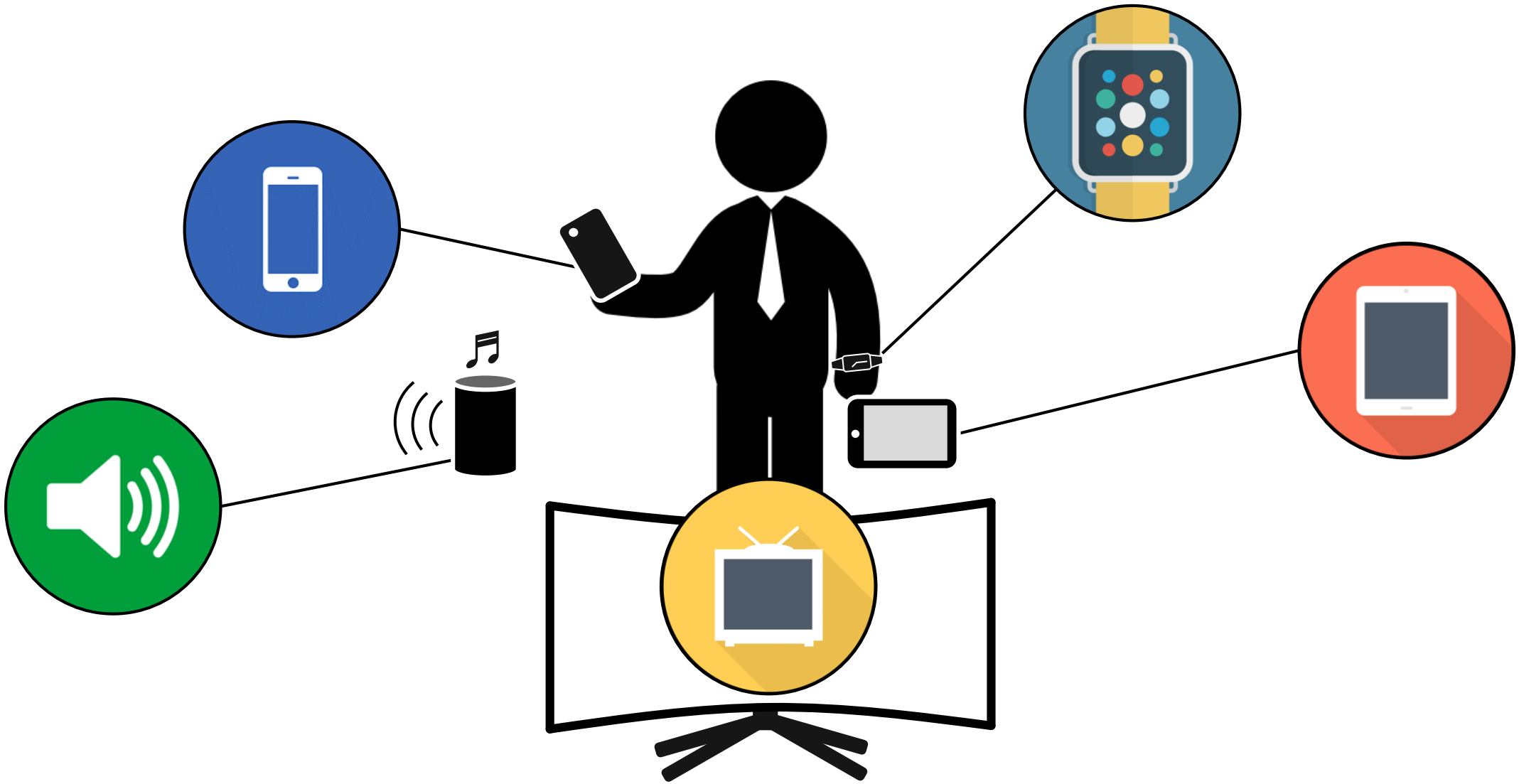
DynaMix:
Dynamic Mobile Device Integration for Efficient Cross-device Resource Sharing

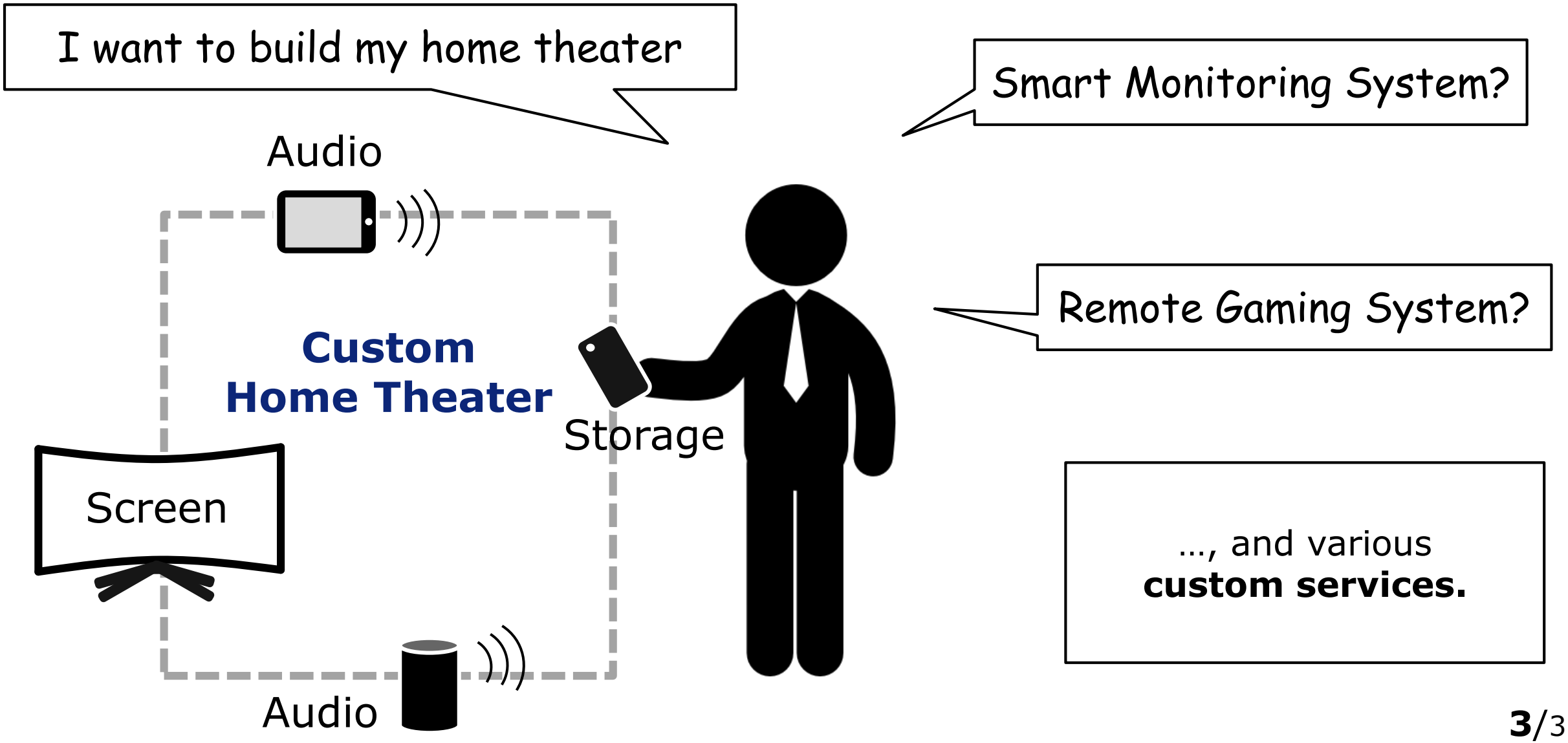
Dongju Chae[1], Joonsung Kim[2], Gwangmu Lee[2], Hanjun Kim[1], Kyung-Ah Chang[3], Hyogun Lee[3], and Jangwoo Kim[2]

[1]Dept. of Computer Science and Engineering, POSTECH
[2]Dept. of Electrical and Computer Engineering, Seoul National University
[3]Samsung Electronics

Increasing Number of IoT Devices



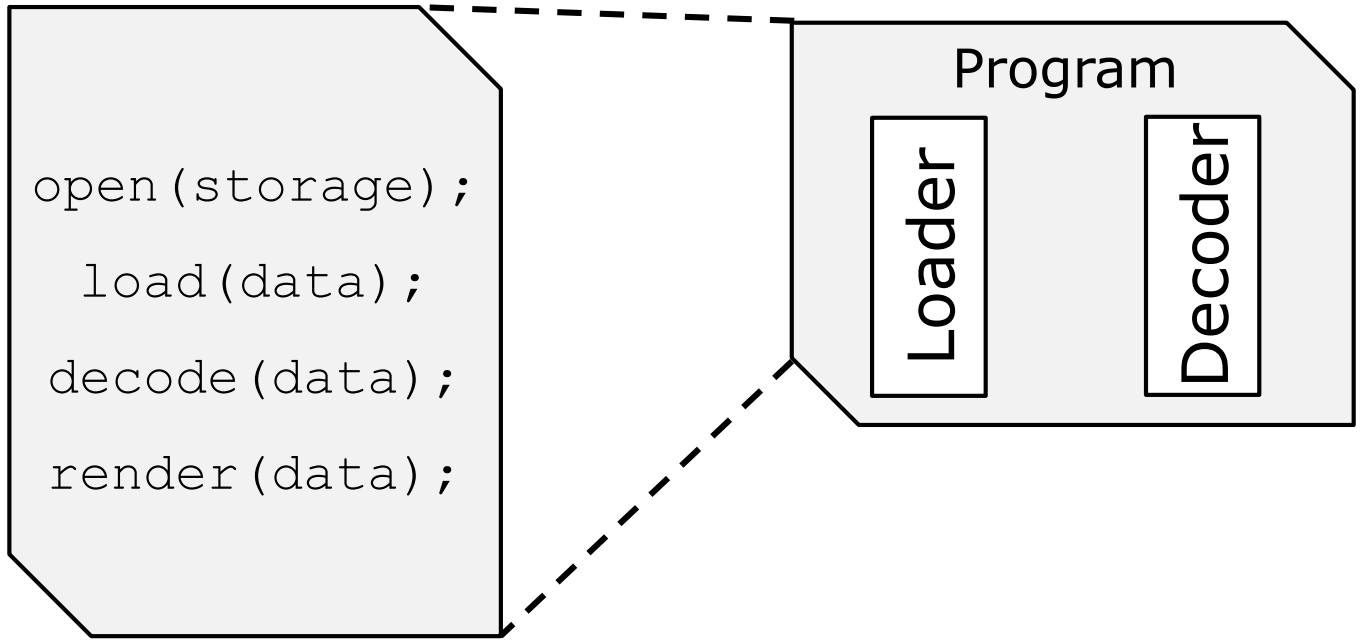
Sharing Multi-device Enables Many Services



Index

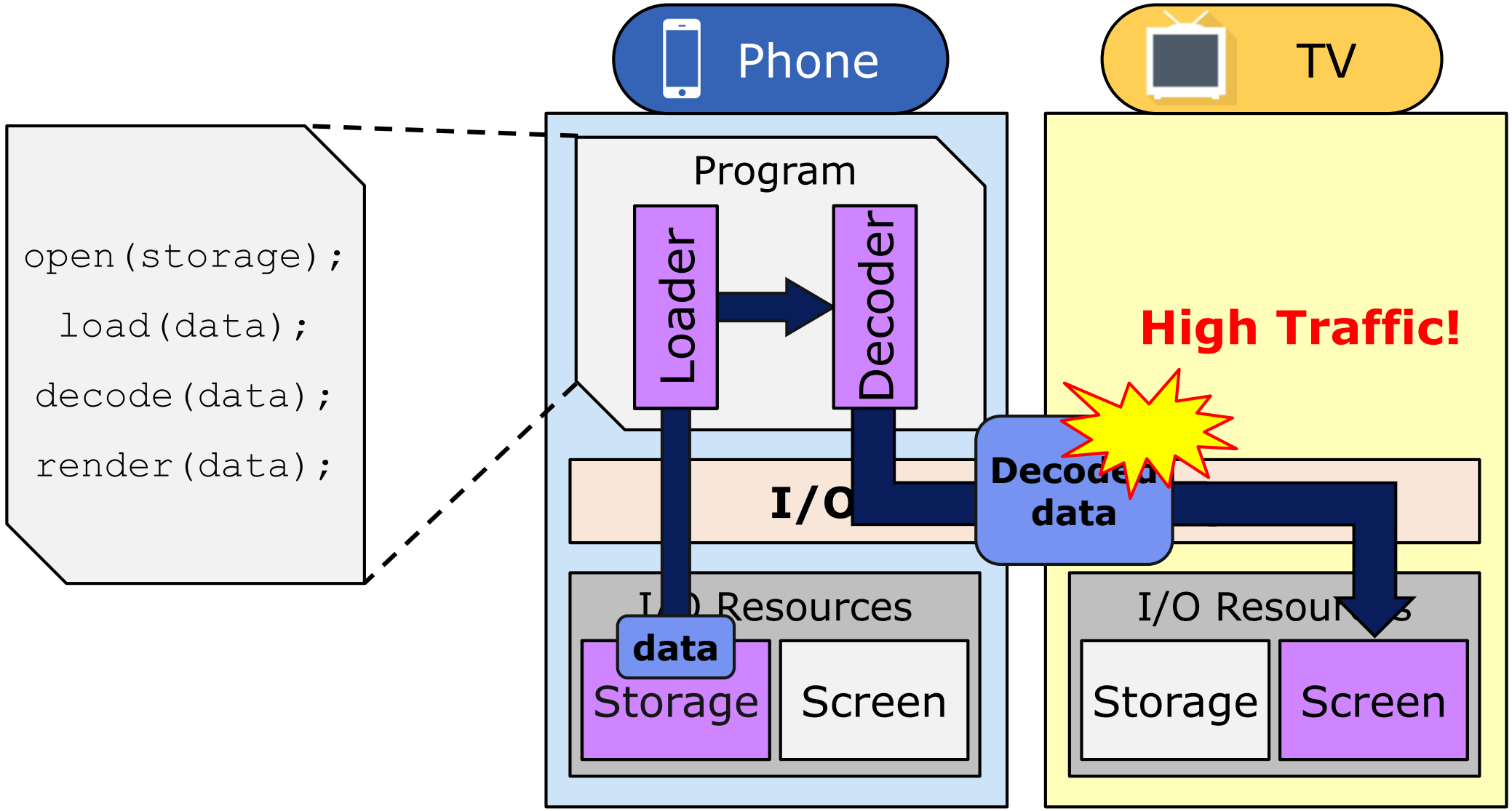
- **Limitations of existing schemes**
  - I/O request forwarding
  - Manual programming
- **DynaMix: Efficient dynamic resource sharing**
- **Evaluation**
- **Conclusion**

Running Example: Home Theater



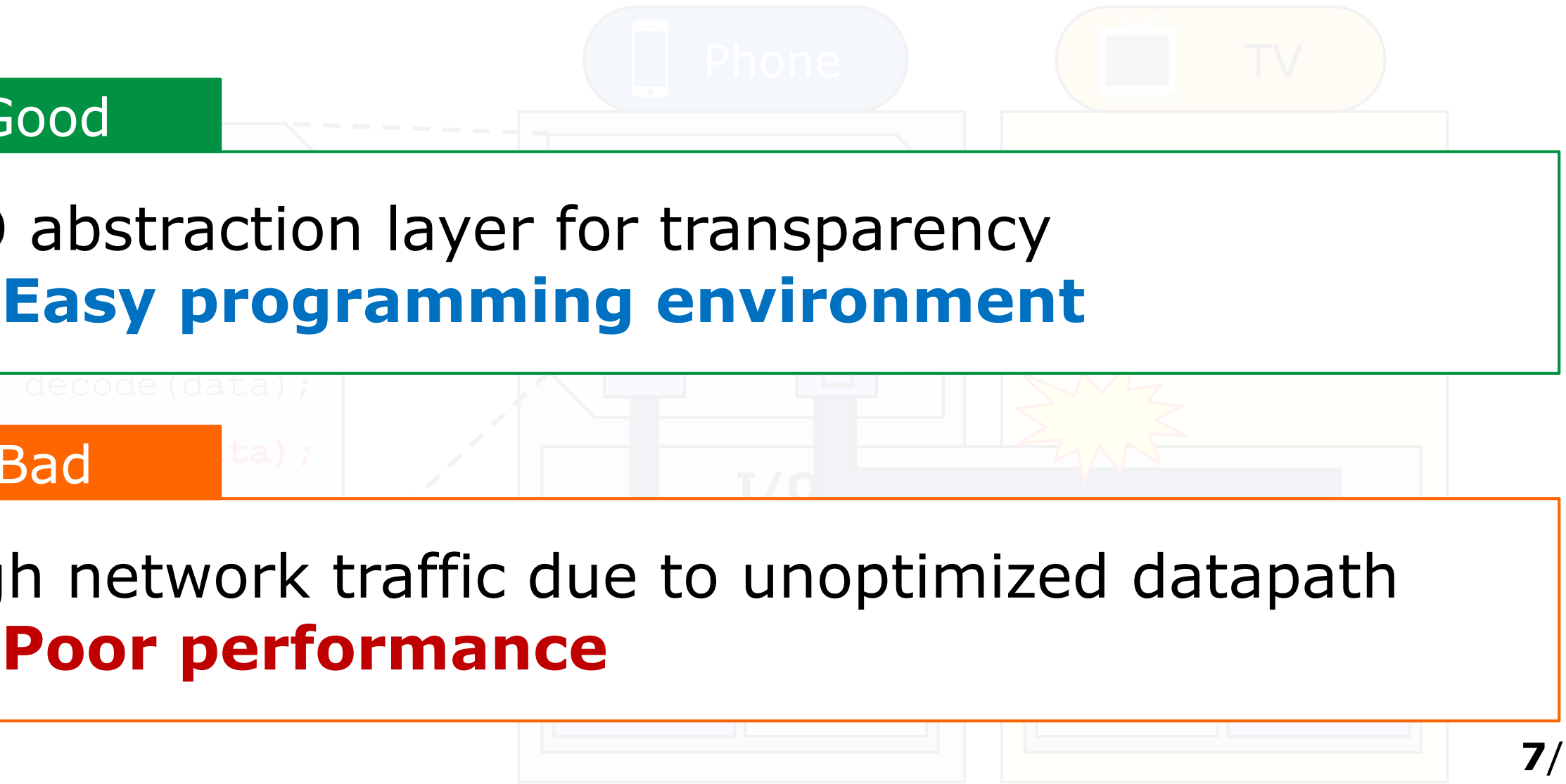
(1) I/O Request Forwarding

- with the example of ***home theater***



(1) I/O Request Forwarding

- with the example of ***home theater***



Good

I/O abstraction layer for transparency

✓ **Easy programming environment**

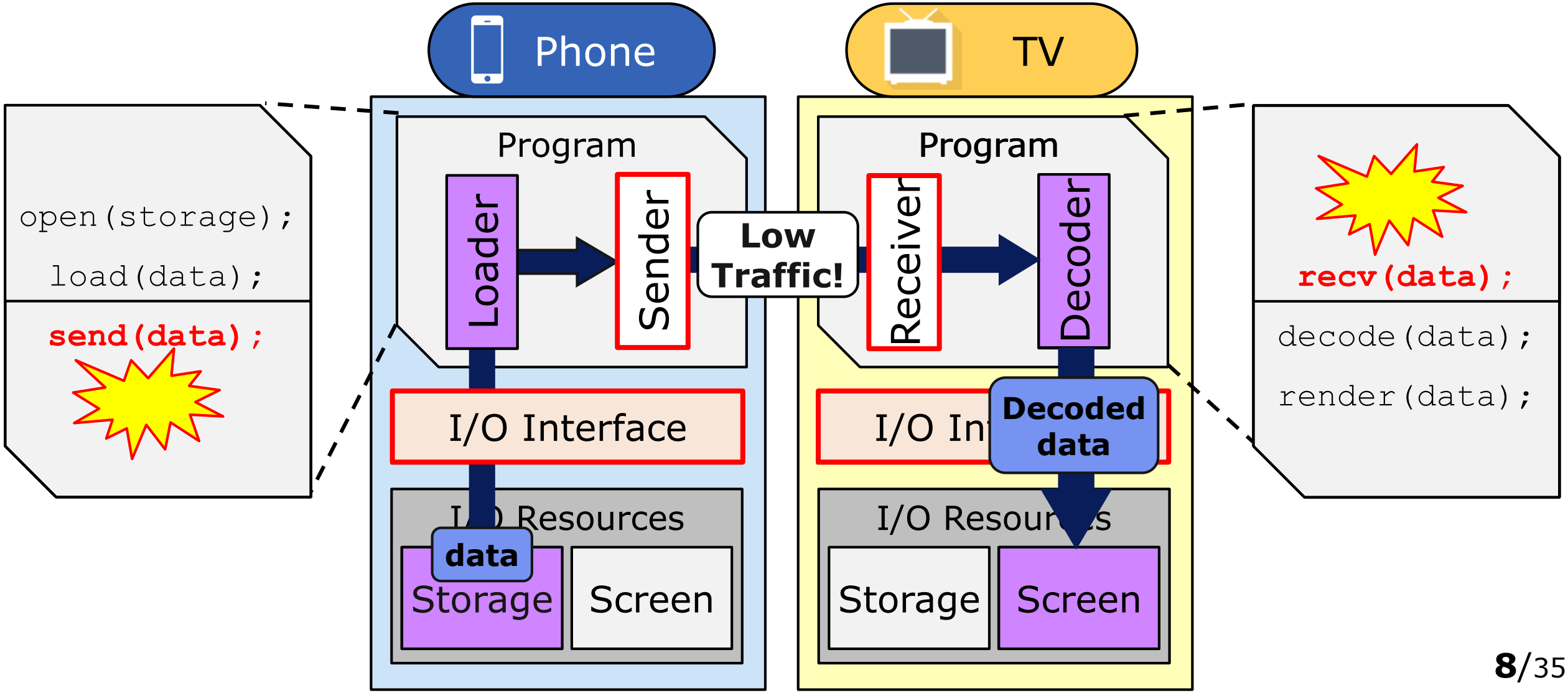
Bad

High network traffic due to unoptimized datapath

✓ **Poor performance**

(2) Manual Programming

- with the example of ***home theater***



(2) Manual Programming

- with the example of ***home theater***

Good

Device-aware task partitioning

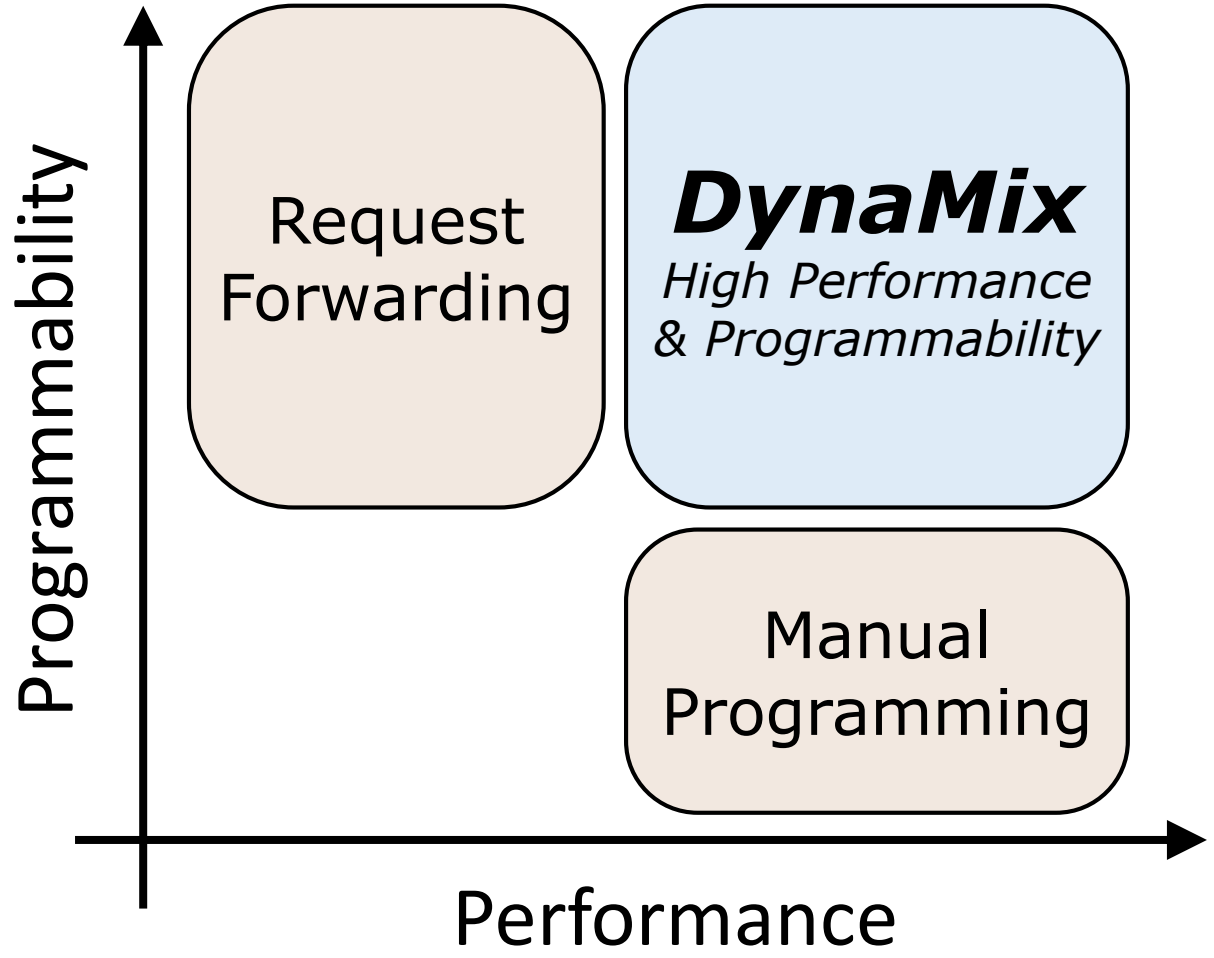
✓ **Good performance**

Bad

Hand-tuned multi-device application

✓ **High programming effort**

Design Goals



Index

- Limitations of existing schemes
- **DynaMix: Efficient dynamic resource sharing**
  - Key ideas
  - Architecture & Implementation
- Evaluation
- Conclusion

Key Ideas of *DynaMix*

1) Transparent & Wide Resource Integration

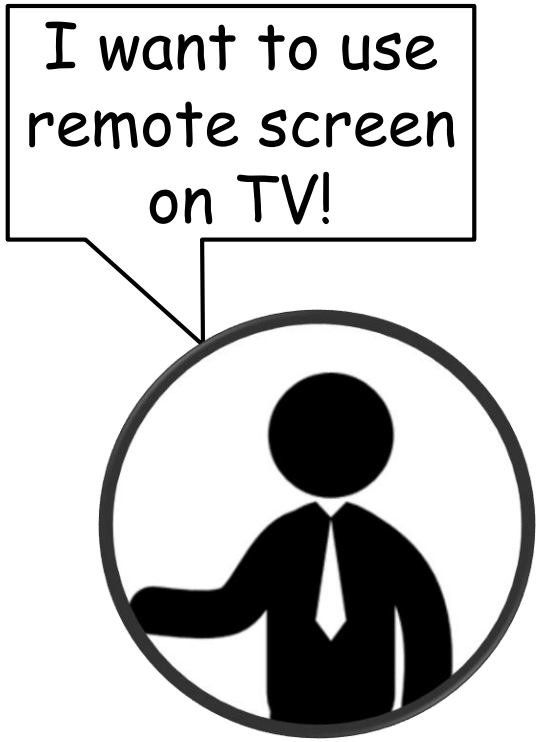
- Kernel-level resource integration
  - ➔ Transparent integration (easy programmability)
- Sharing multiple types of resources (i.e., CPU, Memory, I/O)
  - ➔ Wide resource coverage

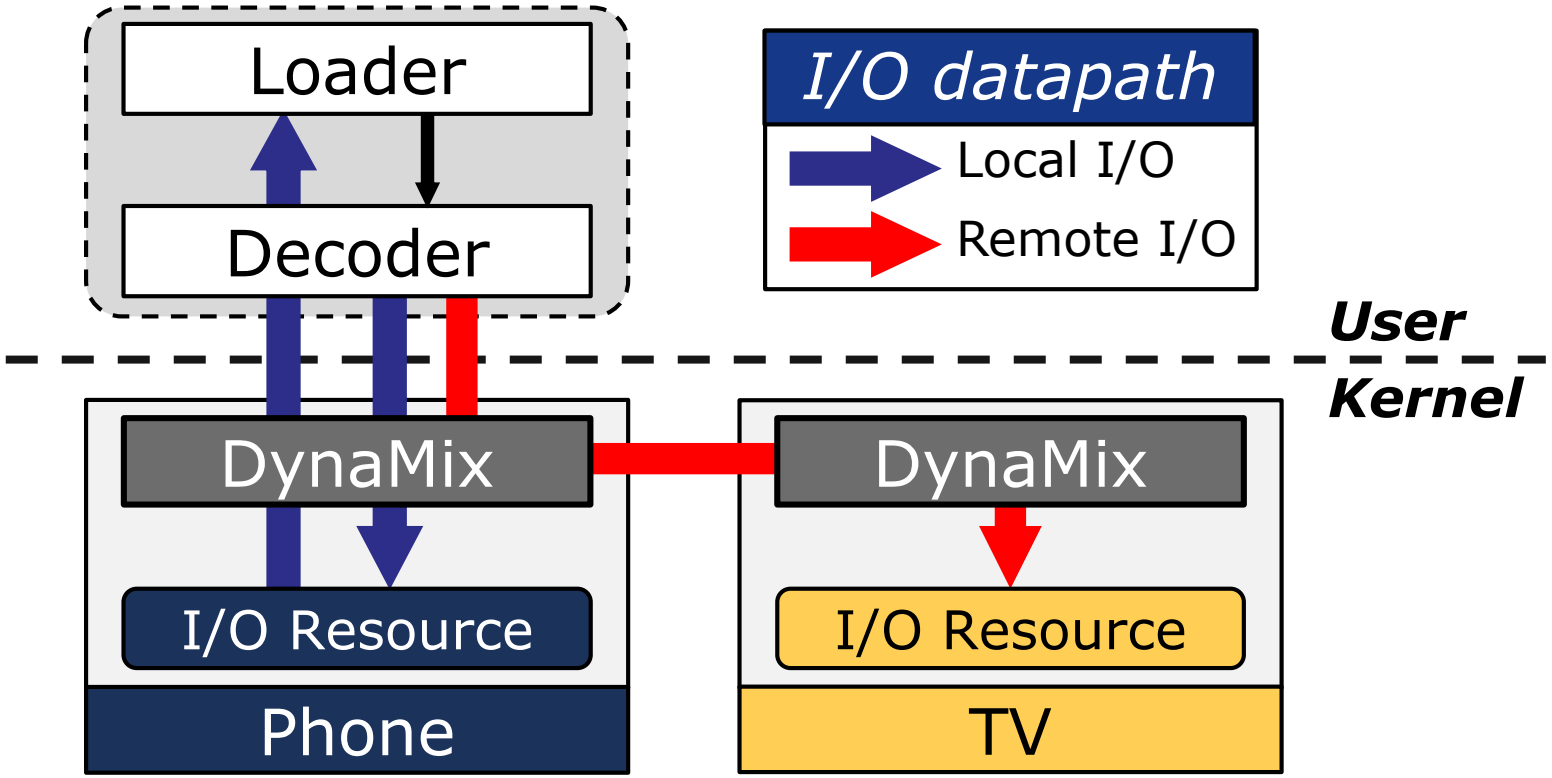
2) Resource-aware Dynamic Task Redistribution

- Contention detection based on resource usage
- Performance estimation for migration scenarios
  - ➔ Optimized performance

Key Ideas of *DynaMix* (1/2)

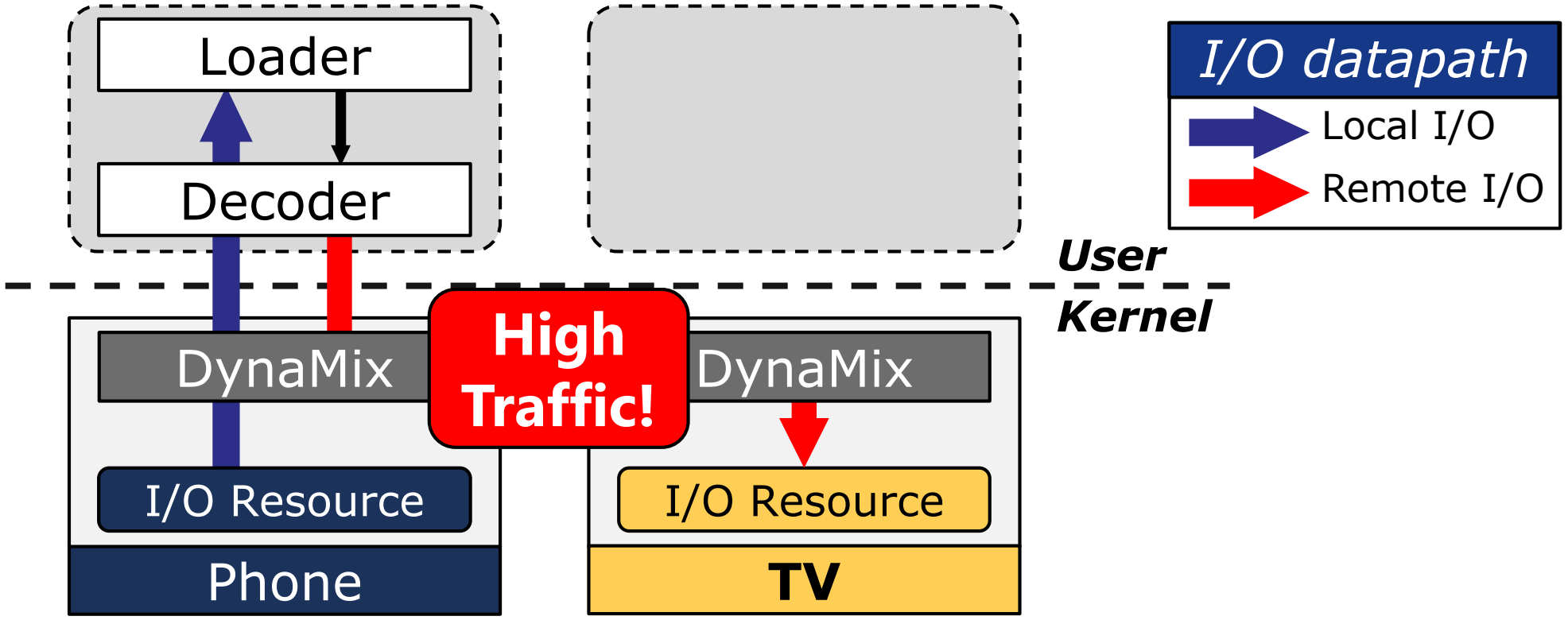
1) Transparent & Wide Resource Integration





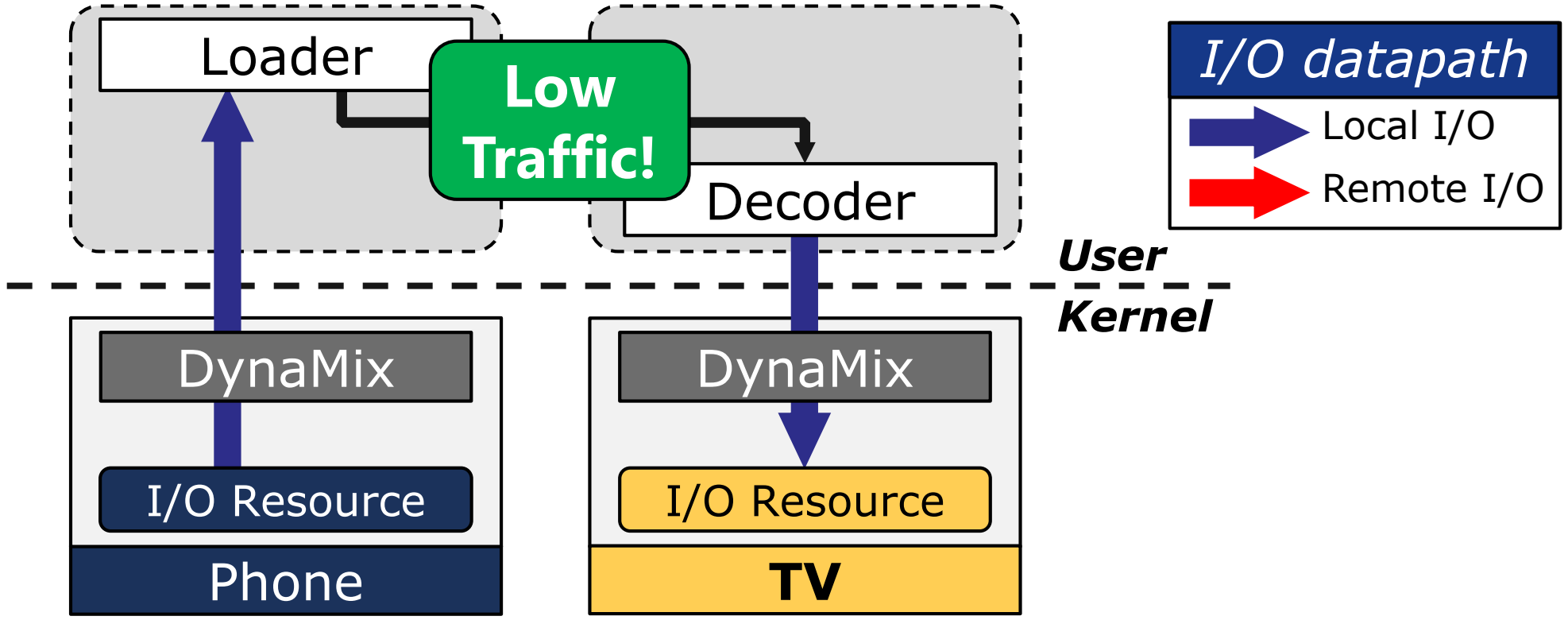
Key Ideas of *DynaMix* (2/2)

2) Resource-aware Dynamic Task Redistribution

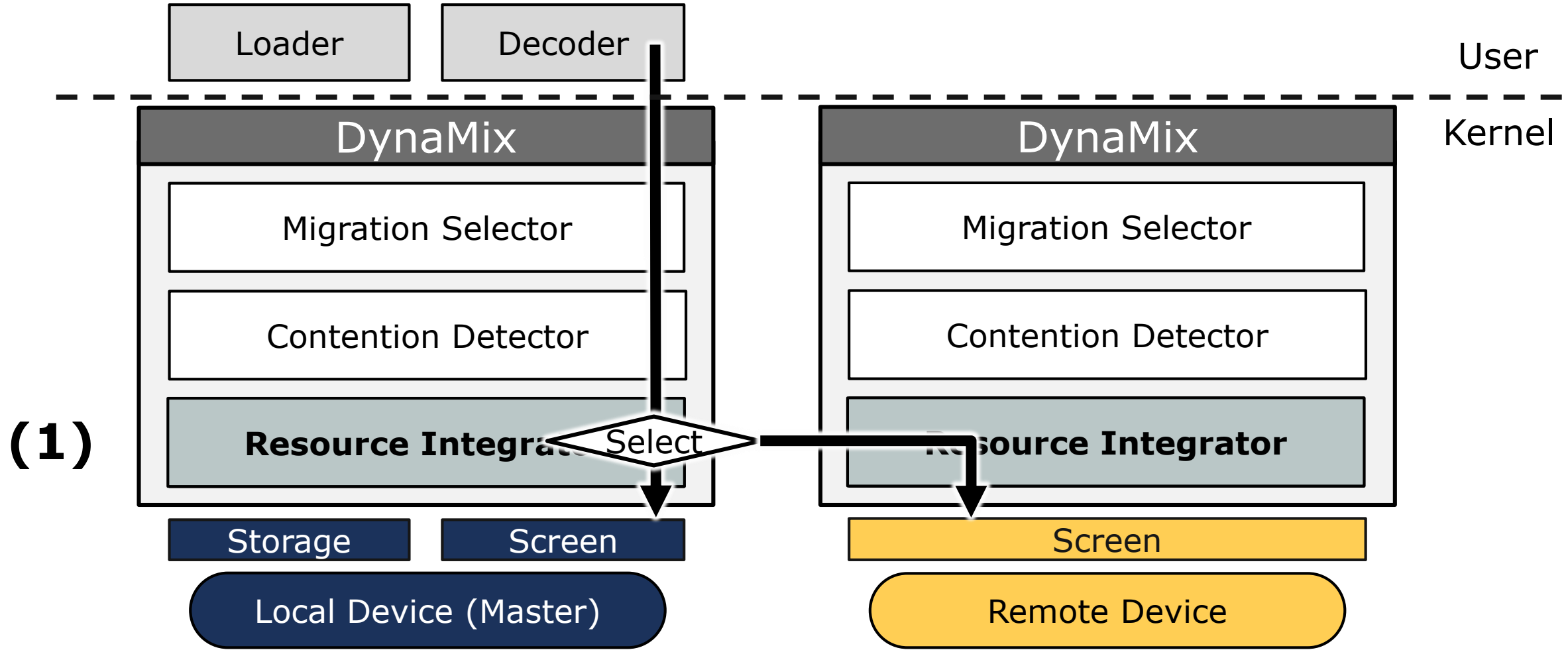


Key Ideas of *DynaMix* (2/2)

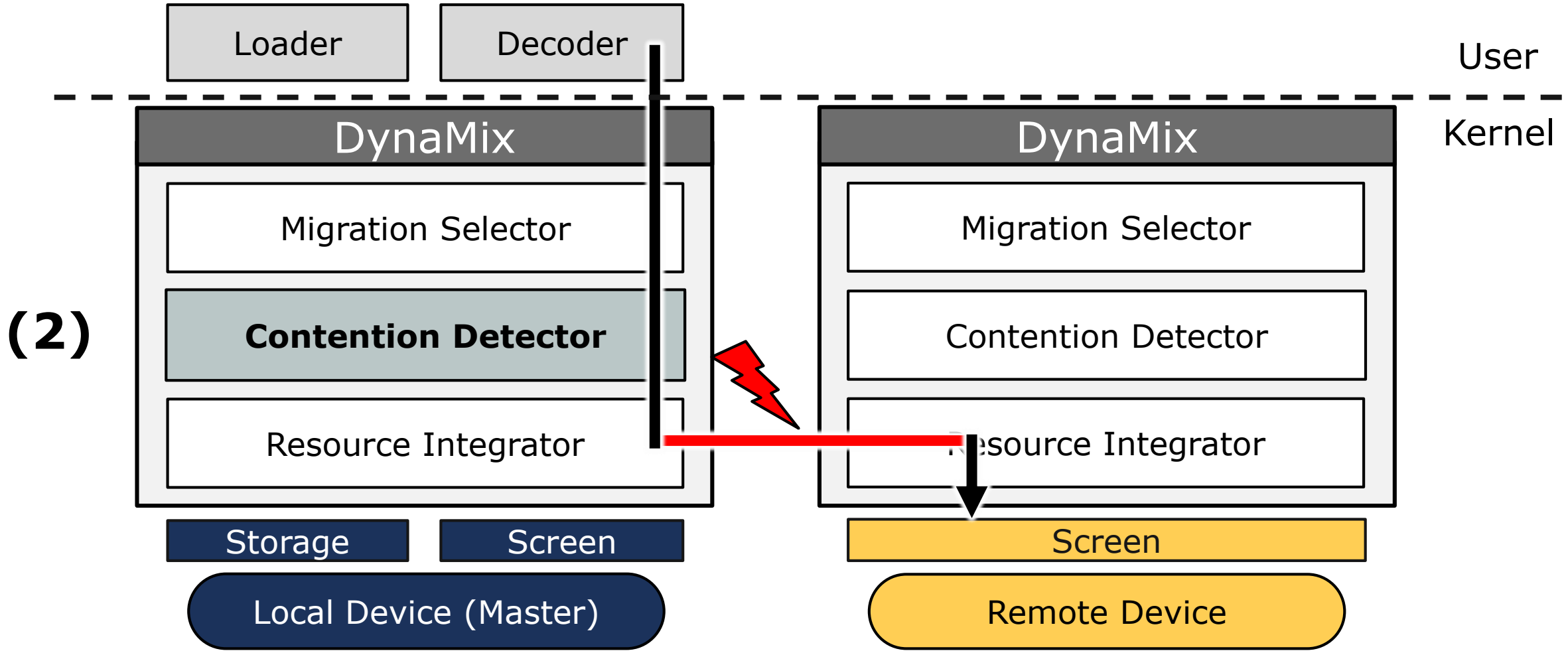
2) Resource-aware Dynamic Task Redistribution



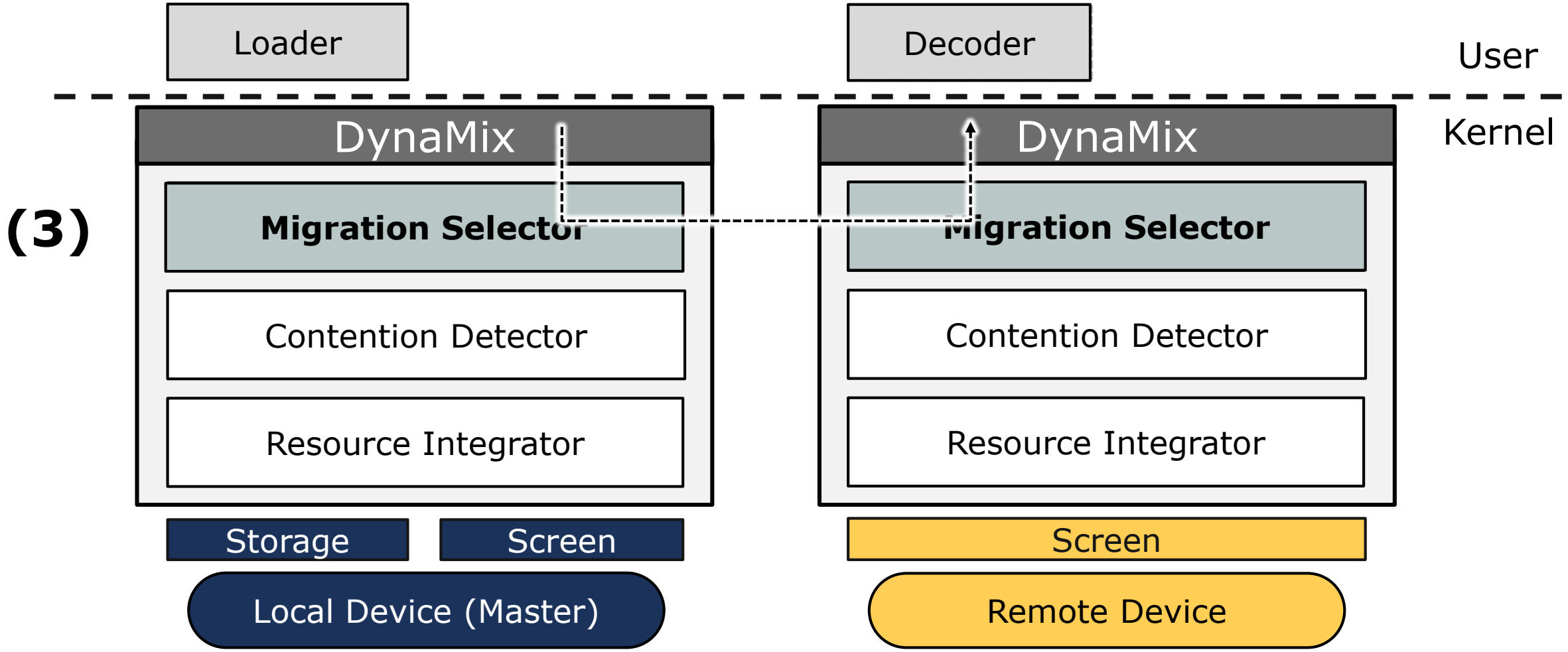
DynaMix Architecture



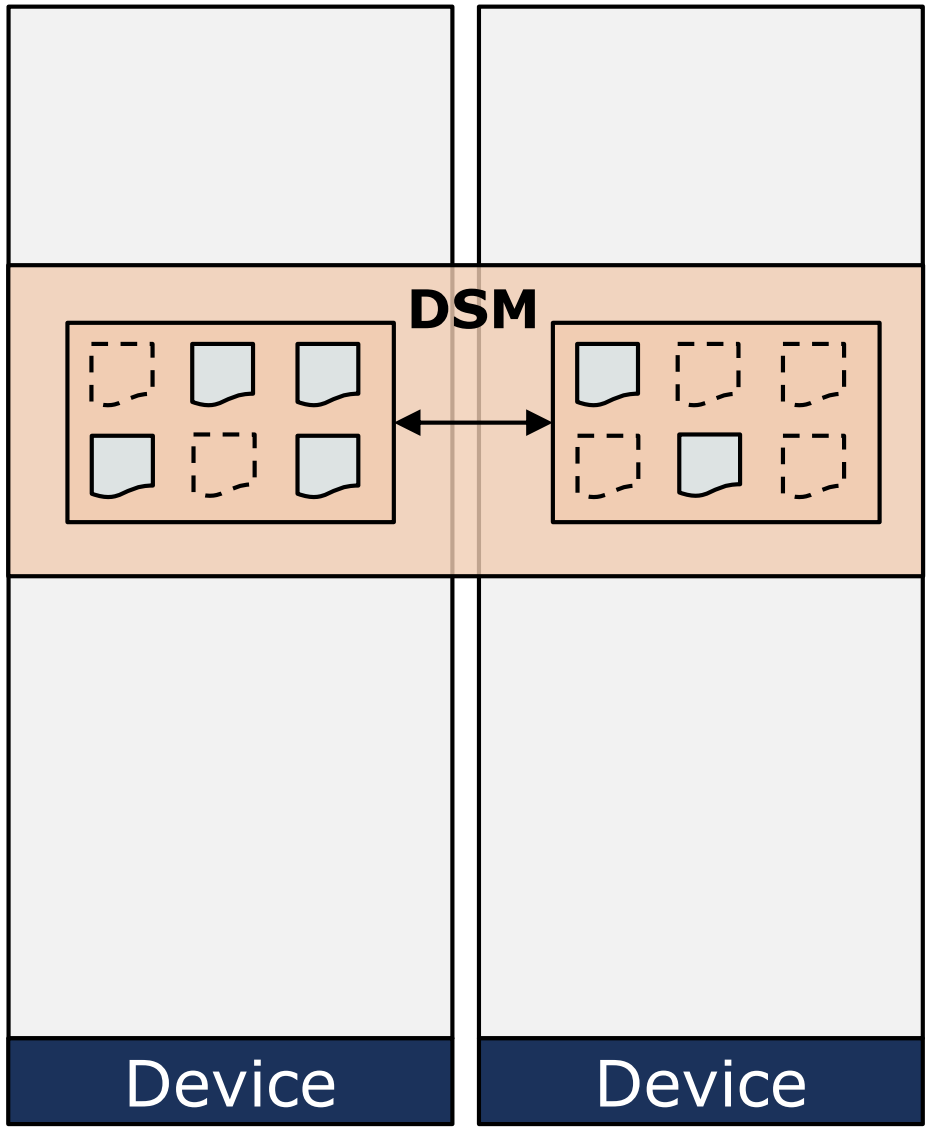
DynaMix Architecture



DynaMix Architecture

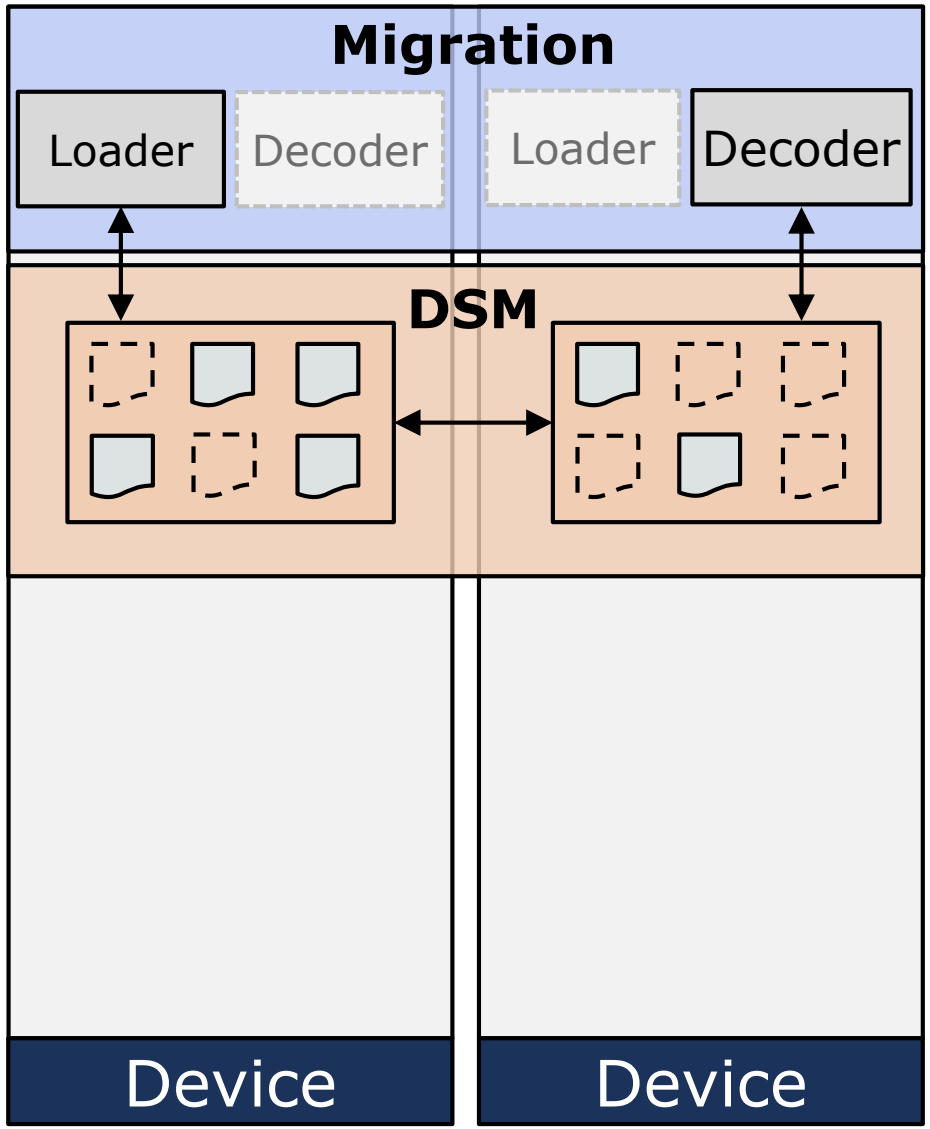


(1) Resource Integrator – (Memory, CPU, I/O)



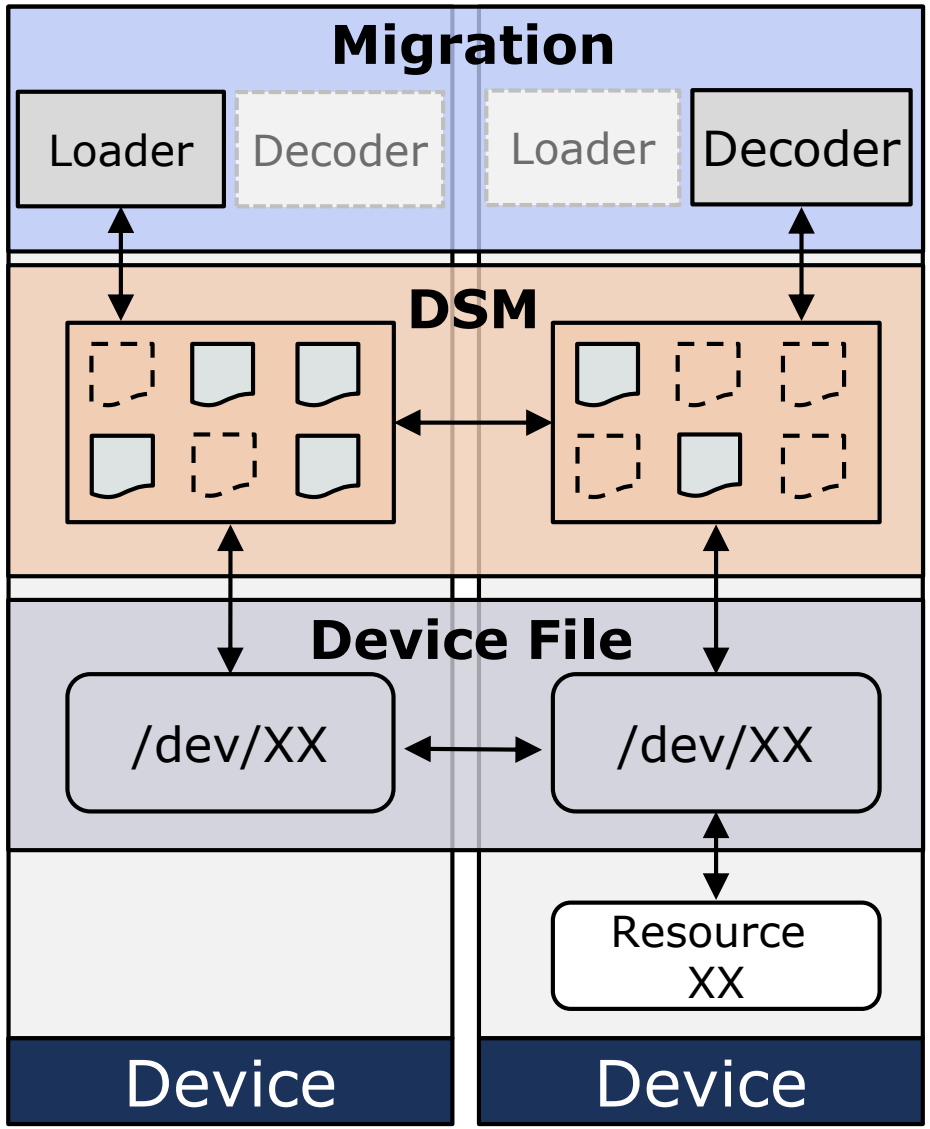
- **Memory integration**
  - Distributed shared memory (DSM)
- **Three perf. optimizations**
  - Lazy release consistency model
  - Page-level coherency block
  - Memory prefetching

(1) Resource Integrator – (Memory, **CPU**, I/O)



- **CPU integration**
  - Thread migration
- **Optimizations**
  - Thread group granularity
  - Clone-based migration
  - Transparent live migration

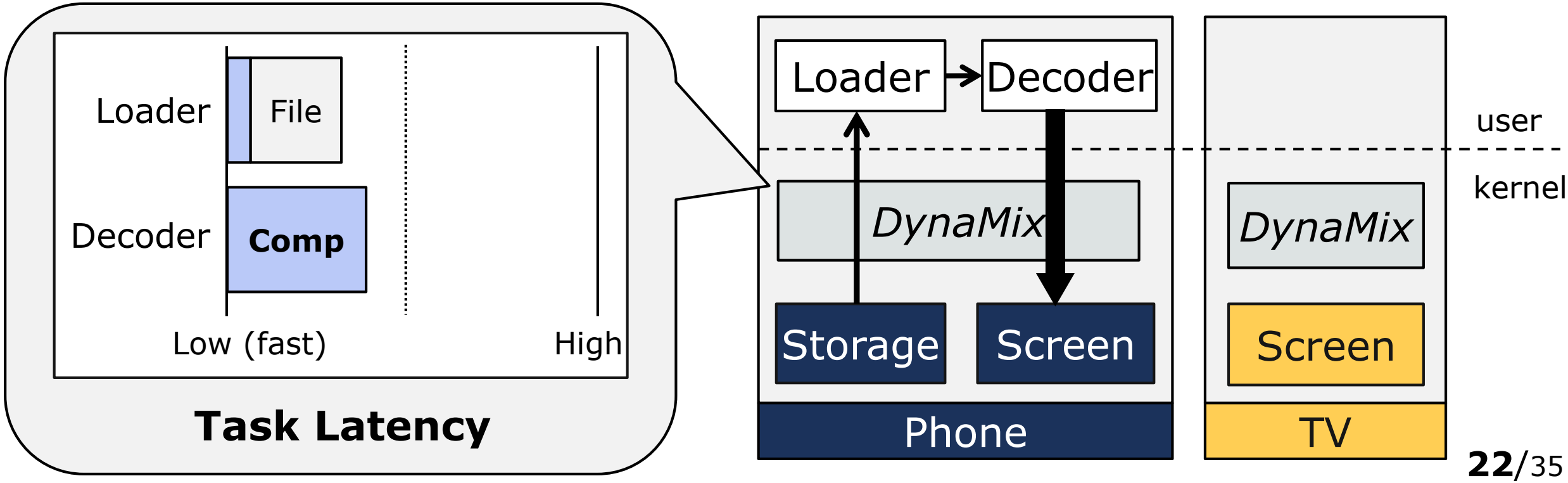
(1) Resource Integrator – (Memory, CPU, I/O)



- **I/O resource integration**
  - Request forwarding (e.g., device file)
- **Optimizations**
  - Data compression
  - Platform-assisted handling

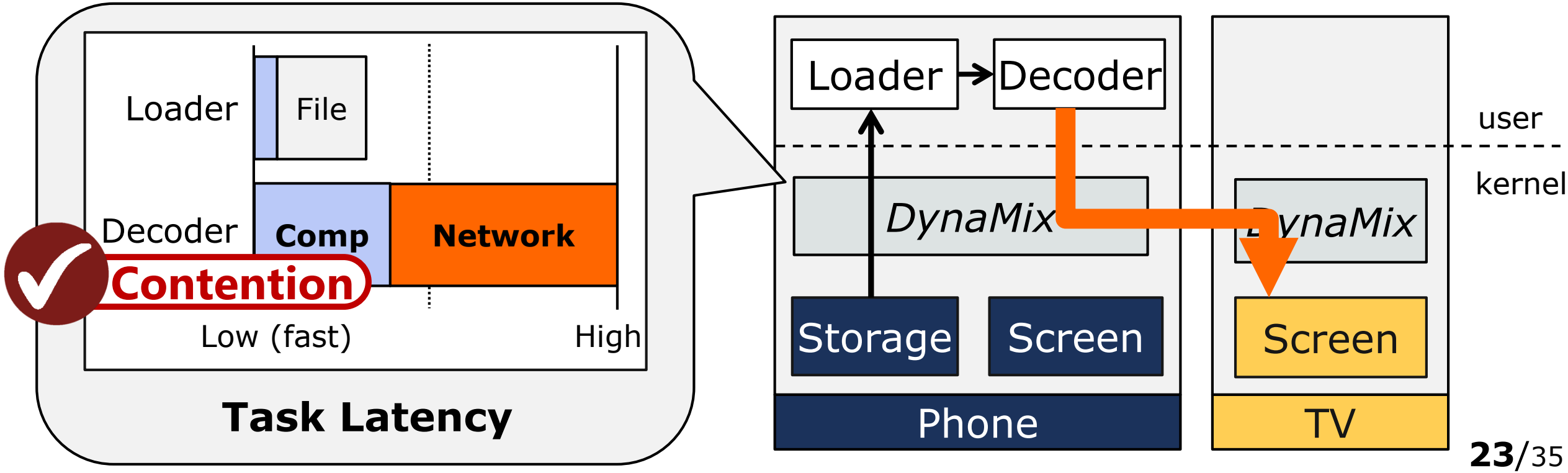
2) Contention Detector

- **Calculate the slowdown of each thread**
  - Collect per-thread resource usage (e.g., CPU, network, …)
  - Measure a stall time due to resource access



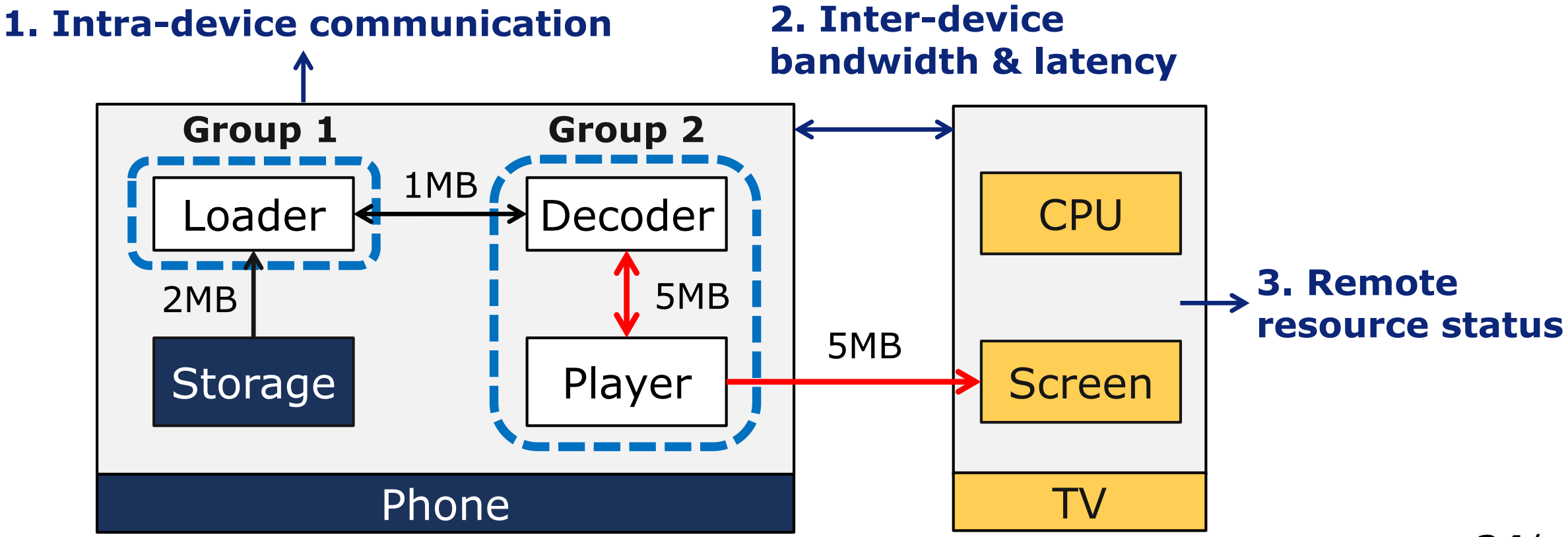
2) Contention Detector

- **Calculate the slowdown of each thread**
  - Collect per-thread resource usage (e.g., CPU, network, …)
  - Measure a stall time due to resource access



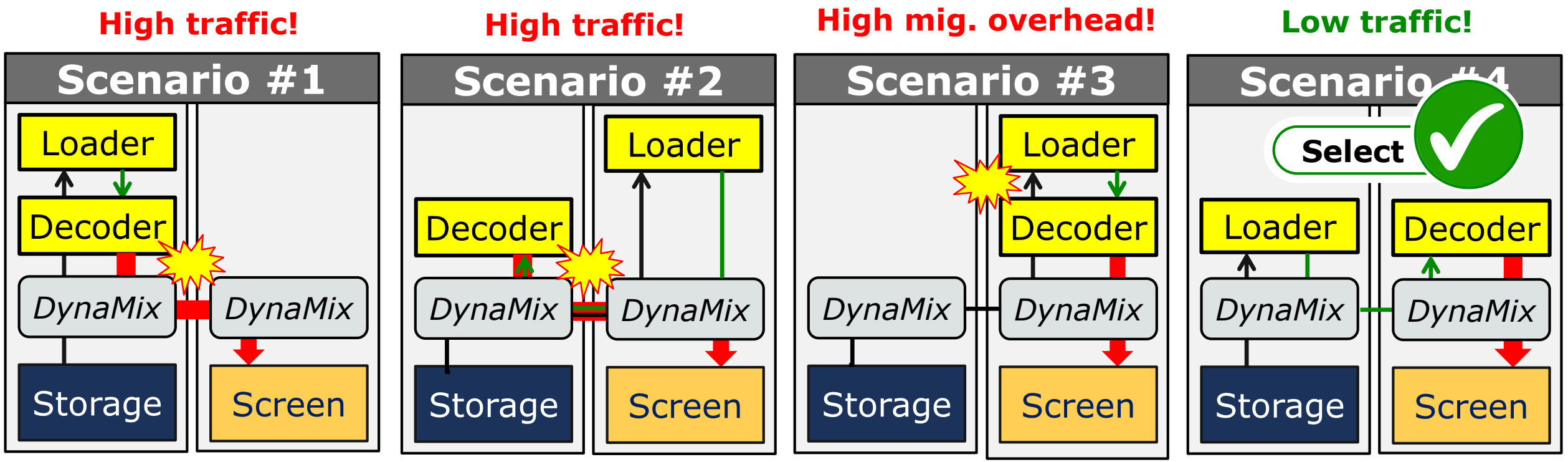
3) Migration Selector

- **Estimate the performance tradeoff after migration**
  - Utilize intra-/inter-device information to decide migration targets



3) Migration Selector

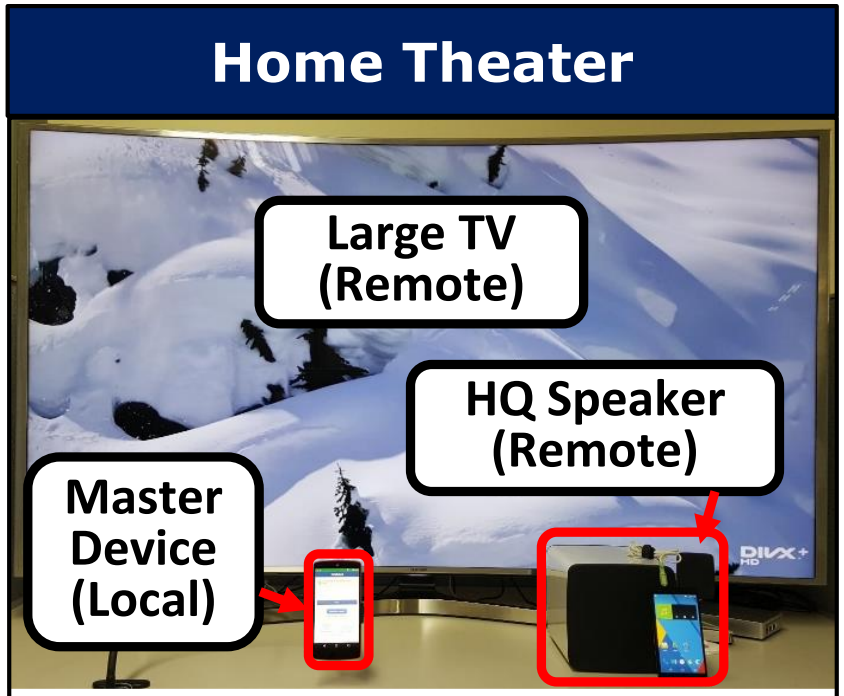
- **Estimate the performance tradeoff after migration**
  - Utilize intra-/inter-device information to decide migration targets
  - Calculate the tradeoffs of all possible migration scenarios



Index

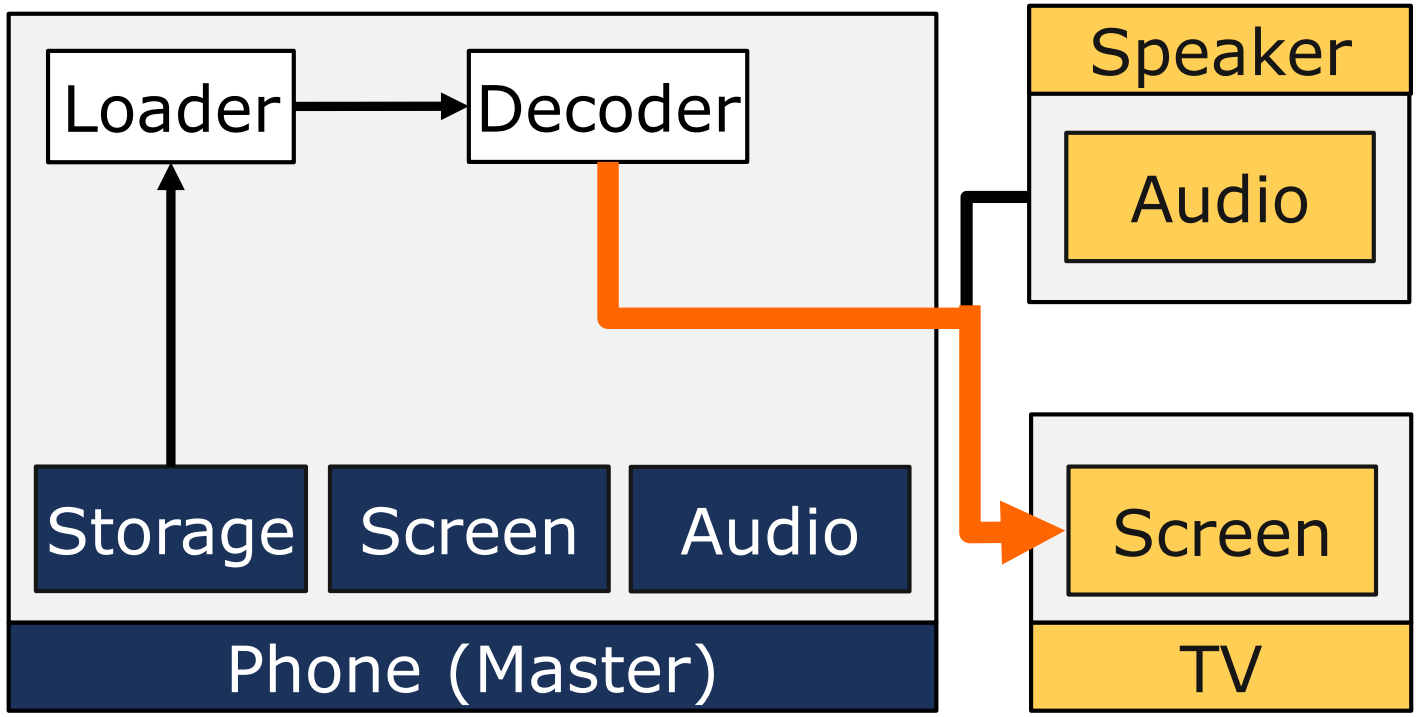
- Limitations of existing schemes
- DynaMix: Efficient dynamic resource sharing
- **Evaluation**
  - Scenario #1: Home theater
  - Scenario #2: Smart monitoring
- Conclusion

Evaluation Setup: Home theater



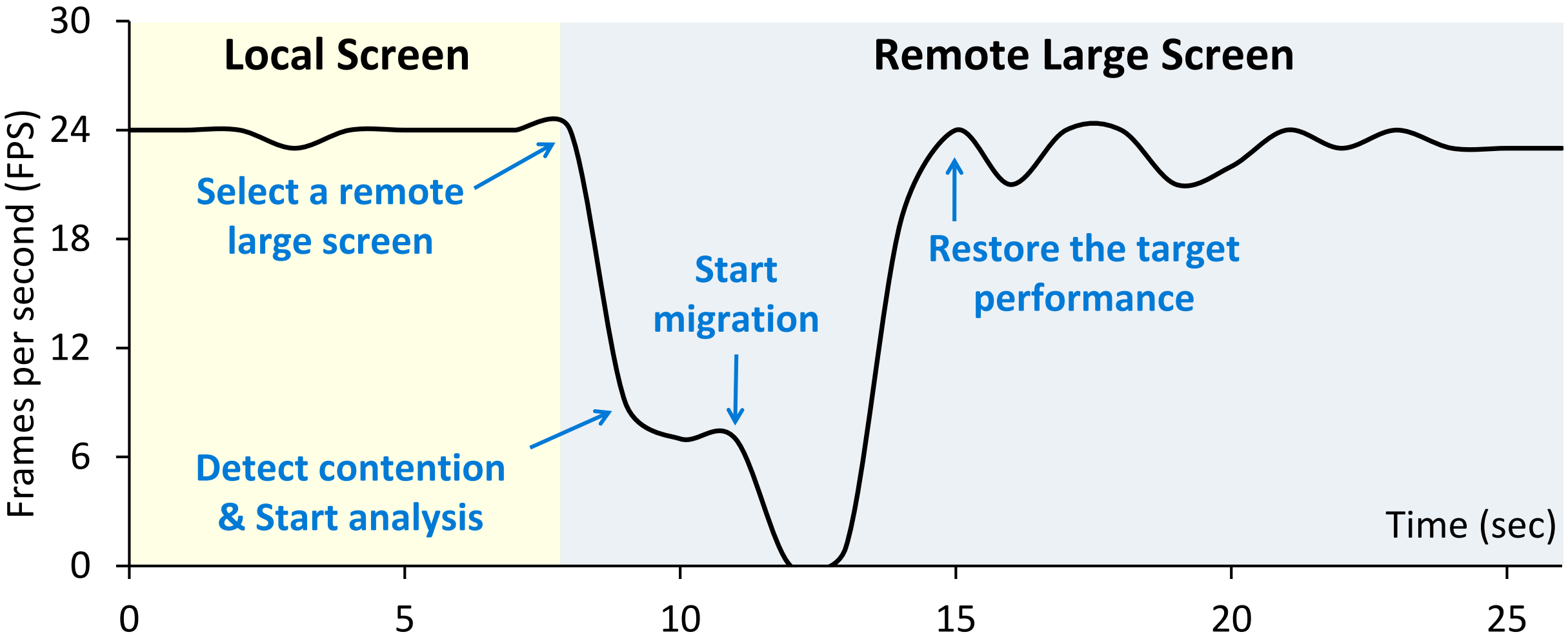
Android-based smartphones (5.1.1)
- Nexus 5 (master)
- Nexus 4 (connected to a speaker)

Tizen-based smart TV (2.3)



Performance Timeline

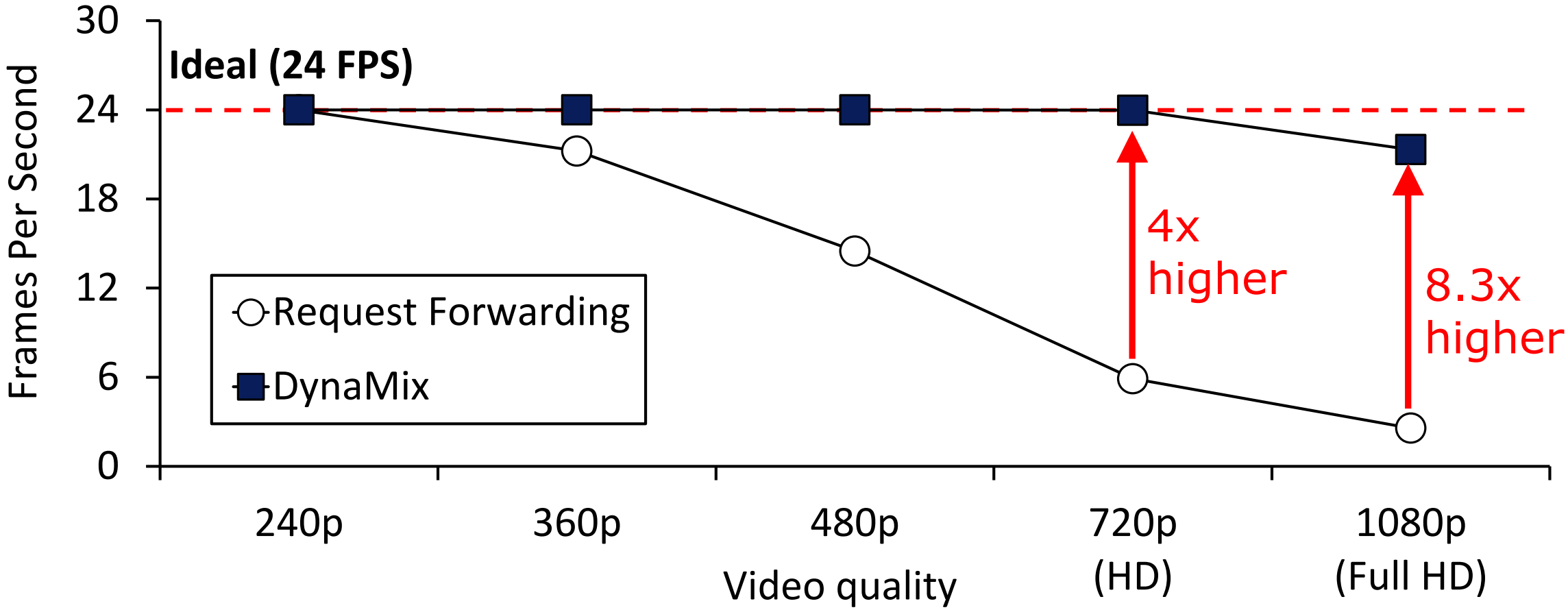
- Resource reconfiguration (local screen → remote large screen)



Throughput Improvement

- Measured FPS for the home theater scenario

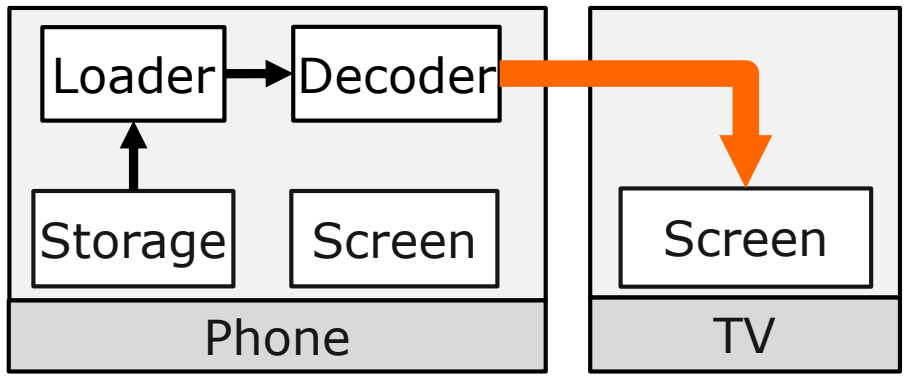
- **Achieve (or closely) the target performance goal**



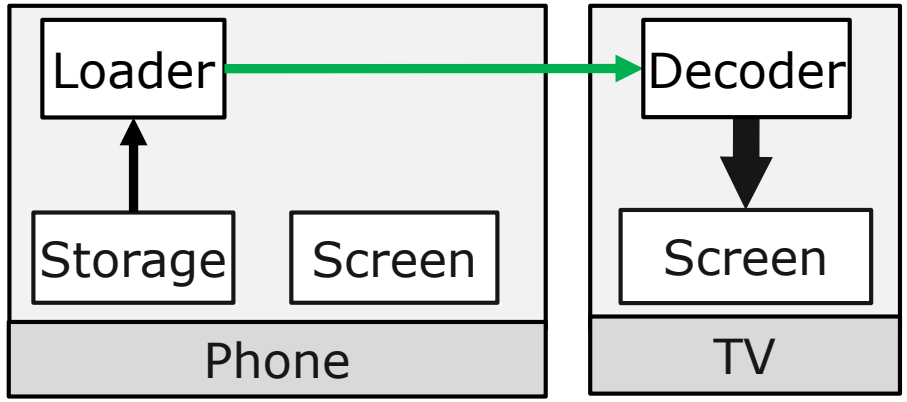
Minimized Network Stall Time

- Per-frame latency analysis for each datapath

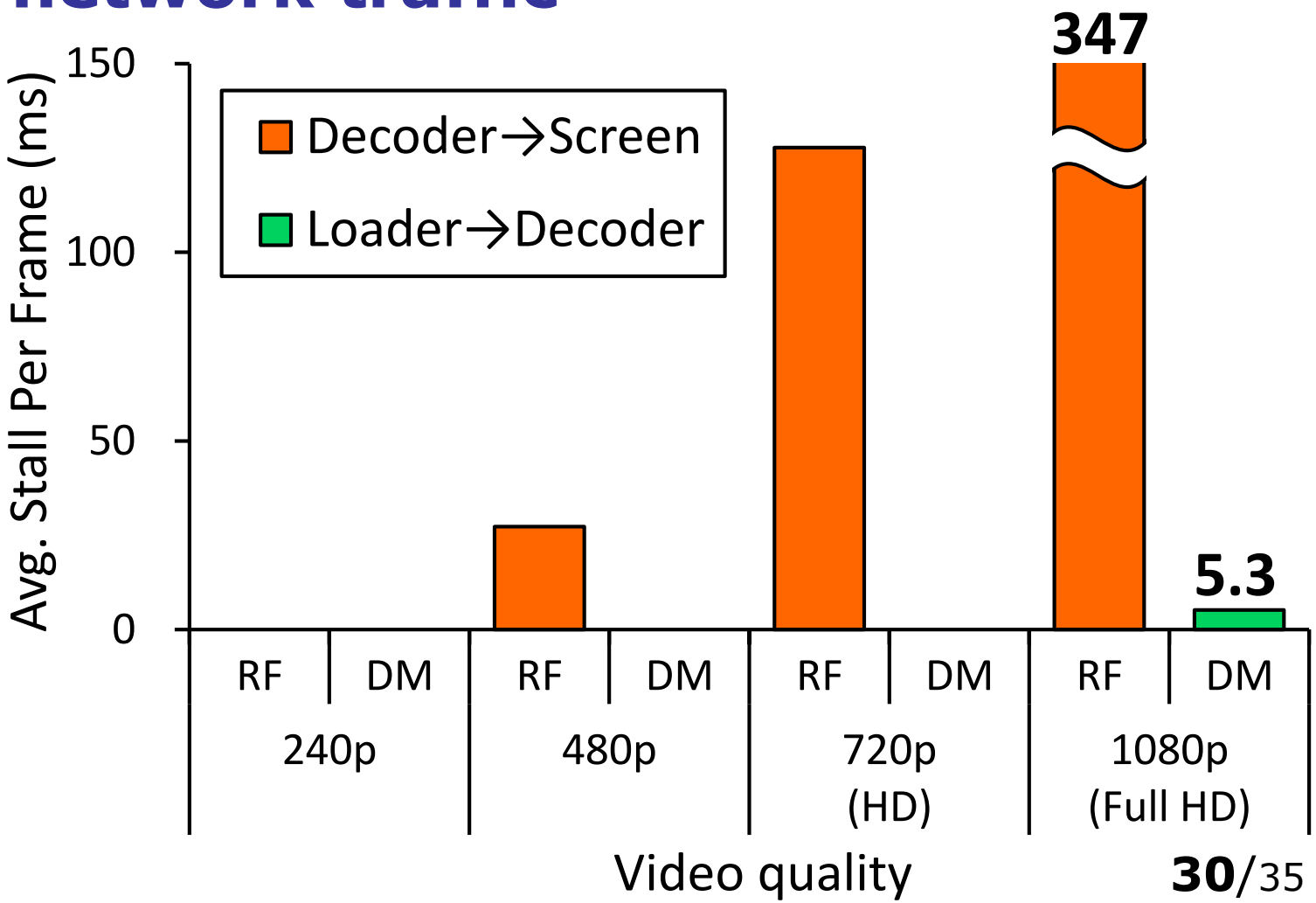
- **Reduce the exposed network traffic**



Request Forwarding (RF)



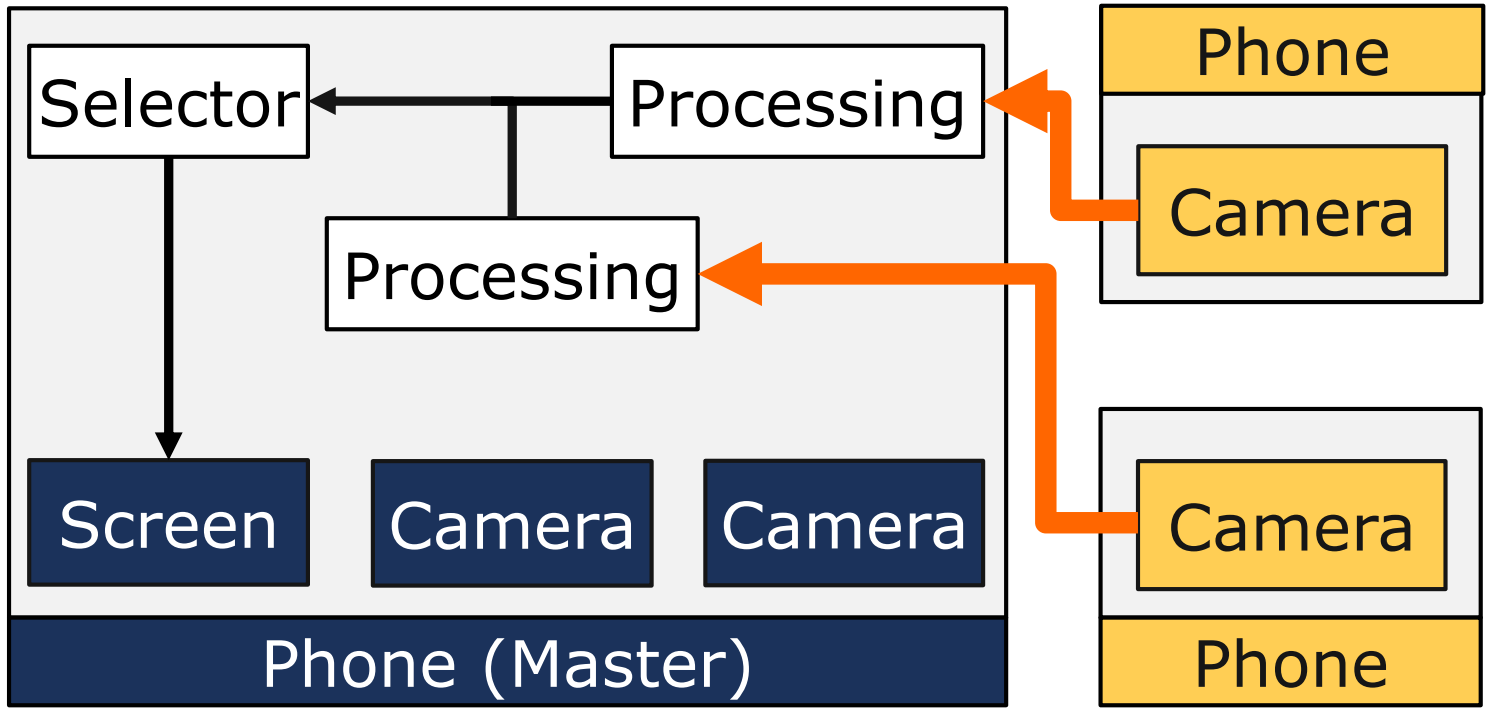
DynaMix (DM)



Evaluation Setup: Smart Monitoring

Smart Monitoring

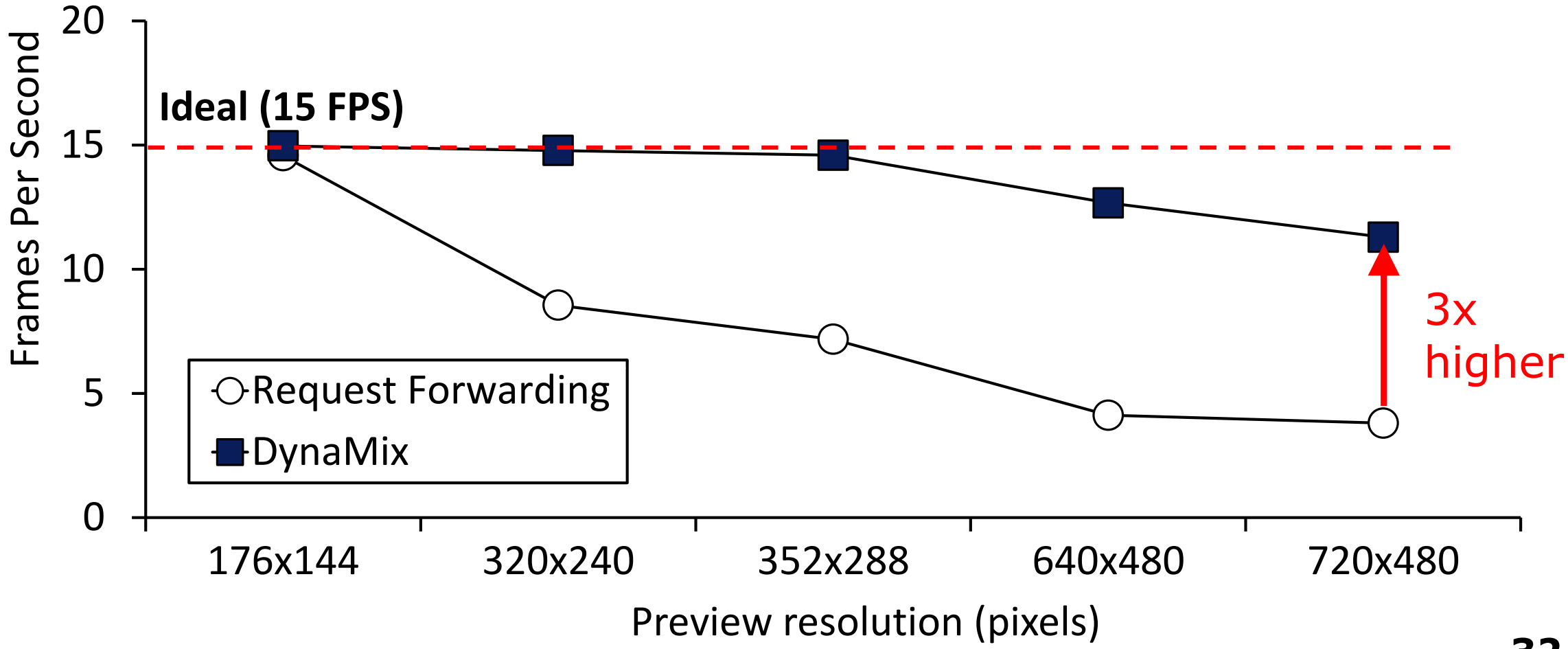
Android-based smartphones (5.1.1)
- Nexus 5 (master)
- Nexus 5 (camera) x 3



Throughput Improvement

- As the camera preview resolution increases

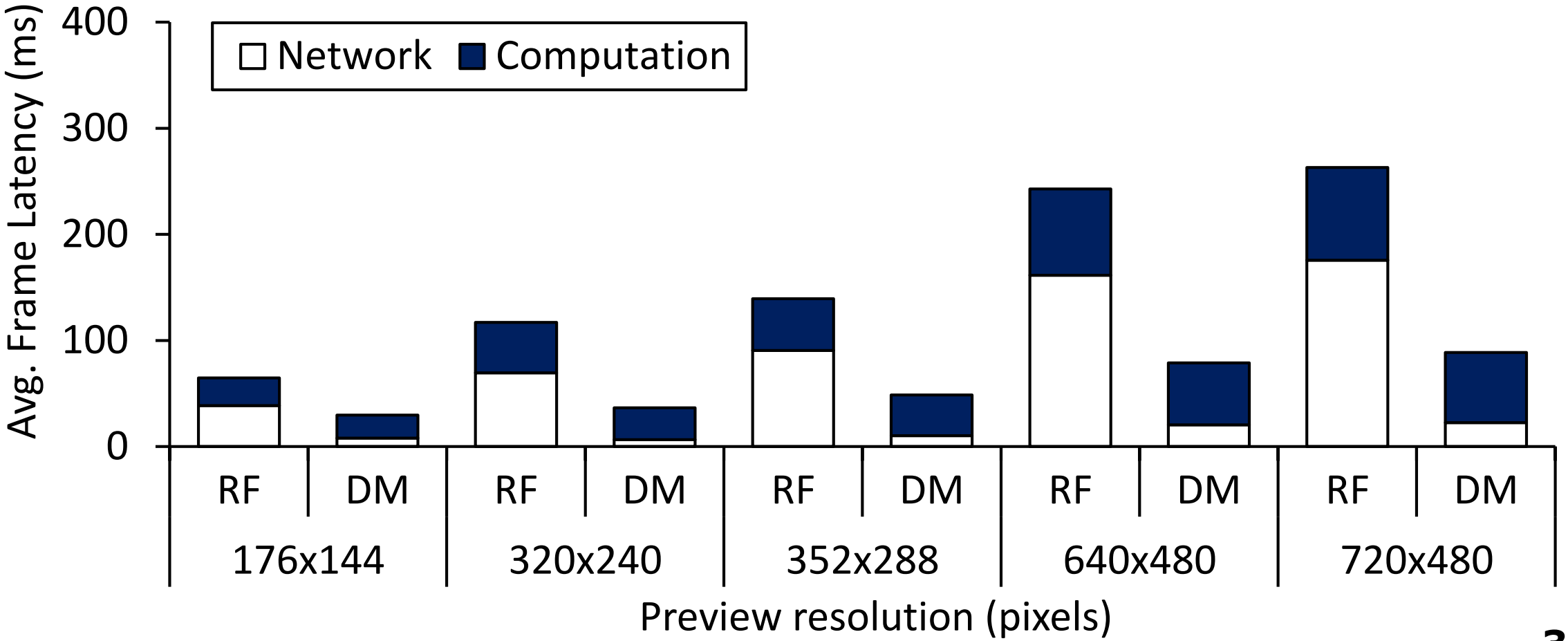
- **Achieve higher throughput than Request Forwarding**



Computation Bottleneck

- Per-frame detection latency

- **Potential to achieve higher throughput with faster CPUs**



Conclusion

- **DynaMix: Efficient cross-device resource sharing**
  - Transparent resource integration for diverse resources
  - Resource-aware dynamic task redistribution
- **Implementation on top of Android/Tizen devices**
  - Achieve target performance for multi-device services
  - e.g., 8.2 FPS → 24 FPS on the home theater scenario

DynaMix:

Dynamic Mobile Device Integration for Efficient Cross-device Resource Sharing

Thank You!

Dongju Chae
torr55@postech.ac.kr
Department of Computer Science & Engineering
Pohang University of Science and Technology (POSTECH)