

A Feed Bundle Protocol for Scuttlebutt

Bachelor Thesis

Natural Science Faculty of the University of Basel
Department of Mathematics and Computer Science
Computer Networks
<http://cn.dmi.unibas.ch/>

Examiner: Prof. Dr. Christian Tschudin
Supervisor: Prof. Dr. Christian Tschudin

Jannik Jaberg
jannik.jaberg@unibas.ch
2017-054-370

02.07.2020

Acknowledgments

I would like to thank Prof. Dr. Christian Tschudin for giving me the opportunity to work together with him on this thesis. He supported me during the whole process of planning and developping this thesis with his valuable and constructive suggestions and guidance, especially in these special times. In addition, I thank Christopher Scherb and Claudio Marxer for supporting my work with essential feedback. Finally, I want to express my gratitude to my whole family and friends for supporting me in any part of this thesis.

Abstract

Mini recap of the whole thesis

- environnement
- problem
- goal
- outcome

Table of Contents

Acknowledgments	ii
Abstract	iii
1 Introduction	1
1.1 Secure Scuttlebutt	1
1.2 Motivation	1
1.3 Goal	1
1.4 Outline	1
2 Secure Scuttlebutt	2
3 Concepts and Architecture	3
3.1 Contracts	3
3.1.1 Client-ISP Contract	3
3.1.2 ISP-Server Contract	3
3.2 Feeds	5
3.3 Remote Procedure Call	6
3.3.1 Send Request	6
3.3.2 Read Request	6
3.3.3 Send Result	6
3.3.4 Read Result	6
3.4 Introducing and Detrucing	7
3.4.1 Introducing	7
3.4.2 Detrucing	7
3.5 Replication	8
3.6 Bundling	9
3.6.1 Adapted Introducing and Detrucing	9
3.6.2 Multiplexing and Demultiplexing	9
3.7 Outlook	9
3.7.1 P2P ISP Nodes	9
3.7.2 Contract between ISPs	9
4 Implementation	10

4.1	Feeds	10
4.2	RPC	10
4.3	Introducing	10
4.4	Replication	10
4.5	Multiplexing	10
5	Evaluation	11
5.0.1	Indexing - Keep Track of Progress	11
5.0.2	Stability - Robustness	11
6	Conclusion and Future Work	12
7	Body of the Thesis	13
7.1	Structure	13
7.1.1	Sub-Section	13
7.1.1.1	Sub-Sub-Section	13
7.2	Equations	13
7.3	Tables	13
7.4	Figures	14
7.5	Packages	14
8	Conclusion	15
	Bibliography	16
	Appendix A Appendix	17
	Declaration on Scientific Integrity	18

1

Introduction

1.1 Secure Scuttlebutt

What is Secure Scuttlebutt - describe on network layer and what application it is suited for.

The little story of the founder

Importance

1.2 Motivation

difference to centralized internet as is right now Onboarding Problem data on wire problem

1.3 Goal

Take SSB and bring it into a business perspective.

Solve or reduce onboarding problem

Reduce load on wire since ssb replicates every feed to every peer after each addition - probably better only one feed and distribute to peers over 'centralized' instance.

1.4 Outline

First better description on SSB

Then concepts and architecture of FBP with respect to SSB.

Closer look on code level how it is solved.

in the Evaluation what are problems with these approached strategies and how to fix them or other ideas.

Finally the conclusion and future work that showed during the process

2

Secure Scuttlebutt

More detailed overview about SSB. Feed distribution problem General onboarding problem
Paralellism between pubs and ISPs.

3

Concepts and Architecture

3.1 Contracts

First and foremost of the whole connectivity, protocol and bundling, the business contracts between the nodes has to be declared. Throughout the whole process of the these contracts and relationships between each participant got questioned and modified, since this is the most important aspect or basic building block for the whole thesis.

3.1.1 Client-ISP Contract

In the first version of the FBP there are only two participants: Client and ISP. Before connecting, a contract between these two parties has to be closed. In this version it consists of the public key, private key, public key of the opponent and the feed ids of the ISP and Client feed. The public keys act as the name of each client or isp, the private keys are obviously used to write to the feed. The feed IDs act as identifier for in which feed has to be written or from which feed has to be read.

Client-Contract	value	ISP-contract	value
actual public key:	cli001	actual public key:	isp001
actual private key:	*****	actual private key:	*****
actual feed ID:	cli001_isp001	actual feed ID:	isp001_cli001
ISP public key	isp001	Client public key:	cli001
ISP feed ID	isp001_cli001	Client feed ID:	cli001_isp001

3.1.2 ISP-Server Contract

Same as in the Client-ISP Contract.

ISP-contract	value	Server-Contract	value
actual public key:	isp001	actual public key:	ser001
actual private key:	*****	actual private key:	*****
actual feed ID:	isp001_ser001	actual feed ID:	ser001_isp001
Server public key:	ser001	ISP public key:	isp001
Server feed ID:	ser001_isp001	ISP feed ID:	isp001_ser001

Clearly seen, that client and server are 'connected' over isp001 - initial idea was a peer to peer isp network where the ISP-Company distributes the feeds and packages in a defined way - so actually isp could stand for swisscom and the number 001 the actual connectivity provider stations which means cli001 has a contract with isp342 and server has a contract with isp903. And swisscom has internal routing to pass information from isp342 to isp903. Result:

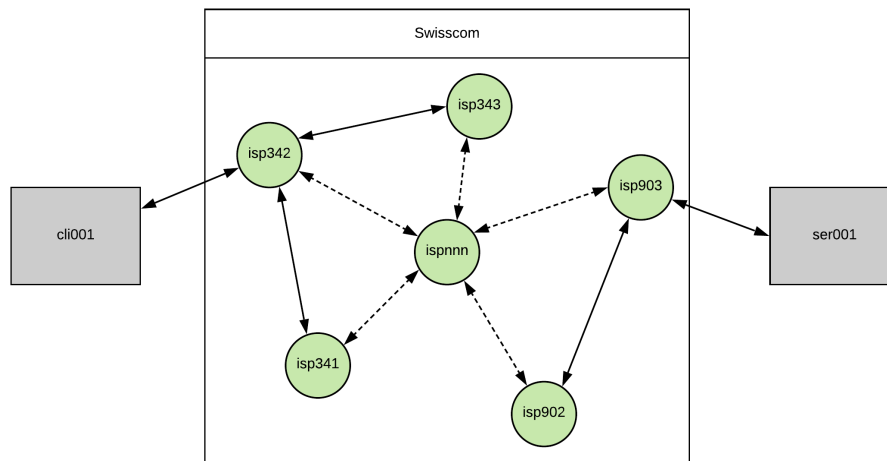


Figure 3.1: A simplified contract network.

Since now a contract is established, the client should hold information for the ISP and vice versa. This leads to the Feeds.

3.2 Feeds

Feeds are append only logs/files. Which means there can only be written to or read from. There is no option to delete entries except by deleting the whole file. So a feed acts as database for all the requests made or information given in the context of the connection between client and isp.

more detailed information about schema of feed, hashchain, sequence and content

also give context to keys - cryptography - signing

role of the feeds in fbp and introducing the idea of feed pairs. Given in the contract these feed IDs are already defined and each node knows where to store or write its own information and read the peers information. Result:

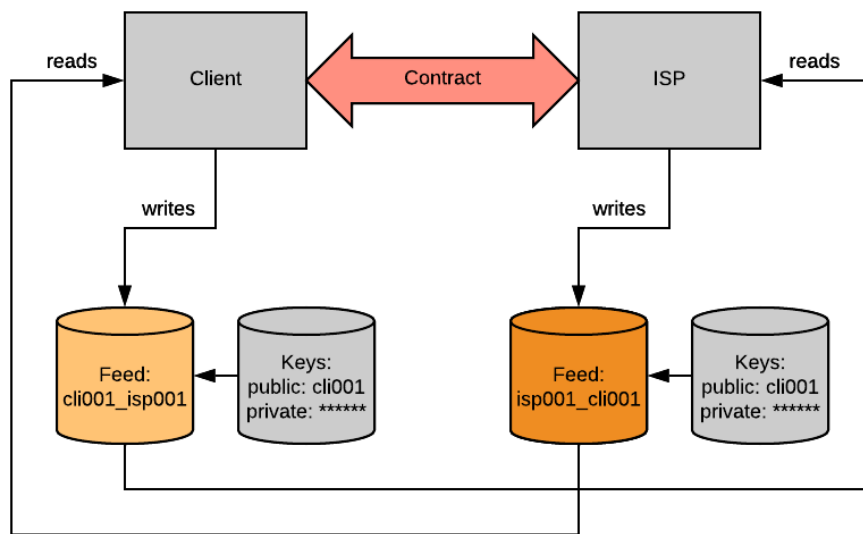


Figure 3.2: Full contract between client and ISP with feeds.

Having this setup the next step is to have a possibility to communicate, so the client can request information from the isps real database

3.3 Remote Procedure Call

General explanation of RPC.

3.3.1 Send Request

Format of request and api of method

3.3.2 Read Request

Format of request and api of method

3.3.3 Send Result

Format of request and api of method

3.3.4 Read Result

Format of request and api of method

Resulting schema of the API and descriptions given above:

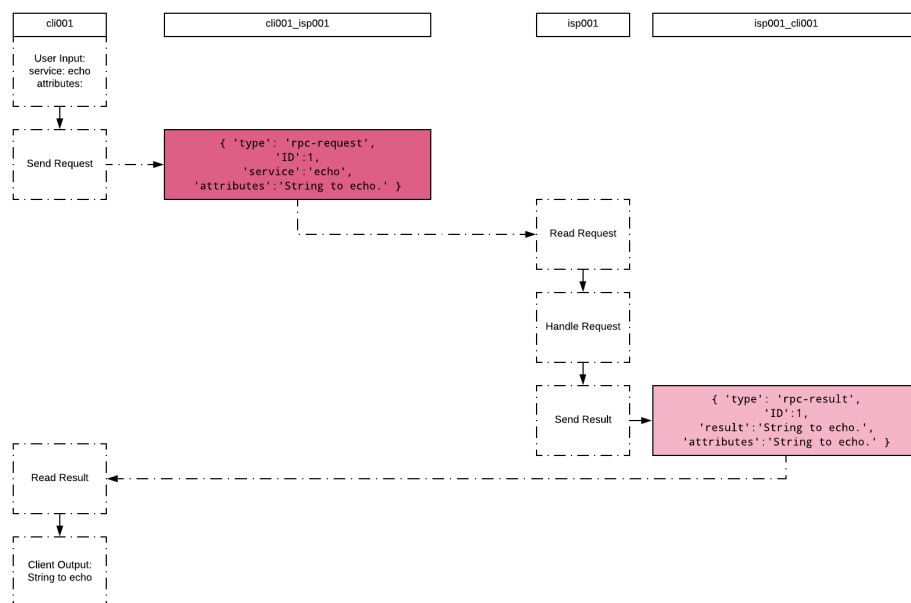


Figure 3.3: A Remote Procedure Call by cli001 to isp001.

Picture description: -j will be split into above sections

Seen in the picture above, client cli001 makes a request of the echo service with the attribute: String to echo. This is passed to the send_request function which assigns a unique ID for this request, resulting from the already given ids in the feed pair. Also merges type, id, service, and attributes into a request (a defined datastructure), writes appends this as a new log entry to the feed and saves the ID for keeping track of open requests. isp001 now detects a change on the feed cli001_isp001 and invokes the read_request method which takes

the request appart and evaluates the service. Now the request gets handled and the wanted value returned. After that send result is invoked. It is similar to the send request, difference is the type and the result is also appended to the result datastructure. Now again, writing a new log entry to the feed with the given content. cli001 is notified that the isp001_cli001 feed is changed. so read result is invoked and the result gets either to the program, where it was requested or printed to the client.

3.4 Introducing and Detrucing

3.4.1 Introducing

General Idea - onboarding to a server. Instead of following server - introduce to server. differs from pub sub pattern in the way that the server has the choice to accept or decline a client. in either way accept or decline an answer is provided to the client.

procedure - cli001 sends rpc request with service:introduce, attribute:ser001 to isp001. isp001 now makes rpc request with information about client and the fact, taht it wants to introduce itself to ser001 since they have a contract. if ser001 accepts introduce and creates from information sub feed - feed with communication direct between client and server. result is contract information for client given by server: feed ID etc. else ser001 declines, so result is no contract. isp001 gets the result and passes this result to the initial rpc request from the client. client now gets result - also builds subfeed according to contract and connection is established. now if client wants to use a service from server it only writes the request in the corresponding feed and procedure is the same as described in RPC Section.

important addition, only client can introduce, so only client can ask server for a contract. server has no information about any cient in the network at initial state.

3.4.2 Detrucing

Detrucing as a newly invented word by me, since normally after you introduced yourself to a person there is no way to make this unhappened. it acts the same for unfollow in a pub sub domain. But in contrary to the introduce both parts of the contract can detrue.

precisely either client or server can send rpc request to isp service detrue which gets propagated to opponent and results in terminating the whole contract. -i deleting keys and feeds. Important addition: after detrue client can again introduce to server.

Resulting picture:

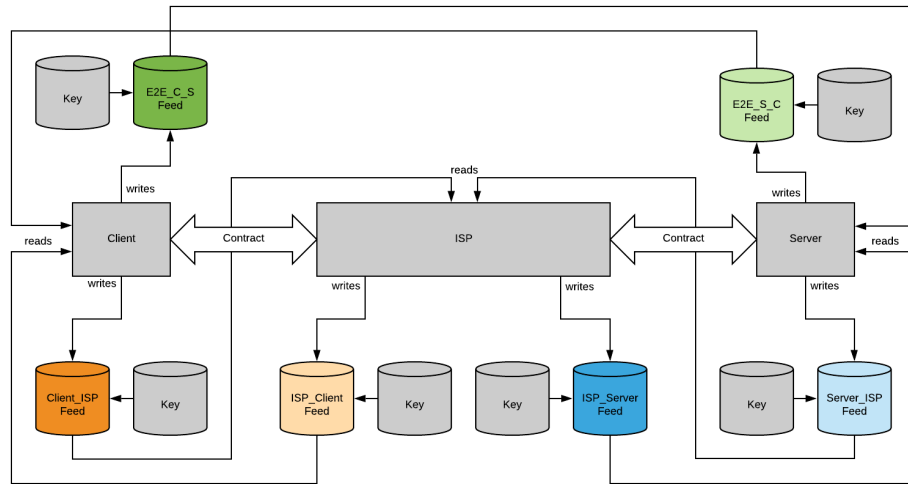


Figure 3.4: State after accepted introduce from cli001 to ser001

3.5 Replication

locally in testing environment this above works. for real world application there needs to be a replication mechanism which propagates feeds to destination.
feed replication works at each write to the feed, to the defined location.

API

could be erweitert to many destinations with an easy effort
resulting picture:

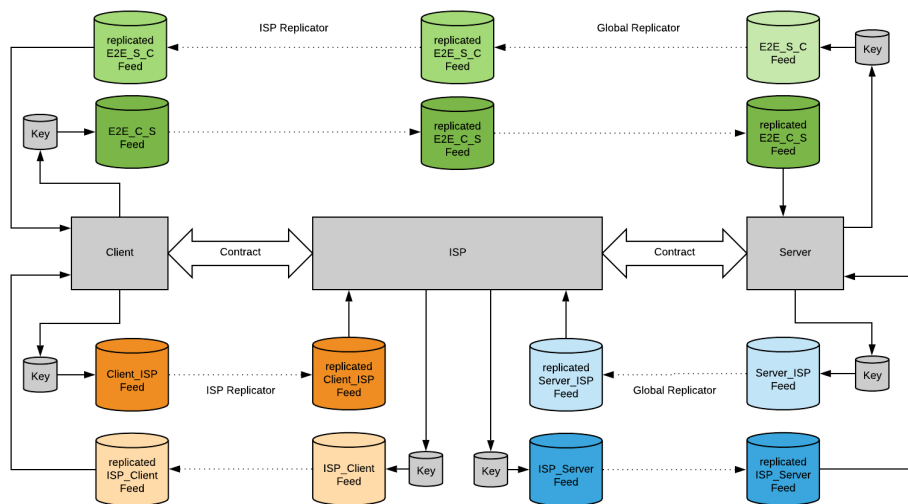


Figure 3.5: Replicated Feeds

seen that e2e feed pair between client and server now are replicated over isp node.
now bundling can be provided.

3.6 Bundling

taking again a look at the real world problem the isp has arbitrary many clients and many of them want to cumminicate with the same server. Idea: bundling all e2e feeds to same server into isp-server feed. -i load on wire problem

3.6.1 Adapted Introducing and Detrucing

introducing to server same - after server accept server generates both feeds request and result feed - sends contract to isp and isp generates result feed and replicates to cli - finally client generates request feed and replicates to isp.

3.6.2 Multiplexing and Demultiplexing

requests from client same way to isp. but isp does not replicate feed anymore. it takes whole log entry, signed by client and mulutplexes into a new log entry with a mux type content and signs this - mutliplexing. at server, server takes this mux type and extracts whole request feed entry and appends bytes to request feed from client - demultiplexing. handles request and writes result to result feed. from there the whole entry gets again multiplexed in the ispser feed. at isp isp demux entry and also appends bytewise to result feed. replicates to cli and cli can use result.

Resulting Schema:

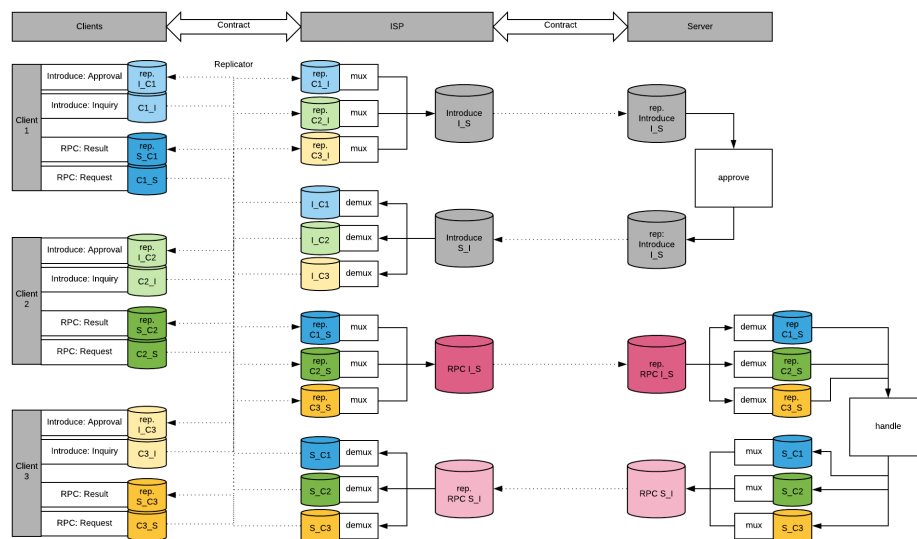


Figure 3.6: multiplexing

3.7 Outlook

3.7.1 P2P ISP Nodes

3.7.2 Contract between ISPs

4

Implementation

4.1 Feeds

Given by Prof. Tschudin explain structure

4.2 RPC

API with all methods in context of feeds and peers Adapted from RPyC so in the service class all services can be defined and have no impact on

4.3 Introducing

how is it implemented

4.4 Replication

Replicator Class that is given to each 'feed' so every time on this feed is operated by wr replicate to predefined location

4.5 Multiplexing

mux package idea is to only pass through the request then write in 'replication' feed and work from there. also for answering channel

5

Evaluation

5.0.1 Indexing - Keep Track of Progress

using sequence numbers for keeping track and ID in entries is only

5.0.2 Stability - Robustness

6

Conclusion and Future Work

7

Body of the Thesis

This is the body of the thesis.

7.1 Structure

7.1.1 Sub-Section

7.1.1.1 Sub-Sub-Section

Paragraph

Even Sub-Paragraph This is the body text. Make sure that when you reference anything you use labels and references. When you refer to anything, you normally capitalise the type of object you reference to, e.g. Section 7.1 instead of section 7.1. You may also just use the `cref` command and it will generate the label, e.g., for Section 7.1, we did not specify the word “Section”.

Hint: Try to structure your labels as it is done with `sec:my-label` and `fig:machine`, etc.

7.2 Equations

A Turing Machine is a 7-Tuple:

$$M = \langle Q, \Gamma, b, \Sigma, \delta, q_0, F \rangle \quad (7.1)$$

A Turing Machine is a 7-Tuple even if defined in the text, as in $M = \langle Q, \Gamma, b, \Sigma, \delta, q_0, F \rangle$.

7.3 Tables

Some tables can also be used as shown in Table 7.1¹. Remember that tables might be positioned elsewhere in the document. You can force positioning by putting a `ht!` in the definition.

¹ Table captions are normally above the table.

Table 7.1: Frequency of Paper Citations. By the way: Make sure to put the label always after the caption, otherwise \LaTeX might reference wrongly!

Title	f	Comments
The chemical basis of morphogenesis	7327	
On computable numbers, with an application to the ...	6347	Turing Machine
Computing machinery and intelligence	6130	

7.4 Figures

Figures are nice to show concepts visually. For organising well your thesis, put all figures in the Figures folder. Figure 7.1 shows how to insert an image into your document. Figure 7.2 references a figure with multiple sub-figures.

Missing: Description figure.

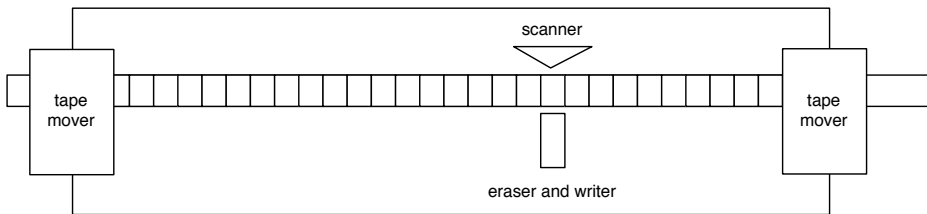
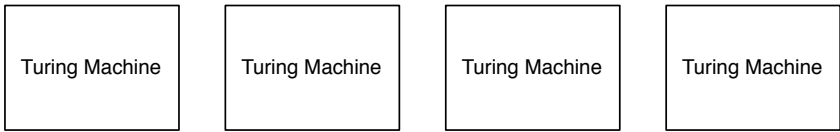


Figure 7.1: A Turing machine.



(a) Turing Machine 1 (b) Turing Machine 2 (c) Turing Machine 3 (d) Turing Machine 4

Figure 7.2: Plots of four Turing machines

7.5 Packages

These packages might be helpful for writing your thesis:

- caption** to adjust the look of your captions
- glossaries** for creating glossaries (also list of symbols)
- makeidx** for indexes and the back of your document
- algorithm**, **algorithmicx**, **algpseudocode** for adding algorithms to your document

8

Conclusion

This is a short conclusion on the thesis template documentation. If you have any comments or suggestions for improving the template, if you find any bugs or problems, please contact me.

Good luck with your thesis!

Bibliography



Appendix

Declaration on Scientific Integrity

Erklärung zur wissenschaftlichen Redlichkeit

includes Declaration on Plagiarism and Fraud
beinhaltet Erklärung zu Plagiat und Betrug

Author — Autor

Jannik Jaberg

Matriculation number — Matrikelnummer

2017-054-370

Title of work — Titel der Arbeit

A Feed Bundle Protocol for Scuttlebutt

Type of work — Typ der Arbeit

Bachelor Thesis

Declaration — Erklärung

I hereby declare that this submission is my own work and that I have fully acknowledged the assistance received in completing this work and that it contains no material that has not been formally acknowledged. I have mentioned all source materials used and have cited these in accordance with recognised scientific rules.

Hiermit erkläre ich, dass mir bei der Abfassung dieser Arbeit nur die darin angegebene Hilfe zuteil wurde und dass ich sie nur mit den in der Arbeit angegebenen Hilfsmitteln verfasst habe. Ich habe sämtliche verwendeten Quellen erwähnt und gemäss anerkannten wissenschaftlichen Regeln zitiert.

Basel, 02.07.2020

Signature — Unterschrift