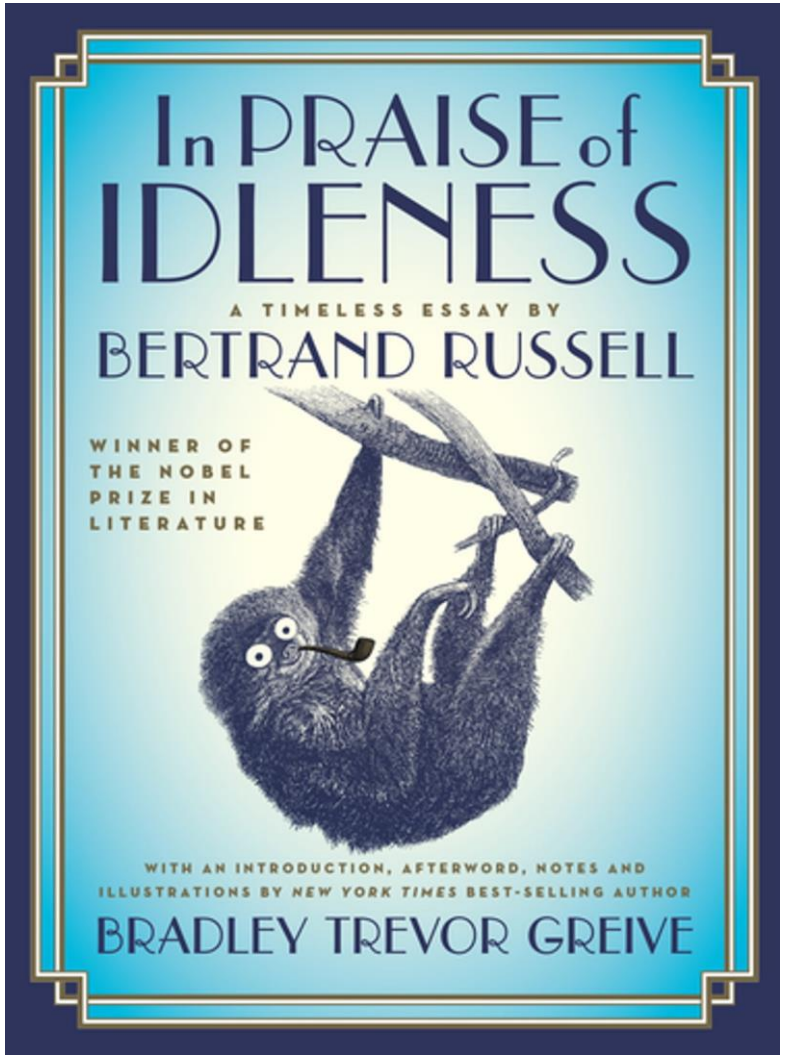
LazyTAP:
In Praise of Laziness

Mohammad M. Ahmadpanah Daniel Hedin Andrei Sabelfeld

October 21, 2022



- *"... the road to happiness and prosperity lies in an organized diminution of work."*

- *"We should increase leisure and produce* ***only*** *what makes for better lives."*

Trigger-Action Platform (TAP)

- Connecting otherwise unconnected services and devices
- Trigger comes, the app performs an action
- Popular TAPs: IFTTT, Zapier, Power Automate

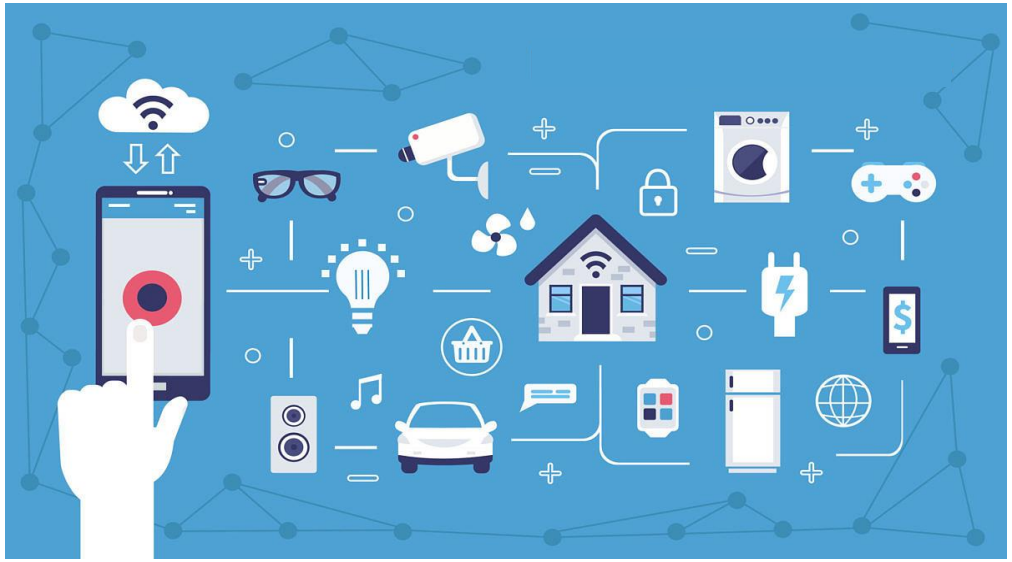
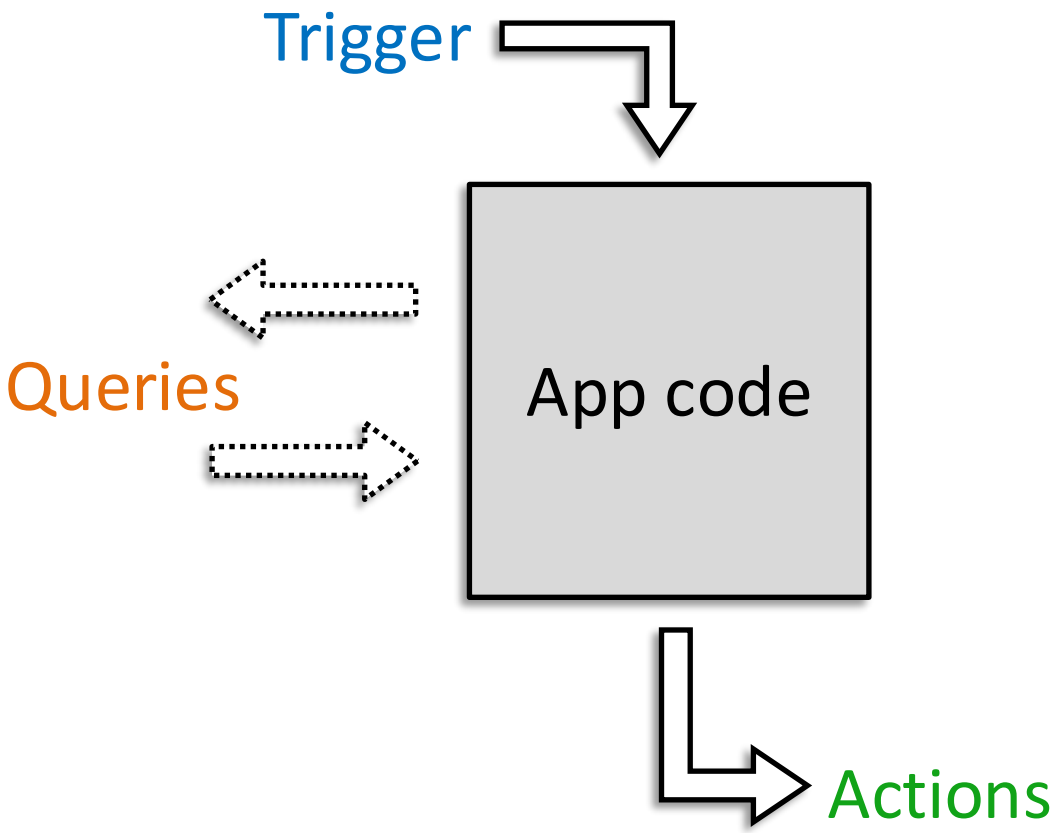


Image: © Irina Strelnikova / Adobe Stock

IFTTT apps



IFTTT examples (1)

Trigger

```
// app filtercode
if (Email.sendIftttAnEmail.From == "supervisor") {
    Slack.postToChannel.setMessage(
        Email.sendIftttAnEmail.Subject +
        Email.sendIftttAnEmail.Body +
        Email.sendIftttAnEmail.AttachmentUrl);
} else {
    Slack.postToChannel.skip();
}
```

Action

IFTTT examples (2)

Trigger

```
if (GoogleCalendar.historyOfEventFromSearchStarts.length == 0) {
    Slack.postToChannel.skip();
}
else {
    Slack.postToChannel.setMessage(
        Email.sendIftttAnEmail.Subject +
        Email.sendIftttAnEmail.Body +
        Email.sendIftttAnEmail.From +
        GoogleCalendar.historyOfEventFromSearchStarts[0].Title);
}
```

("today", Email.From)

Query

historyOfEventFromSearchStarts

Action

IFTTT examples (2)

```
if (GoogleCalendar.historyOfEventFromSearchStarts.length == 0) {
    Slack.postToChannel.setMessage("no important meeting!");
}
else {
    Slack.postToChannel.setTitle(
        GoogleCalendar.historyOfEventFromSearchStarts[0].Title);
    Slack.postToChannel.setMessage("next meeting starts at” +
        GoogleCalendar.historyOfEventFromSearchStarts[0].Starts);
}
```

("important","today")

historyOfEventFromSearchStarts

IFTTT examples (2)

```
if (GoogleCalendar.historyOfEventFromSearchStarts.length == 0) {
    Slack.postToChannel.setMessage("no important meeting!");
}
else {
    Slack.postToChannel.setTitle(
        GoogleCalendar.historyOfEventFromSearchStarts[0].Title);
    Slack.postToChannel.setMessage("next meeting starts at” +
        GoogleCalendar.historyOfEventFromSearchStarts[0].Starts);
}
```

("important","today")

historyOfEventFromSearchStarts

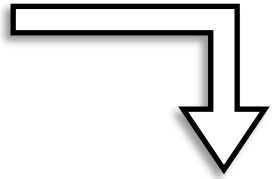
IFTTT examples (2)

```
let location = GoogleCalendar.historyOfEventFromSearchStarts[0].Where;
if (location != 'office') {
    Slack.postToChannel.setMessage("First meeting is not in office!");
} else {
    Slack.postToChannel.setTitle(
        GoogleCalendar.historyOfEventFromSearchStarts[0].Title);
    Slack.postToChannel.setMessage("First office meeting starts at" +
        GoogleCalendar.historyOfEventFromSearchStarts[0].Starts);
}
```

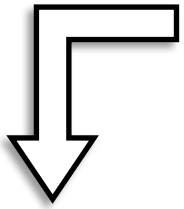
("important","today")

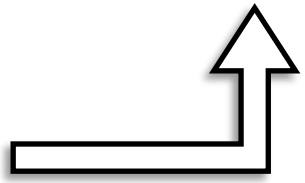
historyOfEventFromSearchStarts

IFTTT examples (3)

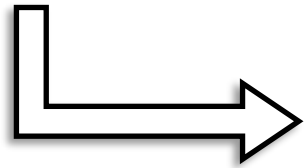


```
let index = Math.floor((Math.random() * Trakt.recommendedMovies.length))
IfNotifications.sendRichNotification.setMessage(
        “This is the movie you’d like to watch this weekend: ” +
        Trakt.recommendedMovies[index].MovieTitle)
```

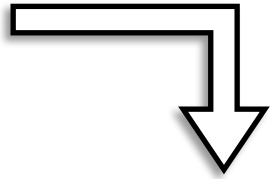




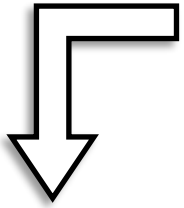
recommendedMovies

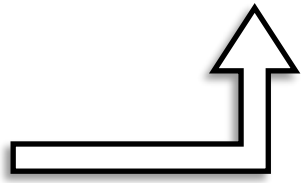


IFTTT examples (3)

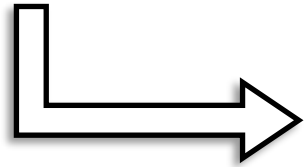


```
var index = Math.floor((Math.random() * Trakt.recommendedMovies.length))
IfNotifications.sendRichNotification.setMessage(
        “This is the movie you’d like to watch: ” +
        Trakt.recommendedMovies[index].MovieTitle)
```

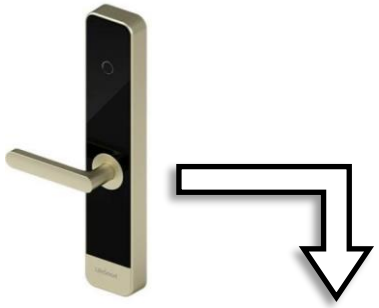




recommendedMovies



IFTTT* examples (4)



```
let events = GoogleCalendar.historyOfCalendarEventBeginnings("work", "01:00");
if (events.length != 0) {
    let parkingLocation = Yelp.searchBusiness(events[0].Where, "parking");
    if (parkingLocation.length != 0) {
        AndroidDevice.startNavigation.setQuery(parkingLocation[0].BusinessAddress);
    }
} else {
    AndroidDevice.startNavigation.skip();
}
```

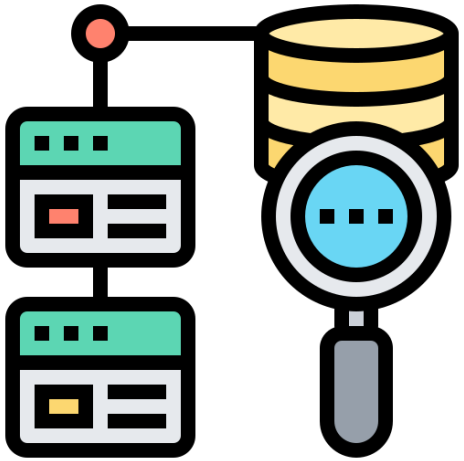
“work”,
“01:00”

historyOfCalendarEventBeginnings

(calendarEvent[0].Where,
“parking”)

BusinessAddress

Queries



- Additional data *source* of apps
- In IFTTT:
  - Can be multiple for app
  - (In)dependent on *trigger* data
    - query chain is not possible
  - First fetch *all* the queries, then execute the app code
- A general TAP:
  - Queries dependent on another data source (trigger/query)
    - Might support query chain

Data minimization

- Only necessary data should be collected for the specific purpose the user consented

- IFTTT's approach: Attribute-level overprivilege
  - Fetching *all* the attributes from trigger/query services *no matter* what the app needs!
  - Email attributes: {`AttachmentTemporaryUrl`, `From`, `Body`, `BodyHTML`, `Subject`, `AttachmentUrl`, `ReceivedAt`}

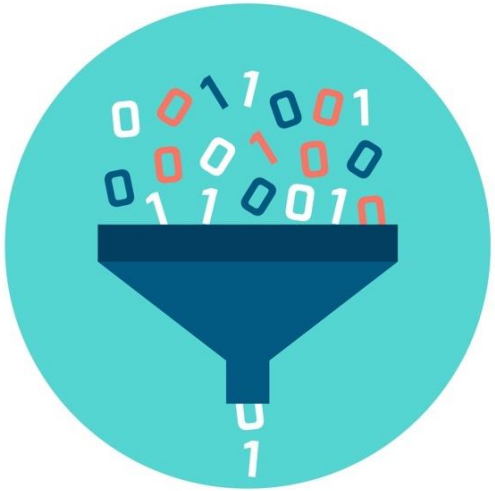
- Goal: Transmit the *minimal* set of attributes *required* for an app per execution

minTAP [USENIX'22]

- Threat model: untrustworthy TAP
  - May deviate from the protocols to steal *more* user data than needed
- Minimizing attributes of *trigger* data
- Modes: Static and Dynamic
  - Static: Aggregates all attributes present in the app code
  - Dynamic: Runs the app code on the trigger service to identify attributes accessed by the code in the execution
  - Static minTAP can support apps with queries (dynamic cannot)
  - Dynamic minTAP can suppress trigger if action will be skipped

minTAP (cont.)

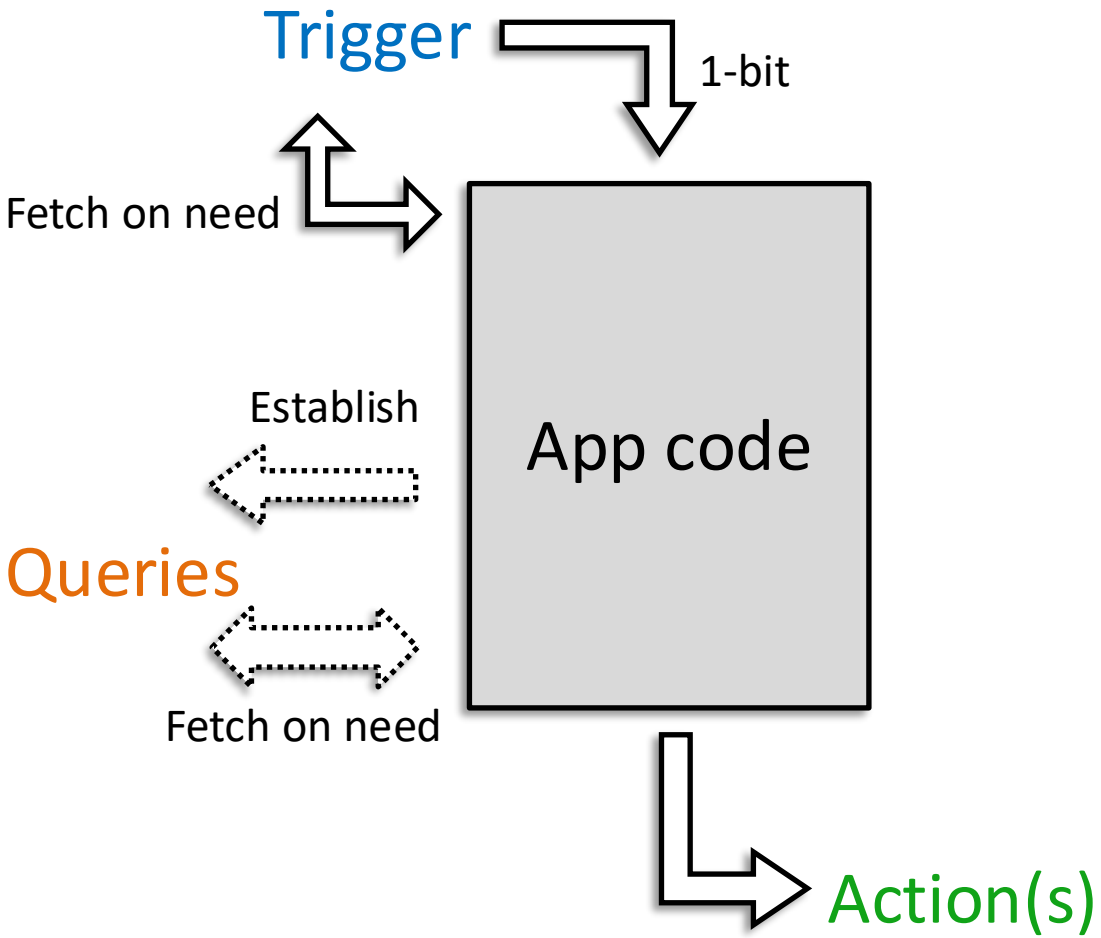
- Notions of data independence
  - Value changes of a subset of attributes has **no** effect on the output
  - Regular: Output never depends on what values an attribute takes
  - Run: Output depends on particular values of certain attributes
- Static minTAP brings regular independence while dynamic minTAP brings run independence



LazyTAP: Data minimization by design

- Threat model: honest-but-curious TAP
  - **Do not deviate from the protocol but will attempt to learn all possible information from legitimately received events**
- Fetch-on-need
  - In execution, an attribute is fetched from service only if it is needed
- Data source unification
  - Trigger and queries are treated *similarly*
  - One-bit event from trigger is sent, the required attributes are fetched by the TAP

LazyTAP


Trigger
1-bit
Fetch on need
App code
Establish
Queries
Fetch on need
Action(s)

LazyTAP (cont.)

- Which parts are changing?
  - TAP: Executes app when 1-bit trigger received, then fetching data *on-the-fly*
  - Trigger/Query services: A shim layer should be added; can be done using trusted mediators
- App code remains as is
  - A lazy version of the app is generated, using the same APIs

LazyTAP: Implementation

- Elements:
  - RemoteTrigger: fetch-on-need trigger data
  - LazyQuery: deferred computation of query, providing fetch-on-need query data
  - LazyProjection: delayed projection *via thunking*
- Fetch-on-need by thunking
  - Using proxies or accessor properties, queries as classes
- Queries can be dependent on *any* data source (trigger or query)

LazyTAP: Demo

- Example 2 and 3 (in JS)

Evaluation

App id	Distinctive pattern	Total attributes (IFTTT)	Static minTAP	LazyTAP
003ecs	Trigger-dep. query, skip on query attribute	8 + (7 * GoogleCalendarLength)	5	2 | 5
004scpn	Query chain trigger-indep., skip on query	4 + (7 * GoogleCalendarLength) + (7 * YelpLength)	4	1 | 3 | 4
jUy5if7H	Nondetermin. indep. query, skip on time	3 + (7 * TraktLength)	TraktLength+1	1 | 2 | 3 | 4
MRm9VBxG	Nondetermin. indep. queries, trigger in action default values, no skip	4 + (7 * TraktLength) + (7 * YelpLength)	3 + TraktLength + YelpLength	4

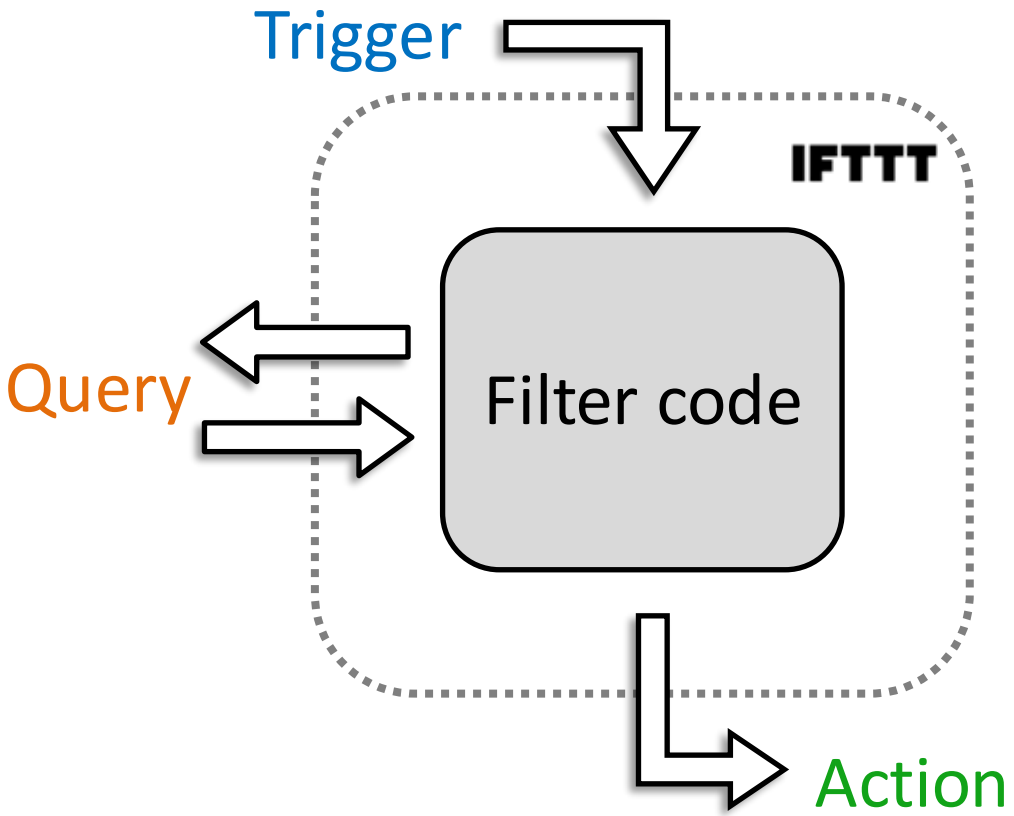
LazyTAP vs. minTAP

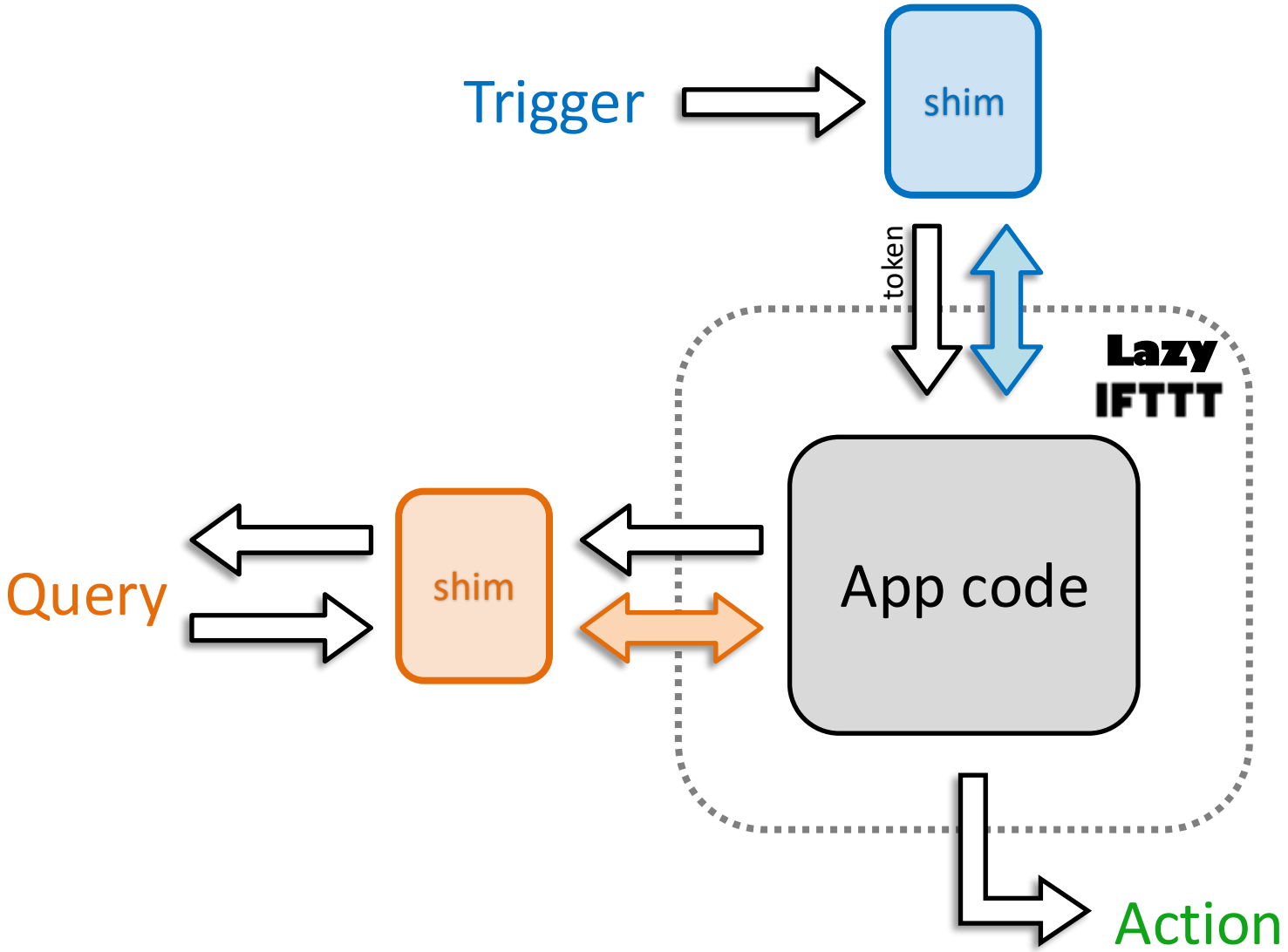
Approach	Trust **model** of TAP	Apps without queries	Apps with queries (incl. chains)
Static minTAP	Untrustworthy	Regular independence	Regular independence
Dynamic minTAP	Untrustworthy	Run independence + No attribute if skip/timeout occurs	N/A
LazyTAP	Honest but curious	Run independence + Real-time behavior	Run independence + Real-time behavior (incl. arrays, nondeterminism, etc.)

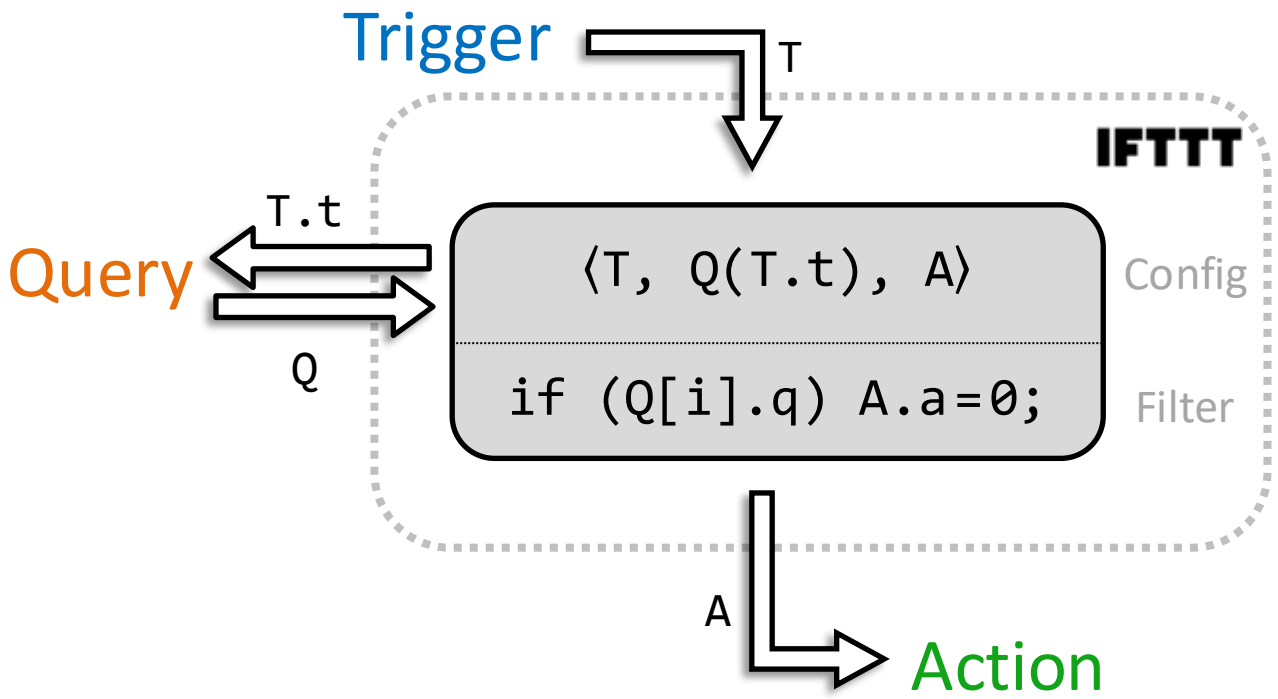
LazyTAP takeaways

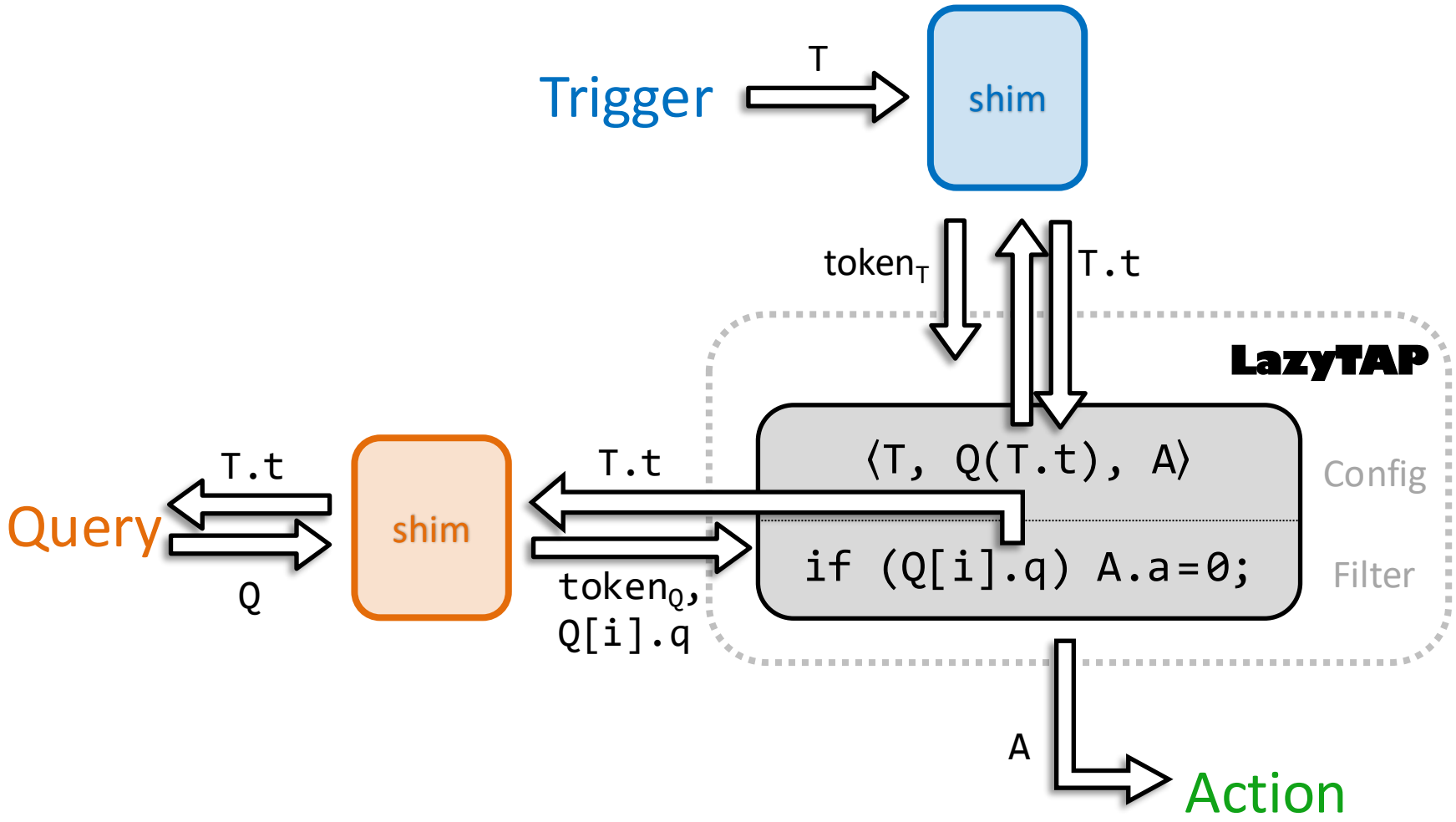
Be *lazy*, be *minimized!*

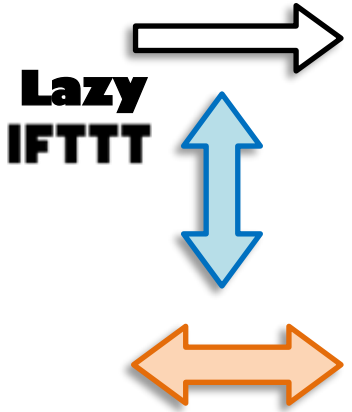
- Data minimizatio nt queries
- Run independen cl. arrays and nondeterminism)
- Changes:
  - Trigger/query se sted caches
    - No code exeucti
  - TAP: supporting fetch-on-need and generating lazy version of the app


Trigger
IFTTT
Query
Filter code
Action


Trigger
shim
token
Lazy
IFTTT
Query
shim
App code
Action


Trigger
T
IFTTT
T.t
Query
⟨T, Q(T.t), A⟩
Config
Q
if (Q[i].q) A.a=0;
Filter
A
Action





App config

Filter code