Securing Your Journey
to the Cloud

Taxonomy of Differential Compression

LIWEI REN, PH.D, Trend Micro™

SNIA SDC 2015, Santa Clara, California, Sept, 2015

Background

- **Liwei Ren, Ph.D**
  - Research interests
    - Data security, network security, data compression, math modeling & algorithms.
  - Major works
    - 10+ academic papers;
    - 20+ US patents granted, and a few more pending;
    - Co-founded a data security company in Silicon Valley with successful exit.
  - Education
    - MS/BS in mathematics, Tsinghua University, Beijing
    - Ph.D in mathematics, MS in information science, University of Pittsburgh
- **Trend Micro™**
  - Global security software company with headquarter in Tokyo, and R&D centers in Silicon Valley, Nanjing and Taipei;
  - One of top security software vendors.
  - A leader in cloud security.

Agenda

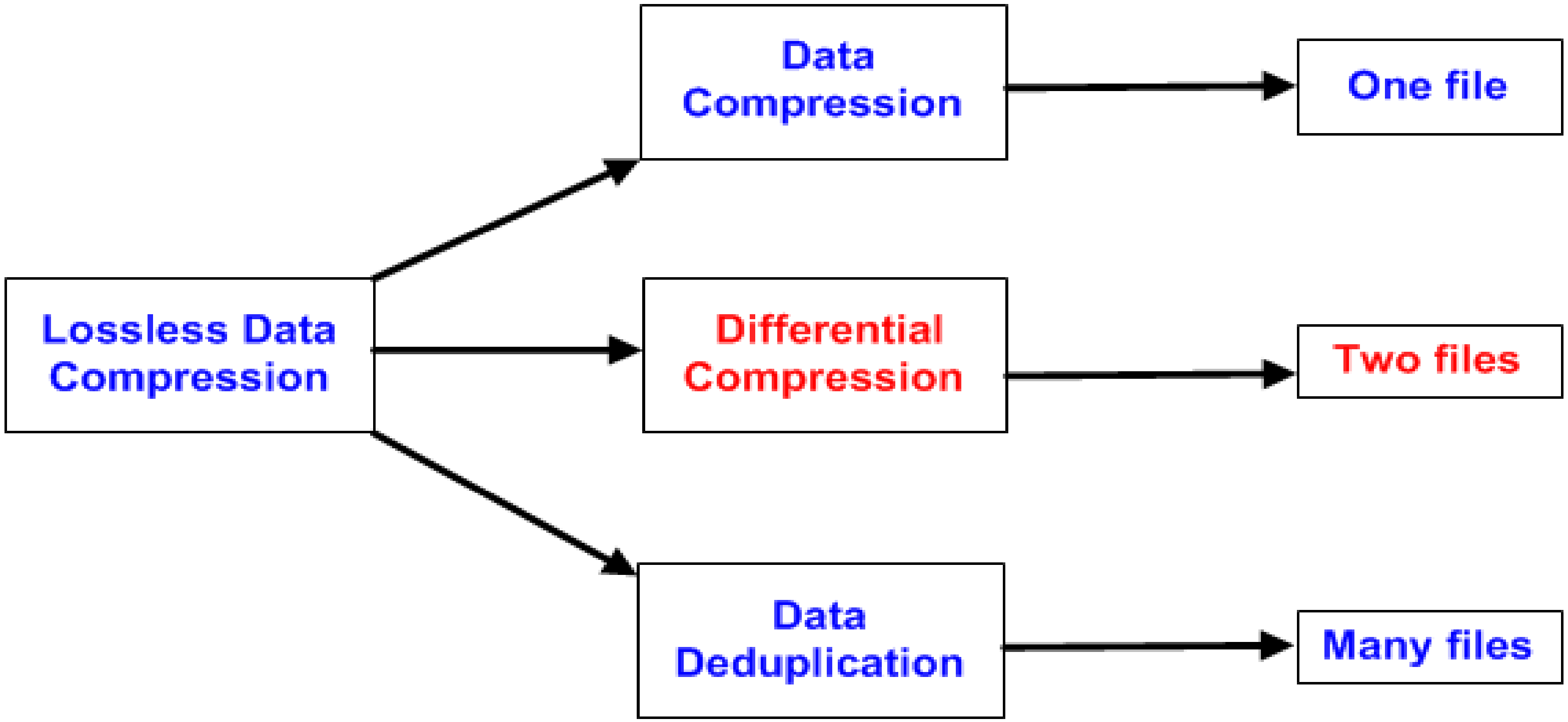
- Introduction
- A Math Model for Describing File Differences
- Categorizing Differential Compression
- Advanced Topics
- Summary

Introduction

- **Objectives for this sharing:**
  - Understand what differential compression is AND its applications.
  - Learn a mathematical model for describing differential compression
  - Know categories of differential compression
  - Be aware of a few advanced topics

Introduction

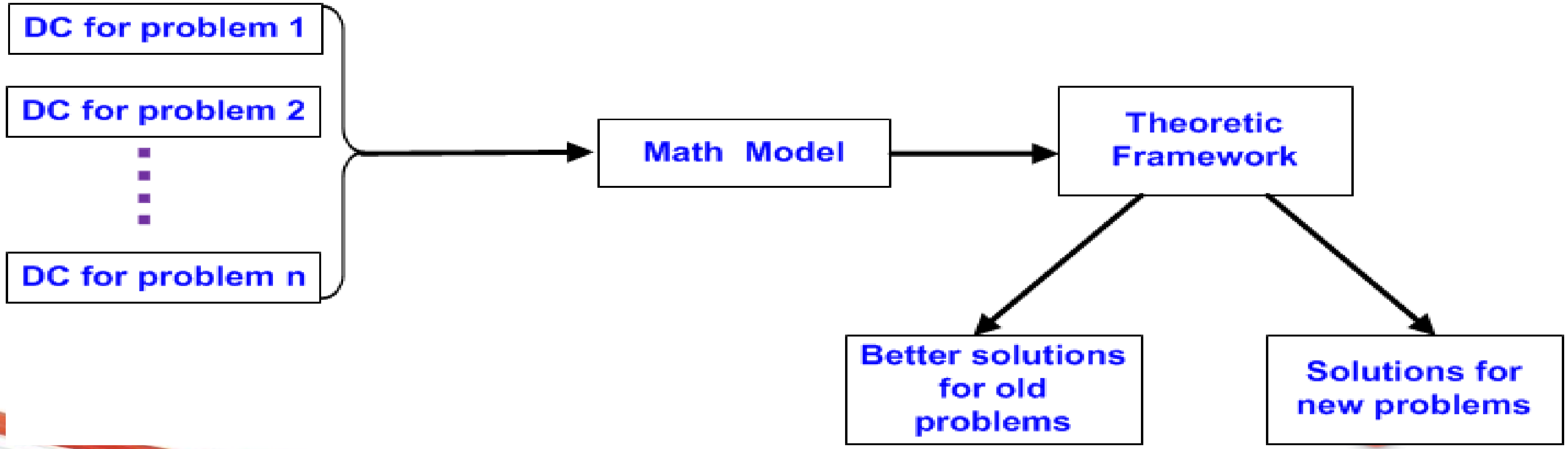
- Lossless data compression --- three categories



- Two purposes:
  - Network data transfer acceleration
  - Storage space reduction

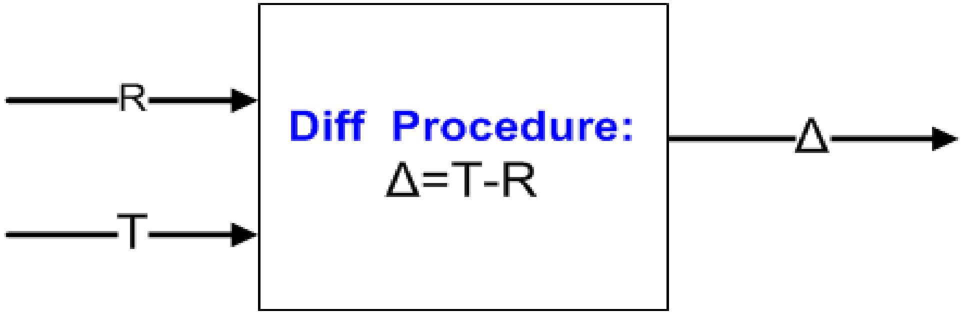
Introduction

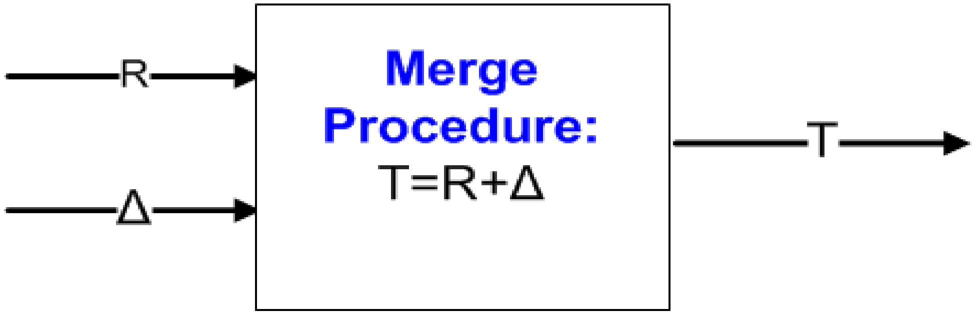
- **Today we talk about *Differential Compression* (DC).**
- **Why do I talk about differential compression?**
  - I have been designing various algorithms for differential compression since 2002 for a few domains:
    - FOTA (Firmware Over The Air) for mobile phones
    - Incremental update of data files for security software.
    - File synchronization & transfer over WAN
    - Differential compression for executable files
    - …
  - It is a time to summarize various problems & techniques in a systemic view:
    - I may write an academic book on this.



Introduction

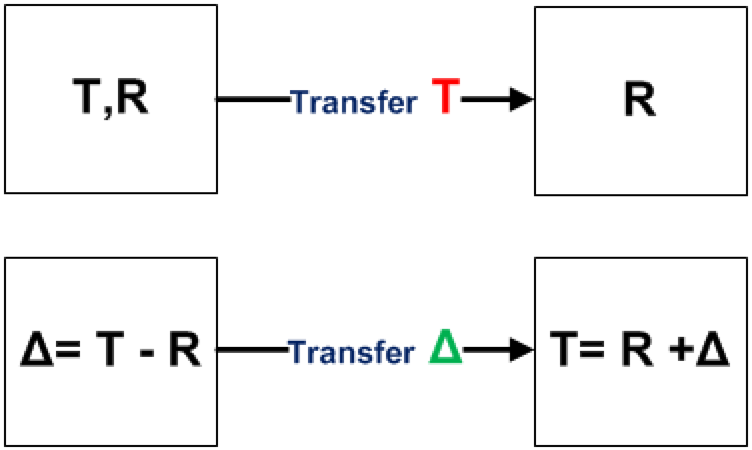
- **What is differential compression?**





Introduction

- **To reduce network bandwidth cost :**



Lower bandwidth cost

- **To reduce storage cost:**

Lower storage cost

Introduction

- **Applications:**
  - Data backup
    - Remote data backup
  - Revision control systems
  - Software vulnerability & patch management
    - FOTA (firmware over the air)
    - Malware signature update
  - File synchronization and transfer
  - Distributed file systems
  - Cloud data migration

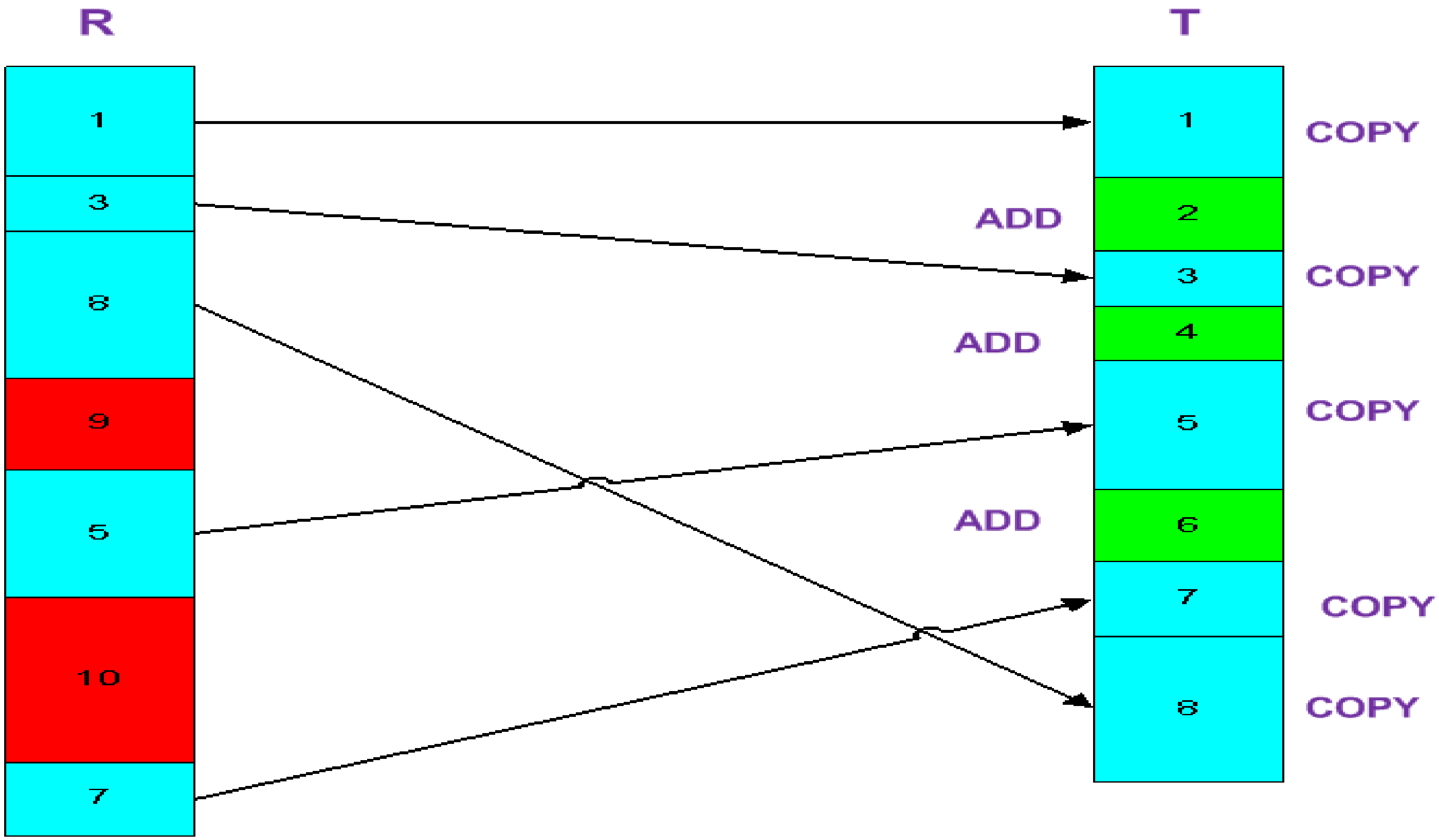
A Math Model for Describing File Differences

- **In the formal presentation Δ = T – R, what do we mean by "-" and Δ?**
- **There are a few approaches to describe DIFF.**
  - Here is one.

- **<u>Diff Model:</u>** A math model to describe the "differences" of T and R:
  - **Δ** is basically a procedure that transforms reference file R to target file T.
    - To be specific, **Δ** is a sequence of string edit operations for reconstructing T from R.
  - Two edit operations COPY & ADD :
    - **COPY (addrSrc, size ,addrDest)** --- to copy a block of data from reference file to target file.
    - **ADD (dataBlock, size ,addrDest)** --- to add a block of data to the target file.

A Math Model for Describing File Differences

- Look at an example:



A Math Model for Describing File Differences

For better illustration, let us assume:

- Block 1 has **100** bytes
- Block 2 has **60** bytes
- Block 3 has **50** bytes
- Block 4 has **50** bytes
- Block 5 has **120** bytes
- Block 6 has **60** bytes
- Block 7 has **70** bytes
- Block 8 has **150** bytes
- Block 9 has **80** bytes
- Block 10 has **180** bytes

- **A sequence of edit operations:**
  1. COPY <0,100,0>
  2. ADD <2^{nd} block,60,100>
  3. COPY <100,50,160>
  4. ADD <4^{th} block,50,210>
  5. COPY <380,120,260>
  6. ADD <6^{th} block,60,380>
  7. COPY <680,70,440>
  8. COPY <150,150,510>

- This sequence is **Δ.**

A Math Model for Describing File Differences

- **To optimize the presentation of Δ, if we arrange the edit operations in an ascending order of addrDest, all addrDest are not required for explicit presentation.**
- **We can rewrite two operations in following formats:**
  - COPY <addrSrc, size>
  - ADD <dataBlock, size>

Δ is presented by:

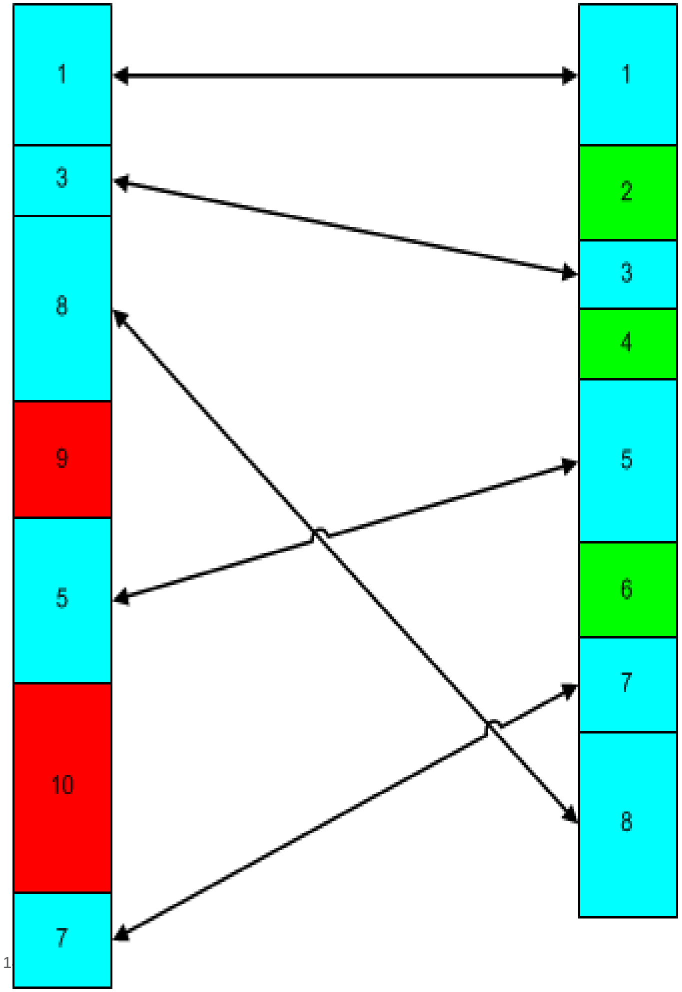
1. COPY <0,100>
2. ADD <2^{nd} block,60>
3. COPY <100,50>
4. ADD <4^{th} block,50>
5. COPY <380,120>
6. ADD <6^{th} block,60>
7. COPY <680,70>
8. COPY <150,150>

A Math Model for Describing File Differences

- **Two tasks remains to be solved:**
  1. How to create Δ, i.e., the sequence of the edit operations?
  2. How to encode Δ into a file (we refer to it as DIFF package) ?

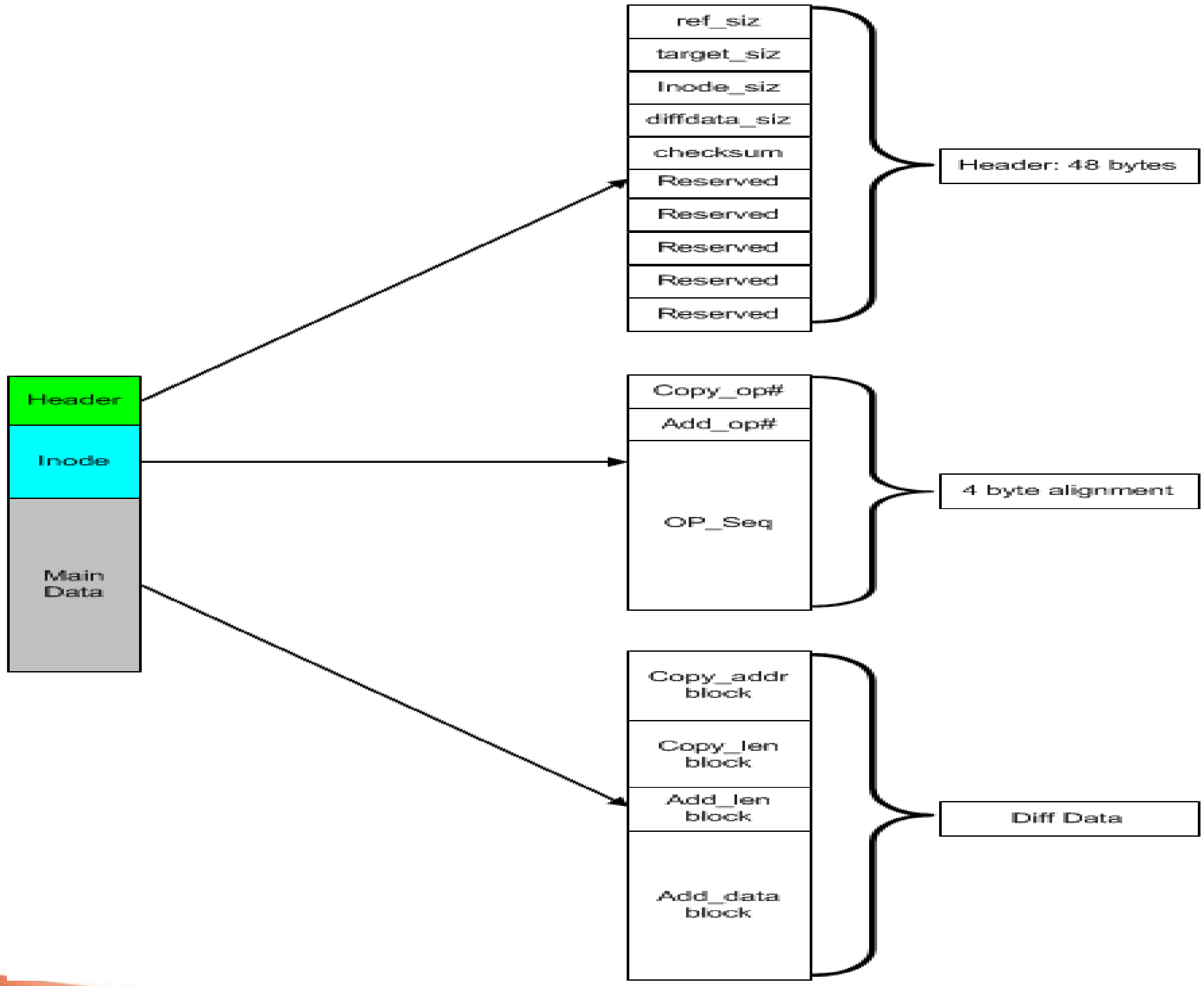
- **The top task is an effective algorithm to identify the common blocks, e.g., the blocks {1,3,5,7,8} shown in the right side.**

- **I don’t think I should talk about algorithms at this conference… the details may take half an hour.**



A Math Model for Describing File Differences

Designing a diff package: an example:



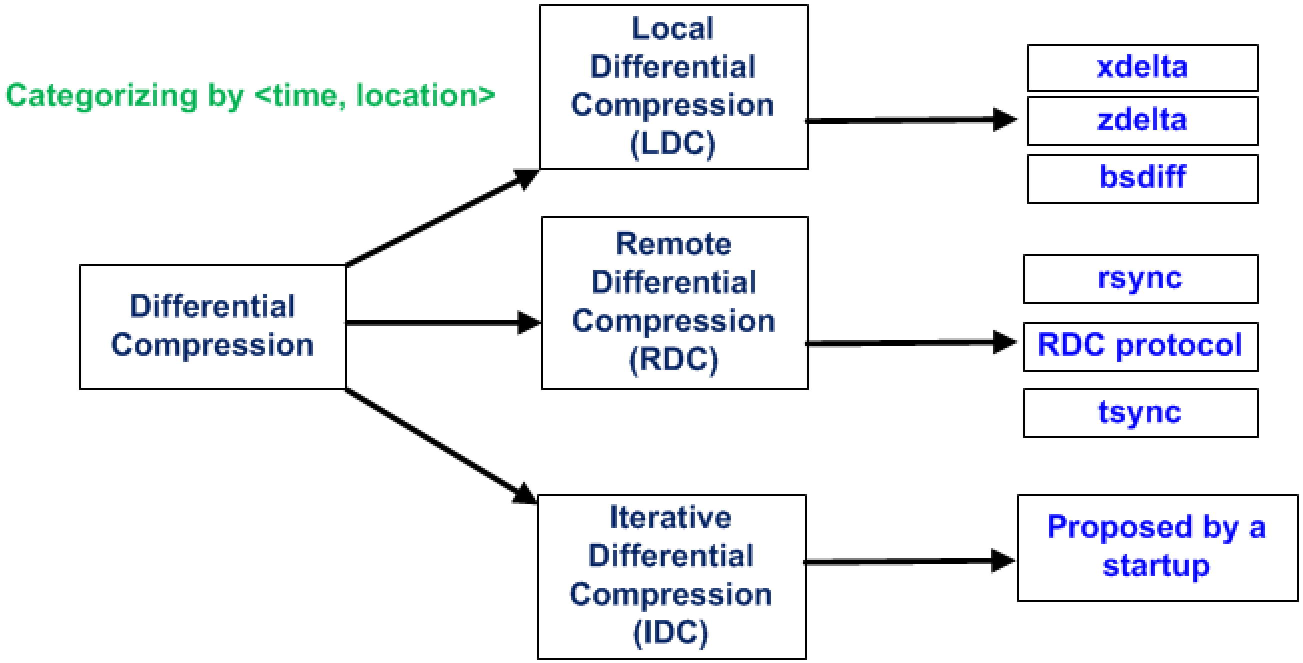
A Math Model for Describing File Differences

- **We answered two basic questions:**
  - What is differential compression?
  - How to describe differential compression mathematically?

- **A few questions remained:**
  - How to design an algorithm for differential compression?
  - How to measure the efficiency of an algorithm?
    - We need to introduce a cost model.
  - Can we design the most efficient algorithm in terms of the cost model?

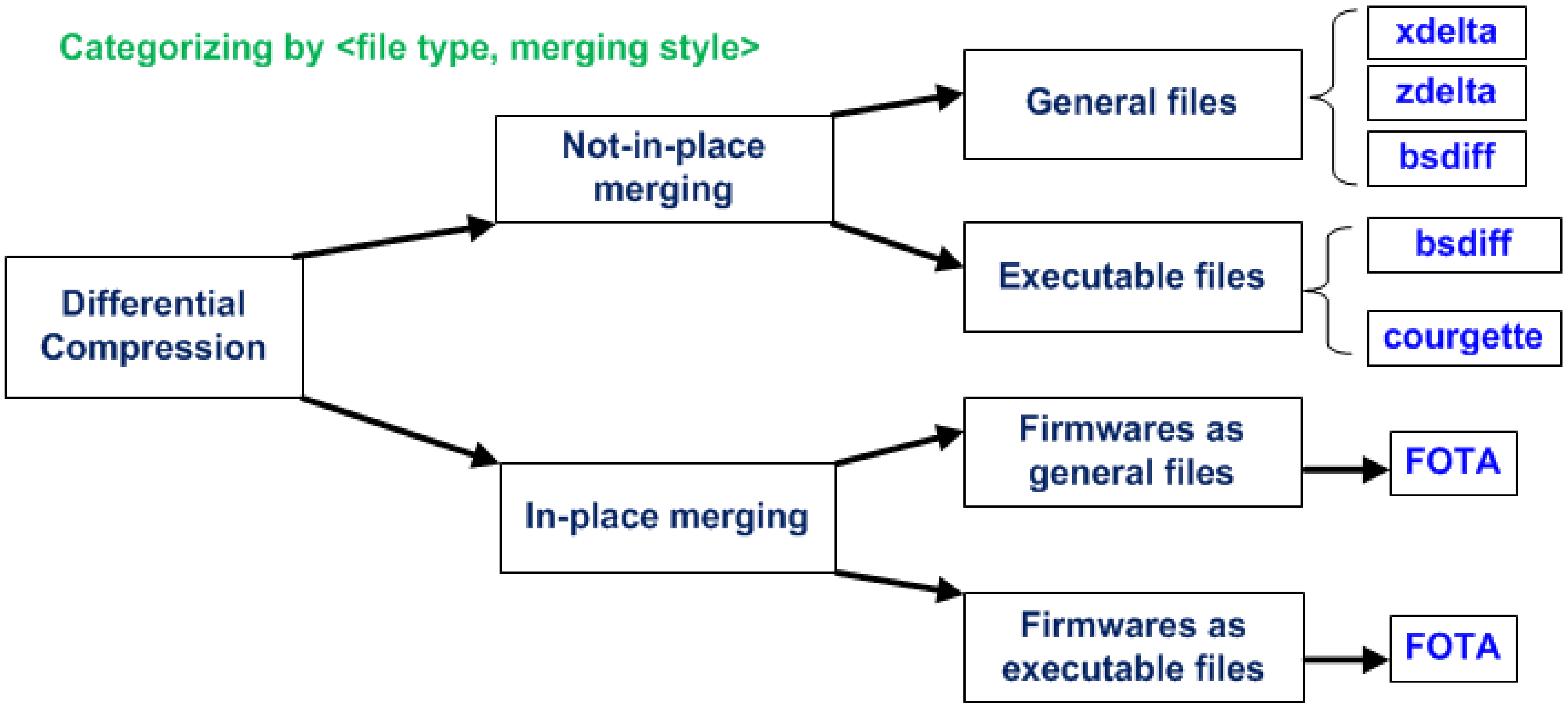
Categorizing Differential Compression

- Due to applications, differential compression can be categorized into different ways.



Categorizing Differential Compression

- **Continued:**



Categorizing Differential Compression

- **Summary:**

	LDC	RDC	IDC
General File	YES	Yes	Yes
Executable file	Yes	No Study Yet	No Study Yet
General firmware	Yes	No Study Yet	No Study Yet
Executable firmware	Yes	No Study Yet	No Study Yet

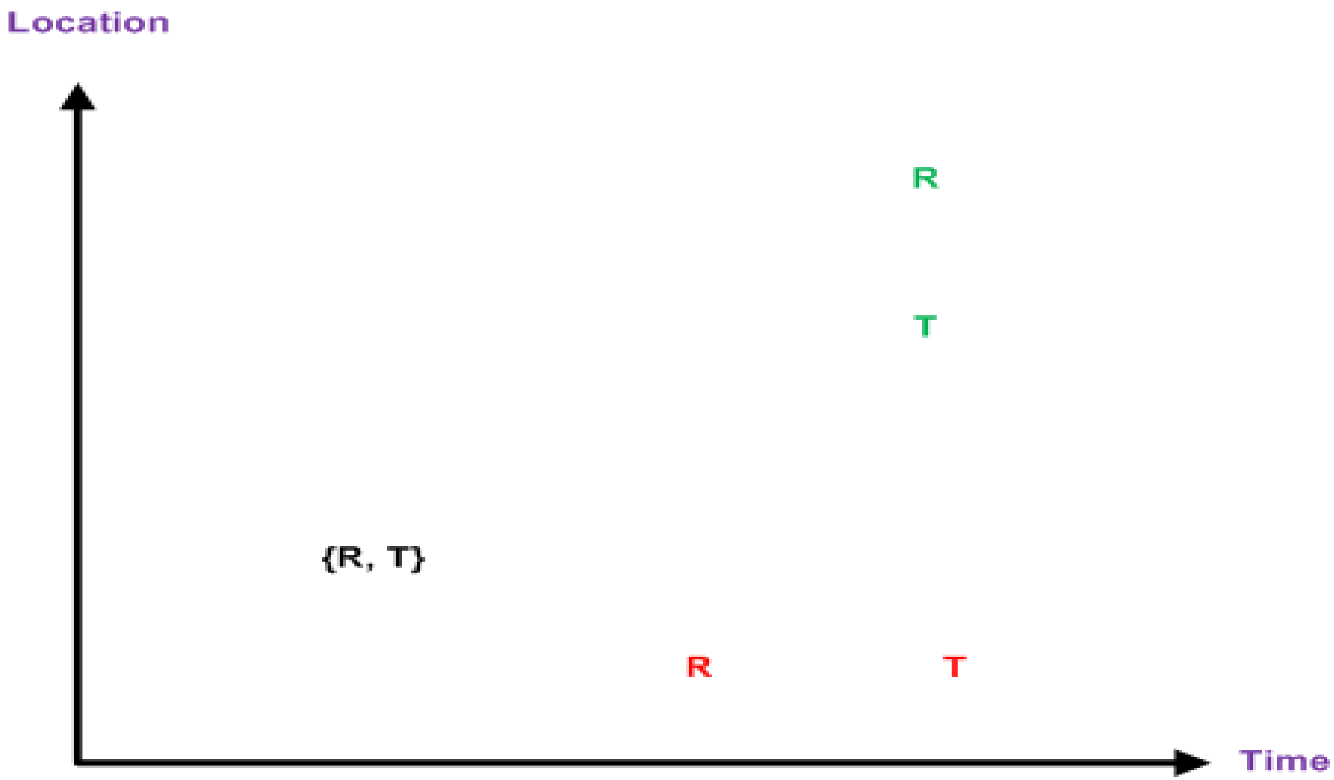
Advanced Topics

- **Let us investigate three topics in depth:**
  1. LDC vs RDC vs IDC   for general files
  2. LDC for executable files
  3. LDC for in-place merging

Advanced Topics

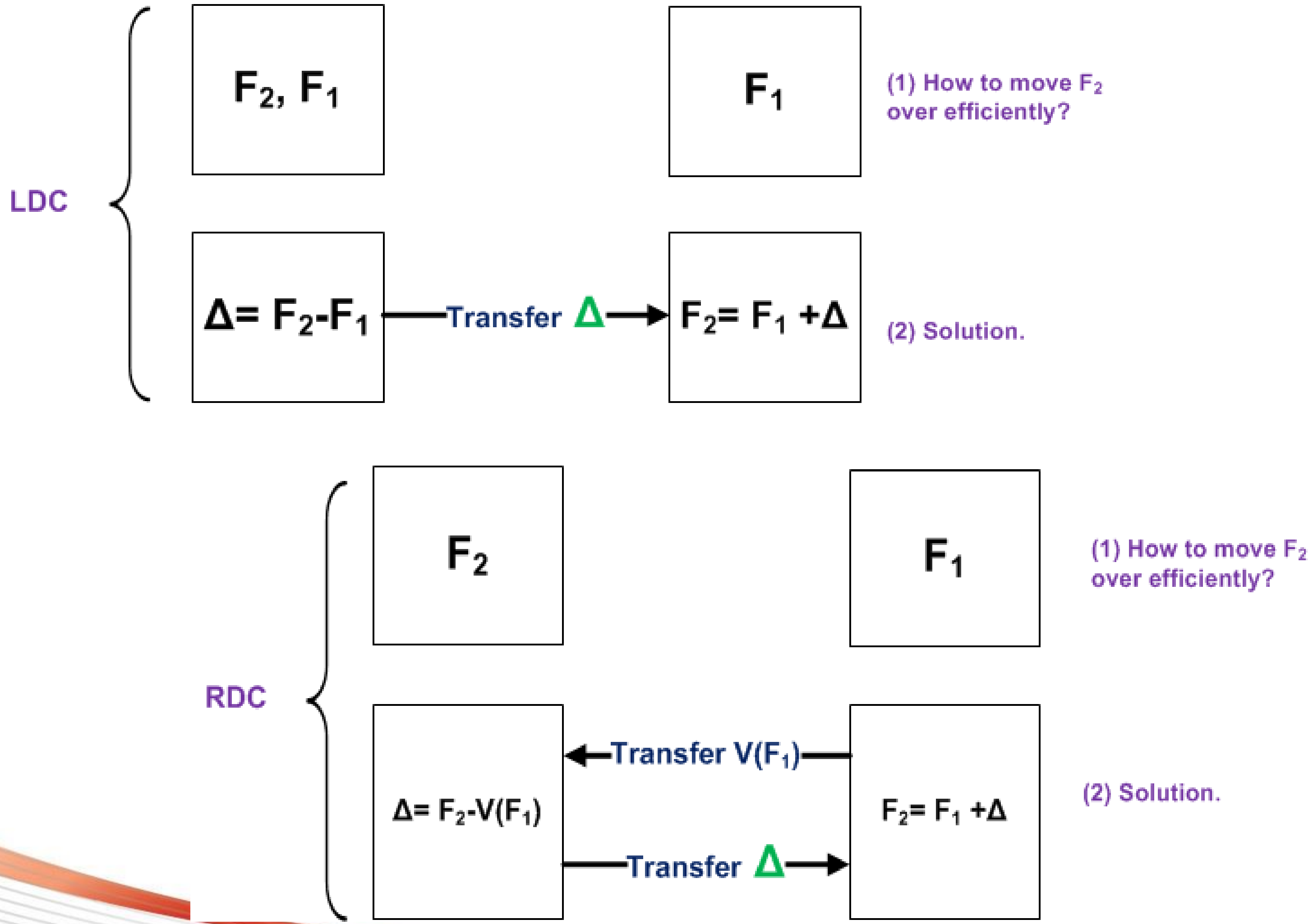
LDC vs RDC vs IDC : use cases

- How to implement Δ=T-R for three different cases in term of space & time?
- **LDC** : both R and T appear in the ***same location*** at the ***same time***.
- **RDC** : R and T appear in ***different locations*** at the ***same time***.
- **IDC** : R and T appear in the ***same location*** at ***different times***.



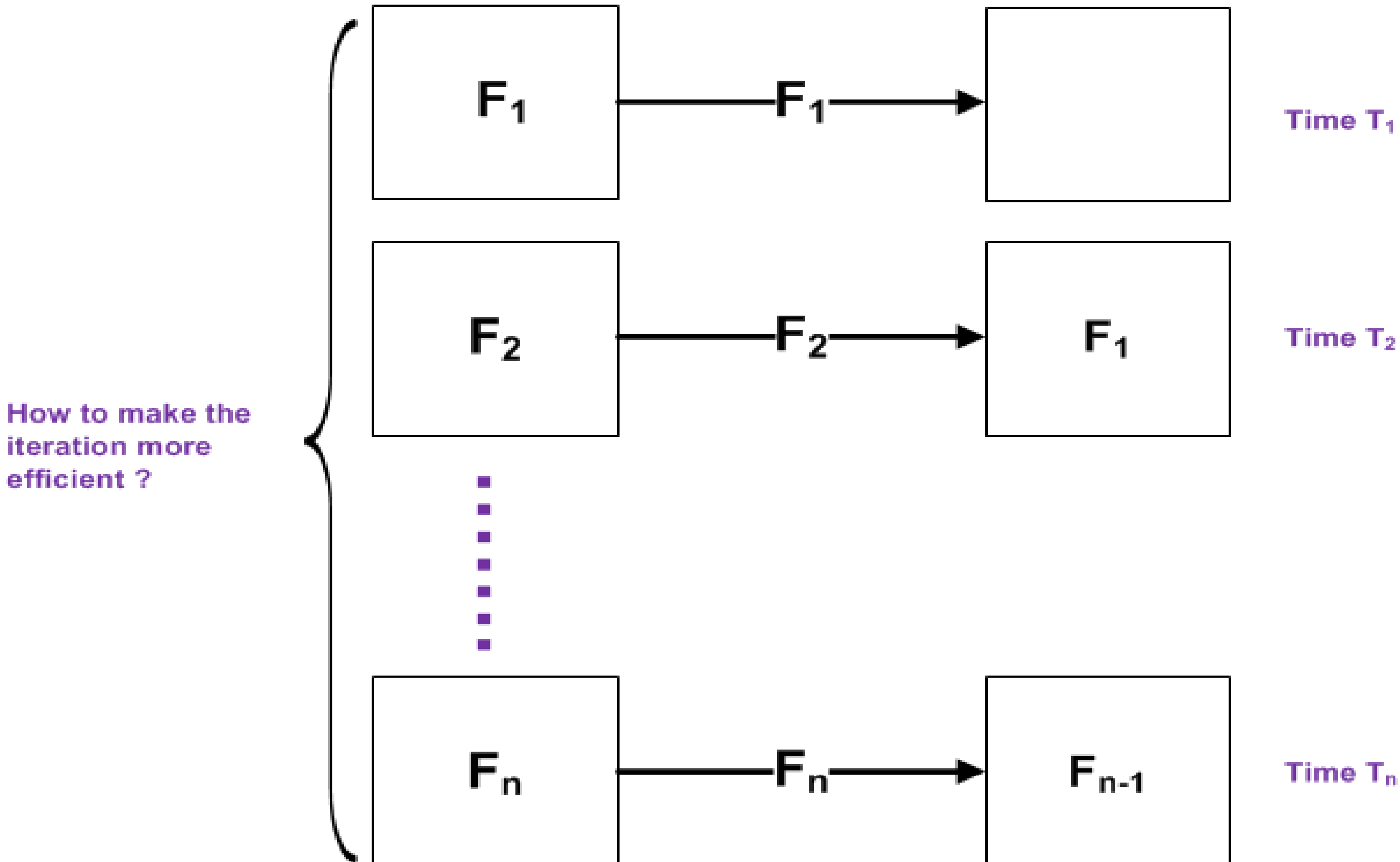
Advanced Topics

- LDC vs RDC vs IDC : architecture



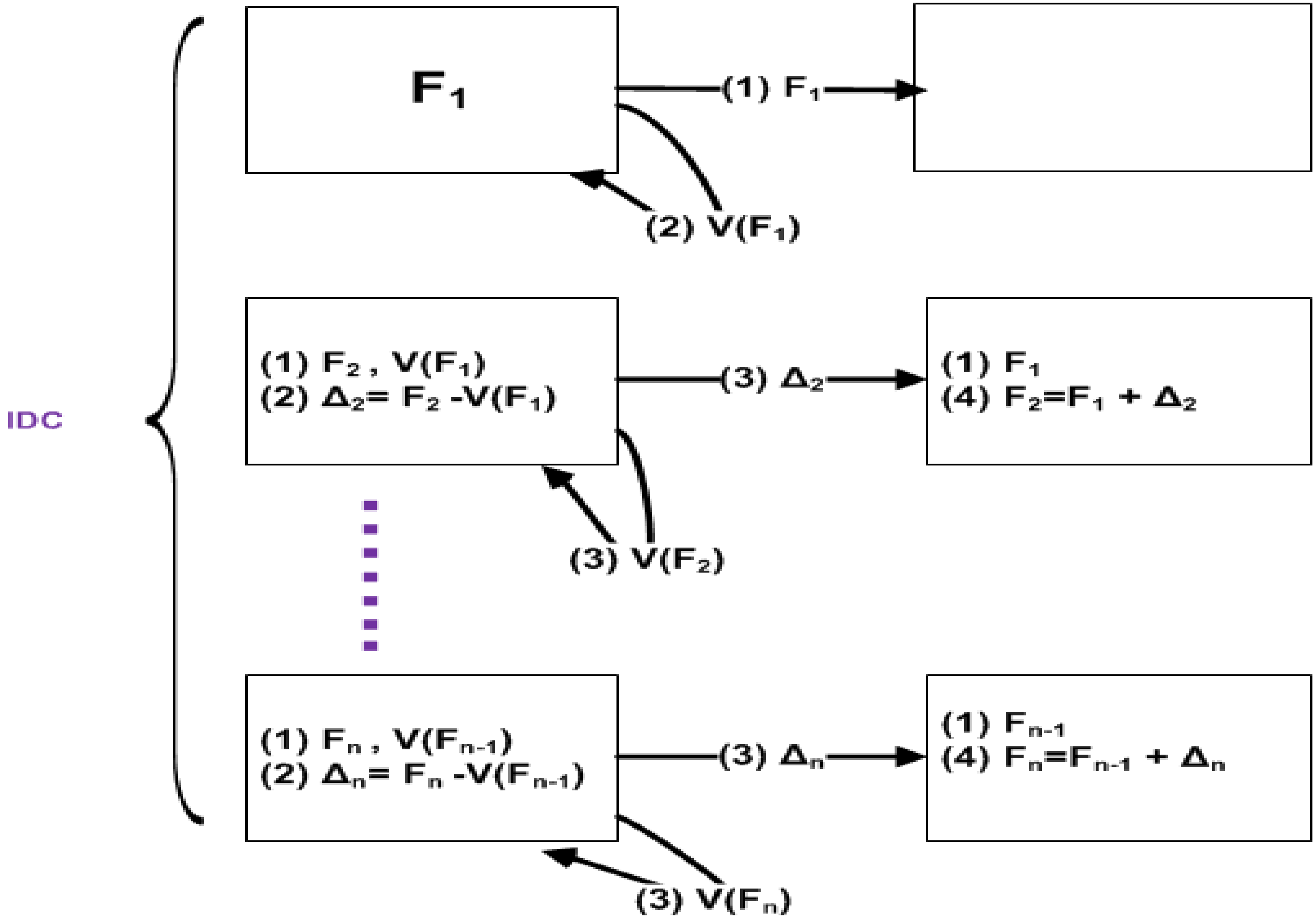
Advanced Topics

- **LDC vs RDC vs IDC :**



Advanced Topics

- **LDC vs RDC vs IDC :**

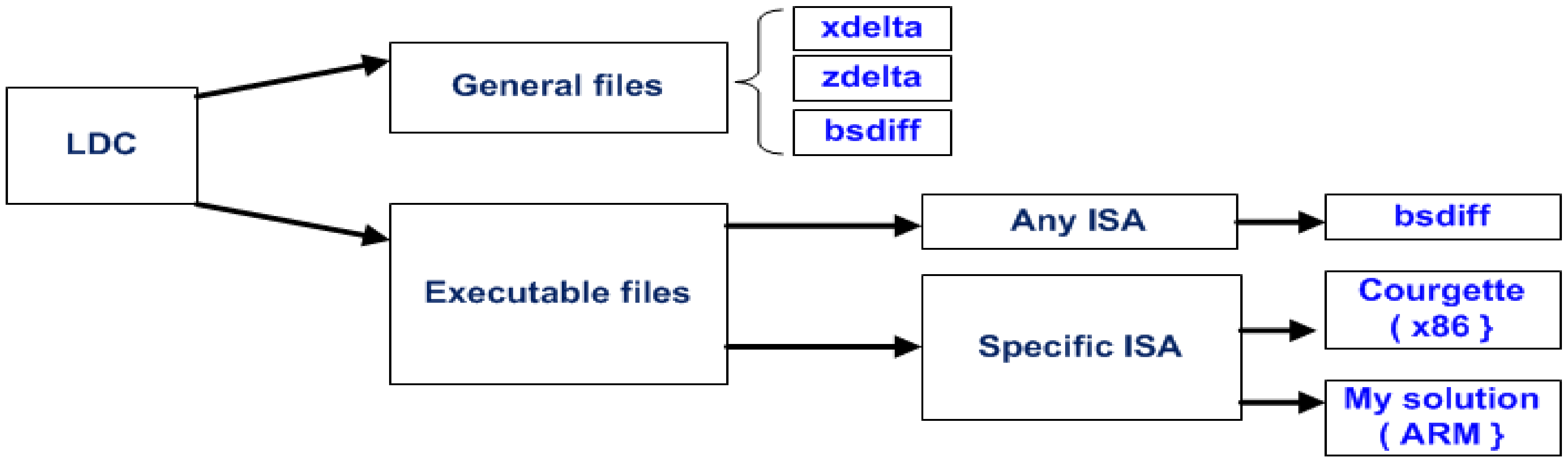


Advanced Topics

- **Notes:**
  - **V(F)** = view of F :
    - It is a data structure that summarizes a file in an abstract yet efficient way. It takes much less space than original file.
    - This concept VIEW was proposed by a local startup to describe the IDC scheme.
      - I found it applies to RDC scheme too.
    - There are different implementations of VIEW. For example, to describe rsync and RDC protocols, we would have two different VIEWs for the same file.
  - $\mathbf{F_2 = F_1 + \Delta}$
    - where $\Delta = F_2 - V(F_1)$ instead of $\Delta = F_2 - F_1$
      - This makes RDC & IDC possible.

Advanced Topics

- LDC for executable files:

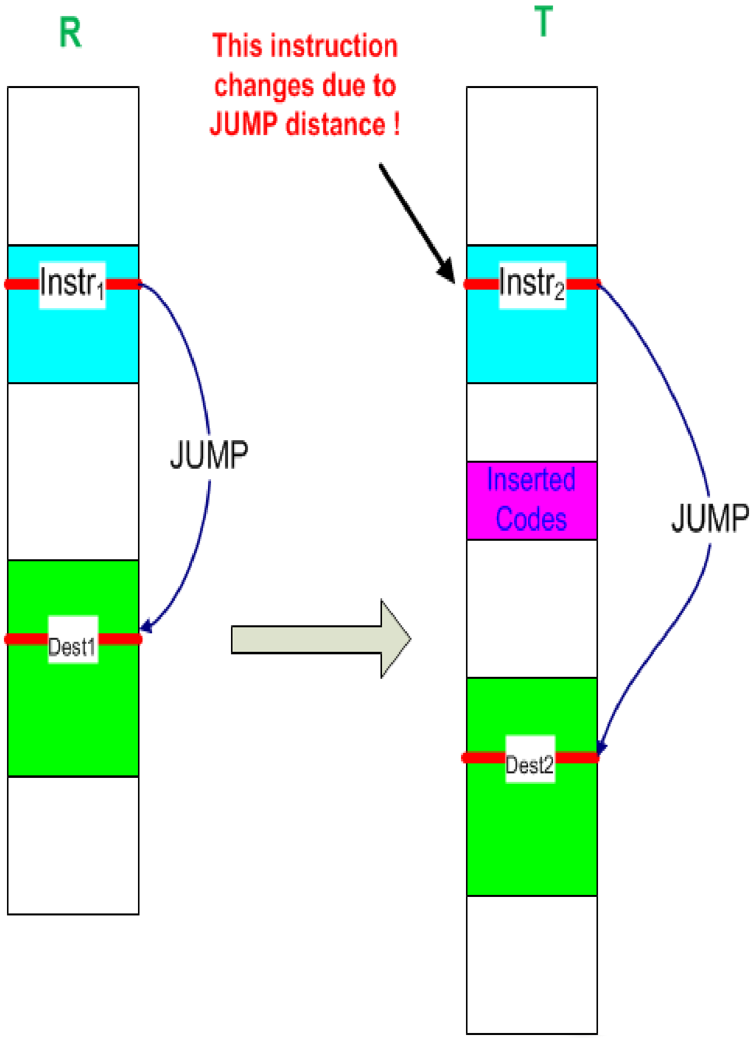


ISA = Instruction Set Architecture

Advanced Topics

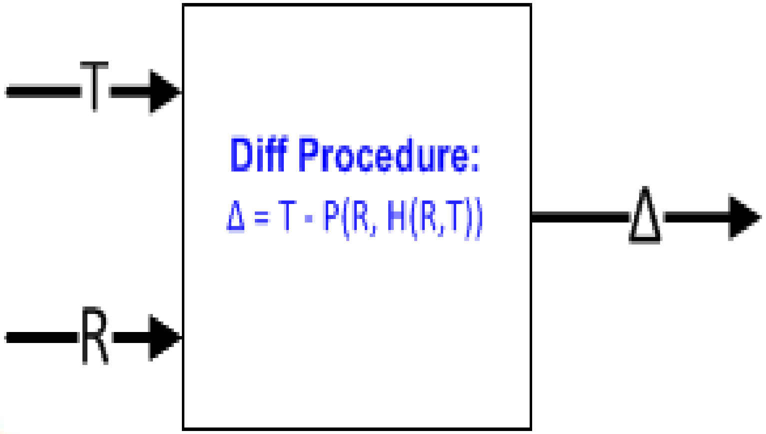
LDC for executable files: for a specific ISA

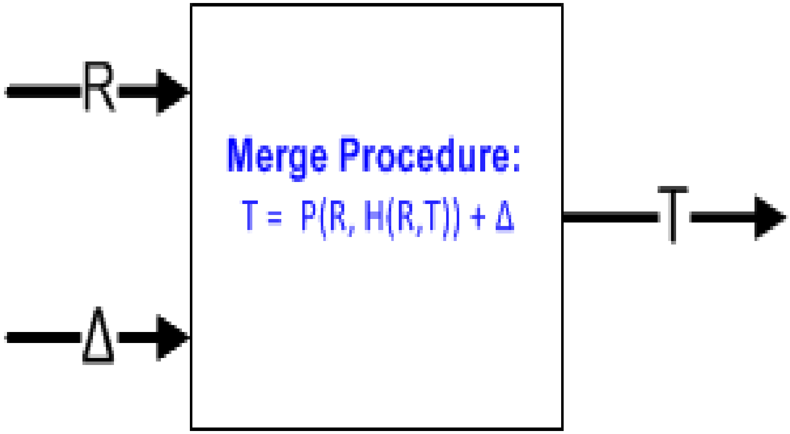
- Differential compression algorithms identify changes between files.
- General files: all changes are just changes.
- Executable files:
  - Primary change: instructions are altered due to source code changes.
  - Secondary change: an instruction is altered at the byte level due to code change happening at other addresses.
  - We use JUMP as an example to illustrate the concept. An JUMP instruction is a few bytes that encode the distance between the source and destination.



Advanced Topics

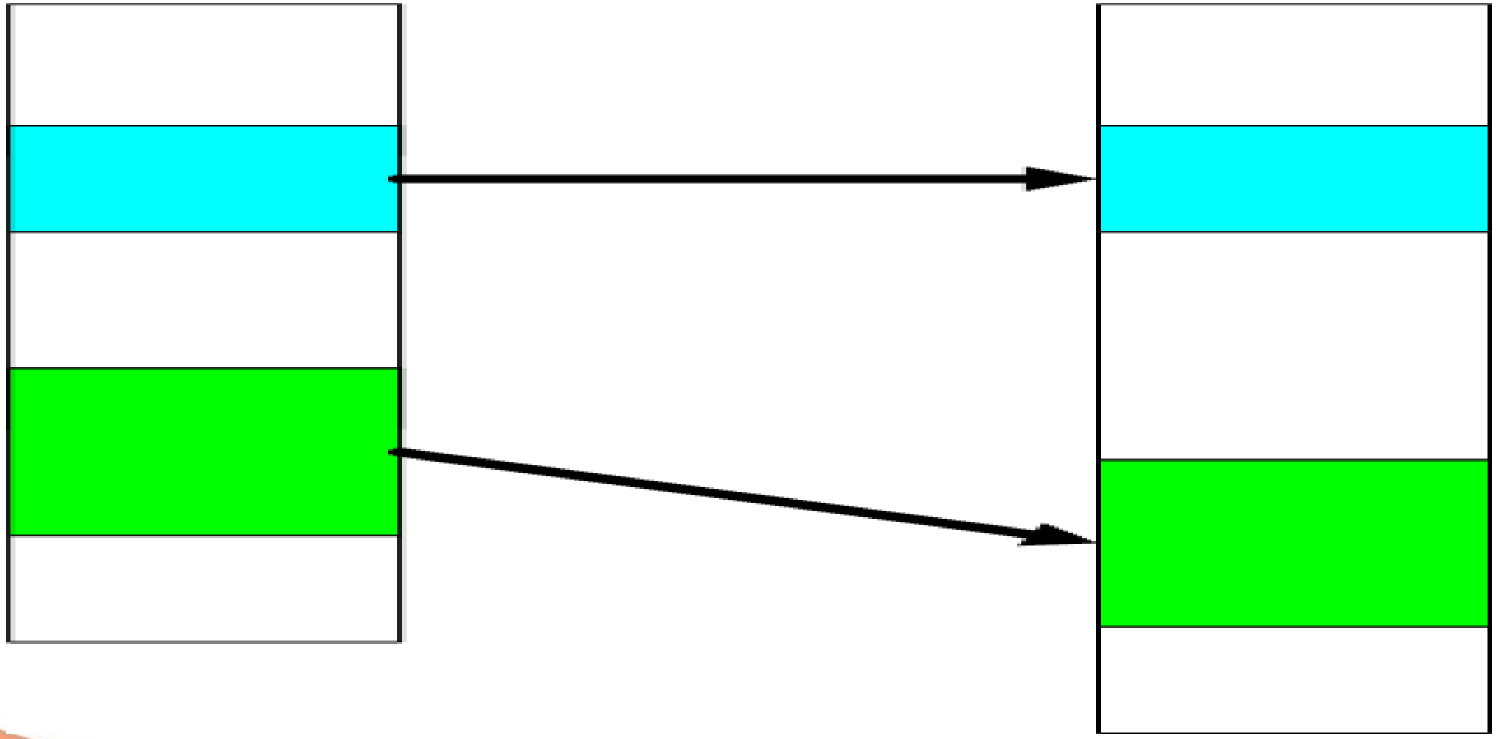
- **LDC for executable files: how to reduce the diff?**
  - A mathematical model is necessary.
  - For example: removing secondary change for JUMP:
    - The secondary code change causes instr1 ≠ instr2
    - Given the file R, if we can derive instr2 from instr1, we can replace instr1 in R with instr2.
    - For all such instructions in R, we can do the same substitution, we transfer R into another file and denote it as P(R,H(R,T)) where H stands for **hints**. We have the new formal presentation:





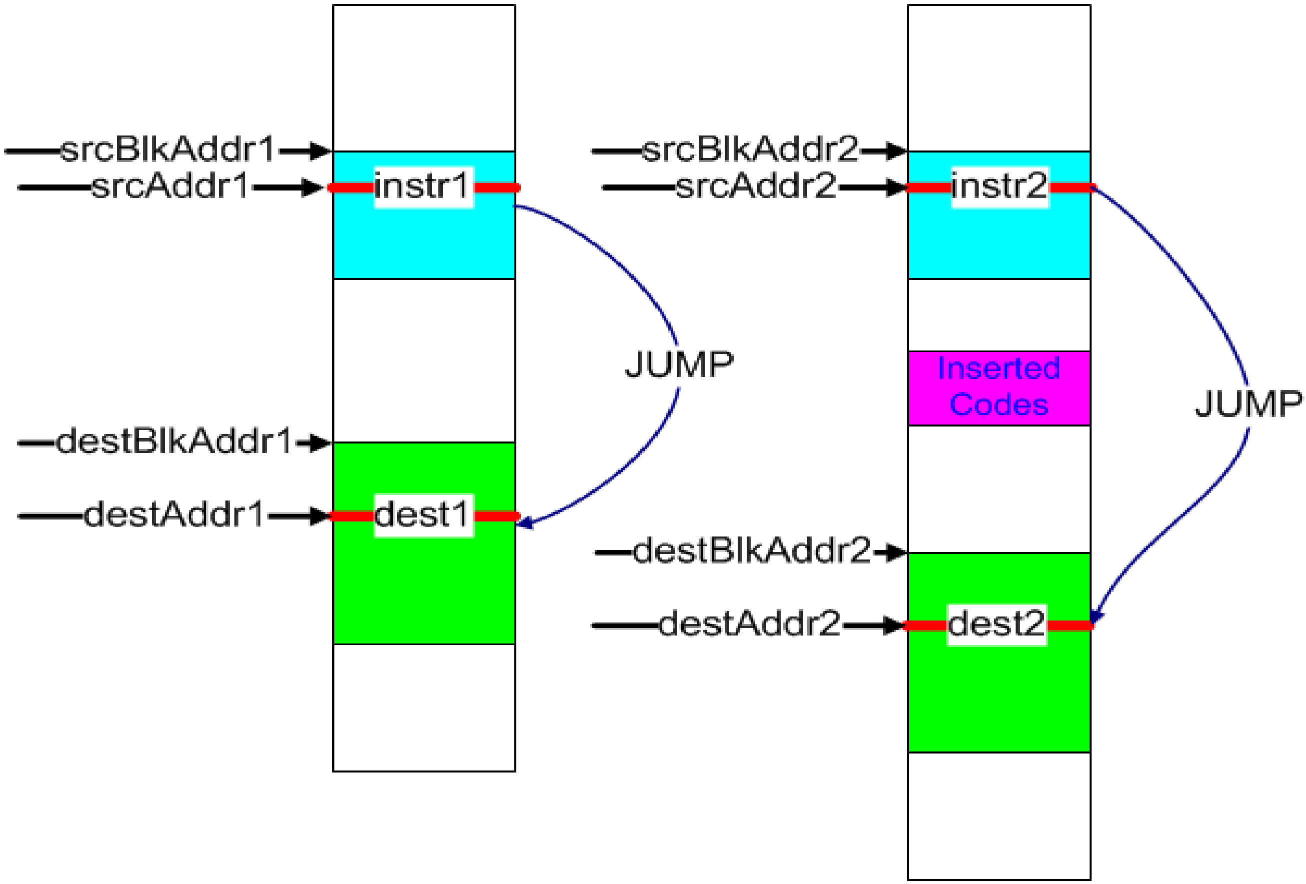
Advanced Topics

- **LDC for executable files**
- **Mathematical Modeling:**
  - Lets start with ***common code blocks*** between two versions:
    - Assume we can identify them with symbol tables or code alignment algorithms.



Advanced Topics

- **A Mathematical Model:**
  - How to derive a new JUMP instruction from an old one?



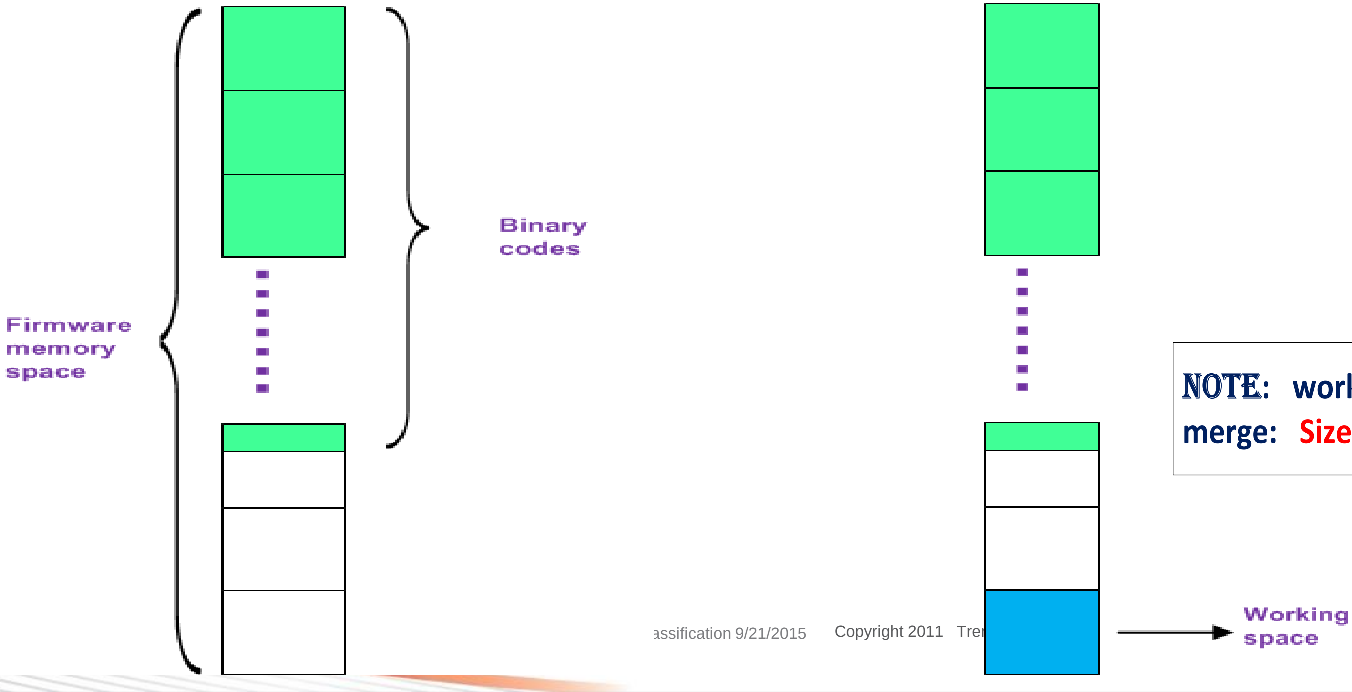
instr2 = Decode(instr1) + (destBlkAddr2 – destBlkAddr1) – (srcBlkAddr2 - srcBlkAddr1)

Advanced Topics

- **Mathematical Modeling:**
  - How to derive a new JUMP instruction from an old one?
    - instr2 = Encode(destAddr2 – srcAddr2)
    - destAddr2 – srcAddr2 = (destAddr1 – srcAddr1) + (destAddr2 – srcAddr2) - (destAddr1 – srcAddr1) = Decode(instr1) + (destAddr2 – destAddr1) – (srcAddr2 - srcAddr1) = Decode(instr1) + (destBlkAddr2 – destBlkAddr1) – (srcBlkAddr2 - srcBlkAddr1)
  - **instr2 = Decode(instr1) + (destBlkAddr2 – destBlkAddr1) – (srcBlkAddr2 - srcBlkAddr1)**
  - We can do the similar to other instructions such as data pointers.
  - All these instructions such as JUMP or data pointers are called ***profitable instructions***.
  - A solution is an algorithm that identifies all the profitable instructions and removes all secondary changes accordingly.

Advanced Topics

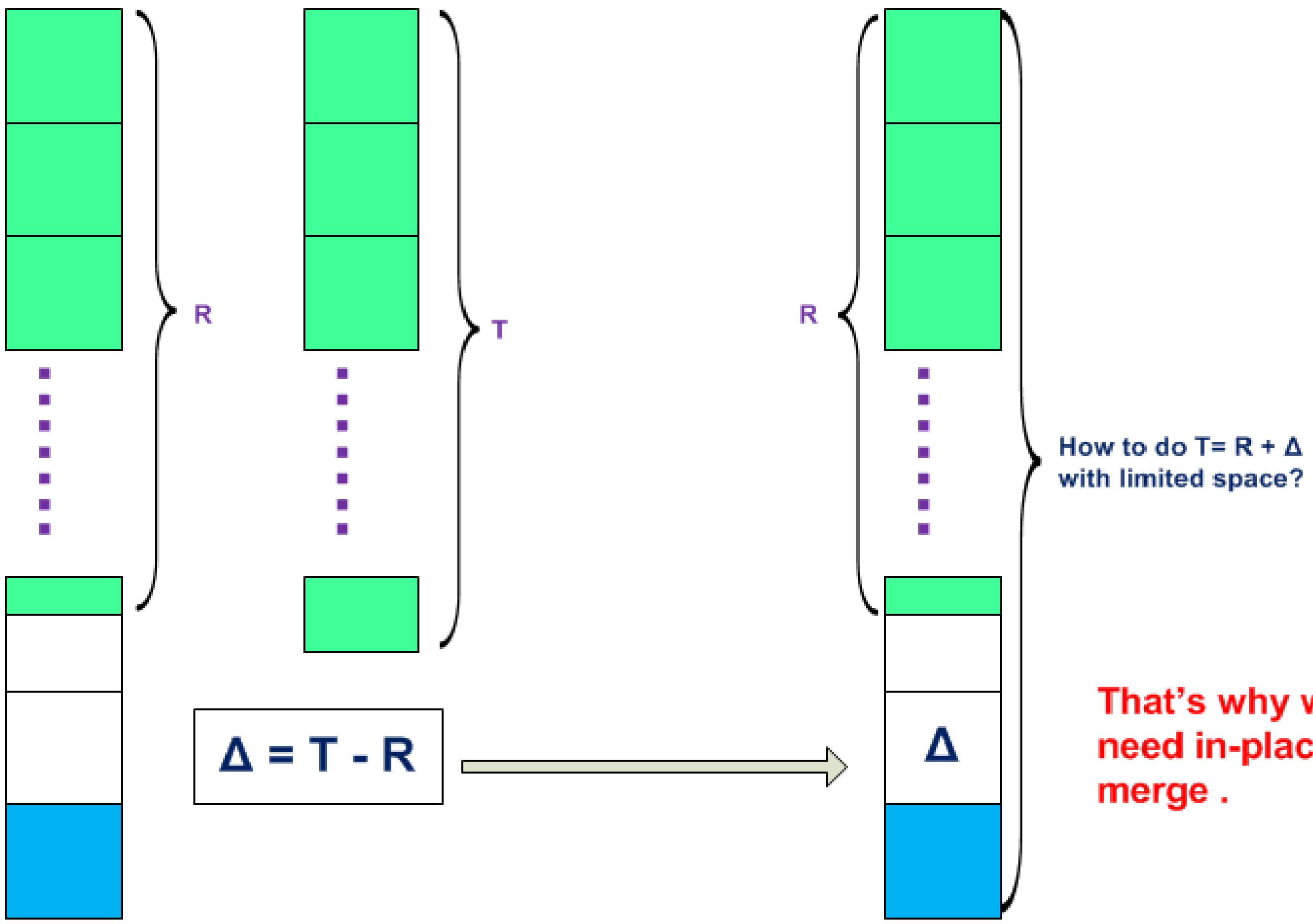
- **LDC for in-place merging: the 3rd topic**
- **Use case:**
  - Mobile phone FOTA (Firmware Over The Air).
  - A firmware can be considered as a file contained in a sequence of code blocks (of fixed sizes).
  - A phone has limited memory space for firmware updating.
    - Updating must be implemented using ***in-place*** algorithm.



NOTE: work space required for general merge: Size(R)+Size(T)+Size(Δ)

Advanced Topics

- **LDC for in-place merging**



That's why we need in-place merge .

Advanced Topics

- **LDC for in-place merging:**
  - block based differential compression
    - block dependency between two versions of firmware
    - Topological sorting to create a sequence of block number based on block precedence.
  - In-place merging
    - Block based merging
    - Block writing based on the sequence of block number.

That is a very interesting technique!

Summary

- **Background of differential compression**
- **A mathematical model for differential compression**
- **Categorizing differential compression from two perspectives:**
  - <time, location>
  - <file type, merging style>
- **Three advanced topics:**
  - Comparing three differential compression schemes
  - Differential compression of executable files
  - In-place file merging with LDC

Q & A

THANK YOU FOR YOUR ATTENTION!

Any questions!

Email: liwei_ren@trendmicro.com
Home page: https://pitt.academia.edu/LiweiRen