FLUID: Flexible User Interface Distribution for Ubiquitous Multi-device Interaction

Sangeun Oh[*], Ahyeon Kim[*], Sunjae Lee[*], Kilho Lee[*],
Dae R. Jeong[*], Steven Y. Ko[†], and Insik Shin[*]

[*] KAIST

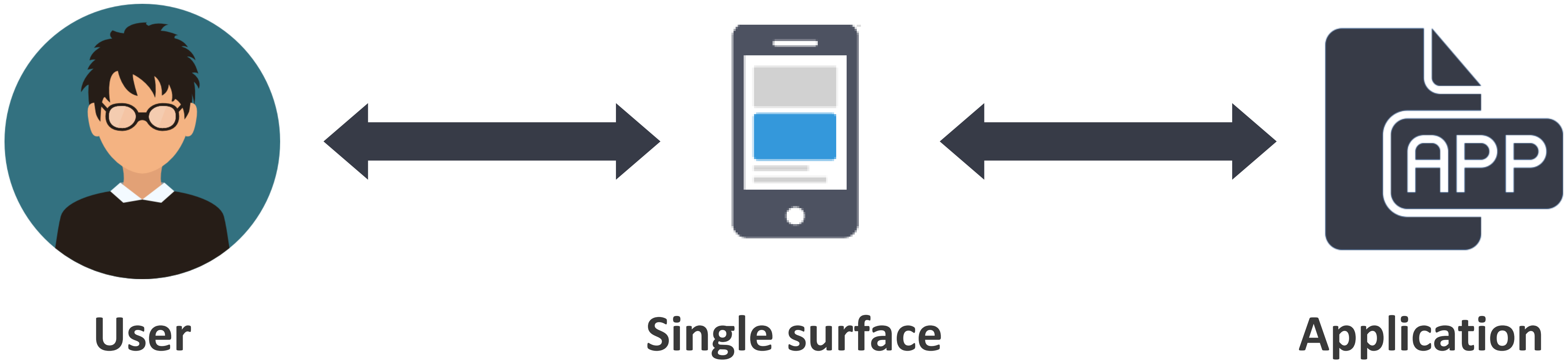
[†] University at Buffalo
The State University of New York

Various surfaces become pervasive!

- Smart devices have various surfaces with different shapes & sizes
  - From smartwatch to smart TV
  - Foldable screen (Samsung Galaxy Fold) / dual screen (LG V50)

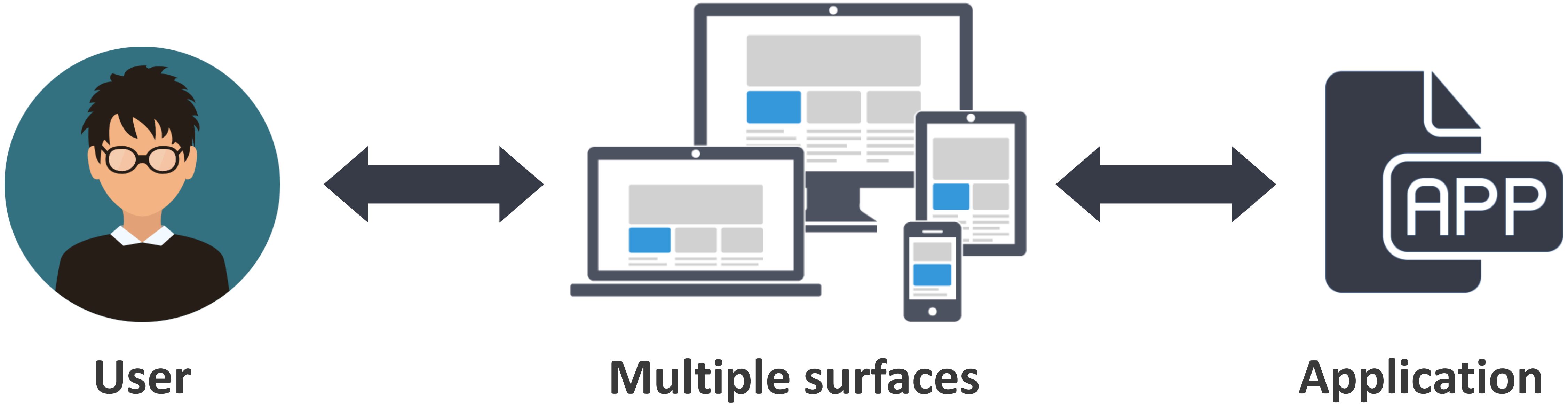
Potential for multi-surface interaction

- The trend can change how users interact with applications
  - Using only single surface → **Using multiple surfaces concurrently**



Potential for multi-surface interaction

- The trend can change how users interact with applications
  - Using only single surface ➔ **Using multiple surfaces concurrently**



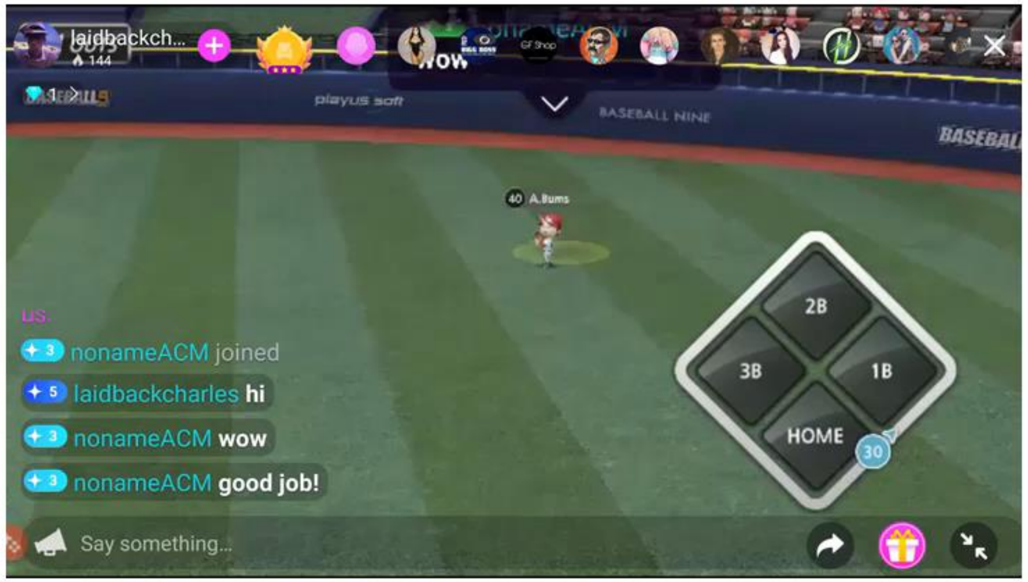
Use case: live streaming

Twitch

LiveMe

Instagram

VK live



3 nonameACM joined

5 laidbackcharles hi

3 nonameACM wow

3 nonameACM good job!

Existing solutions

- **Customized apps**
  - Extra engineering efforts
  - **Low applicability**
- **Screen mirroring**
  - **Low flexibility**
    - Supports only full screen
  - **Low responsiveness** for high resolutions
- **App migration**
  - **Low flexibility**
    - Supports only full screen
    - Cannot support concurrent usage

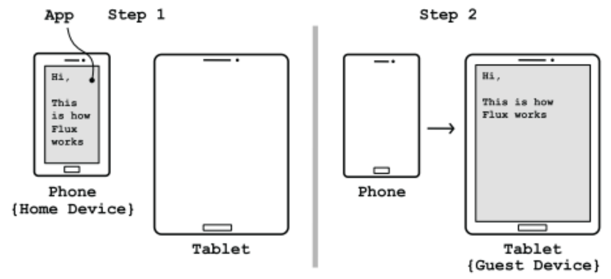
Google docs

Netflix

Smartwatch apps

Vysor

Chromecast



Flux [EuroSys'15]

Research goal

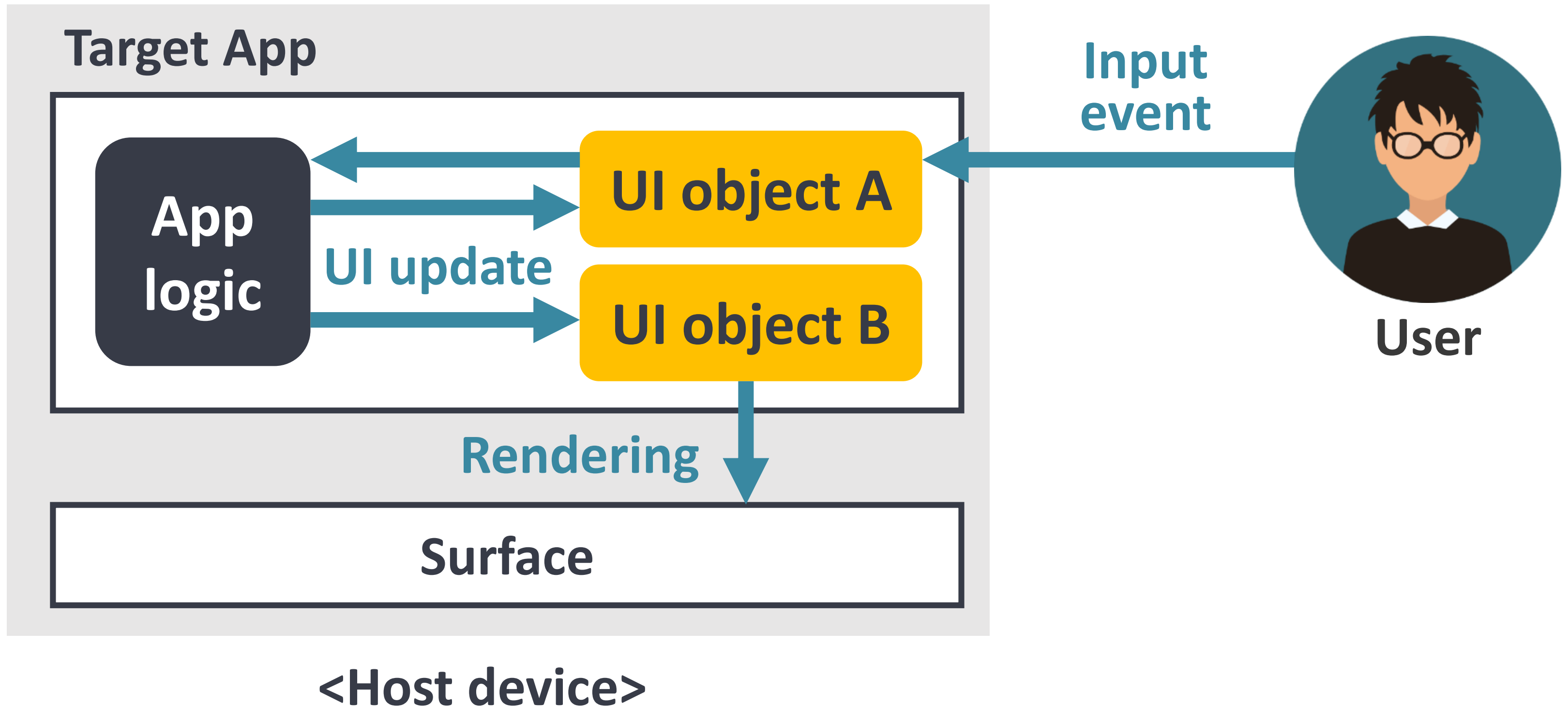
- Design a new mobile platform that supports multi-surface interaction by distributing UI objects to different devices
  - in a **flexible**, **transparent** and **responsive** manner

FLUID

(FLexible UI Distribution)

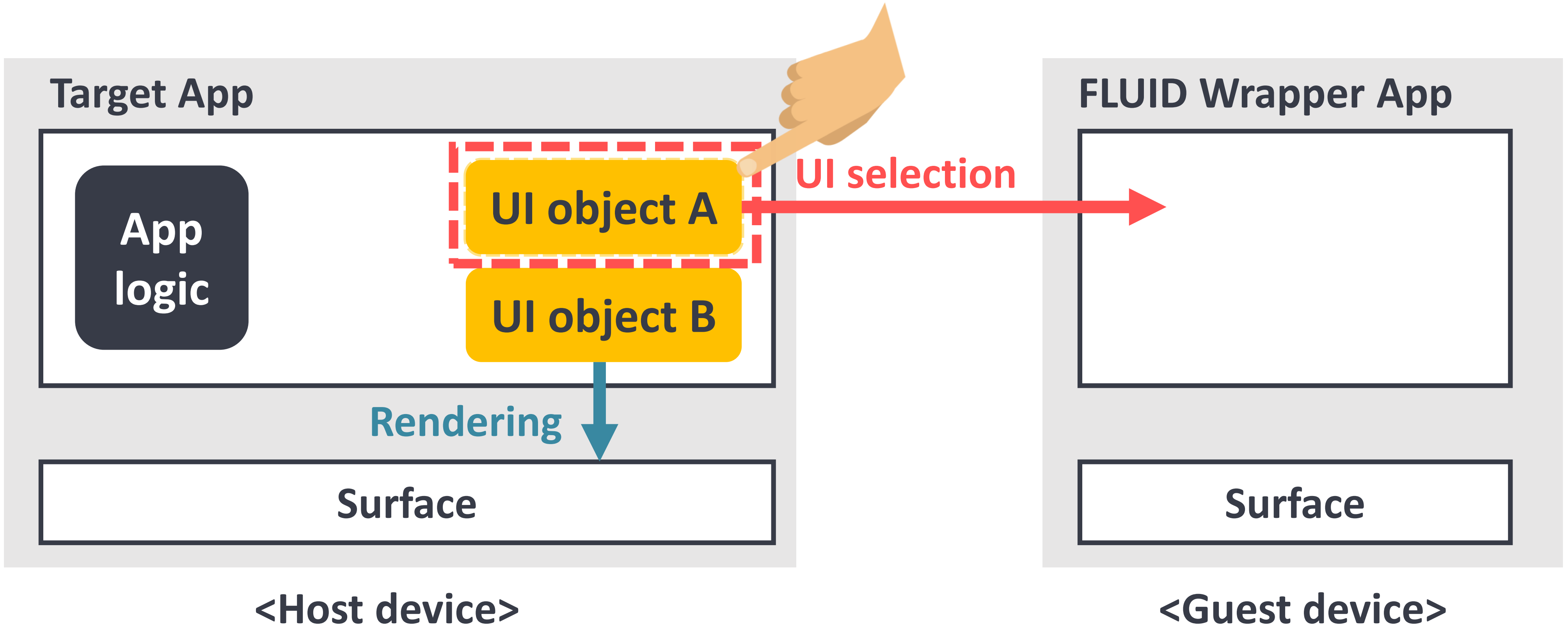
FLUID overview

- **Key idea**: separation between app logic & UI parts



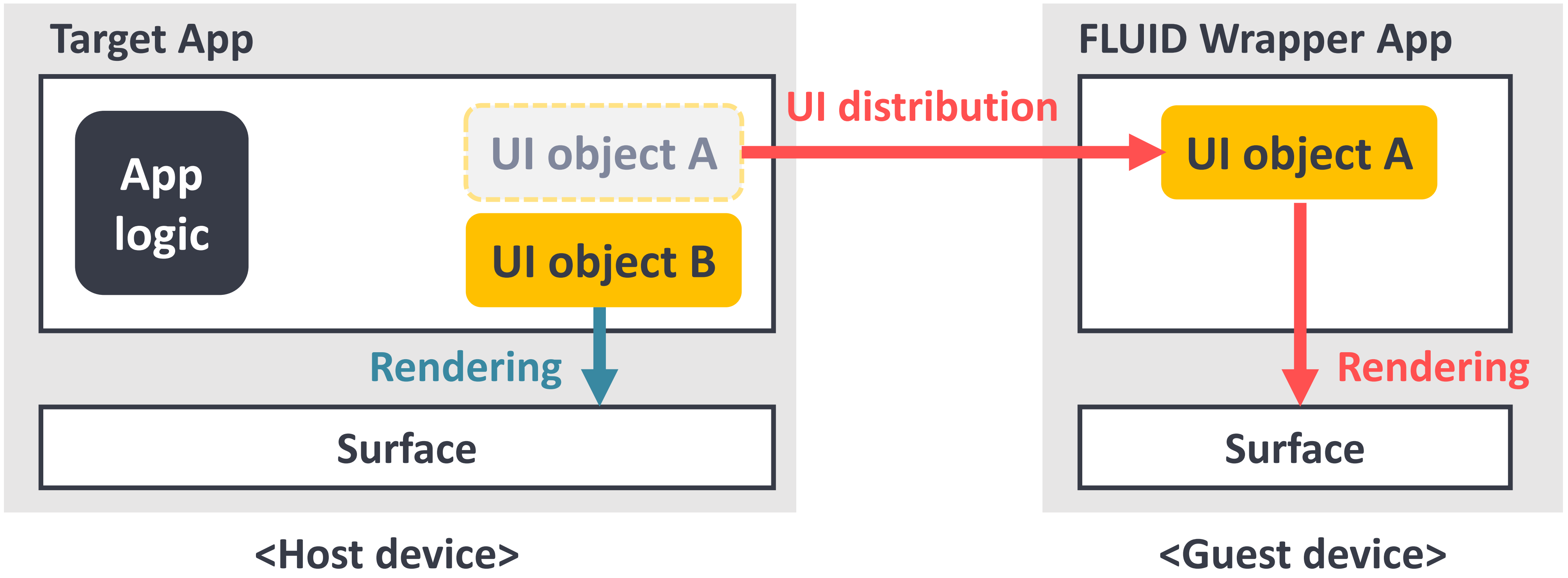
FLUID overview

- **Key idea**: separation between app logic & UI parts
  1) Distributing target UI objects to remote devices and rendering them



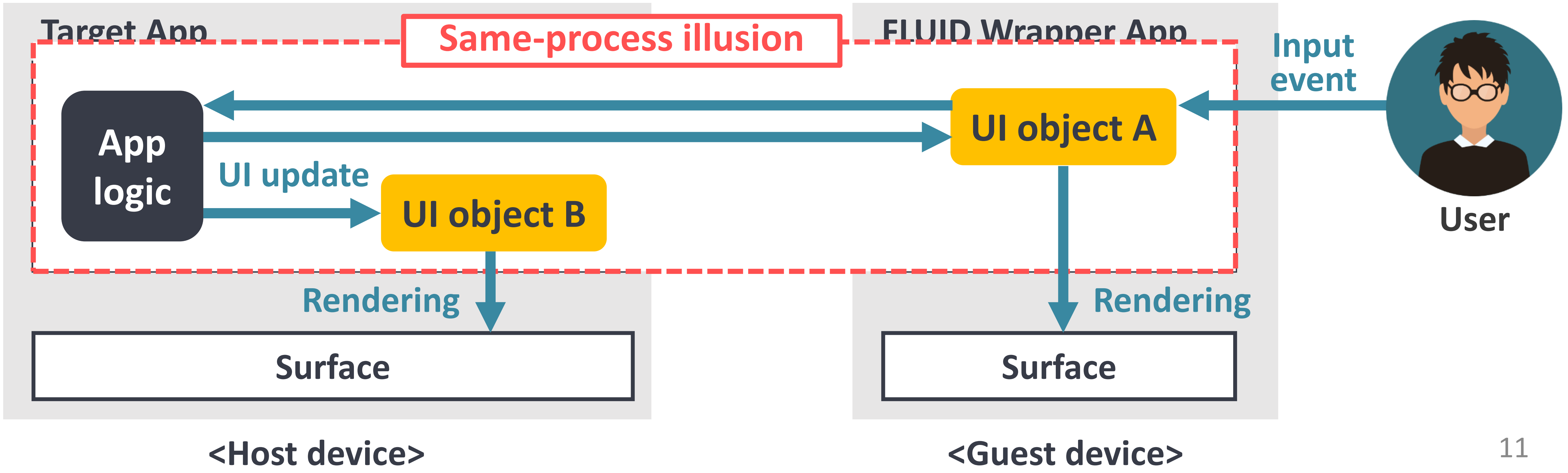
FLUID overview

- **Key idea**: separation between app logic & UI parts
    1) Distributing target UI objects to remote devices and rendering them



FLUID overview

- **Key idea**: separation between app logic & UI parts
  1) Distributing target UI objects to remote devices and rendering them
  2) Giving an illusion as if app logic and UI objects were in the same process

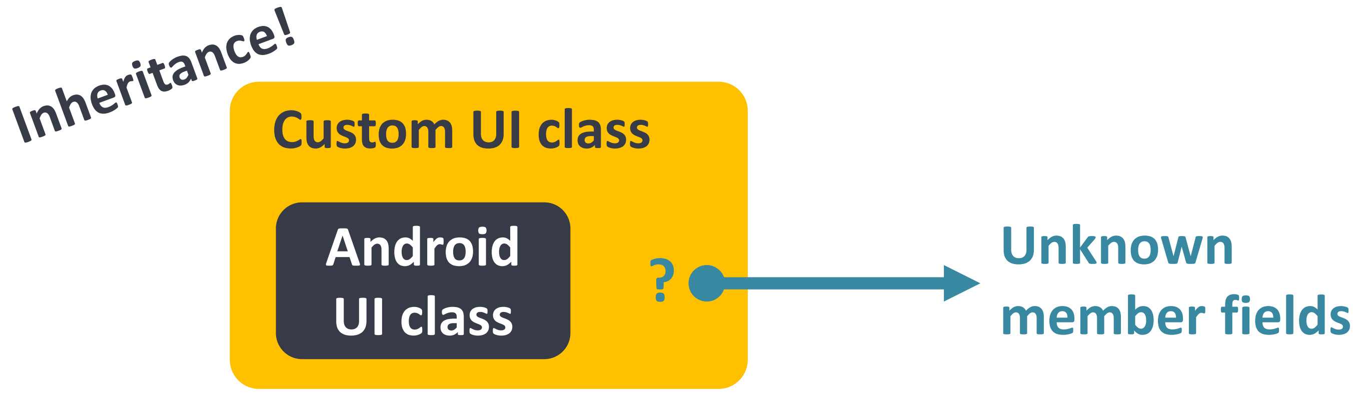


Why is FLUID good?

- **Flexibility**
  - Allow users to control multiple surfaces as they want via fine-grained UI distribution
- **Transparency**
  - Support legacy apps without any modification to them
  - Develop new multi-surface apps under the existing programming model
- **Responsiveness**
  - Require less network transmission when moving UIs instead of full screen

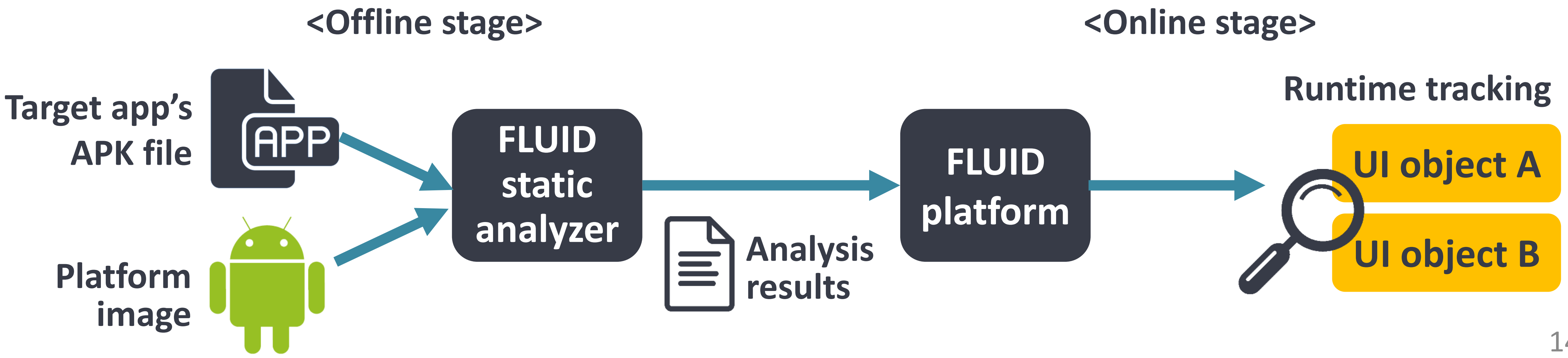
Problems

- **P1**. How to split & distribute UI objects?
  - Transmits minimum graphical states needed for UI rendering
    - To reduce network overhead
  - However, <u>it is unknown which graphical states app-specific custom UIs use</u>



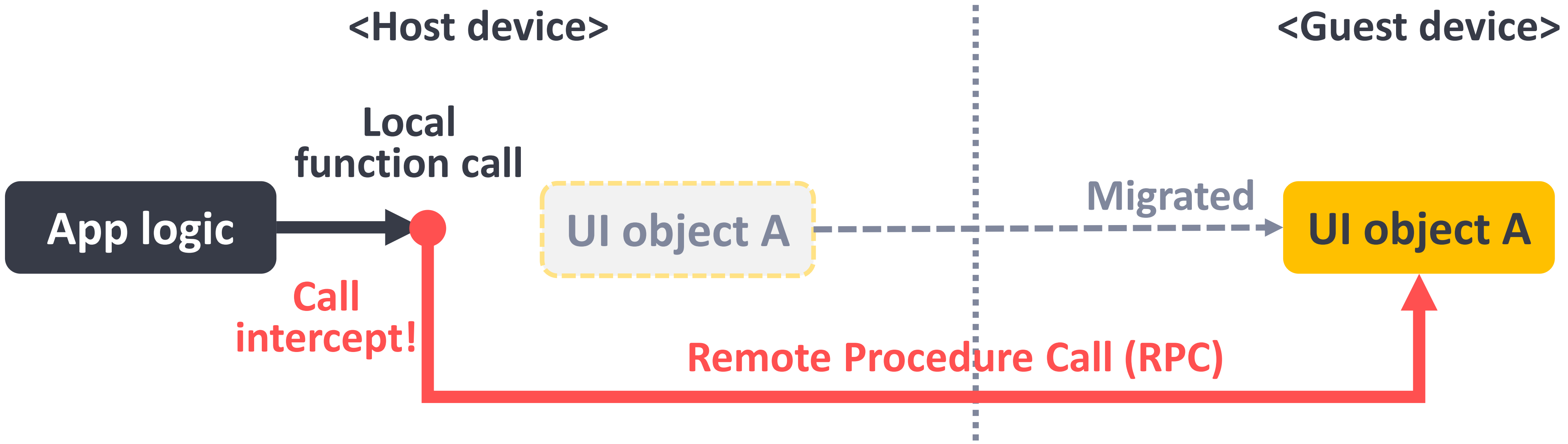
Problems

- **P1**. How to split & distribute UI objects?
  - Transmits minimum graphical states needed for UI rendering
    - To reduce network overhead
  - However, it is unknown which graphical states app-specific custom UIs use
  - **Our solution**: Selective UI distribution



Problems

- **P2**. How to maintain interaction between app logic & UI objects?
  - Such interaction is achieved via local function calls
    - e.g., *TextView.setText()*, *ImageView.setImageResource()*, etc.
  - However, <u>local functions cannot be executed across devices</u>
  - **Our solution**: transparent RPC transformation in *Android VM (ART)*



Evaluation environment

- Implemented FLUID prototype based on Android 8.1 (Oreo)
- Used Google Pixel XL (smartphone) & Pixel C (tablet)
  - Phone-to-phone
  - Phone-to-tablet
  - Tablet-to-phone
- On the Same WiFi network

Pixel XL

Pixel C

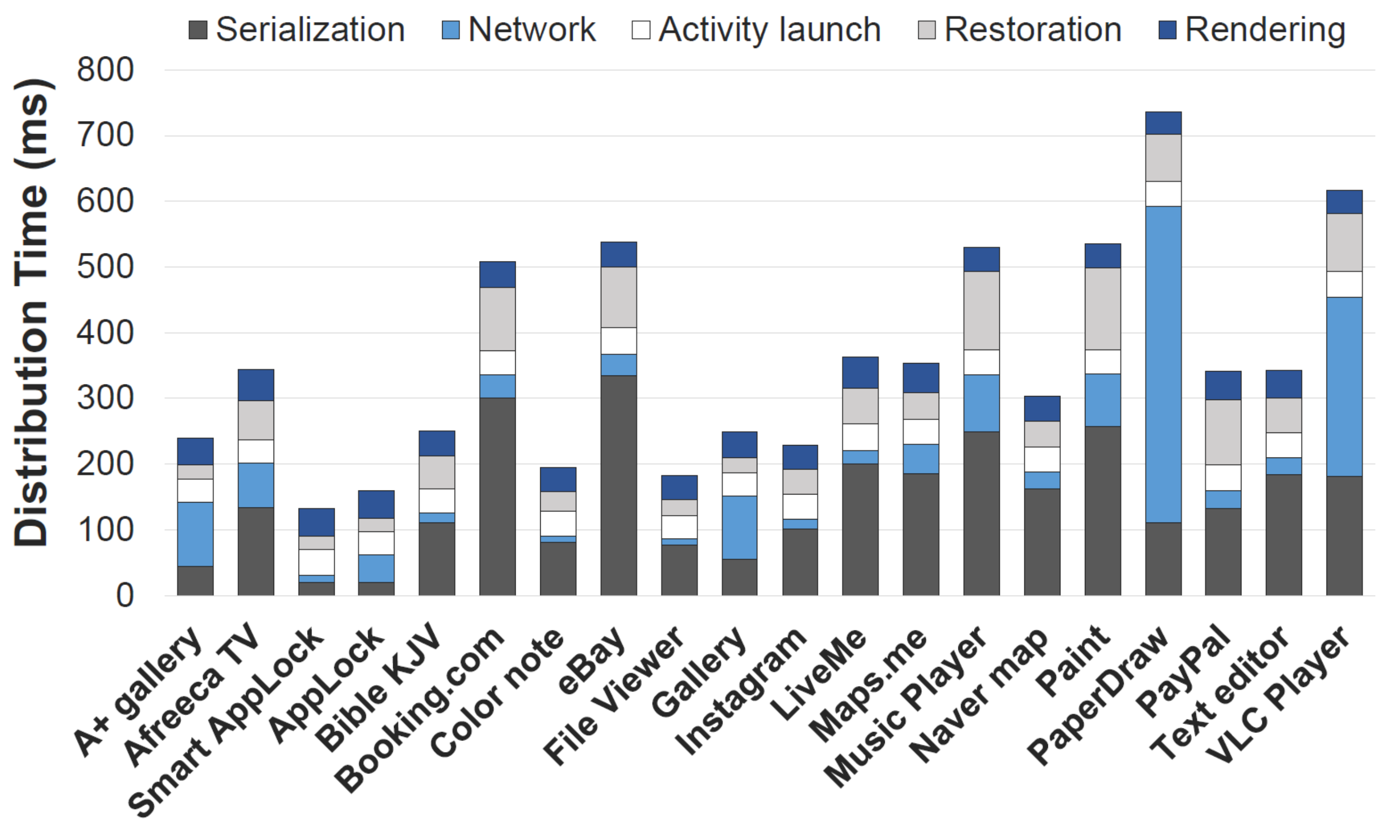
App coverage

- Using 20 legacy apps for 10 multi-surface scenarios
  - All legacy apps use their own custom UIs
- FLUID can support various legacy apps successfully

Use case scenario	UI type	App name
Login with personal device	Editor	Instagram
		Paypal
Fill in information collaboratively	Text, editor	eBay
		Booking.com
Chatting with different device while broadcasting	Button, editor	LiveMe
		Afreeca TV
Search destination with different device	Button, editor	Naver map
		Maps.me
Control media with different device	Seek bar, button	VLC Player
		Music Player
Control painting tool with different device	Scroll, image, button	PaperDraw
		Paint
Sharing photo to public device	Image	Gallery
		A+ Gallery
Unlock pattern with personal device	Pattern lock	Smart app lock
		AppLock
Read document with different device	Text, scroll	File Viewer
		Bible KJV
Edit text on different device	Editor	Color note
		Text editor

UI distribution time

- It ranges from 132 to 735ms ➔ Fast enough for interactive use



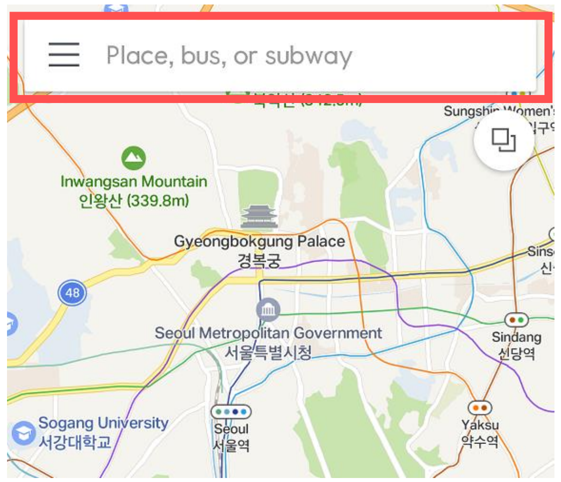
Network transfer over time

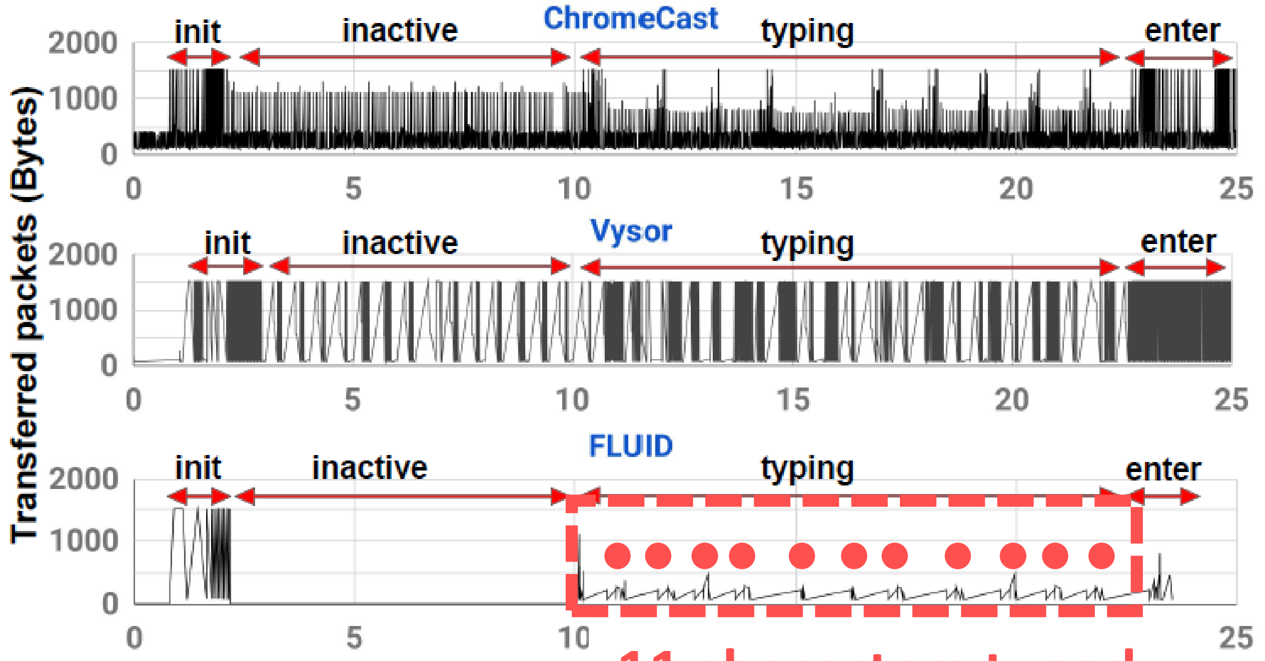
- Comparing transfer pattern of FLUID and other mirroring techniques
  - Under the same scenario that a user types destination (11 characters) into the search box UI of Naver map

Chromecast

Vysor

Search box UI



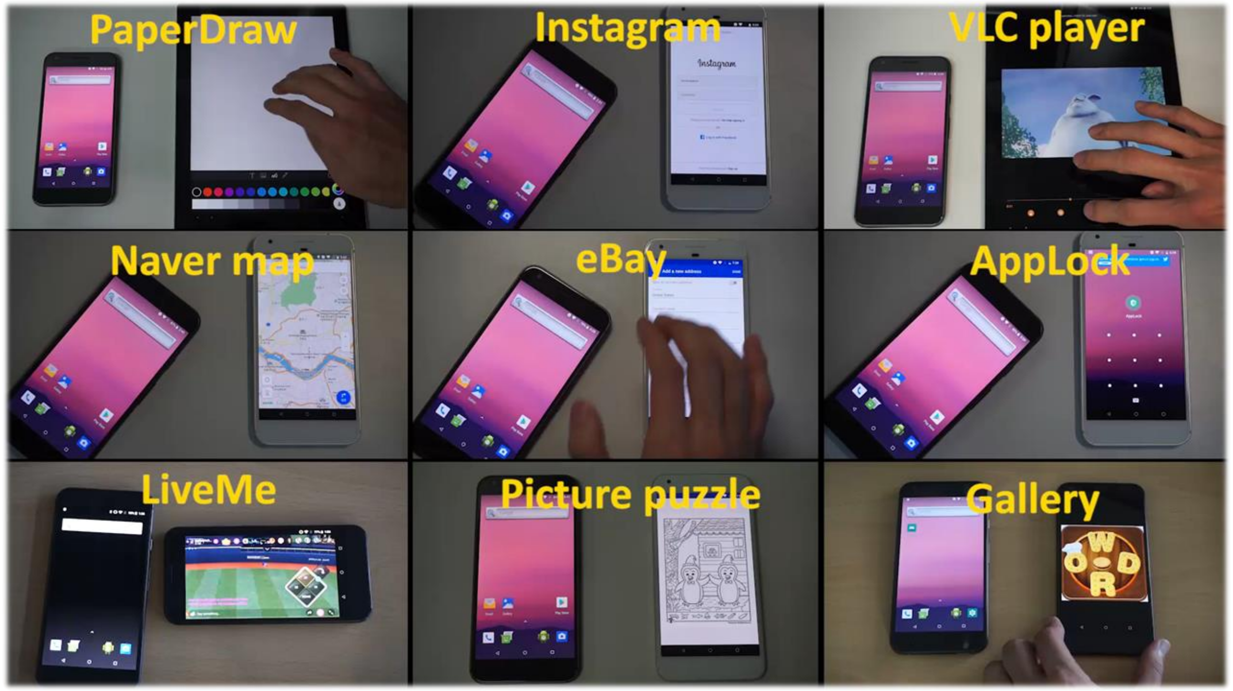


11 characters typed

Conclusion

- Designed & implemented **FLUID**
  - Separation between app logic & UIs
  - Evaluated with 20 legacy apps for 10 multi-surface scenarios

- Expect FLUID to accelerate development of creative and useful apps to provide novel UX



Thank you!

Visit cps.kaist.ac.kr/fluid for more information:)

KAIST

University at Buffalo
The State University of New York