

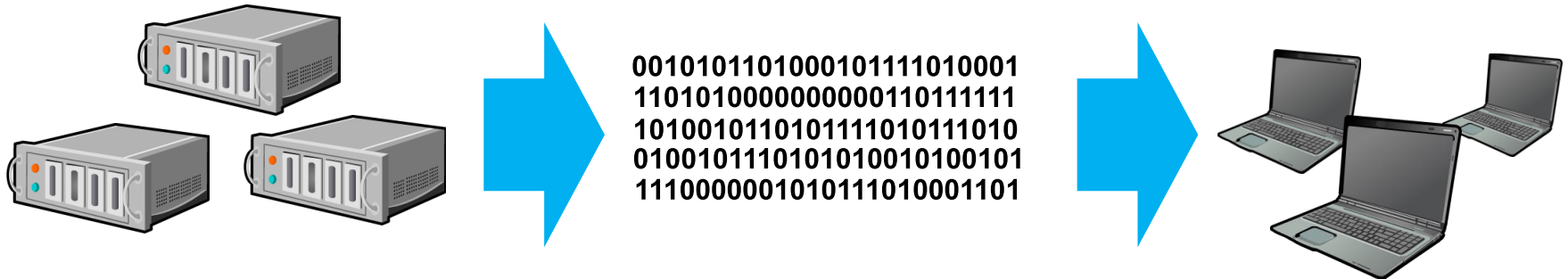
# **SMB2.2 Advancements for WAN**

**Molly Brown, Mathew George**  
**Windows File Server Team**  
**Microsoft Corporation**

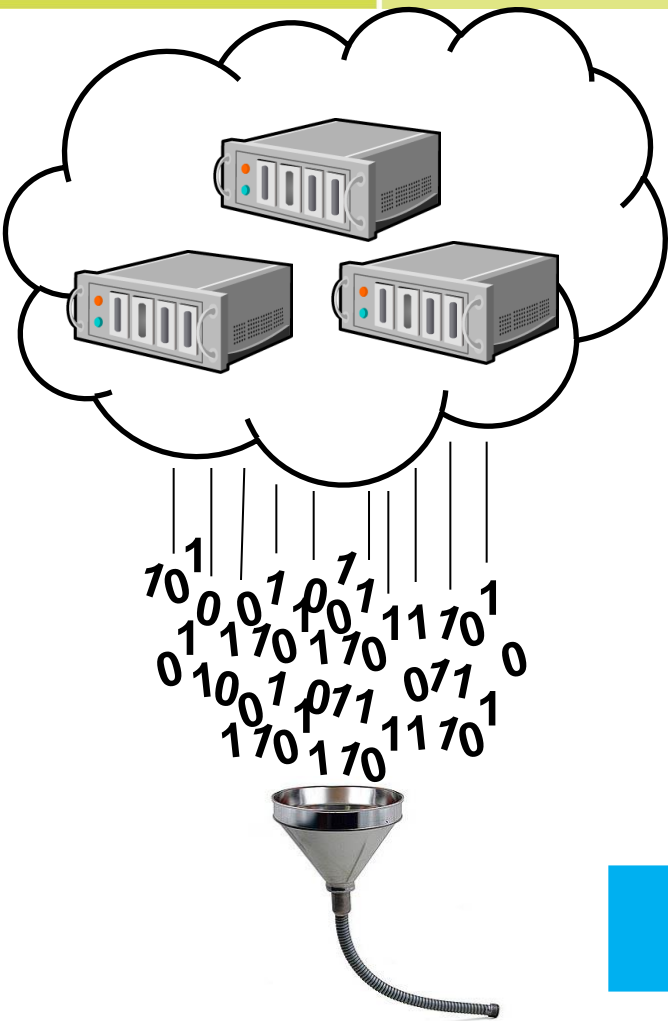
- ❑ BranchCache
  - ❑ Overview of BranchCache
  - ❑ Overview of improvements in BranchCache v2
  - ❑ Changes to SMB 2.2 protocol for BranchCache
- ❑ Directory Leases
  - ❑ Overview of Leasing Concepts
  - ❑ Semantics of Directory Leases
  - ❑ Changes to SMB 2.2 protocol
  - ❑ Implementation guidelines
  - ❑ Analysis & Observations

# BranchCache Problem Background

- When servers are close, bandwidth is free, applications are fast and everyone is happy



# BranchCache Problem Background



- ❑ When servers are far, bandwidth costs money, applications slow down and people get frustrated
- ❑ Nobody likes waiting

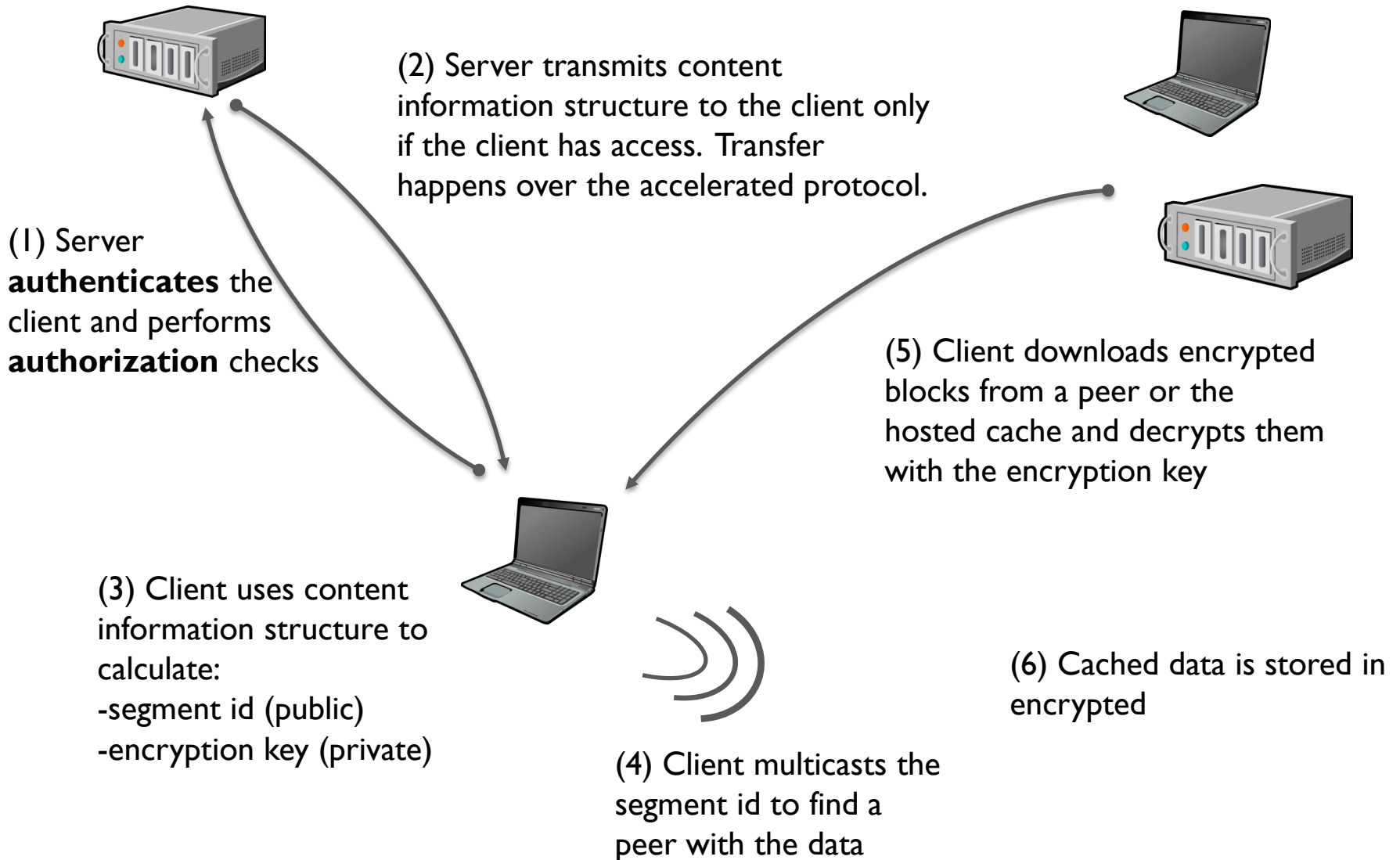


1010010110101111010111010

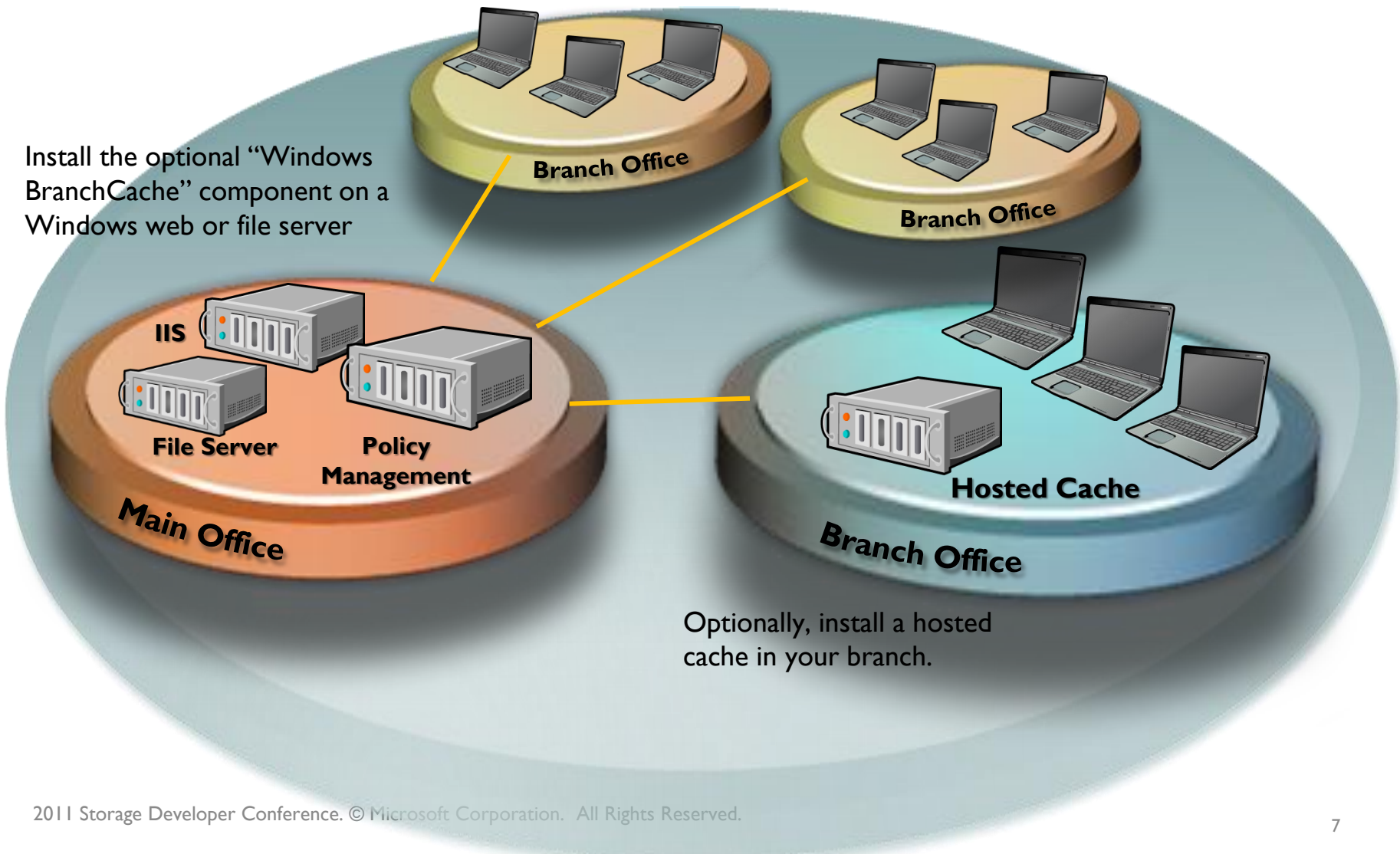


- Optimize end-to-end protocols
  - Maintains protocol integrity
  - Benefits from on-going protocol optimizations
  - Optimizes SSL, IPSec, SMB signing, HTTP, BITS and SMB

# BranchCache Keeps Data Secure



# BranchCache Deployment Options



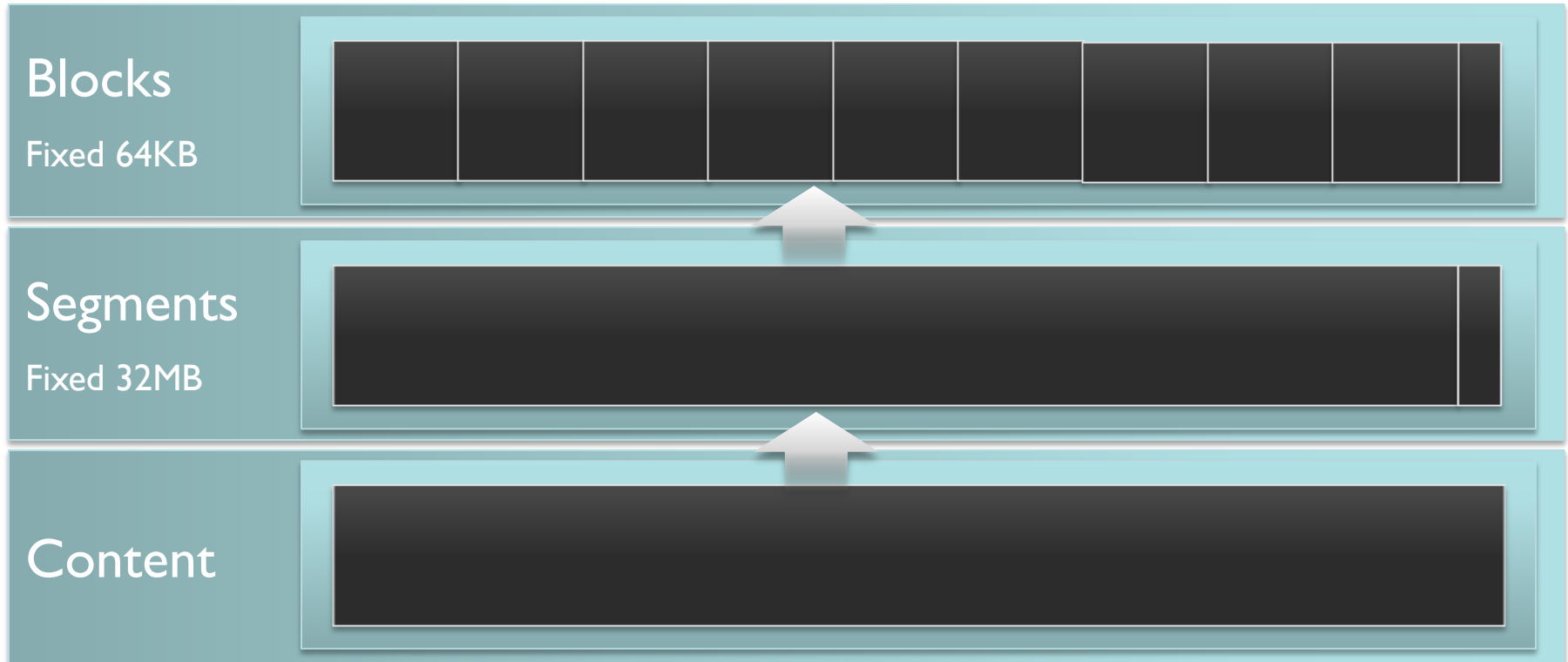
Install the optional “Windows BranchCache” component on a Windows web or file server

Optionally, install a hosted cache in your branch.

# BranchCache Improvements for Windows 8

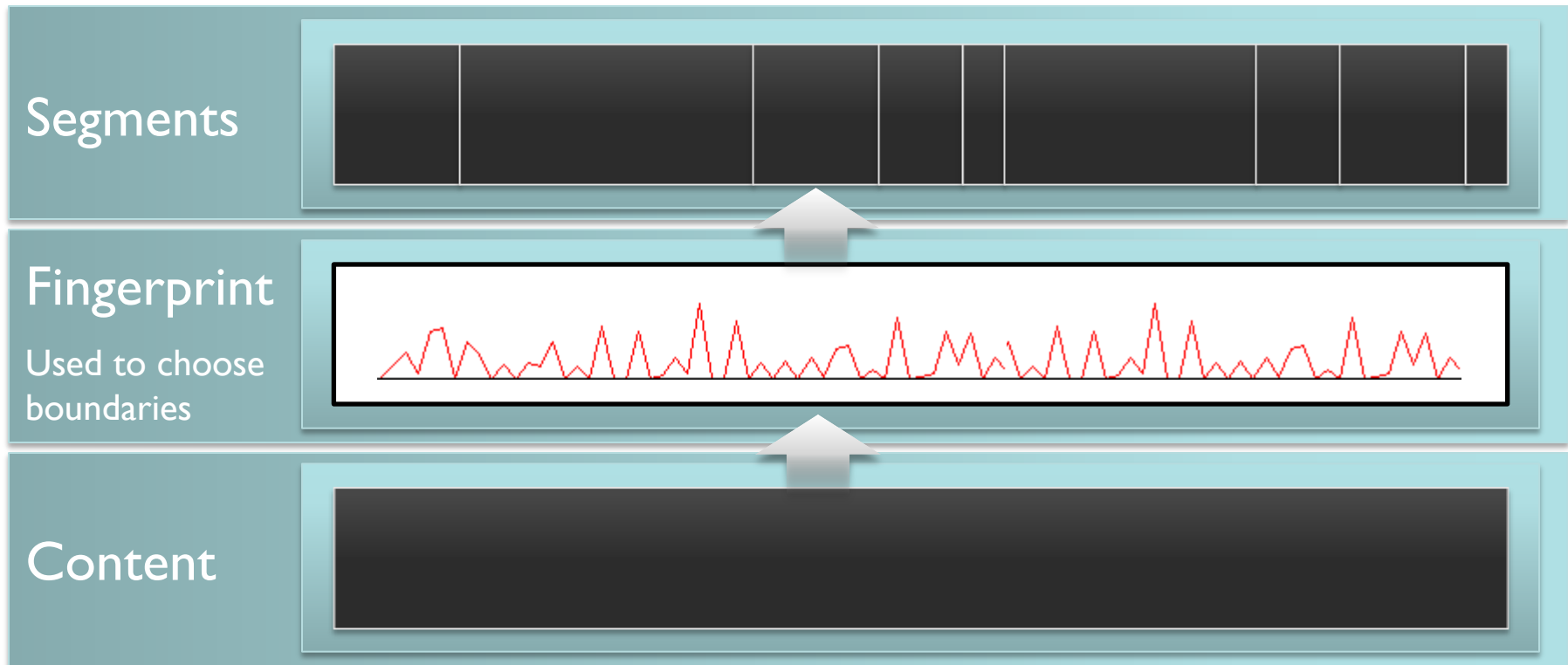
- ❑ BranchCache protocol level improvements
  - ❑ Variable sized segments to increase shared chunks within and across files
- ❑ SMB 2.2 protocol level improvements
  - ❑ Allows partial hash retrieval to improve initial download efficiency for large files
- ❑ Windows implementation improvements
  - ❑ Improved management
  - ❑ Support for larger branches

# BranchCache v1 Content Chunking



- ❑ Fixed size 32MB segments, consisting of fixed size 64KB blocks
- ❑ Both segments and blocks were hashed
- ❑ Publication and discovery at the segment level
- ❑ Retrieval done at the block level

# BranchCache v2 Content Chunking



- ❑ Segments are no longer broken down further into blocks
  - ❑ Windows implementation note: target 32-128KB sized segments
- ❑ Hashing, publication and discovery at segment level
- ❑ Approach increases likelihood of finding shared segments

- ❑ In v1, hash request offset and length are relative to Content Information itself
  - ❑ SMB has no way to determine how this mapped to actual file content, so full Content Information had to be retrieved
- ❑ In v2, hash request offset and length are relative to the file data
  - ❑ SMB can now retrieve the portion of the Content Information corresponding to a particular region of a file

- ❑ SMB/BranchCache integration remains the same
- ❑ A Windows 8 client will request v2 hashes when:
  - ❑ The file server is running a v2 capable SMB 2.2 server,
  - ❑ Windows 8 client-side BranchCache policy setting to force clients to request v1 hashes is **not** enabled, and
  - ❑ If the client is configured in hosted cache mode, clients must be configured to use v2 hashes; by default, v1 hashes are used by hosted cache

# TREE\_CONNECT Response

## SMB 2.2

Frame Number	Protocol Name	Description
49	SMB2	SMB2:C TREE CONNECT (0x3), Path=\\192.168.1.11\bc_share,
51	SMB2	SMB2:R TREE CONNECT (0x3), TID=0x5,
52	SMB2	SMB2:C CREATE (0x5), Da(...), Sh(RWD), DH2Q+MxAc+QFid, File=NULL@#52,
53	SMB2	SMB2:C OPEN (0x5), File=NULL@#53, File=NULL@#53,

### Frame Details

Frame: Number = 51, Captured Frame Length = 138, MediaType = ETHERNET

```
⊕ Ethernet: Etype = Internet IP (IPv4), DestinationAddress: [00-1C-C4-3C-CD-8F], SourceAdc
⊕ Ipv4: Src = 192.168.1.11, Dest = 192.168.1.9, Next Protocol = TCP, Packet ID = 51, Tc
⊕ Tcp: Flags=...AP..., SrcPort=Microsoft-DS(445), DstPort=54886, PayloadLen=84, Seq=284
⊕ SMBOverTCP: Length = 80
⊖ SMB2: R TREE CONNECT (0x3), TID=0x5,
  --- SMBIdentifier: SMB
  ⊕ SMB2Header: R TREE CONNECT (0x3), TID=0x0005, MID=0x0007, PID=0xFEFF, SID=0x0001
  ⊖ RTreeConnect: 0x1
    --- StructureSize: 16 (0x10)
    --- ShareType: Disk (0x1)
    --- Reserved: 0 (0x0)
    ⊖ ShareFlags: 24576 (0x6000)
      --- SHI1005_FLAGS_DFS: (.....0)
      --- SHI1005_FLAGS_DFS_ROOT: (.....0.)
      --- Reserved_bits3_4: (.....00..)
      --- SHAREFLAGS_CACHING_TYPE: (.....0000)
      --- SHI1005_FLAGS_RESTRICT_EXCLUSIVE_OPENS: (.....0....)
      --- SHI1005_FLAGS_FORCE_SHARED_DELETE: (.....0.....)
      --- SHI1005_FLAGS_ALLOW_NAMESPACE_CACHING: (.....0.....)
      --- SHI1005_FLAGS_ACCESS_BASED_DIRECTORY_ENUM: (.....0.....)
      --- SHI1005_FLAGS_FORCE_LEVELIII_OPLOCK: (.....0.....)
      --- SHI1005_FLAGS_ENABLE_HASH_V1: (.....1.....)
      --- SHI1005_FLAGS_ENABLE_HASH_V2: (.....1.....)
      --- Reserved_bits14_32: (0000000000000000.....)
```

Share supports v1 and v2 hash retrieval

# SRV\_READ\_HASH Request

## SMB 2.2

Frame Summary - SMB2

Frame Number	Protocol Name	Description
82	SMB2	SMB2:C IOCTL (0xb), FID=0x1100000011, 0x00094264 - Unknown File System Control Code,
83	SMB2	SMB2:R -NT Status: System - Error, Code = (13) STATUS_INVALID_PARAMETER_IOCTL (0xb),
84	SMB2	SMB2:C IOCTL (0xb), FID=0x1100000011, 0x001441BB - FSCTL_SRV_READ_HASH - Request to read set of hashes for ...

Frame Details

Frame: Number = 84, Captured Frame Length = 202, MediaType = ETHERNET

- Ethernet: Etype = Internet IP (IPv4), DestinationAddress: [00-1C-C4-3D-D0-F3], SourceAddress: [00-1C-C4-3D-D0-F3]
- Ipv4: Src = 192.168.1.9, Dest = 192.168.1.11, Next Protocol = TCP, Packet ID = 189, Total IP Len = 60
- Tcp: Flags=...AP..., SrcPort=54886, DstPort=Microsoft-DS(445), PayloadLen=148, Seq=3229757545
- SMBOverTCP: Length = 144
- SMB2: C IOCTL (0xb), FID=0x1100000011, 0x001441BB - FSCTL\_SRV\_READ\_HASH - Request to read set of hashes for a specific file

- SMBIdentifier: SMB
- SMB2Header: C IOCTL (0xb), TID=0x0005, MID=0x0019, PID=0xFEFF, SID=0x0001
- CIoctl:
  - StructureSize: 57 (0x39)
  - Reserved: 0 (0x0)
  - CtlCode: 0x001441BB - FSCTL\_SRV\_READ\_HASH - Request to read set of hashes for a specific file
  - FileId: Persistent: 0x1100000005, Volatile: 0x1100000011
  - InputOffset: 120 (0x78)
  - InputCount: 24 (0x18)
  - MaxInputResponse: 0 (0x0)
  - OutputOffset: 120 (0x78)
  - OutputCount: 0 (0x0)
  - MaxOutputResponse: 64512 (0xFC00)
  - Flags: (00000000000000000000000000000001) FSCTL request
  - Reserved2: 0 (0x0)
- SMB2IoctlSrvReadHashReq:
  - HashType: SRV\_HASH\_TYPE\_PEER\_DIST - the hash is requested for branch caching.
  - HashVersion: SRV\_HASH\_VER\_2 - Branch cache version 2
  - HashRetrievalType: SRV\_HASH\_RETRIEVE\_FILE\_BASED
  - Length: 1297629232 (0x4D5B3600)
  - Offset: 0 (0x0)

- Request v2 hashes
- Offset specified is relative to the file (only valid with SRV\_HASH\_VER\_2)

# SRV\_READ\_HASH Response

## SMB 2.2

### Frame Summary - SMB2

Frame Number	Protocol Name	Description
87	SMB2	SMB2:R IOCTL (0xb), File=Windows XP Pro Gold x86 FPP.vhd@#72, 0x001441BB - FSCTL_SRV_READ_HASH - Reque
155	SMB2	SMB2:A OPLOCK BREAK (0x12),
160	SMB2	SMB2:A OPLOCK BREAK (0x12)

### Frame Details

```
Frame: Number = 87, Captured Frame Length = 1514, MediaType = ETHERNET
Ethernet: Etype = Internet IP (IPv4), DestinationAddress: [00-1C-C4-3C-CD-8F], SourceAddress
Ipv4: Src = 192.168.1.11, Dest = 192.168.1.9, Next Protocol = TCP, Packet ID = 67, Total
Tcp: Flags=...A...., SrcPort=Microsoft-DS(445), DstPort=54886, PayloadLen=1460, Seq=28452
SMBOverTCP: Length = 64648
SMB2: R  IOCTL (0xb), File=Windows XP Pro Gold x86 FPP.vhd@#72, 0x001441BB - FSCTL_SRV_R
  SMBIdentifier: SMB
  SMB2Header: R IOCTL (0xb), TID=0x0005, MID=0x0019, PID=0xFEFF, SID=0x0001
  RIoCtl:
    StructureSize: 49 (0x31)
    Reserved: 0 (0x0)
    CtlCode: 0x001441BB - FSCTL_SRV_READ_HASH - Request to read set of hashes for a spec:
    FileId: Persistent: 0x1100000005, Volatile: 0x1100000011
    InputOffset: 112 (0x70)
    InputCount: 24 (0x18)
    OutputOffset: 136 (0x88)
    OutputCount: 64512 (0xFC00)
    Flags: 0 (0x0)
    Reserved2: 0 (0x0)
    InputData: Binary Large Object (24 Bytes)
  SMB2IoCtlSrvReadHash:
    FileBasedHashDataInfo:
      FileDataOffset: 0 (0x0)
      FileDataLength: 84592142 (0x50AC60E)
      BufferLength: 64486 (0xFB6E)
      Reserved: 0 (0x0)
    Header:
      HashType: SRV_HASH_TYPE_PEER_DIST - the hash is requested for branch caching.
      HashVersion: SRV_HASH_VER_2 - Branch cache version 2
      SourceFileChangeTime: 07/29/2011, 06:59:10.682340 UTC
      SourceFileSize: 1297823232 (0x4D5B3600)
```

Starting offset and length of file data that is described by the hashes returned

V2 hashes returned

# Summary of Protocol Changes

- ❑ Content Identification (MS-PCCRC)
  - ❑ Changes to support variable size segments
- ❑ Discovery (MS-PCCRD), Hosted Cache Offer (MS-PCHC)
  - ❑ Changes to support discovery and upload of smaller sized blocks without dramatic increase to LAN traffic
- ❑ SMB 2.2 protocol extensions for BranchCache (MS-SMB2)
  - ❑ Minor changes to support hash retrieval for file ranges
  
- ❑ Preview SMB protocol documentation will be posted this week at <http://msdn.microsoft.com/en-us/library/ee941641.aspx>
- ❑ Full BranchCache related protocol documentation will be available at beta of Windows Server 8

# **SMB 2.2**

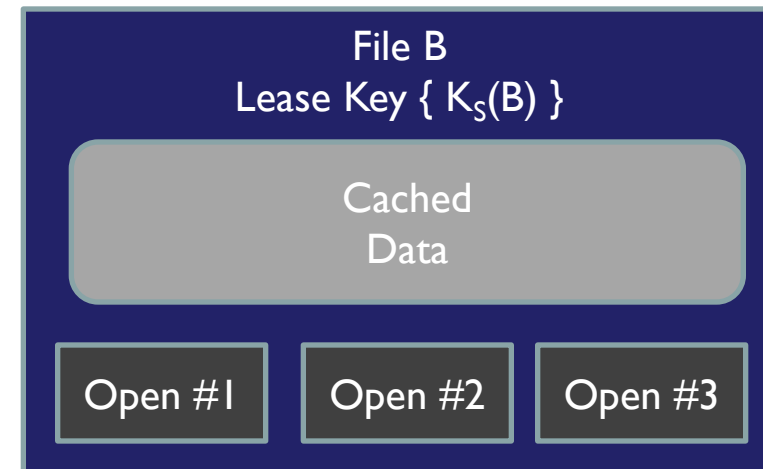
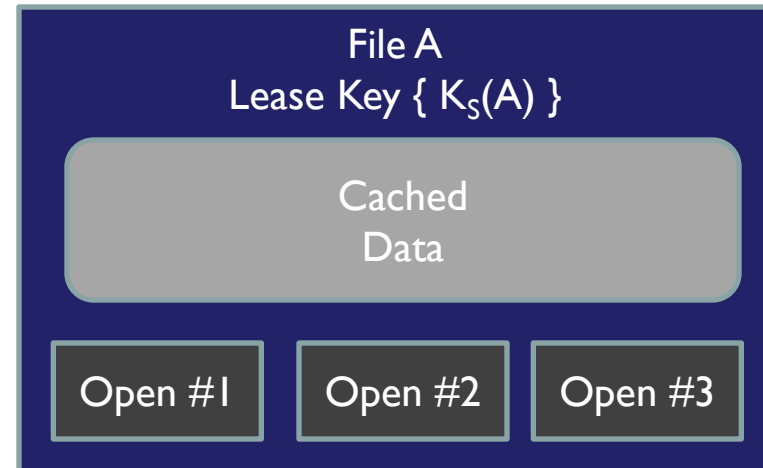
## **Directory Leasing**

Mathew George

- ❑ Move from a “best effort” metadata cache to a “near consistent” metadata cache.
- ❑ Optimize for the following scenarios
  - ❑ Home directory workload (read/write, no sharing.)
  - ❑ Publication scenarios (read only, sharing)
- ❑ Allow clients to cache directory metadata for extended periods of time.
  - ❑ Round trip reduction
  - ❑ Latency reduction
  - ❑ Server scalability

# Leasing Concepts

- ❑ Granular lease levels
  - ❑ H – Handle lease (shared)
  - ❑ R – Read lease (shared)
  - ❑ W – Write lease (exclusive)
- ❑ Multiple opens to the same file from the same client are tied together using a single “lease key”.
  - ❑ Avoids lease breaks due to changes made by the same client.
  - ❑ Single coherent view of cache.
- ❑ File lease key ( $K_S$ )



# What is a Directory Lease ?

- ❑ A Directory Lease protects
  - ❑ Directory Handles
  - ❑ Directory metadata
    - ❑ Applies only to immediate children of a directory.
- ❑ Directory lease levels
  - ❑ R lease protects the directory metadata
    - ❑ No acknowledgement needed for a R break.
  - ❑ H lease protects directory handles.
    - ❑ Without H leases, the R lease is of no value.
    - ❑ Needs acknowledgement
  - ❑ W leases are not granted.

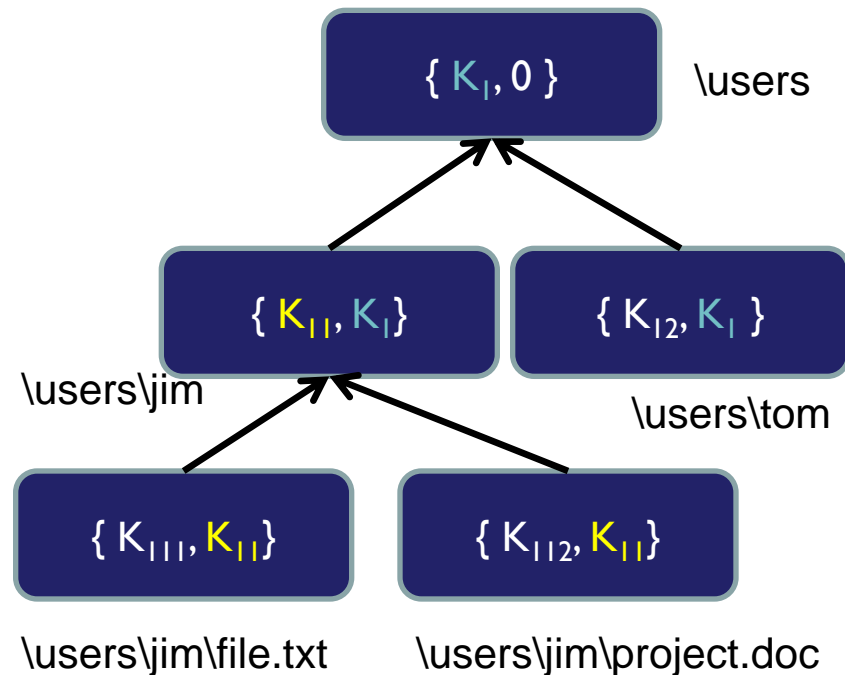
# What causes Directory Lease breaks?

- ❑ When directory metadata gets updated ( R lease break)
  - ❑ Creation of new children.
  - ❑ Rename of immediate child file/directory.
  - ❑ Deletion of immediate child.
    - ❑ Effective only when the last handle to child is closed !!
  - ❑ Modification of immediate children
    - ❑ (Usually) happens only when the modifying child handle is closed.
- ❑ When the directory handle itself causes a sharing conflict
  - ❑ Semantics are identical to H leases on a file.
  - ❑ Another conflicting open to the directory.
  - ❑ Rename / deletion of a parent directory.

# Directory Lease Keys

- ❑ Directory lease keys ( $K_p$ ) tie child opens to its parent (i.e. containing) directory handle.
  - ❑ All opens by the same client to files/subdirectories within a given directory share the same “directory lease key”.
- ❑ Client provides the parent – child association by supplying a key pair  $\{K_{\text{Self}}, K_{\text{Parent}}\}$  for each open handle.
- ❑ Lease on a directory open ( $p$ ) is broken via
  - ❑ Operation on child open ( $h$ ) : if  $K_p(h) \neq K_S(p)$
  - ❑ Operation on self open ( $h$ ) : if  $K_S(h) \neq K_S(p)$

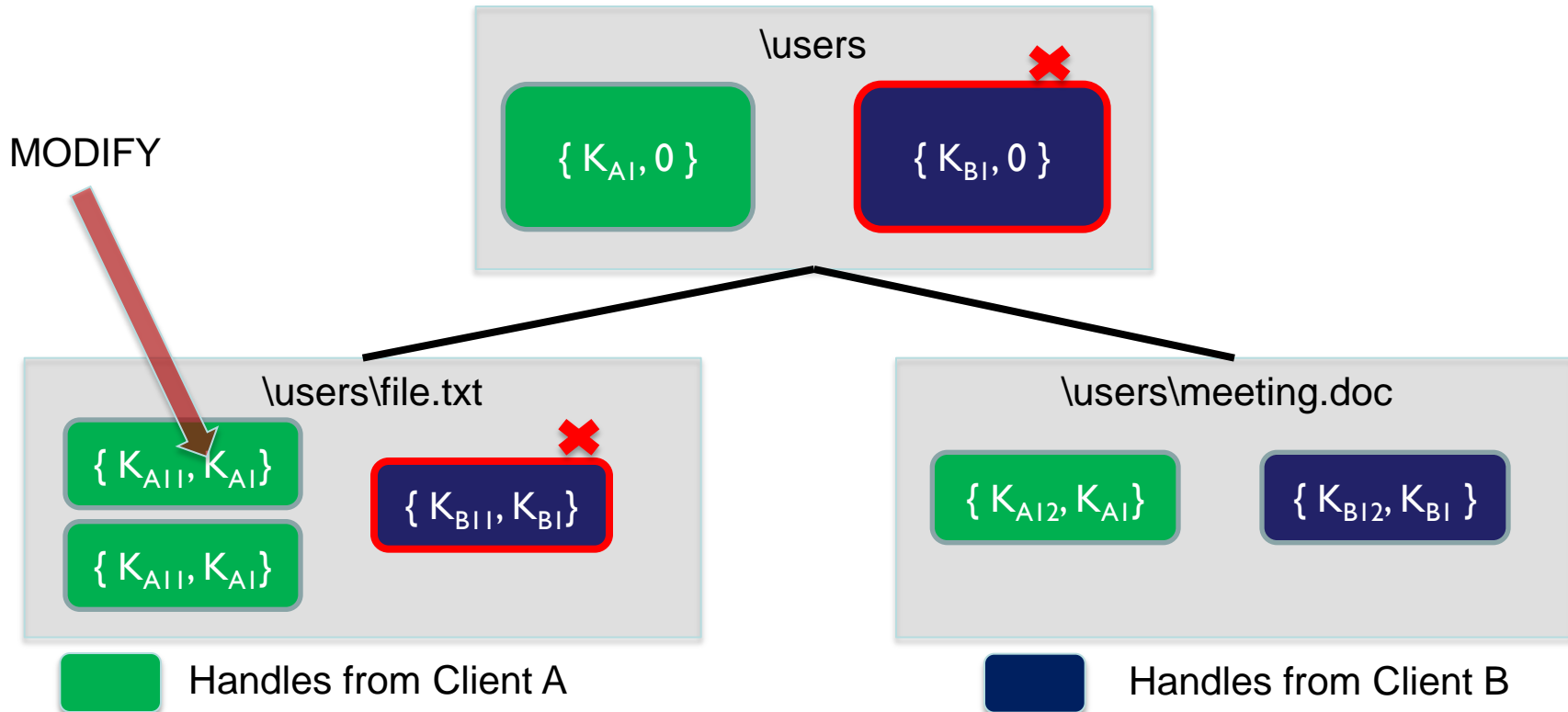
# Lease Key Hierarchy : Client View



- Directory lease keys ( $K_p$ ) tie child opens to its parent (containing) directory handle.
  - All opens by the same client to files / subdirectories within a given directory share the same “directory lease key”  $K_p$
- Client provides the parent – child association by supplying a key pair  $\{K_{Self}, K_{Parent}\}$  for each open handle.

# Lease processing : Server View

- ❑ Lease on a directory open ( $p$ ) is broken when:
  - ❑ Operation on child open ( $h$ ) : if  $K_p(h) \neq K_S(p)$
  - ❑ Operation on self open ( $h$ ) : if  $K_S(h) \neq K_S(p)$



# Changes to the SMB 2.2 protocol

- ❑ New lease V2 create context.
  - ❑ Adds the directory / parent lease key.
  - ❑ New flag to indicate presence of a directory lease key.
- ❑ No changes to lease break / acknowledgement messages.

```
#define SMB2_LEASE_FLAG_PARENT_LEASE_KEY_SET 0x04
```

```
typedef struct {
```

```
    GUID    LeaseKey;           // Unique ID which identifies owner of the lease.
    DWORD   LeaseState;        // Combination of R, W, H bits.
    DWORD   Flags;             // Optional: flags.
    INT64   LeaseDuration;     // Reserved. Must be ZERO.
    GUID     ParentLeaseKey;   // Lease key for parent directory.
    USHORT  Epoch;             // Current lease epoch number stored by the client.
    USHORT  Reserved;
```

```
} SMB2_ECP_REQUEST_LEASE_v2, // Client->Server.
   SMB2_ECP_GRANTED_LEASE_v2; // Server->Client.
```

# SMB 2.2 Client Implementation Choices

- ❑ Directory cache lifetime is tied to lifetime of the directory lease.
  - ❑ Windows 8 clients will delay-close directory handles with an H lease open for up to 10\* minutes. (\* may change.)
  - ❑ Only one handle is delay-closed per user.
- ❑ Client maintains multiple views of the same directory under the same lease.
  - ❑ Access Based Enumeration
  - ❑ All views are updated / invalidated together.
- ❑ Client can re-use delay-closed directory handles.
  - ❑ Must reset state using the SMB2\_REOPEN / SL\_RESTART\_SCAN flag in the query directory request.

# Interesting Windows Semantics

- ❑ When a file is deleted, the entry for the file is removed from the parent directory only when the last handle to the file is closed.
  - ❑ Which parent key is to be used to process the lease break?
    - ❑ Last closed handle?
    - ❑ Handle which set the delete disposition?
  - ❑ Windows implementation favors correctness over performance.
    - ❑ If parent key of handle on which delete disposition was set does not match the parent key of last closed handle, all leases on the parent directory are broken.
    - ❑ A problem only if multiple clients have open handles to a file being deleted.

- ❑ Lazy update of directory metadata can delay directory lease breaks
  - ❑ NTFS maintains 2 copies of file metadata – primary copy in the MFT and a duplicate in the parent directory.
  - ❑ Lazy update is done when a child handle is closed or via explicit metadata modifying operations (Set Info).
  - ❑ Directory metadata is immediately updated when a new file is created or when a file is renamed.
- ❑ Directory metadata for hard links are updated only when the file is opened via the hard link.
  - ❑ Indefinite delay for directory metadata to be updated.

# What are the gains?

- During SDC 2010, we modeled potential frame reduction for common FSCT scenarios using directory leases and estimated ~ 36% reduction in metadata frames.

Directory Oplocks	Change Notify	Close	Create	QueryDir
CmdLineFileDelete		-2	-3	-2
CmdLineFileDownload		-2	-2	-2
CmdLineFileUpload		-1	-2	
CmdLineNavigate		-3	-4	-4
ExplorerDragDropFileDownload		-4	-7	-2
ExplorerDragDropFileUpload		-6	-8	-4
ExplorerFileDelete	-2	-13	-14	-10
ExplorerNavigate		-8	-11	-4
ExplorerSelect		-1	-1	-2
WordEditAndSave		-3	-6	
WordFileClose	-2	-17	-19	-8
WordFileOpen		-21	-25	-14

# What are the gains?

- Preliminary measurements using a pre-release Windows 8 build show significant frame reduction for individual FSCT scenarios.

Scenario	Leasing Off	Leasing On	Change
ExplorerUpload	25	10	-60.0%
ExplorerNavigate	26	17	-34.6%
WordClose	63	44	-30.2%
ExplorerDelete	55	40	-27.3%
WordEditAndSave	99	82	-17.2%
WordOpen	88	73	-17.0%
CmdLineDelete	19	16	-15.8%
ExplorerDownload	26	22	-15.4%
CmdLineNavigate	10	9	-10.0%
CmdLineDownload	12	11	-8.3%
ExplorerSelect	13	12	-7.7%
CmdLineUpload	11	11	0.0%

# I/O rate during FSCT home folders workload run with 10,000 users.

<b>SMB2 Command</b>	<b>Leasing Off</b>	<b>Leasing On</b>	<b>Change</b>
Create	5737	4431	-22.8%
Close	5036	3887	-22.8%
Read	3377	3185	-5.7%
Write	769	706	-8.2%
QueryInfo	3288	3161	-3.9%
QueryDirectory	2789	1846	-33.8%
SetInfo	604	607	0.4%
Ioctl	1168	1169	0.1%
ChangeNotify	219	197	-9.9%
Cancel	132	131	-1.0%
BreakLease	0	113	N/A
AckBreakLease	0	4	N/A
Negotiate	48	0	-100.0%
SessionSetup	97	0	-100.0%
Logoff	49	0	-100.0%
TreeConnect	383	344	-10.2%
TreeDisconnect	375	333	-11.2%
Total	24071	20112	-16.4%

# Key Takeaways

- ❑ If you are implementing an SMB2 server (or a SMB2 protocol accelerator), supporting directory leases can benefit a broad spectrum of metadata-heavy workloads.
  - ❑ Accelerators can cache much more metadata for longer time intervals than typical clients.
  - ❑ Better consistency guarantees.
- ❑ Minimal changes to the protocol (and the backend filesystem.)
  - ❑ Relatively easy bang for the buck !