Discussion Section 8

Spring 2014

Assignment 5, Problem 1

Problem Statement

(a) Does it satisfy **monotonic read consistency**?
(b) Does it satisfy **monotonic write consistency**?

L1	$W(x_1)$				$WS(x_1;x_3)$
L2		$WS(x_1;x_2)$	$R(x_2)$	$R(x_1)$	

(c) Does it satisfy **read-your-writes**?
(d) Does it satisfy **writes-follow-read**?

L1		$WS(x_1;x_2)$	$R(x_2)$	
L2	$W(x_1)$			$WS(x_2;x_3)$

Assignment 5, Problem 1

Monotonic Read/Write Consistency

Monotonic Read Consistency:

If a process **reads** the value of a data item *x*, any **successive read** of *x* (at the **same** or a **different** data store) by the same process returns the **same** or a **more recent** value of *x.*

Monotonic Write Consistency:

If a process **writes** to a data item x, that operation must be **completed before** any **successive write** operation to x by the same process

Assignment 5, Problem 1

Monotonic Read/Write Consistency

L1	$W(x_1)$				$WS(x_1;x_3)$
L2		$WS(x_1;x_2)$	$R(x_2)$	$R(x_1)$	

(a) **No**, it does not satisfy **monotonic reads consistency** since process P first reads in L2 $R(x_2)$, i.e., the most recent version of x at that time, and then it reads $R(x_1)$, i. e., an older version of x.

(b) **No**, it does not satisfy **monotonic writes consistency** as the write $WS(x_1;x_3)$ in L1 only considers the initial write $W(x_1)$ in L1 and not the write on L2 $WS(x_1;x_2)$.

Assignment 5, Problem 1

Read-your-writes/Writes-follow-read Consistencies

Read-your-writes Consistency:

The effect of a **write** operation by a process of a data item *x* is **seen by a successive read** operation of *x* by the same process.

Writes-follow-read Consistency:

A **write** operation on a data item *x*, that **follows a read** operation on x by the same process, depends on the **same** or a **more recent** value of *x*

Assignment 5, Problem 1

Read-your-writes/Writes-follow-read Consistencies

L1		$WS(x_1;x_2)$	$R(x_2)$	
L2	$W(x_1)$			$WS(x_2;x_3)$

(c) **Yes**, it does satisfy **read-your-writes** consistency. The read $R(x_2)$ in L1 returns the write of x_2 which in turn considers the effect of the write $W(x_1)$ performed in L2 as indicated by the write operation $WS(x_1;x_2)$ in L1.

(d) **Yes**, it does satisfy **write-follows-reads** consistency as the last write in $WS(x_2;x_3)$ in L2 is consistent with the last value read in L1 $R(x_2)$.

Assignment 6, Problem 3

Distributed by Prof. Moser on Tuesday's lecture

Problems Chapter 9

Key Distribution Center

Q: What are the advantages and disadvantages of using centralized servers for key management?

Advantages

- Simplicity
- Efficient storage and management

Disadvantages

- Server could become a bottleneck
- New keys may not be established if server is compromised

Problems Chapter 9

Session Keys

Q: Does it make sense to restrict the lifetime of a session key? If so, give an example how that could be established.

Ans: Session keys should always have a restricted lifetime because they are easier to break compared to other cryptographic keys.

e.g. Sesame Authentication Protocol (an extension to Kerberos Protocol)

Problems Chapter 9

Access Control List (ACL)

Q: How are ACLs implemented in a UNIX file system?

Ans: Each file can have **three** associated entries: for the owner, for a group that is associated with the file and one for all the other users. For each entry, the access rights can be specified as *read*, *write* and *execute*.

Problems Chapter 10

Interface Definition Language (IDL)

Q: Why is it useful to define the interface of an object in an Interface Definition Language?

- From a practical point of view, it is easier to maintain and understanding objects if a well defined and unambiguous interface is used.
- Additionally, well defined interfaces are very valuable when developing studs and proxies.
- Finally, if an interface is well defined, it can be parsed and stored beforehand, which in turn allows dynamic invocations that can be used to construct at runtime in the client side.

Problems Chapter 10

Exceptions in languages

Q: Java and other languages support exceptions, which are raised when an error occurs. How are the exceptions implemented in RPCs and RMIs?

A: Exceptions will initially occur on the **server side** and as such, the server stub has to catch it and **return a special error code** to the client that initiated the request.
In the **client side**, the client stub has to **also raise an exception**.
This means that exceptions need to also be described in an Interface Definition Language (IDL).