



universität
wien

URI Fragment Identifiers

Uniform Fragment Identification

Wolfgang Jochum

M3A Workshop - Sept. 05, 2007

www.cs.univie.ac.at/wolfgang.jochum



Problem Use Case – Mingled Lessons

The screenshot shows a web browser window displaying a LECTURNITY WebPlayer interface. The window title is "Prof. Dr. Thomas Ottmann: Informatik II - LECTURNITY WebPlayer 1.7.0.p4". The address bar shows the URL "http://132.230.139.59/demos/Ottmann_Hanoi/real/Tuerme_von_Hanoi.html". The main content area displays the title "Die Türme von Hanoi" and a diagram of three towers with disks. A video player in the bottom left shows a professor speaking. The interface includes navigation buttons, a progress bar, and a "LECTURNITY webplayer" logo.



Problem Use Case – Mingled Lessons

design goals of Mingled Lessons

interoperable

open

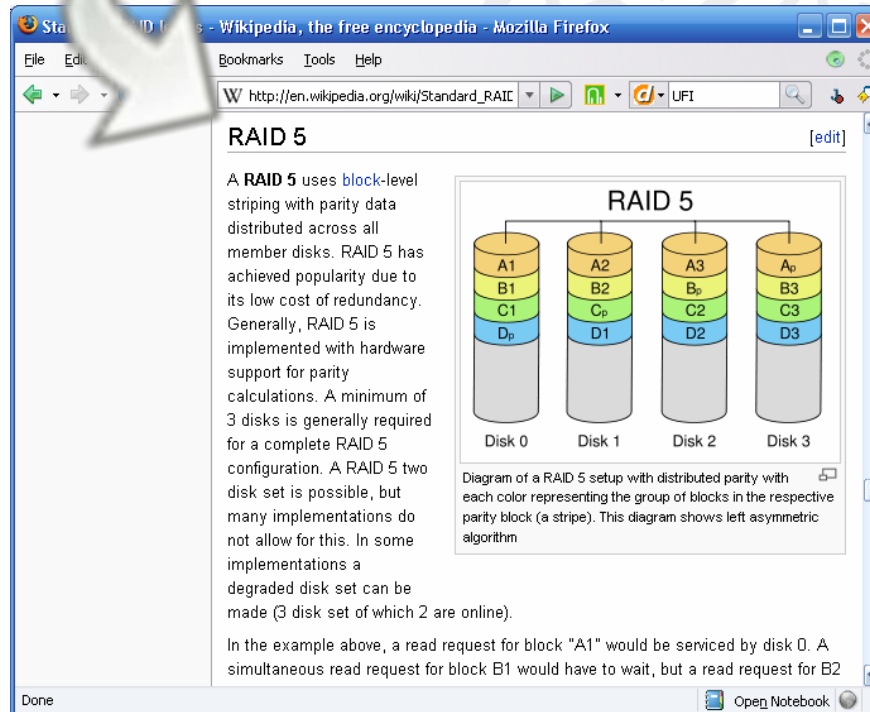
ensure reuse of single and composed media objects

easy integration in existing environments

core component:

uniform fragment identifiers

Problem L.I – link to text document fragment



Standard RAID Levels - Wikipedia, the free encyclopedia - Mozilla Firefox

File Edit View Bookmarks Tools Help

W http://en.wikipedia.org/wiki/Standard_RAIC UFI

RAID 5 [\[edit\]](#)

A **RAID 5** uses **block-level** striping with parity data distributed across all member disks. RAID 5 has achieved popularity due to its low cost of redundancy. Generally, RAID 5 is implemented with hardware support for parity calculations. A minimum of 3 disks is generally required for a complete RAID 5 configuration. A RAID 5 two disk set is possible, but many implementations do not allow for this. In some implementations a degraded disk set can be made (3 disk set of which 2 are online).

RAID 5

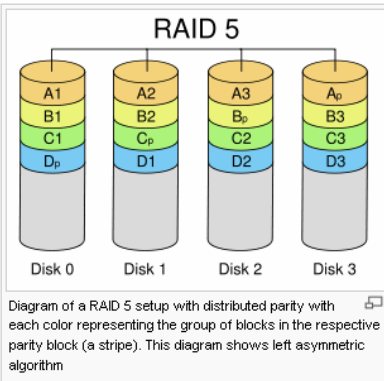


Diagram of a RAID 5 setup with distributed parity with each color representing the group of blocks in the respective parity block (a stripe). This diagram shows left asymmetric algorithm

In the example above, a read request for block "A1" would be serviced by disk 0. A simultaneous read request for block B1 would have to wait, but a read request for B2

Done [Open Notebook](#)



Problem L.I – link to text document fragment



HMTL – [anchors](#)



PDF – [open parameters](#)



MS Word – [bookmarks](#)



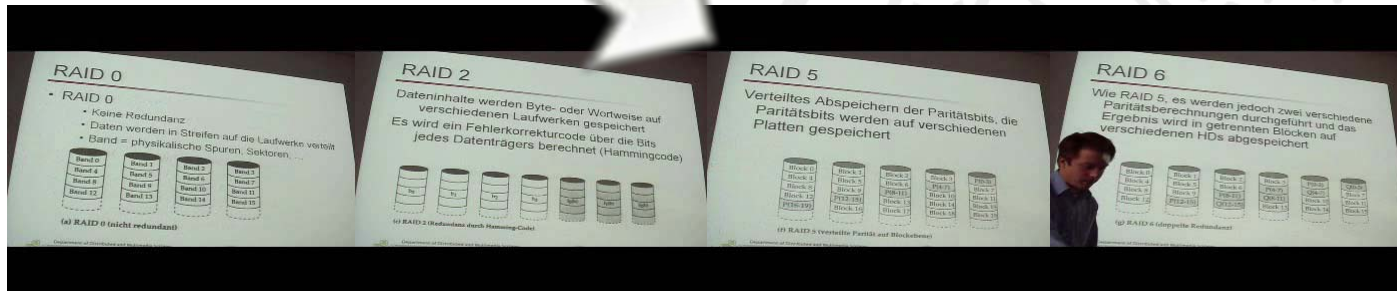
Open Office



Plain Text



Problem L.II – link to video fragment





Problem L.II – link to video fragment



RTSP – npt / smpte / utc



MPEG 21 Part 17 FID



avi , mov, ...



Problem L.III – link to image fragment



RAID 0



RAID 1



RAID 5



RAID 0+1



Problem L.III – link to image fragment

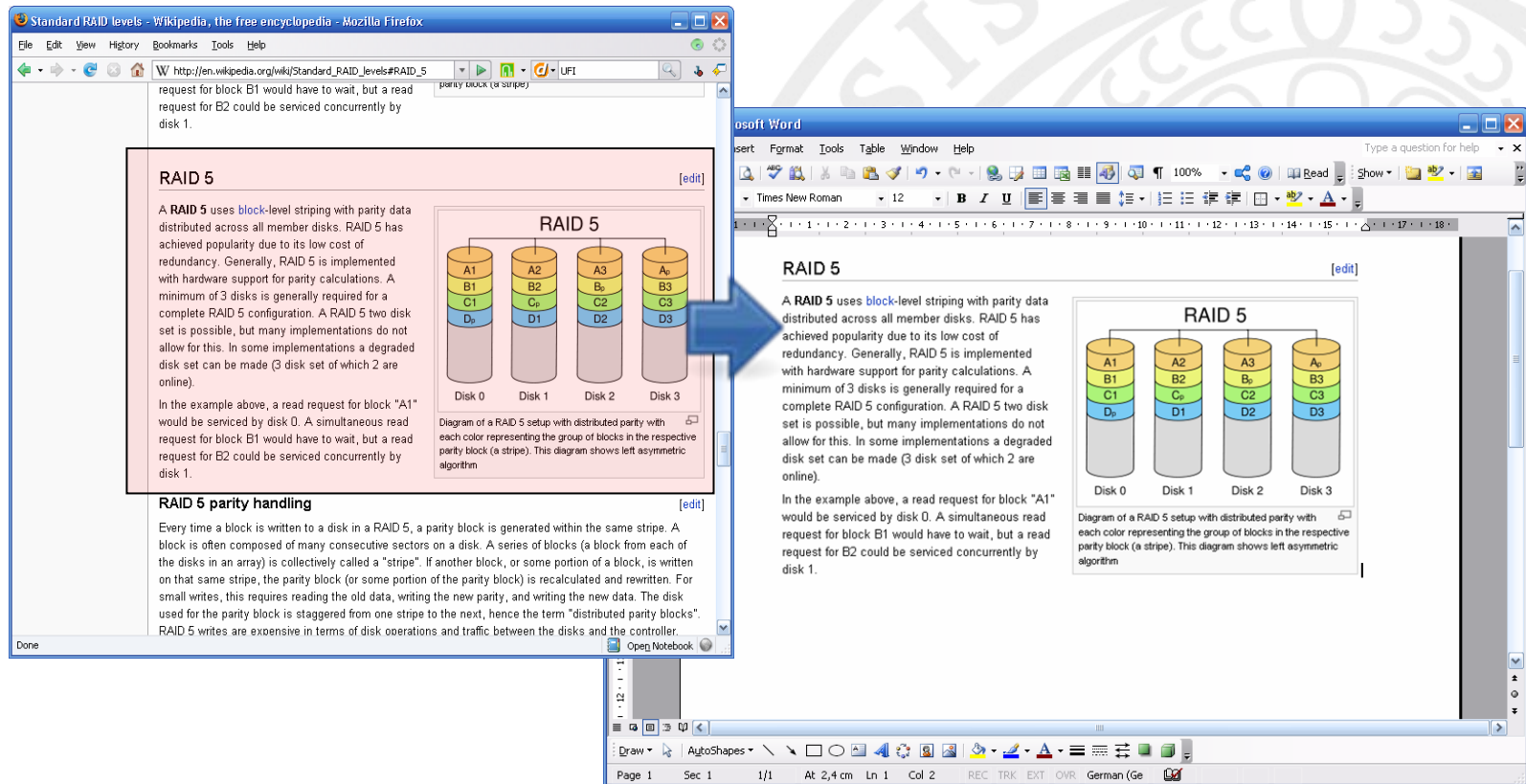


SVG – anchor / view



jpeg, gif, bmp, ...

Problem R.I – reuse text document fragment



The image shows a comparison of text copied from a Wikipedia page into a Microsoft Word document. A blue arrow indicates the direction of copying.

Browser Window (Left): Mozilla Firefox displaying the Wikipedia page for RAID 5. The text includes:

RAID 5

A RAID 5 uses block-level striping with parity data distributed across all member disks. RAID 5 has achieved popularity due to its low cost of redundancy. Generally, RAID 5 is implemented with hardware support for parity calculations. A minimum of 3 disks is generally required for a complete RAID 5 configuration. A RAID 5 two disk set is possible, but many implementations do not allow for this. In some implementations a degraded disk set can be made (3 disk set of which 2 are online).

In the example above, a read request for block "A1" would be serviced by disk 0. A simultaneous read request for block B1 would have to wait, but a read request for B2 could be serviced concurrently by disk 1.

RAID 5 parity handling

Every time a block is written to a disk in a RAID 5, a parity block is generated within the same stripe. A block is often composed of many consecutive sectors on a disk. A series of blocks (a block from each of the disks in an array) is collectively called a "stripe". If another block, or some portion of a block, is written on that same stripe, the parity block (or some portion of the parity block) is recalculated and rewritten. For small writes, this requires reading the old data, writing the new parity, and writing the new data. The disk used for the parity block is staggered from one stripe to the next, hence the term "distributed parity blocks". RAID 5 writes are expensive in terms of disk operations and traffic between the disks and the controller.

Word Document (Right): Microsoft Word containing the copied text from the browser. The text is identical to the browser window, including the RAID 5 diagram and the RAID 5 parity handling section.

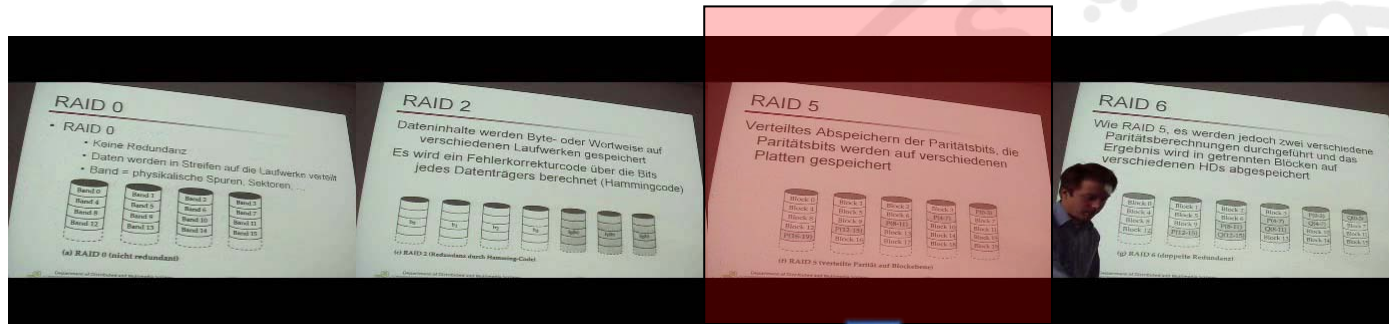
Diagram (Shared): A diagram titled "RAID 5" showing four disks (Disk 0, Disk 1, Disk 2, Disk 3) with data blocks A1, B1, C1, D1 on Disk 0; A2, B2, C2, D2 on Disk 1; A3, B3, C3, D3 on Disk 2; and A4, B4, C4, D4 on Disk 3. The diagram illustrates left asymmetric striping with distributed parity.



Problem R.I – reuse text document fragment

- HTML
- PDF – open parameters
- Text - Internet-Draft
- MS Word
- Open Office

Problem R.II – reuse video fragment



request for block B1 would have to wait, but a read request for B2 could be serviced concurrently by disk 1.

parity block (a stripe)

RAID 5

A RAID 5 uses block-level striping with parity data distributed across all member disks. RAID 5 has achieved popularity due to its low cost of redundancy. Generally, RAID 5 is implemented with hardware support for parity calculations. A minimum of 3 disks is generally required for a complete RAID 5 configuration. A RAID 5 two disk set is possible, but many implementations do not allow for this. In some implementations a degraded disk set can be made (3 disk set of which 2 are online).

In the example above, a read request for block "A1" would be serviced by disk 0. A simultaneous read request for block B1 would have to wait, but a read request for B2 could be serviced concurrently by disk 1.

Diagram of a RAID 5 setup with distributed parity with each color representing the group of blocks in the respective parity block (a stripe). This diagram shows left asymmetric algorithm

RAID 5 parity handling

Every time a block is written to a disk in a RAID 5, a parity block is generated within the same stripe. A block is often composed of many consecutive sectors on a disk. A series of blocks (a block from each of the disks in an array) is collectively called a "stripe". If another block, or some portion of a block, is written on that same stripe, the parity block (or some portion of the parity block) is recalculated and rewritten. For small writes, this requires reading the old data, writing the new parity, and writing the new data. The disk used for the parity block is staggered from one stripe to the next, hence the term "distributed parity blocks". RAID 5 writes are expensive in terms of disk operations and traffic between the disks and the controller.



Problem R.II – reuse video fragment



RTSP – npt / smpte / utc



MPEG 21 Part 17 FID



avi , mov, ...



Problem R.III – reuse image fragment

The image shows a presentation slide with four small images at the top, each labeled with a RAID configuration: RAID 0, RAID 1, RAID 5, and RAID 0+1. The RAID 5 image is highlighted with a red border. A large blue arrow points from the RAID 5 image down to a larger image of a water cooler. Below the images is a text box with the following content:

RAID 5 [edit]

A RAID 5 uses block-level striping with parity data distributed across all member disks. RAID 5 has achieved popularity due to its low cost of redundancy. Generally, RAID 5 is implemented with hardware support for parity calculations. A minimum of 3 disks is generally required for a complete RAID 5 configuration. A RAID 5 two disk set is possible, but many implementations do not allow for this. In some implementations a degraded disk set can be made (3 disk set of which 2 are online).

In the example above, a read request for block "A1" would be serviced by disk 0. A simultaneous read request for block B1 would have to wait, but a read request for B2 could be serviced concurrently by disk 1.

The slide also shows a Microsoft Word interface with a ruler and a drawing toolbar at the bottom.



Problem R.III – reuse image fragment



SVG – anchor / view



jpeg, gif, bmp, ...



Problem - Summary

hypertext link targets are often limited to **whole resources**

reuse of resource fragments is primarily **application dependent**

uniform fragment identification is missing



Approach

current status

HTML

WMP <PARAM name="currentPosition " value="5">

Quicktime <PARAM name="currentPosition " value="00:00:05.0">

SMIL

Continuous Media <object src="raid.avi" clipBegin="5s" />



Approach

proposed approach to increase interoperability

move syntax and semantics of fragment identification
from application to media format

```
<object src="raid.avi#mp(~time('npt','5'))" />
```



Approach – Identify Fragments

application

separate resource

e.g. MPEG-7 Locators

part of the resource identifier

e.g. MPEG-21 Part 17 FID

application

separate Resource

URI Fragment

<object src="raid.avi" clipBegin="5s" />

<object src="raid5s.mp7.xml" />

<object src="raid.avi#mp(~time('npt','5'))" />





Approach – Identify Fragments with URIs

pros

- vast application support for URIs
- fallback to whole resource if fragment is unsupported
- fragments are server/proxy independent
- independent of other resources

cons

- content negotiation needs to be considered
- cumbersome for complex fragments



Approach – Identify Fragments with URIs

How to achieve uniformity for fragment identification?

fragment identification depends on **media type**

provide standards with media type description

web uses **MIME types** to specify media types

fragment identification **SHOULD** be registered with **MIME type**



Requirements

requirements vary depending on application scenarios

media types SHOULD provide a solid basis for fragment identification

media format developers SHOULD reuse existing media type specifications



Requirements

Location of fragment definition

within the target

limits fragment definition to author of resource

robust

part of the fragment identification

independent fragment definition

consider modifications of target



Requirements

Identification Type

measured fragments

nominal fragments

structured fragments

Physical structures (bytes)

Logical structures (headings, tracks, ...)



Requirements

Presentation

What should a client/UA do if a fragment is used???

Highlights -> target class in CSS3

Context removal



Requirements

Context Removal

client

presentation issue

server

uniform way needed

- > Temporal URI: URI query as solution ?
reuse same syntax and semantics of fragment identification
<http://example.com/video.anx?t=15.2/18.7>



Requirements

Robustness

ensure valid fragment identifier

- > explicit definition inside the resource
limits fragment definition to author of resource
- > external algorithm to check changes
md5 hash



Requirements - Summary

Location of fragment definition

Identification Type

Presentation

Context Removal

Robustness





Next Steps

requirements

of application scenarios

specify

general **recommendations** for content types

specifications for subtypes

evaluate

toolset to build “Mingled Lessons”

extend existing applications to support fragment identifiers

evaluate benefits of uniform fragment identification