Aspectual Development of a P2P Secure File Synchronization Application

R. Bertan Gündoğdu, Kaan Onarlıoğlu, Volkan Yazıcı
Bilkent University Department of Computer Engineering
06800, Ankara, Turkey
{*gundogdu,onarliog,vyazici*}*@cs.bilkent.edu.tr*

TAOSD Workshop, 2008

Introduction

P2P Architecture

- Provides scalability, fault-tolerance, and high adaptability to dynamic changes[8].
- No distinction between a client and a server.
- All the nodes in the system are considered "peers".

File Synchronization

- Studies the concept of synchronous distribution of a data state between multiple peers.
- All peers subscribed to a certain network have the same data state.

Key Concerns & Challenges

Key Concerns

- Reliability of the committed state after data transfer.
- Atomicity of the process.
- Authenticity and confidentiality.

Challenges

- Key concerns are cross-cutting.
- Cross-cutting concerns cannot be represented as first class elements by traditional software development paradigms.
  - Scattered concern.
  - Tangled code.
- Decrease software quality.

Aspect-Oriented Software Development

Aspectual Development

- Separation of cross-cutting concerns.
- Identifies and expresses cross-cutting concerns as first class elements.
- Provides additional language abstractions, called aspects.

P2P Secure File Synchronization with ...

- Object-Oriented Software Development
- Aspect-Oriented Software Development

Key Concepts

Peer-to-Peer Communication

- A popular architecture in modern network topology setups.
- Introduces scalability and fault-tolerance concepts.

File Synchronization

- Studies the methodology of data state propagation among multiple endpoints.
- Modifications propagated to other peers automatically.

Implementation Approaches

Master-to-Master Database Replication (Oracle[4], IBM DB2[5], etc.)

- Requires almost hardware level advanced locking.
- Complicated transaction atomicity methodologies.

Distributed Revision Control (git[2], mercurial, bazaar, etc.)

- Provides a perfect environment for working with groups of individuals.
- Impractical to have each node at the same data state by design.

Distributed File System (OpenAFS[10], etc.)

- Runs in kernel level. (Except FUSE[1] implementations.)
- Provides an easy to use traditional file system.

FSync

Overview

- Secure peer-to-peer file synchronization application.
- Synchronization of files across a virtual network.
  - Sync network
- Computers join sync networks.
  - Node
- Multiple network subscription is possible.
- P2P architecture, no master coordinator.
- Nodes issue updates to the shared content.
- Updates propagate automatically.

FSync

Requirements Analyzes

- Sync network management
- Joining/Leaving a sync network
- File synchronization
- Transactional file transfer
- Secure network association
- Secure file transfer
- Persistence

FSync

Object Oriented Design

- FSync: Top-level component.
- SyncNetwork
- Node
- Encryption: Abstract class representing encryption box.
- ConnectionManager: Handles transmission of control messages.
- FileManager: Handles transmission of files.

FSync

Observer Pattern

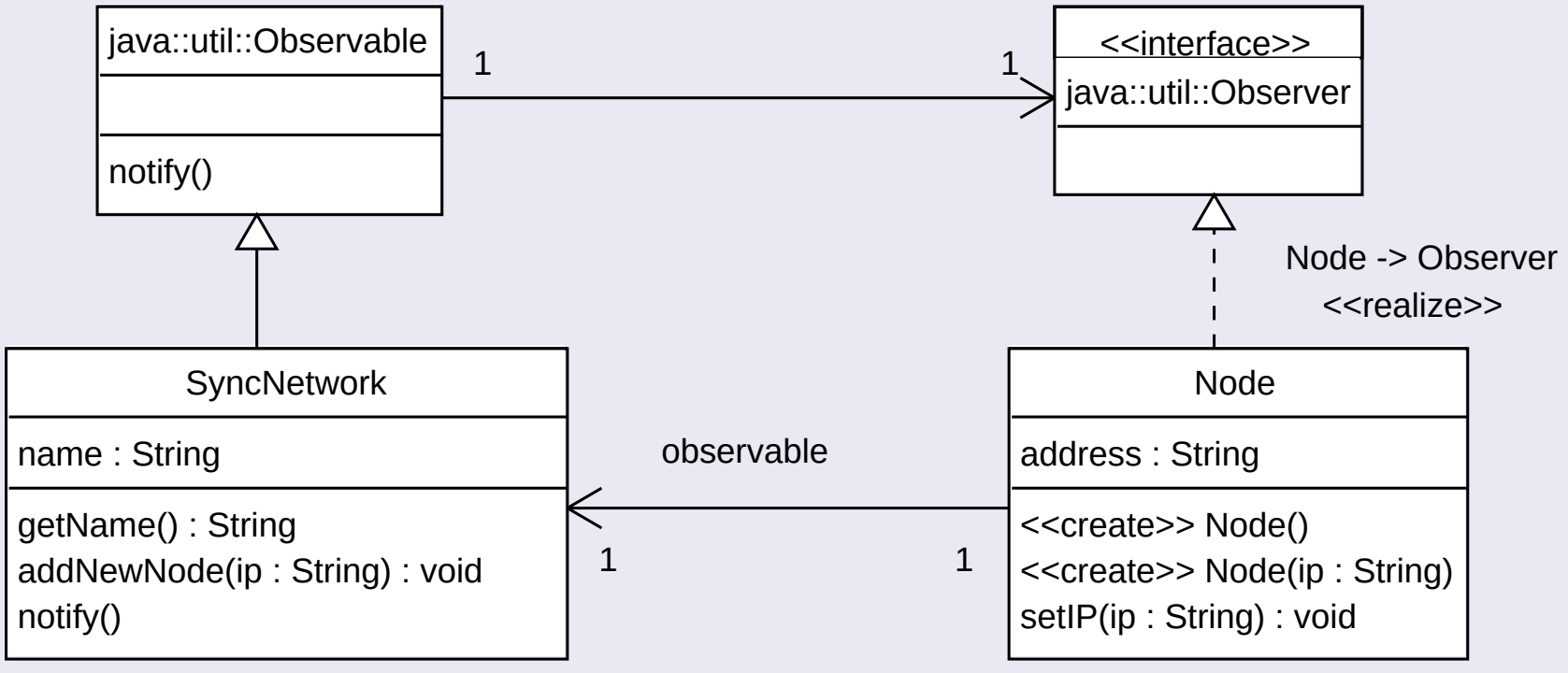


Figure: Application of Observer Pattern in FSync

FSync

Strategy Pattern

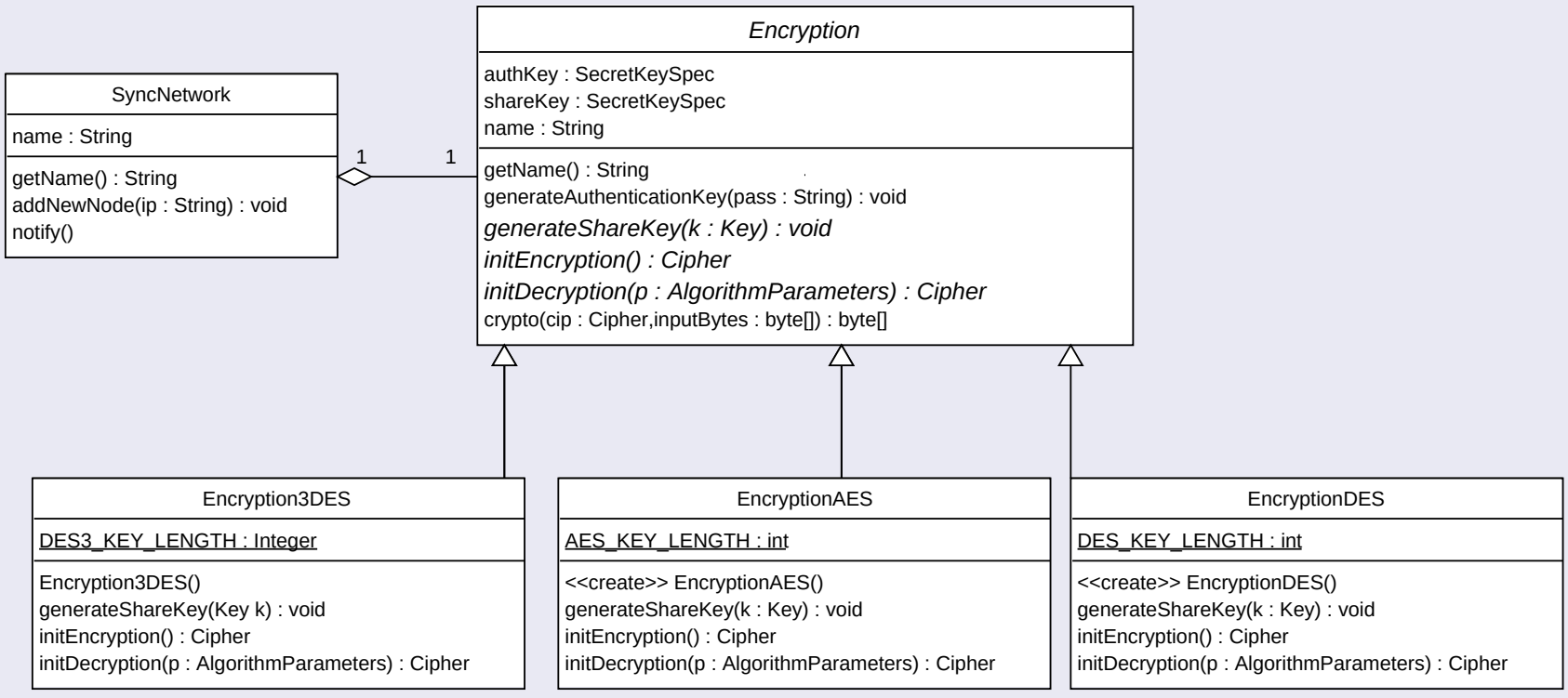


Figure: Application of Strategy Pattern in FSync

Problems with the Object Oriented Approach

Design State

- We have presented a strong object-oriented design.
- Object-oriented design enhanced by patterns.

Problem Statement

- Cross-cutting concerns!
- Due to the inherent inadequacy of the object-oriented approach.

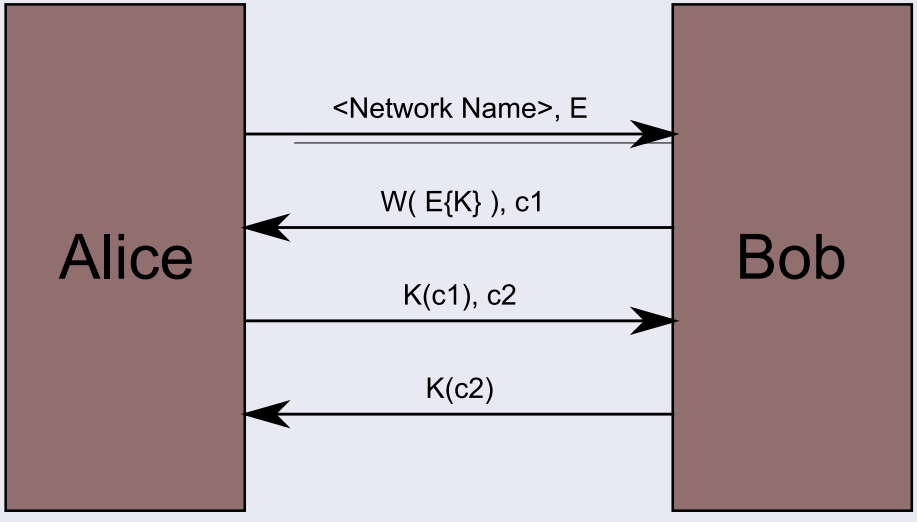
Cross-Cutting Concerns

- Authentication,
- Encryption,
- Transaction & data integrity,
- Persistence,
- Graphical user interface updates & notifications.

Cross-Cutting Concerns

Authentication

- Involves running a cryptography based protocol[3].



E: Per session generated asymmetric public key.
W: Weak symmetric key generated from a public key.
K: Shared public key.
c1, c2: Randomly generated challange

Figure: EKE (Encrypted Key Exchange) Protocol

Cross-Cutting Concerns

Authentication

- Part of the sync network connection protocol.
- Must be implemented within the ConnectionRequestWorker and ConnectionResponseWorker.
- Tangling!
- Scattering!

Change Scenario

- **Scenario:** Replace the EKE protocol with a simple password based login.
- **Impacts:** run() methods in the ConnectionRequestWorker and ConnectionResponseWorker must be modified.

Cross-Cutting Concerns

Encryption

- Involves encrypting the files to be transfer over a physical network.
- Must be implemented in UploadWorker, DownloadWorker, and SyncNetwork.
- Tangling!
- Scattering!

Change Scenario

- **Scenario:** Offer the option to bypass encryption.
- **Impacts:**
  - run() methods of DownloadWorker and UploadWorker must be modified.
  - Constructor of SyncNetwork must be modified.

Cross-Cutting Concerns

Transaction & Data Integrity

- Involves making the synchronization operations atomic and prevent going to an inconsistent state[9].
- Must be implemented in UpdateResponseWorker and UpdateRetryWorker.
- Tangling!
- Scattering!

Change Scenario

- **Scenario:** Instead of retrying the whole update, only request download of erroneous files.
- **Impacts:**
  - run() methods of UpdateResponseWorker and UpdateRetryWorker must be modified.
  - Scope of cross-cutting increased: UpdateRequestWorker.

Cross-Cutting Concerns

Persistence

- Involves persistently storing the latest state of the synchronized folders and the application configuration.
- Saving must be implemented in FSync, SyncNetwork, UpdateRequestWorker, UpdateResponseWorker and Encryption.
- Tangling!
- Scattering!

Change Scenario

- **Scenario:** Add new fields to the SyncNetwork and make them persistent: last update date, last update issuer, version, etc.
- **Impacts:**
  - Add set/get methods for new fields.
  - These methods must be populated with persistence code.
  - Scope of cross-cutting increases with each field.

Cross-Cutting Concerns

GUI Updates and Notifications

- Involves updating the GUI elements according to the state of the application.
- GUI needs to be addressed in almost all of the worker classes and in FSync.
- Tangling!
- Scattering!

Change Scenario

- **Scenario:** Add progress bars to display the synchronization progress.
- **Impacts:**
  - Add progress bar update calls in each of the UpdateResponseWorker, UploadWorker and DownloadWorker.
  - Scope of cross-cutting increases with every new feature.

Aspect-Oriented Programming

Identifying Aspects

- Authentication aspect,
- Encryption aspect,
- Transaction aspect,
- Persistence aspect,
- UIAspect.

Implementation Language

- AspectJ[7],
- Ppularity,
- Good support by development environments.

Aspect-Oriented Programming

Authentication Aspect

```
public aspect AuthenticationAspect {
    // variable declarations
    ...
    // pointcut definitions
    pointcut newOInputStream( ConnectionRequestWorker w):
        set( InputStream ConnectionRequestWorker.from) && this( w);
    pointcut newOutputStream( ConnectionResponseWorker w):
        set( OutputStream ConnectionResponseWorker.to) && this( w);
    // advice definitions
    ...
}
```

Aspect-Oriented Programming

Authentication Aspect

```
public aspect AuthenticationAspect {
    // advice definitions
    after( ConnectionRequestWorker worker)
        : newInputStream( worker) {
            # send E;
            # receive W( E{K}) and  c1;
            # extract K, send K(c1);
            # receive and verify K(c2);
    }

    after( ConnectionResponseWorker worker)
        : newOutputStream( worker) {
            # receive E;
            # send W( E{K}) and  c1;
            # receive and verify K(c1), send K(c2);
    }

    // support methods
    // performs encryption operations
    private byte[] crypto( ... ){...}

    // encrypts a given key
    private byte[] keyWrap( ... ) {...}

    // decrypts a given key
    private Key keyUnwrap( ... ) {...}

    // computes a key from a password
    private SecretKeySpec deriveKey( ... ) {...}
}
```

Aspect-Oriented Programming

Encryption Aspect

```
public aspect EncryptionAspect {
    // variable declarations
    private Cipher DownloadWorker.cipher;
    private Cipher UploadWorker.cipher;
    // pointcut definitions
    pointcut sendIV( UploadWorker w):
        set( ObjectOutputStream UploadWorker.to) && this( w);

    pointcut getIV( DownloadWorker w):
        set( ObjectInputStream DownloadWorker.from) && this( w);
    ...

    // advice definitions
    // initializes the encryption and sends IV
    after( UploadWorker w): sendIV( w) {
        w.cipher = getNetwork( w.name).encyption.initEncryption();
        Parameters params = w.cipher.getParameters();
        w.to.writeObject( params.getEncoded());
    }

    // receives IV and initializes decryption
    after( DownloadWorker w): getIV( w) {
        byte[] param = (byte[])w.from.readObject();
        Encryption encryption = getNetwork( w.name).encryption;
        Parameters ap = Parameters.getInstance();
        ap.init( param);
        w.cipher = encryption.initDecryption( ap);
    }
}
```

Aspect-Oriented Programming

Encryption Aspect

```
public aspect EncryptionAspect {
    // variable declarations
    ...
    // pointcut definitions
    pointcut encryptBuffer( UploadWorker w):
        call( * FileInputStream.read(..))
        && this( w);

    pointcut decryptBuffer( DownloadWorker w):
        call( * ObjectInputStream.readFully(..))
        && this( w);
    ...
    // advice definitions
    // encrypt data buffer
    after( UploadWorker w): encryptBuffer( w) {
        SyncNetwork sn= getNetwork( w.name);
        w.buffer = sn
            .encryption
            .crypto( w.cipher, w.buffer);
    }

    // decrypt data buffer
    after( DownloadWorker w): decryptBuffer(w) {
        SyncNetwork sn= getNetwork( w.name);
    }
}
```

Aspect-Oriented Programming

Encryption Aspect

```
public aspect EncryptionAspect {
    // variable declarations
    private Encryption SyncNetwork.enc;

    // pointcut definitions
    pointcut newSyncNetwork( String pass, Encryption.Method m, SyncNetwork nw):
        execution( SyncNetwork.new(..))
        && this( nw)
        && args( .., pass, m);
    ...
    // advice definitions
    // instantiate an encryption object
    after( String pass, Encryption.Method m, SyncNetwork nw):
        newSyncNetwork( pass, m, nw) {

        if( m == Encryption.Method.DES)
            nw.enc = new EncryptionDES();
        else if( m == Encryption.Method.AES)
            nw.enc = new EncryptionAES();
        # else if...

        nw.enc.generateShareKey();
        nw.enc.generateAuthenticationKey( pass);
    }
}
```

Discussions

Aspect-Oriented Software Development

- Helps separate cross-cutting concerns[6].
- Extends flexibility by modularity.
- Supports quality factors.
- But,
  - One needs to be aware of the detailed uses of every aspect.
  - Vulnerable to the programmer mistakes and result in hard to spot bugs.

Conclusions

Conclusions

- Presented object-oriented design for FSync.
- Identified cross-cutting in concerns in file synchronization systems.
- Enhanced the design by aspect-oriented programming paradigm.
- Experienced that AOSD increases the modularity and quality in file synchronization systems.
- AOSD will mature in time through well established design methodologies and with better tool support.

Bibliography

Fuse: Filesystem in userspace.

http://fuse.sourceforge.net/.

Git - fast version control system.

http://git.or.cz/.

S. M. Bellovin and M. Merritt.

Encrypted key exchange: password-based protocols secure against dictionary attacks.
pages 72–84. IEEE, 1992.

O. Corporation.

Oracle real application clusters 11g technical overview.
http://www.oracle.com/technology/products/database/clustering/pdf/twp_rac11g.pdf, 2007.

L. J. Gu, L. Budd, A. Cayci, C. Hendricks, M. Purnell, and C. Rigdon.

A Practical Guide to DB2 UDB Data Replication V8.
IBM Redbooks, 2002.

W. L. Hursch and C. V. Lopes.

Separation of concerns.
Technical report, 1995.

G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W. G. Griswold.

An overview of aspectj.
pages 327–353. Springer-Verlag, 2001.

D. Malkhi, L. Novik, and C. Purcell.

P2p replica synchronization with vector sets.
SIGOPS Oper. Syst. Rev., 41(2):68–74, 2007.

C. Mastroianni, G. Pirrò, and D. Talia.

A hybrid architecture for content consistency and peer synchronization in cooperative p2p environments.
In *InfoScale '08: Proceedings of the 3rd international conference on Scalable information systems*, pages 1–9, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

C. M. University and I. P. Labs.

Open source implementation of the andrew distributed file system.
http://www.openafs.org/.