# Adaptive Video Streaming for Device-to-Device Mobile Platforms

## Greedy Pull with Minimum Delay

**Students (USC):**
Joongheon Kim, Feiyu Meng, Peiyao Chen, Hilmi E. Egilmez, Dilip Bethanabhotla
**Professors (USC):**
Dr. Andreas F. Molisch, Dr. Giuseppe Caire, Dr. Michael J. Neely, Dr. Antonio Ortega

USC University of Southern California

# Outline

Push-Strategic Device-to-Device Video Streaming

Implementation Limitation of Push-Strategic Streaming

Greedy Pull with Minimum Delay

Conclusions

# Outline

Push-Strategic Device-to-Device Video Streaming

Implementation Limitation of Push-Strategic Streaming

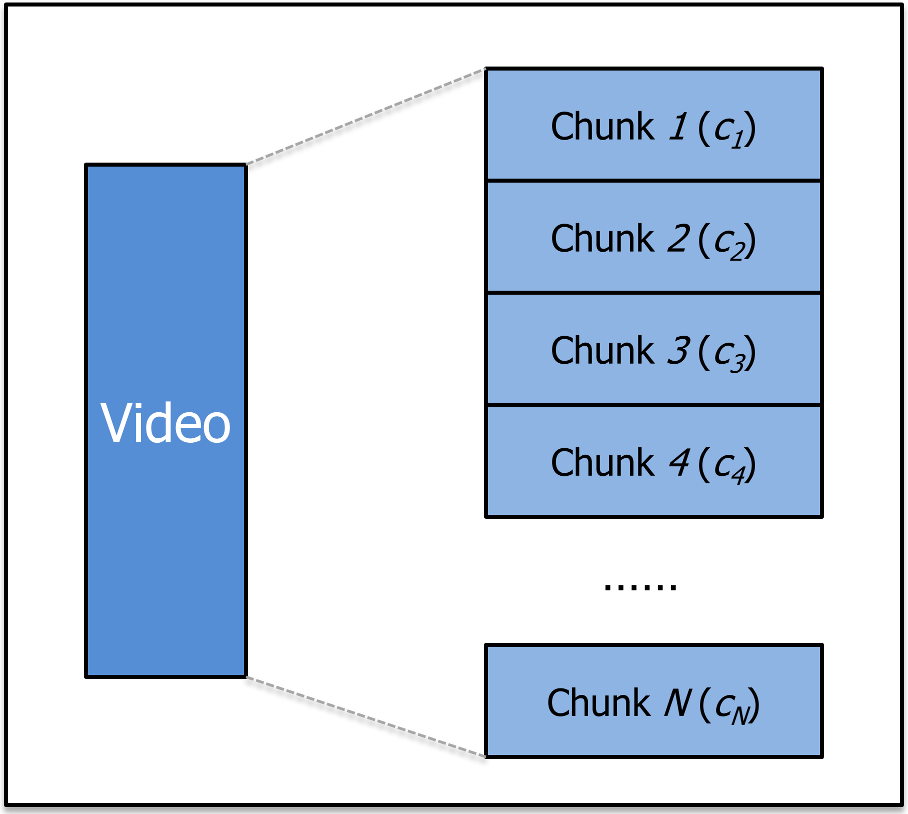
Greedy Pull with Minimum Delay

Conclusions

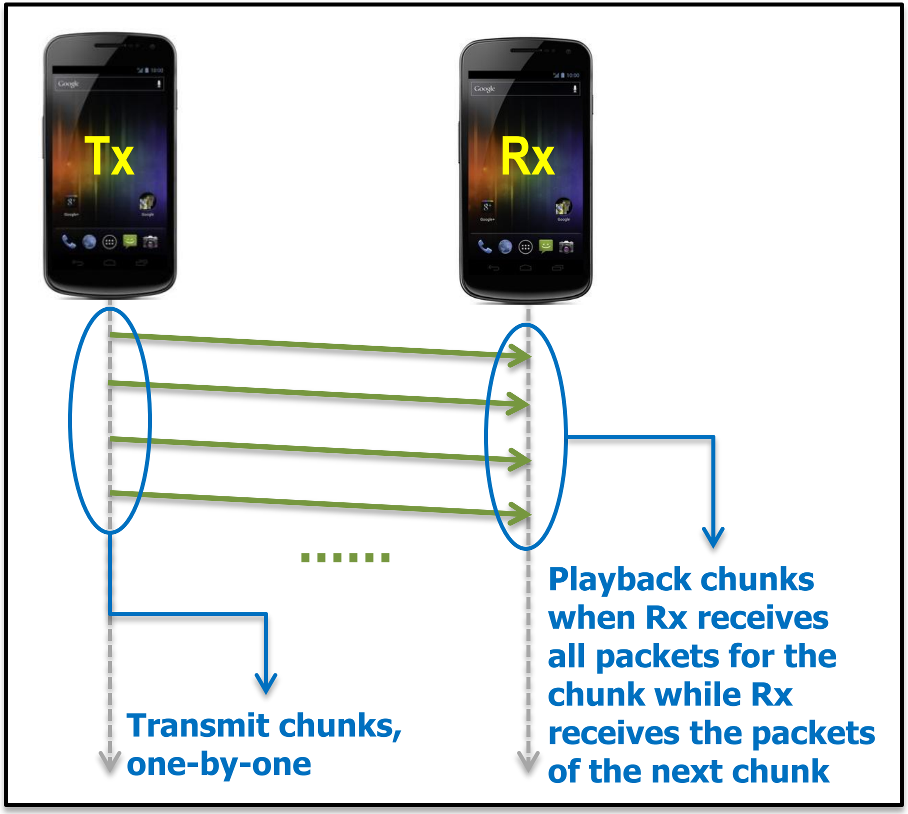
USC University of Southern California

# Video Contents

## Video Contents Data Structure



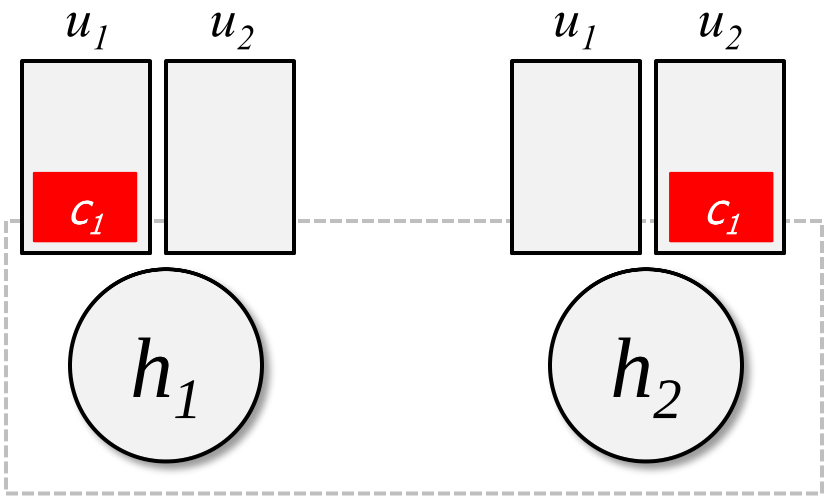
## Streaming



Transmit chunks, one-by-one

Playback chunks when Rx receives all packets for the chunk while Rx receives the packets of the next chunk

# System Model for Push Strategic D2D Video Streaming



**Helpers** (Wireless/Mobile Video Server)

- Operation: **Transmission Scheduling**
  - *Determining which user will be served*
- Maintain Multiple Queues for Individual Users

**Users** (Smartphone Users)

- Operation: **Admission Control**
  - *Determining the quality mode for each chunk based on DPP algorithm for network utility maximization*
- Download chunks from Helpers

**[Reference]** D. Bethanabhotla, G. Caire, and M. J. Neely, "**Joint Transmission Scheduling and Congestion Control for Adaptive Streaming in Wireless Device-to-Device Networks**," *Proc. Asilomar 2012*. (Journal Version: http://arxiv.org/abs/1304.8083)

# Outline

Push-Strategic Device-to-Device Video Streaming

Implementation Limitation of Push-Strategic Streaming

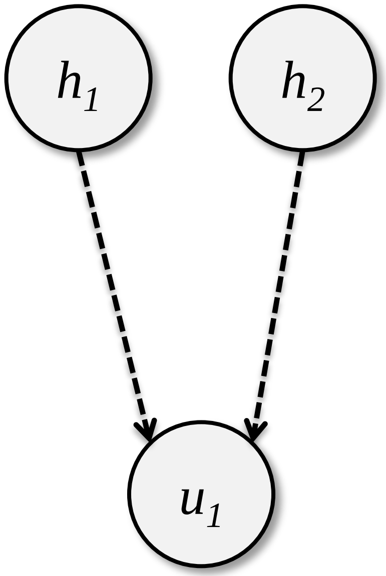
Greedy Pull with Minimum Delay

Conclusions

# Implementation Limitation

Implementation Platform:
**Android Mobile Open Platforms**

## Issue #1 (Scheduling)



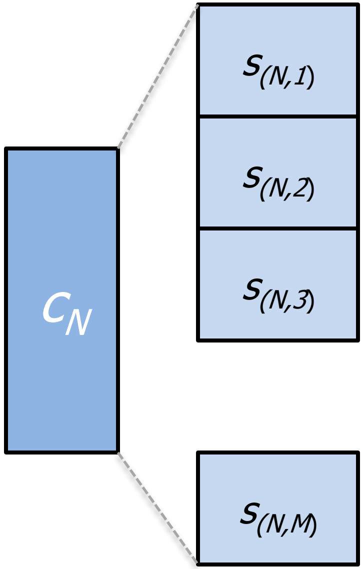
In this scenario, two helpers always select one same user.

In single-channel WiFi, the single user will always *randomly* select one helper.

Then, max-weight scheduling becomes random scheduling.

**(Solution) Greedy Pull for Minimum Delay**

## Issue #2 (Bit-by-Bit Transmission)



In the algorithm, helpers do the bit-level transmission.

But, it is not possible in Android API level.

**(Solution) Defining sub-chunks and doing the transmission in a sub-chunk unit.**

# Outline

Push-Strategic Device-to-Device Video Streaming

Implementation Limitation of Push-Strategic Streaming

Greedy Pull with Minimum Delay

Conclusions

# Basic Concept

**[Greedy Pull with Minimum Delay]**

- **Basic Concept:**

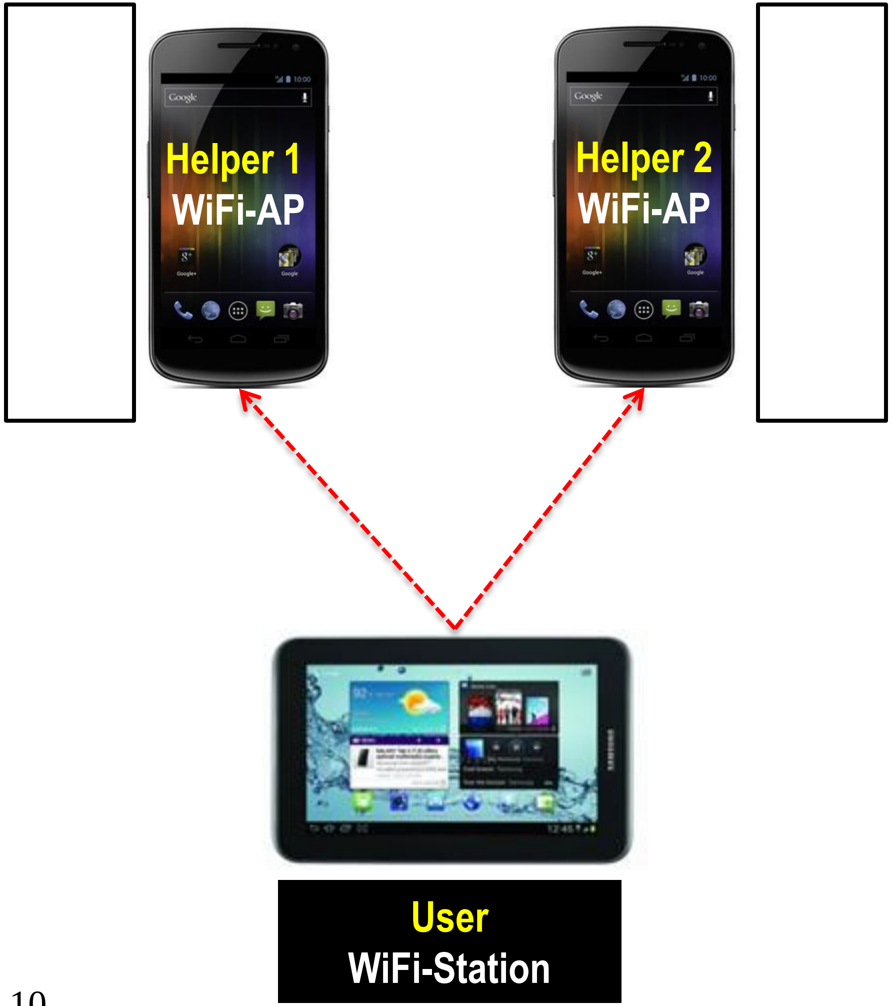
  User receives sub-chunks from helpers.
  In terms of playback order, user knows
  (1) which sub-chunk should be downloaded in next time slot
  (2) who has the sub-chunk among the given helpers.

  Thus, instead of doing transmission scheduling at helpers,
  User can ***request*** desired sub-chunk to the helper
  who has the one (**Greedy Pull for Minimum Delay**).

- **Explanation Settings:**
  - Two Helpers and One User
  - A video consists of 3 Chunks (**c1, c2, c3**) and
    Each chunk consists of 3 Sub-Chunks, i.e.,
    **s11, s12, s13, s21, s22, s23, s31, s32, s33**
    where **s23** means the 3rd subpart of 2nd chunk, etc.
    Assume that the sizes of all sub-chunks are same for
    the simplicity of explanation.

*The quality mode selection is equivalent to
Push-based algorithm. Thus, this example is not including
the function for simplicity.*

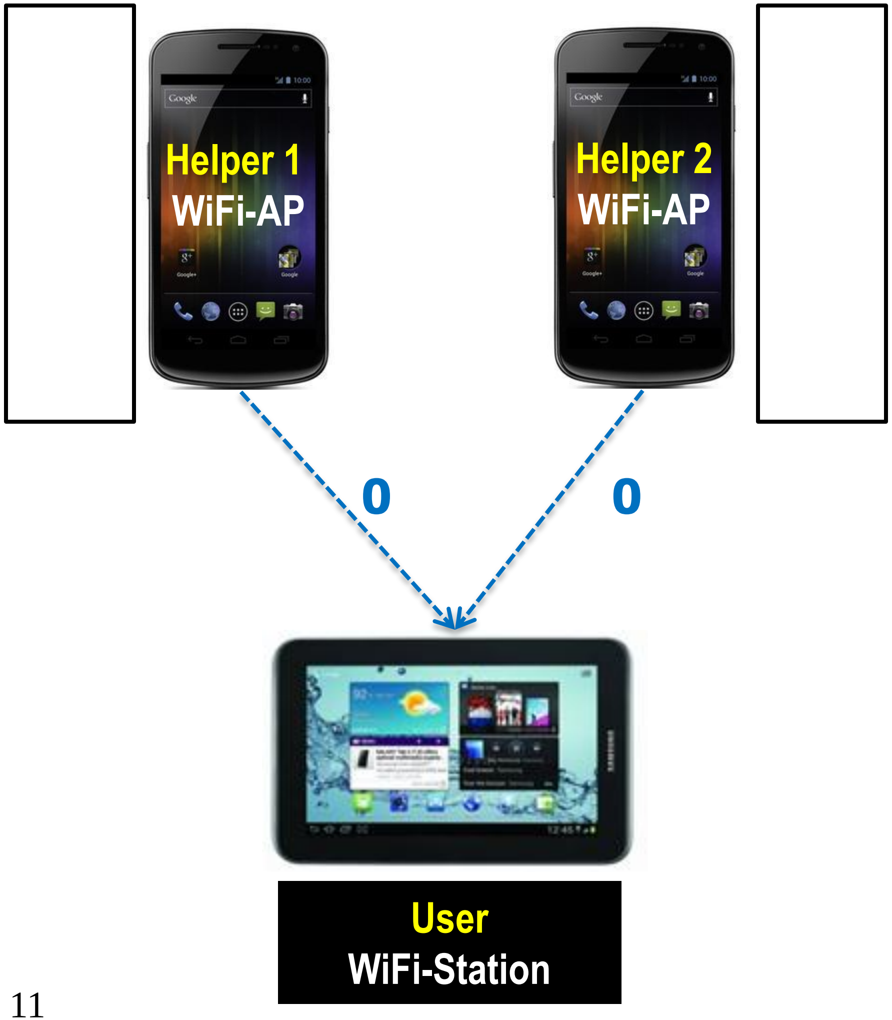
# Operational Example



**[Example-Based Explanation]**

- User requests desired video to two helpers.

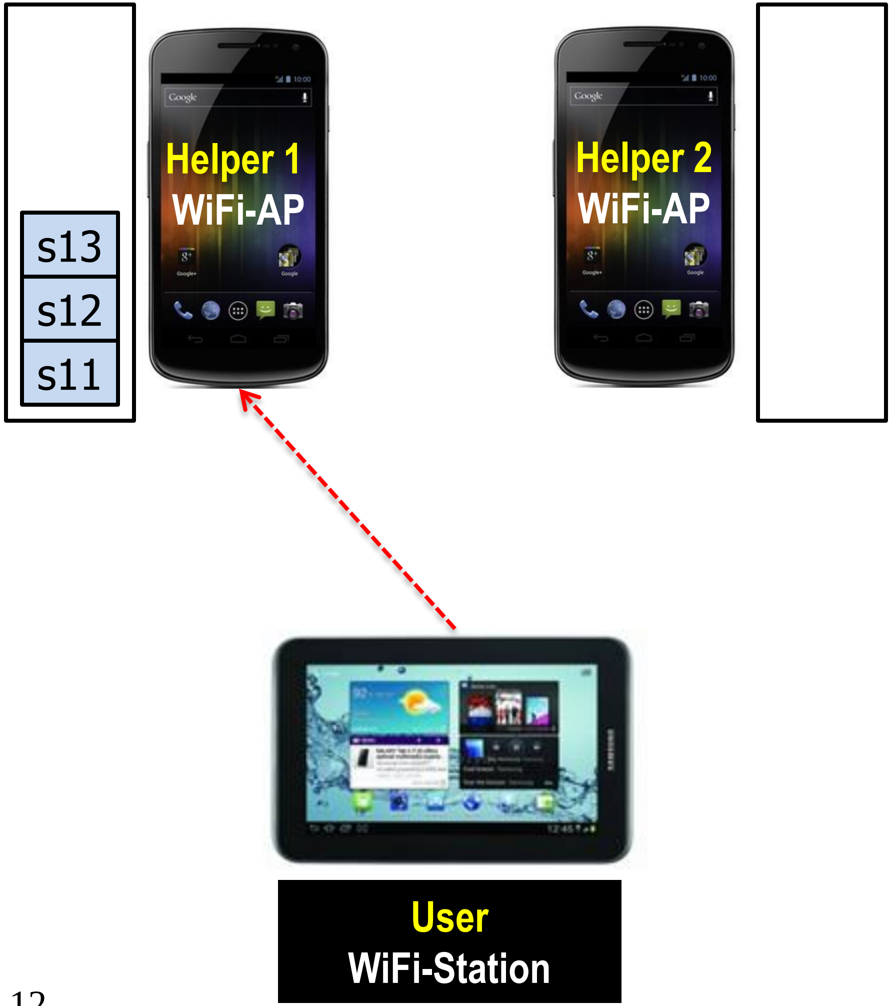
# Operational Example



**[Example-Based Explanation]**

- User requests desired video to two helpers.
- Both helpers will reply zero (which is current backlog size).

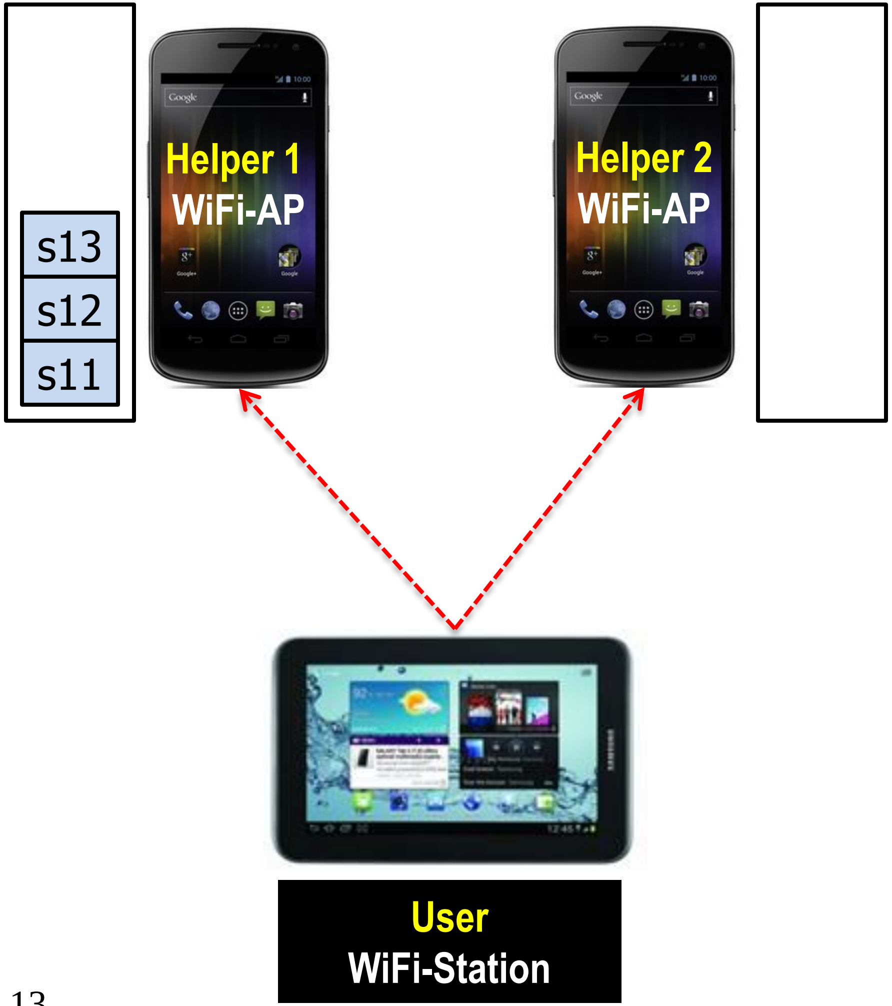
# Operational Example



**[Example-Based Explanation]**

- User requests desired video to two helpers.
- Both helpers will reply zero (which is current backlog size).
- User does the random selection (helper 1 is selected). And the user lets helper 1 know that it should place the sub-chunks of chunk 1 (i.e., s11, s12, s13).

# Operational Example

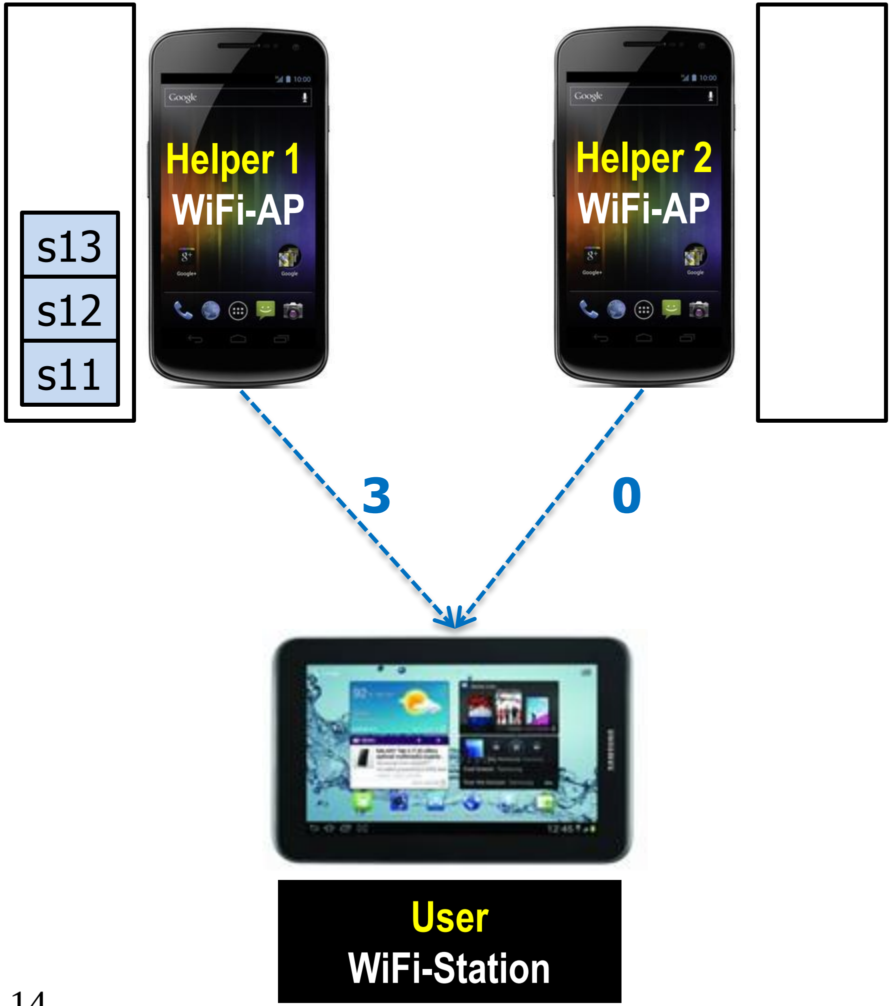


**[Example-Based Explanation]**

- User requests desired video to two helpers.
- Both helpers will reply zero (which is current backlog size).
- User does the random selection (helper 1 is selected). And the user lets helper 1 know that it should place the sub-chunks of chunk 1 (i.e., s11, s12, s13).
- User requests next chunks (i.e., c2) to two helpers.

USC University of Southern California

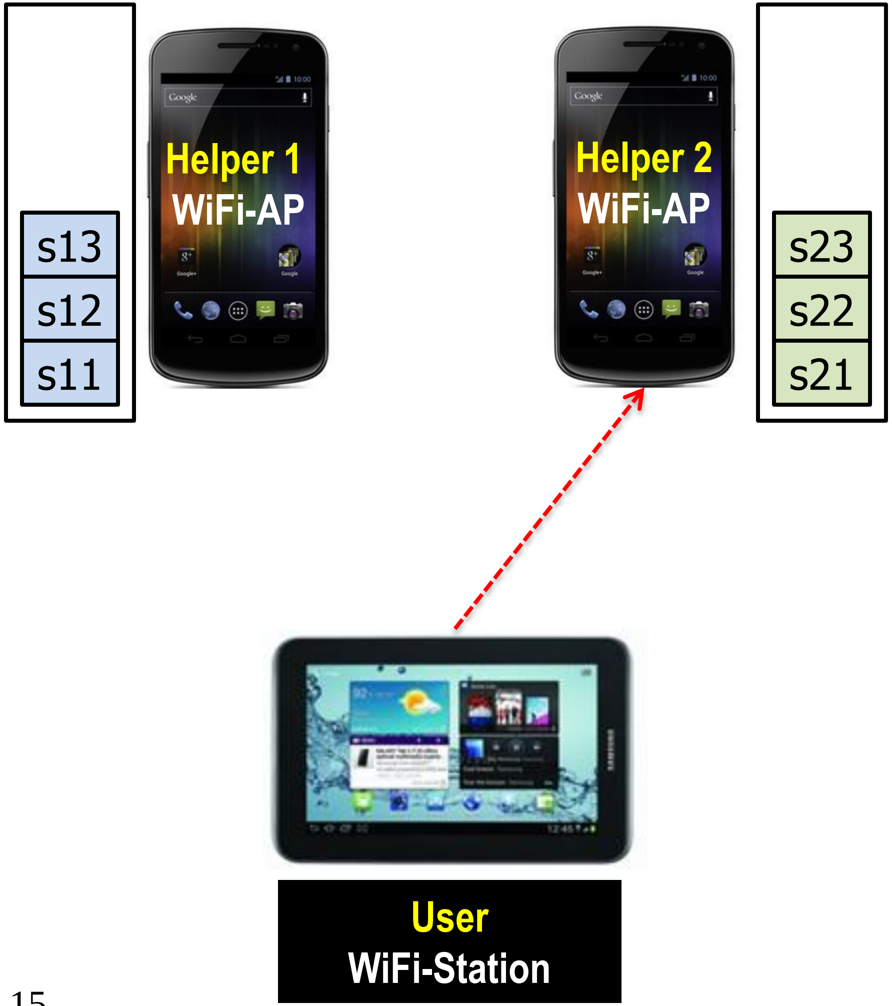
# Operational Example



**[Example-Based Explanation]**

- User requests desired video to two helpers.
- Both helpers will reply zero (which is current backlog size).
- User does the random selection (helper 1 is selected). And the user lets helper 1 know that it should place the sub-chunks of chunk 1 (i.e., s11, s12, s13).
- User requests next chunks (i.e., c2) to two helpers.
- Helper 1 will reply 3 and helper 2 will reply 0.

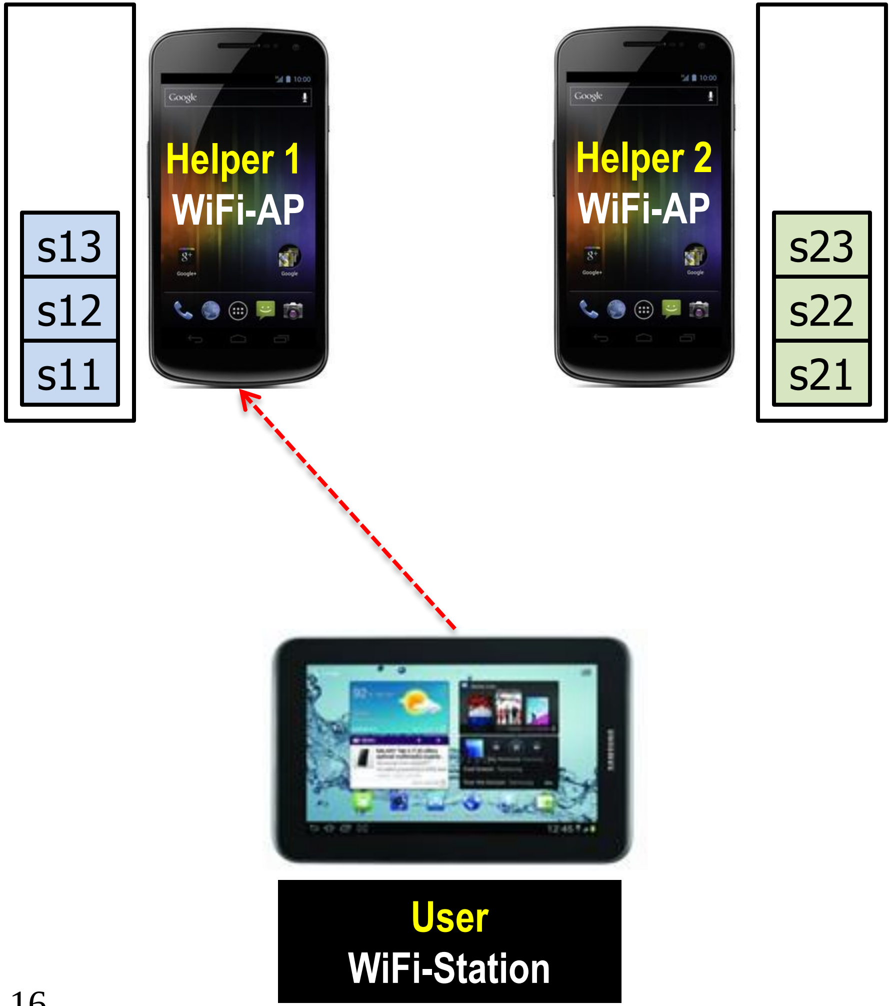
# Operational Example



**[Example-Based Explanation]**

- User requests desired video to two helpers.
- Both helpers will reply zero (which is current backlog size).
- User does the random selection (helper 1 is selected). And the user lets helper 1 know that it should place the sub-chunks of chunk 1 (i.e., s11, s12, s13).
- User requests next chunks (i.e., c2) to two helpers.
- Helper 1 will reply 3 and helper 2 will reply 0.
- User selects the one which has the smallest queue backlog size. Thus, helper 2 is selected. And the user lets helper 2 know that it should place the sub-chunks of chunk 2.

# Operational Example

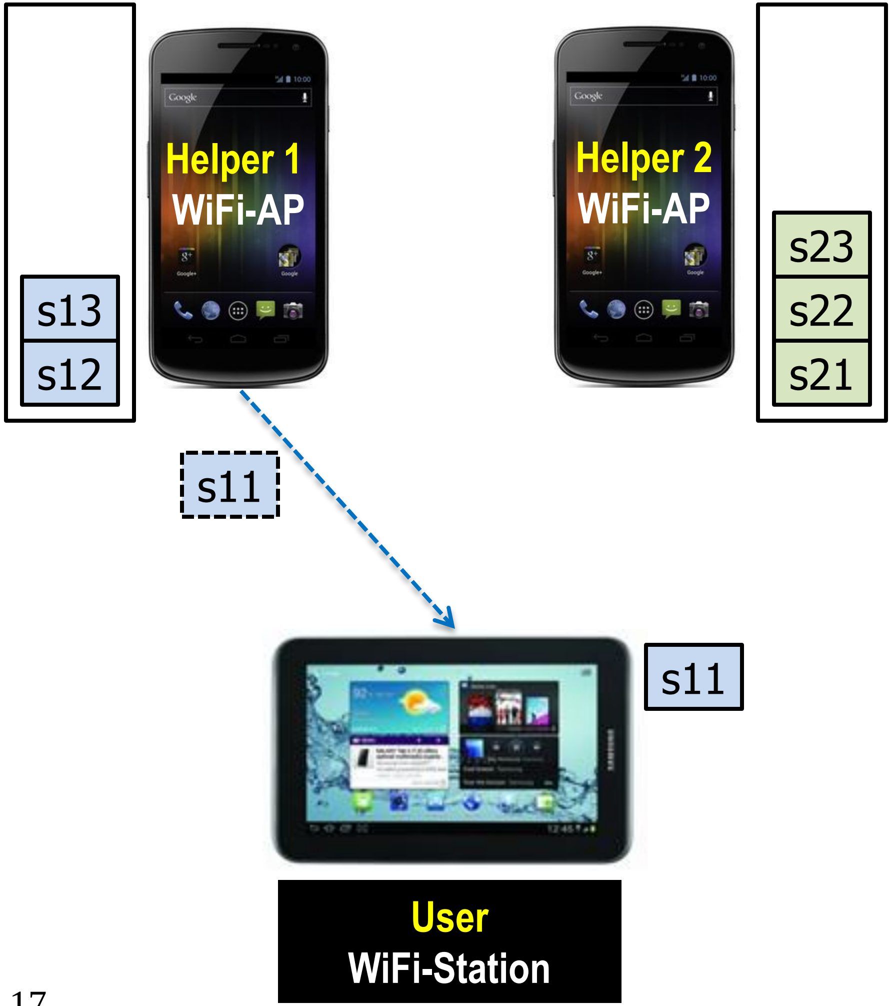


**[Example-Based Explanation]**

- User requests desired video to two helpers.
- Both helpers will reply zero (which is current backlog size).
- User does the random selection (helper 1 is selected). And the user lets helper 1 know that it should place the sub-chunks of chunk 1 (i.e., s11, s12, s13).
- User requests next chunks (i.e., c2) to two helpers.
- Helper 1 will reply 3 and helper 2 will reply 0.
- User selects the one which has the smallest queue backlog size. Thus, helper 2 is selected. And the user lets helper 2 know that it should place the sub-chunks of chunk 2.

- Now, instead of doing transmission scheduling, User selects helper 1 because user needs **s11** in terms of playback order and user knows that helper 1 has the one (**Greedy Pull for Minimum Delay**).

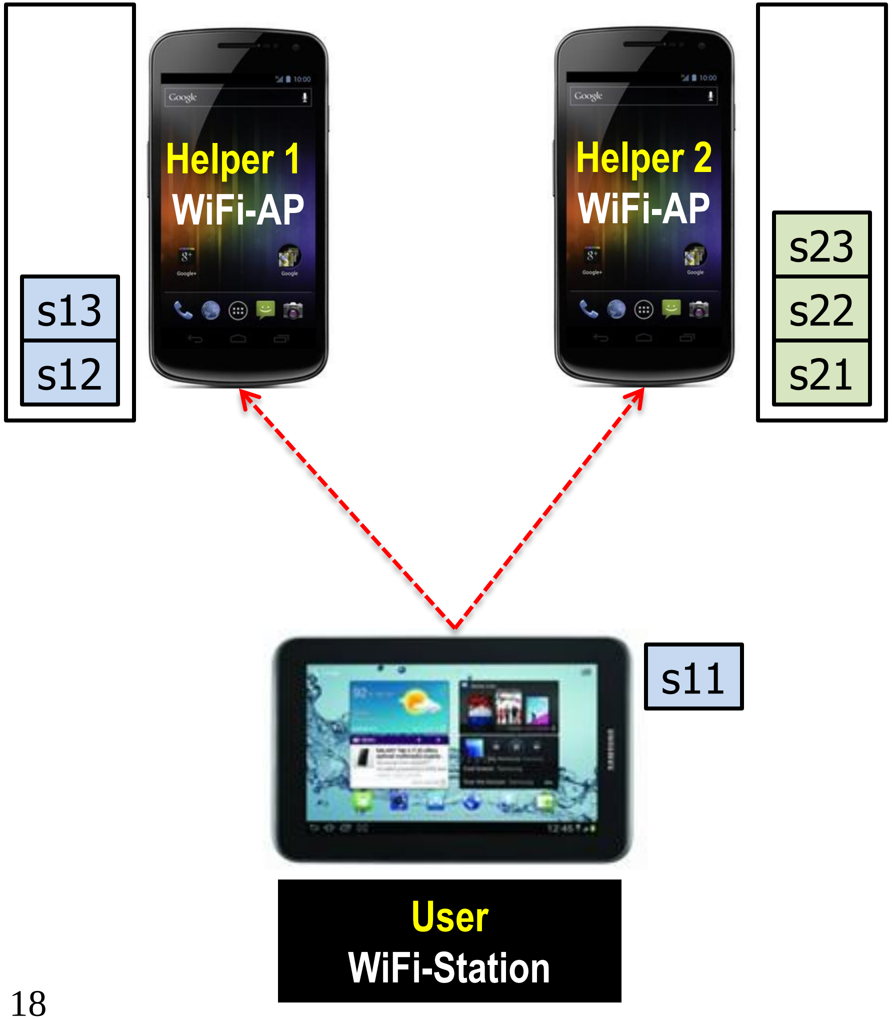
# Operational Example



**[Example-Based Explanation]**

- Helper 1 transmits sub-chunks for the given time. If RSSI is good, then it can transmit more. Now, suppose that the channel is bad, so, helper 1 can transmit only one, i.e., s11.

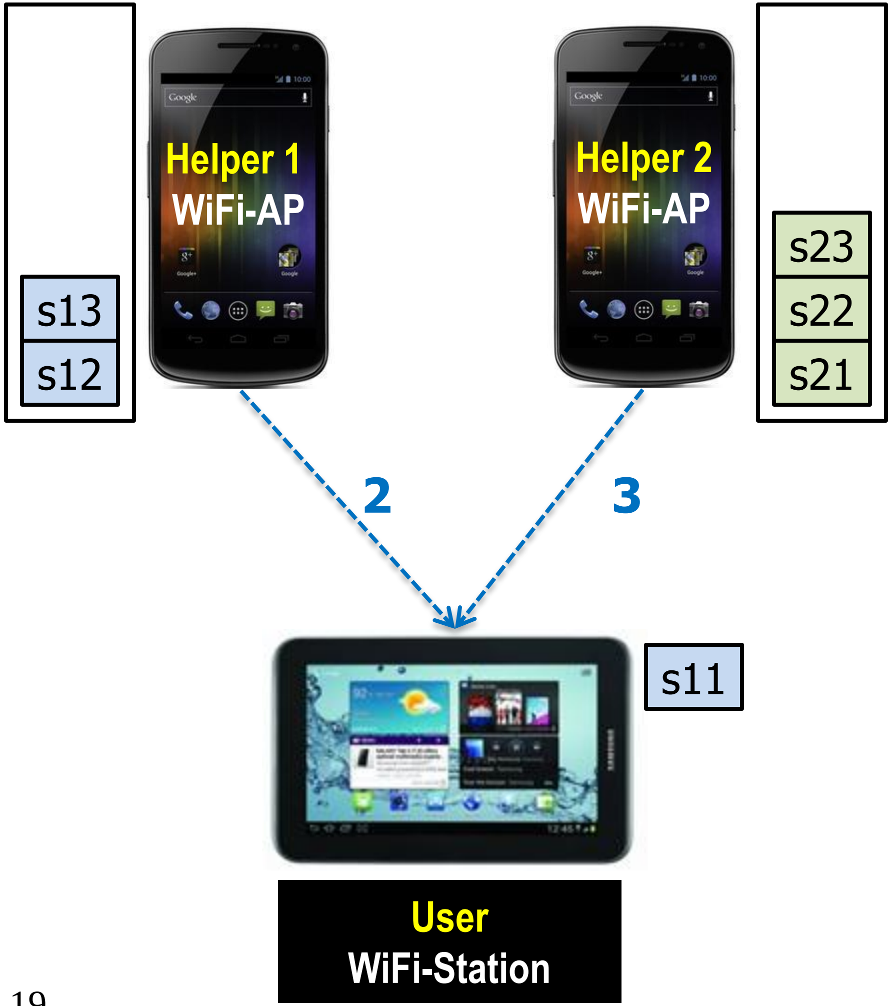
# Operational Example



**[Example-Based Explanation]**

- Helper 1 transmits sub-chunks for the given time. If RSSI is good, then it can transmit more. Now, suppose that the channel is bad, so, helper 1 can transmit only one, i.e., s11.
- User requests next chunk (i.e., c3) to two helpers.

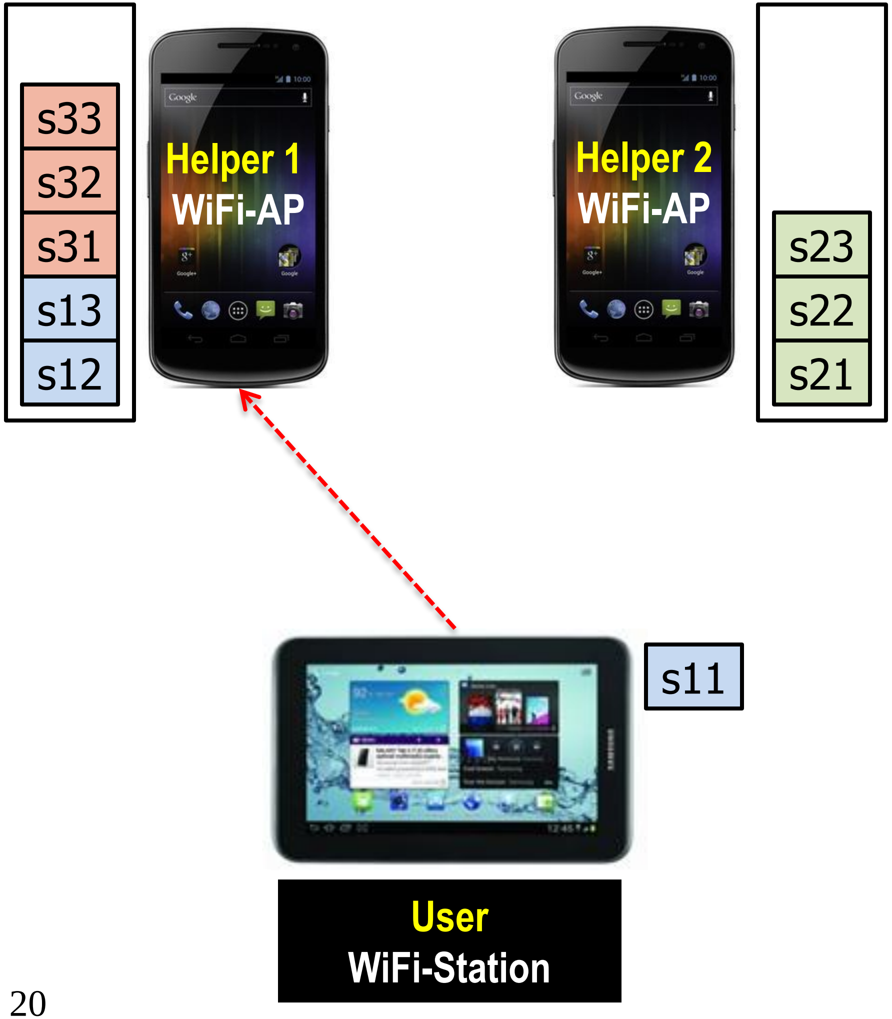
# Operational Example



**[Example-Based Explanation]**

- Helper 1 transmits sub-chunks for the given time. If RSSI is good, then it can transmit more. Now, suppose that the channel is bad, so, helper 1 can transmit only one, i.e., s11.
- User requests next chunk (i.e., c3) to two helpers.
- Helper 1 will reply 2 and helper 2 will reply 3.

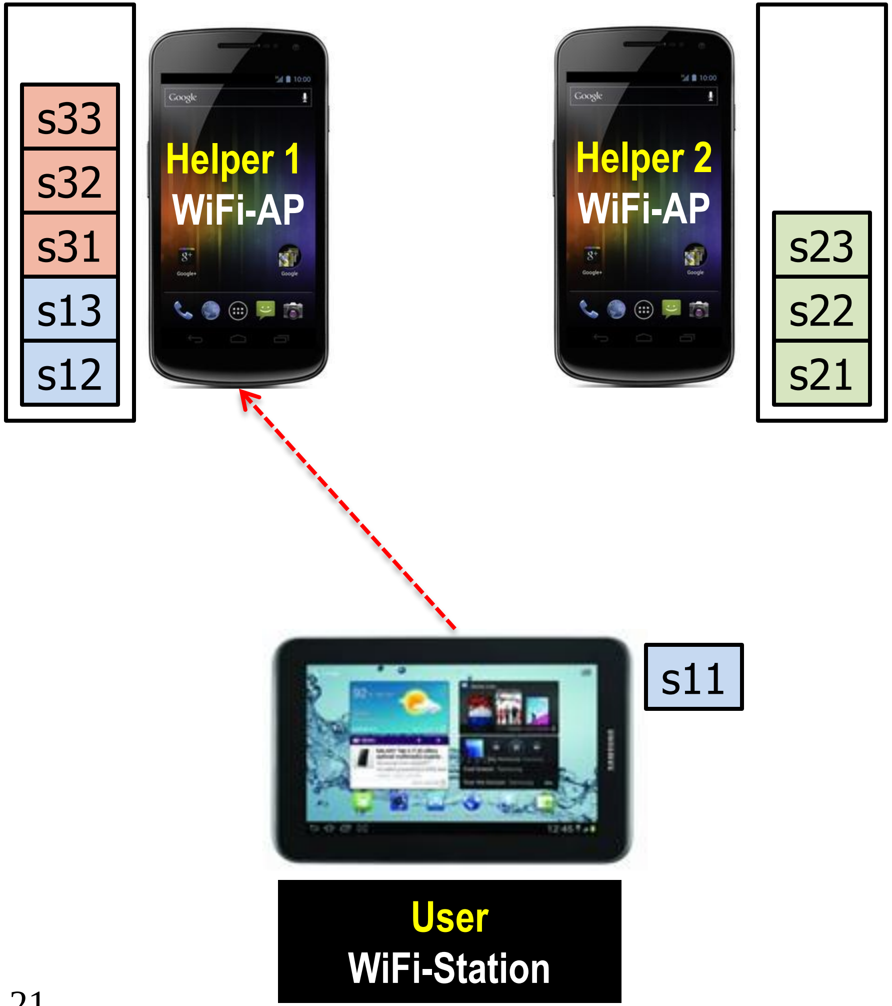
# Operational Example



**[Example-Based Explanation]**

- Helper 1 transmits sub-chunks for the given time. If RSSI is good, then it can transmit more. Now, suppose that the channel is bad, so, helper 1 can transmit only one, i.e., s11.
- User requests next chunk (i.e., c3) to two helpers.
- Helper 1 will reply 2 and helper 2 will reply 3.
- User selects the one which has the smallest queue backlog size. Thus, helper 1 is selected. And the user lets helper 1 know that it should place the sub-chunks of chunk 3.

# Operational Example



s33
s32
s31
s13
s12

Helper 1
WiFi-AP

Helper 2
WiFi-AP

s23
s22
s21

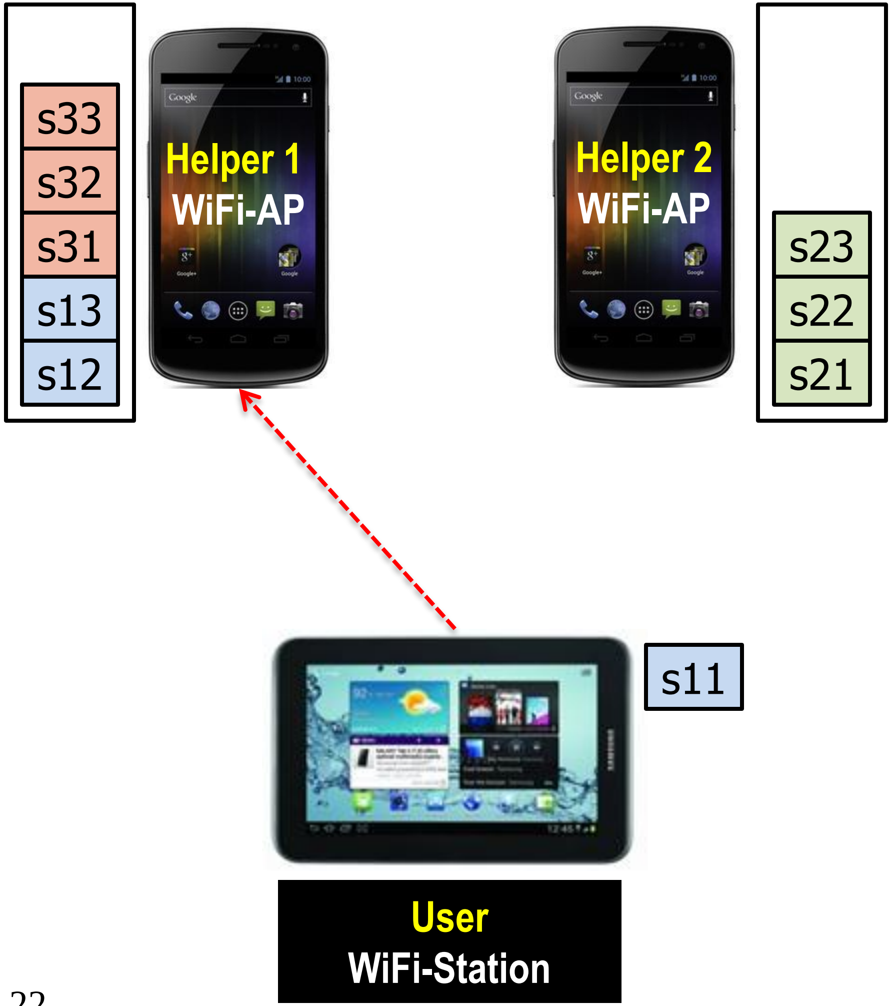
s11

User
WiFi-Station

**[Example-Based Explanation]**

- Helper 1 transmits sub-chunks for the given time. If RSSI is good, then it can transmit more. Now, suppose that the channel is bad, so, helper 1 can transmit only one, i.e., s11.
- User requests next chunk (i.e., c3) to two helpers.
- Helper 1 will reply 2 and helper 2 will reply 3.
- User selects the one which has the smallest queue backlog size. Thus, helper 1 is selected. And the user lets helper 1 know that it should place the sub-chunks of chunk 3.

# Operational Example

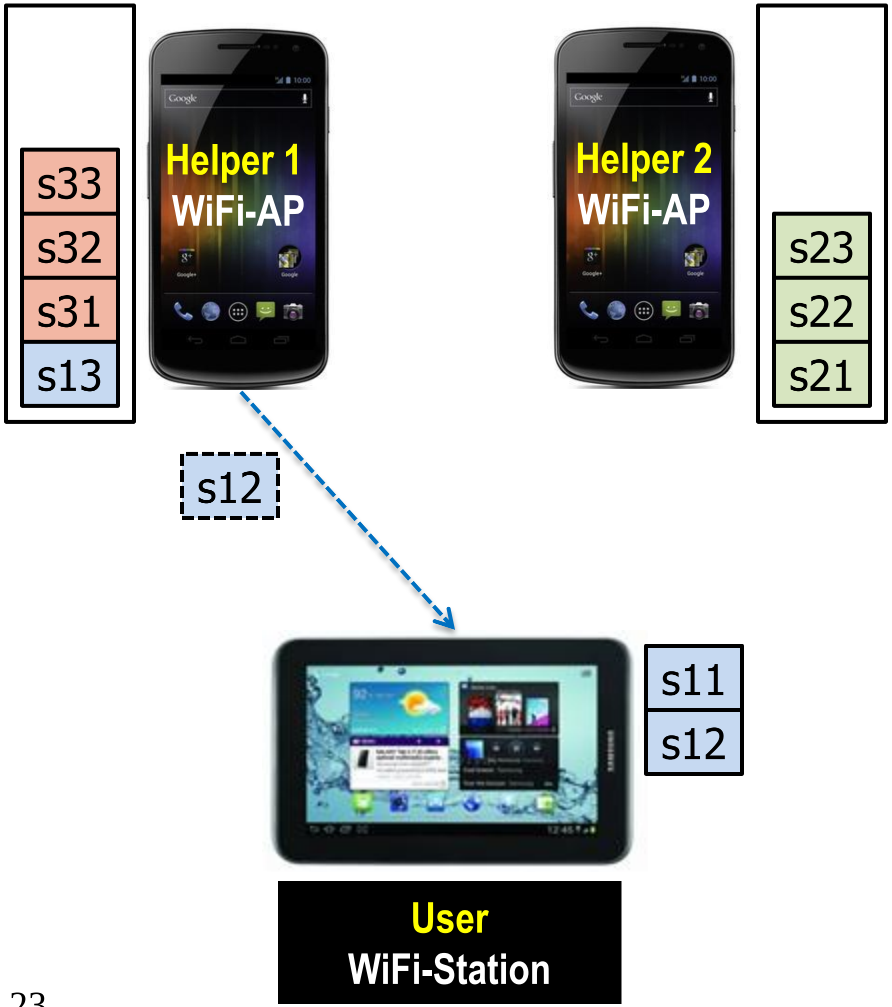


**[Example-Based Explanation]**

- Helper 1 transmits sub-chunks for the given time. If RSSI is good, then it can transmit more. Now, suppose that the channel is bad, so, helper 1 can transmit only one, i.e., s11.
- User requests next chunk (i.e., c3) to two helpers.
- Helper 1 will reply 2 and helper 2 will reply 3.
- User selects the one which has the smallest queue backlog size. Thus, helper 1 is selected. And the user lets helper 1 know that it should place the sub-chunks of chunk 3.

- Now, instead of doing transmission scheduling, User selects helper 1 again because user needs s12 in terms of playback order and user knows that helper 1 has the one (**Greedy Pull for Minimum Delay**).

# Operational Example



s33
s32
s31
s13

Helper 1
WiFi-AP

Helper 2
WiFi-AP

s23
s22
s21

s12

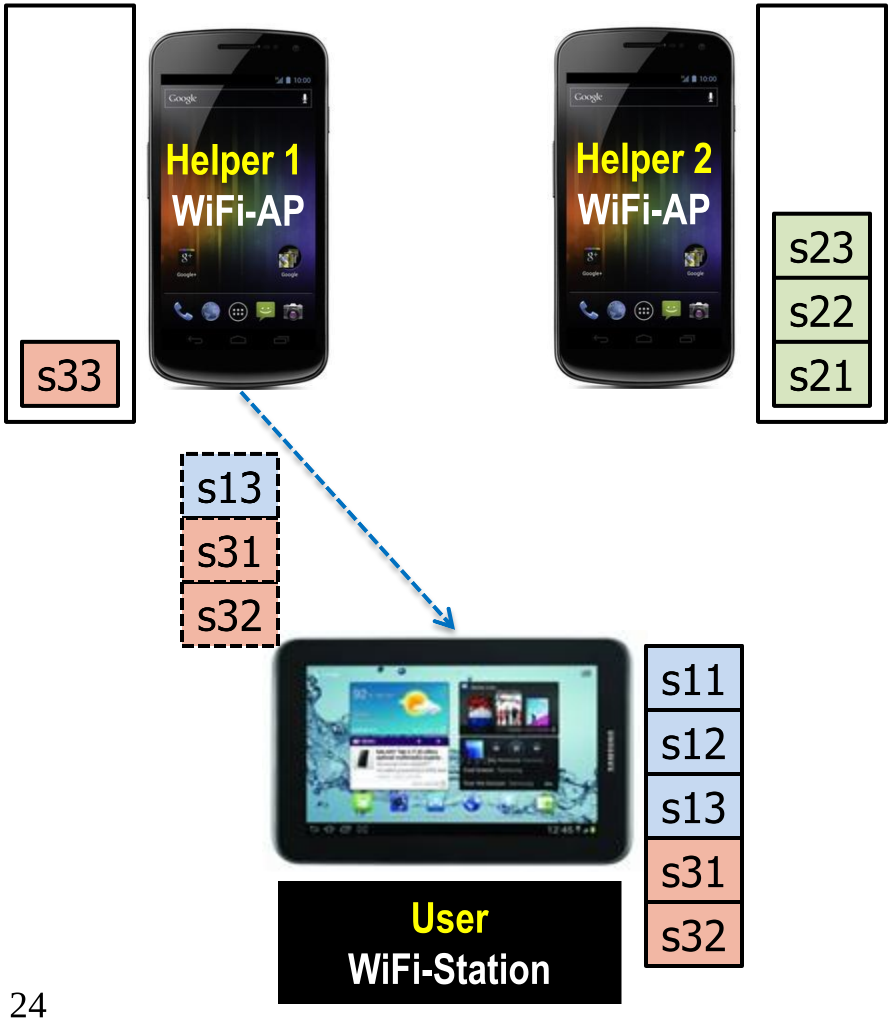
s11
s12

User
WiFi-Station

**[Example-Based Explanation]**

- Helper 1 transmits sub-chunks for the given time. If RSSI is good, then it can transmit more. Now, suppose that the channel is bad, so, helper 1 can transmit only one, i.e., s12.

# Operational Example

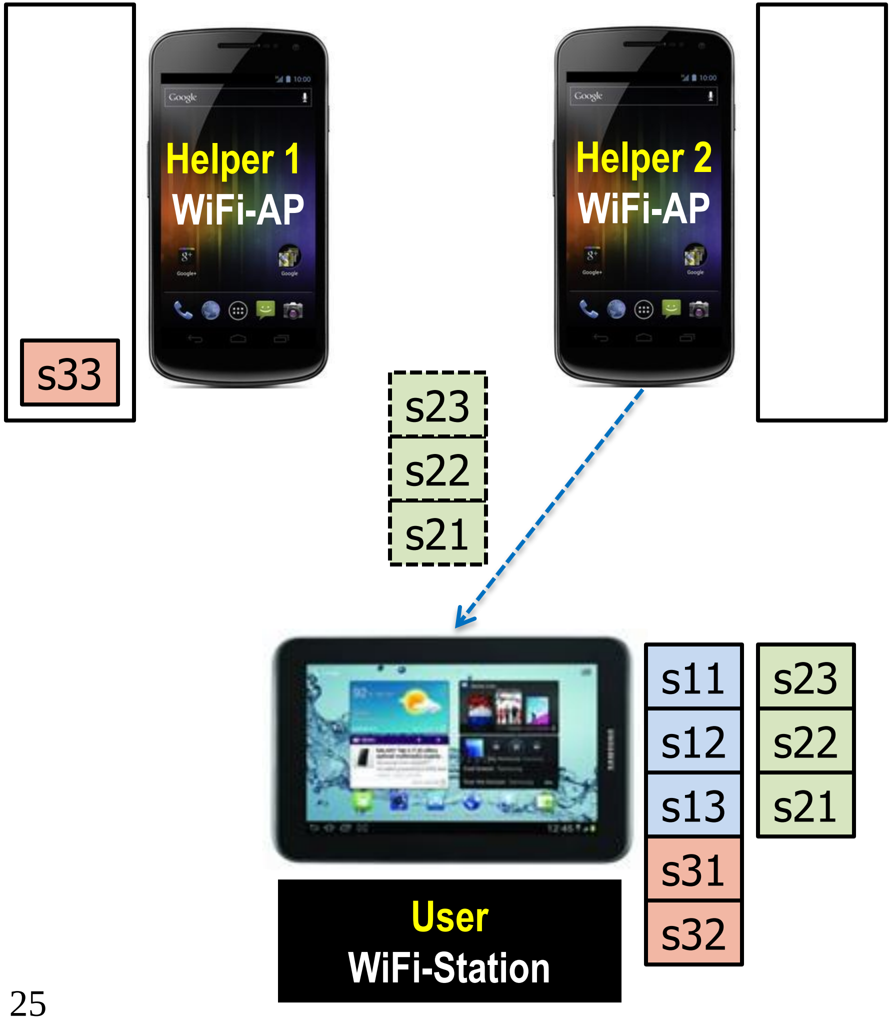


**[Example-Based Explanation]**

- Helper 1 transmits sub-chunks for the given time. If RSSI is good, then it can transmit more. Now, suppose that the channel is bad, so, helper 1 can transmit only one, i.e., s12.
- (User doesn't request chunks because c3 was the last one)

- Now, instead of doing transmission scheduling, User selects helper 1 again because user needs s13 in terms of playback order and user knows that helper 1 has the one (**Greedy Pull for Minimum Delay**).
- Helper 1 transmits sub-chunks for the given time. If RSSI is good, then it can transmit more. Now, suppose that the channel is good, so, helper 1 can transmit three.

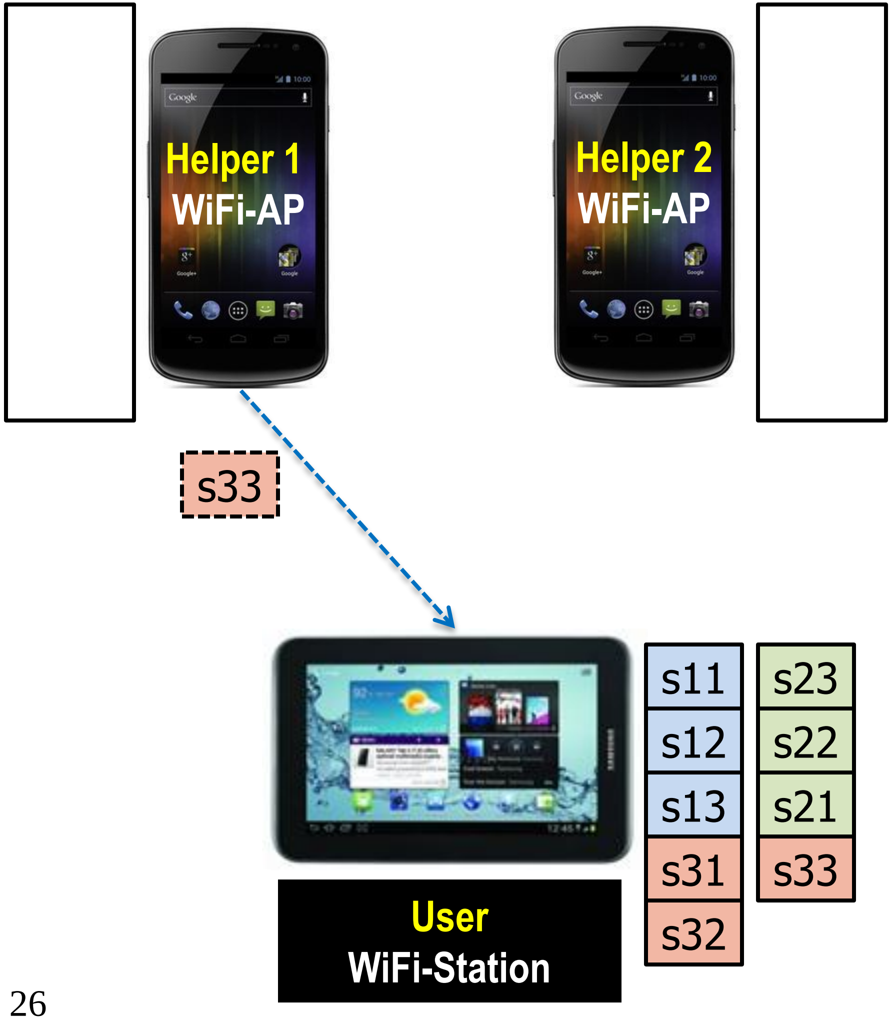
# Operational Example



**[Example-Based Explanation]**

- Now, instead of doing transmission scheduling, User selects helper 2 because user needs **s21** in terms of playback order and user knows that helper 2 has the one (**Greedy Pull for Minimum Delay**).
- Helper 2 transmits sub-chunks for the given time. If RSSI is good, then it can transmit more. Now, suppose that the channel is good, so, helper 2 can transmit three.

# Operational Example



**[Example-Based Explanation]**

- Now, instead of doing transmission scheduling, User selects helper 1 because user needs **s33** in terms of playback order and user knows that helper 1 has the one (**Greedy Pull for Minimum Delay**).
- Helper 1 transmits sub-chunks for the given time. If RSSI is good, then it can transmit more. Now, suppose that the channel is good, so, helper 1 can transmit the all of remaining sub-chunks.

# Outline

Push-Strategic Device-to-Device Video Streaming

Implementation Limitation of Push-Strategic Streaming

Greedy Pull with Minimum Delay

Conclusions

# Conclusions

- Design and Implementation of D2D Video Streaming on top of Android
- Modifying Current Scheme for the Implementation
- Designing a New Scheme for Downloading Chunks from Different Helpers