Unsupervised Detection of Microservice Trace Anomalies through Service-Level Deep Bayesian Networks

Ping Liu[1], Haowen Xu[1], Qianyu Ouyang[1], Rui Jiao[1], Zhekang Chen[4]
Shenglin Zhang[3], Jiahai yang[1,5], Linlin Mo[2], Jice Zeng[2], Wenman Xue[2], Dan Pei[1]

1 清華大學 Tsinghua University

2 WeBank 微众银行

3 南開大學 Nankai University

4 必示 BIZSEER

5 鹏城实验室 PCL Peng Cheng Laboratory

Background

Design

Evaluation

Conclusion

Background

Design

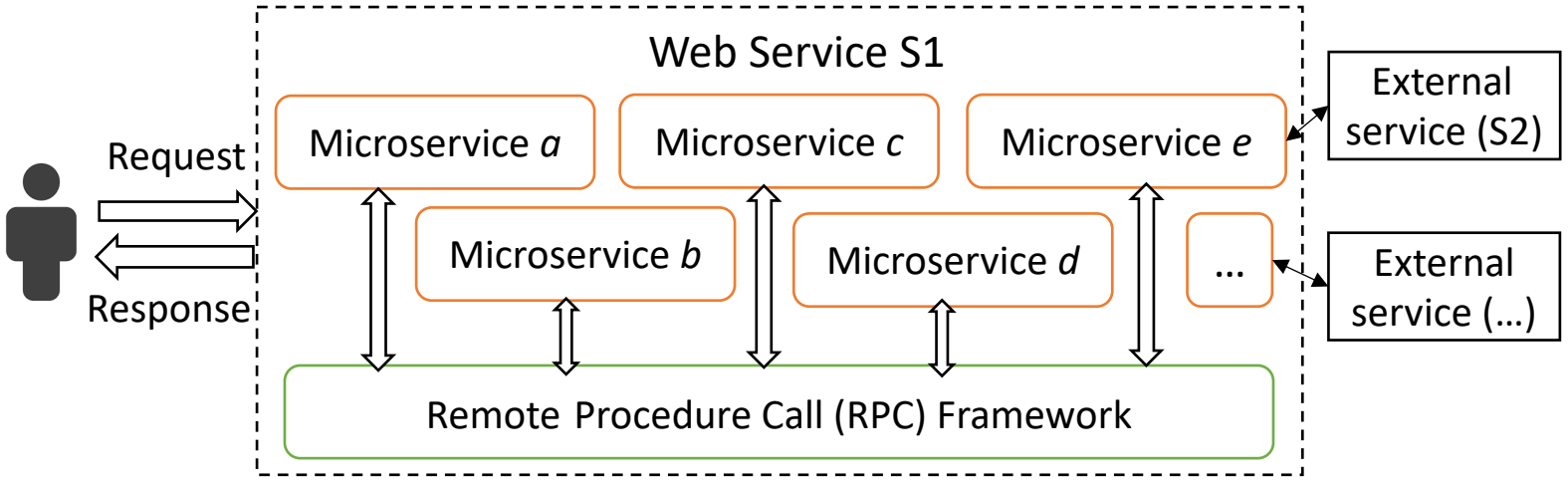
Evaluation

Conclusion

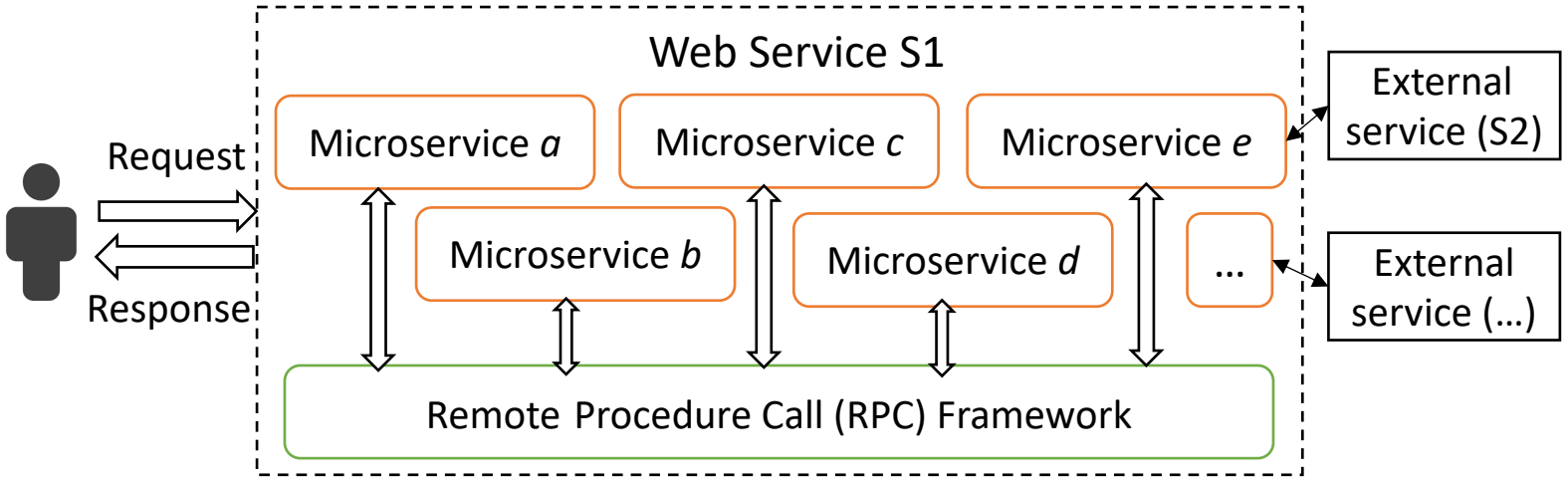
Background

What is a **trace**?

Microservice architecture

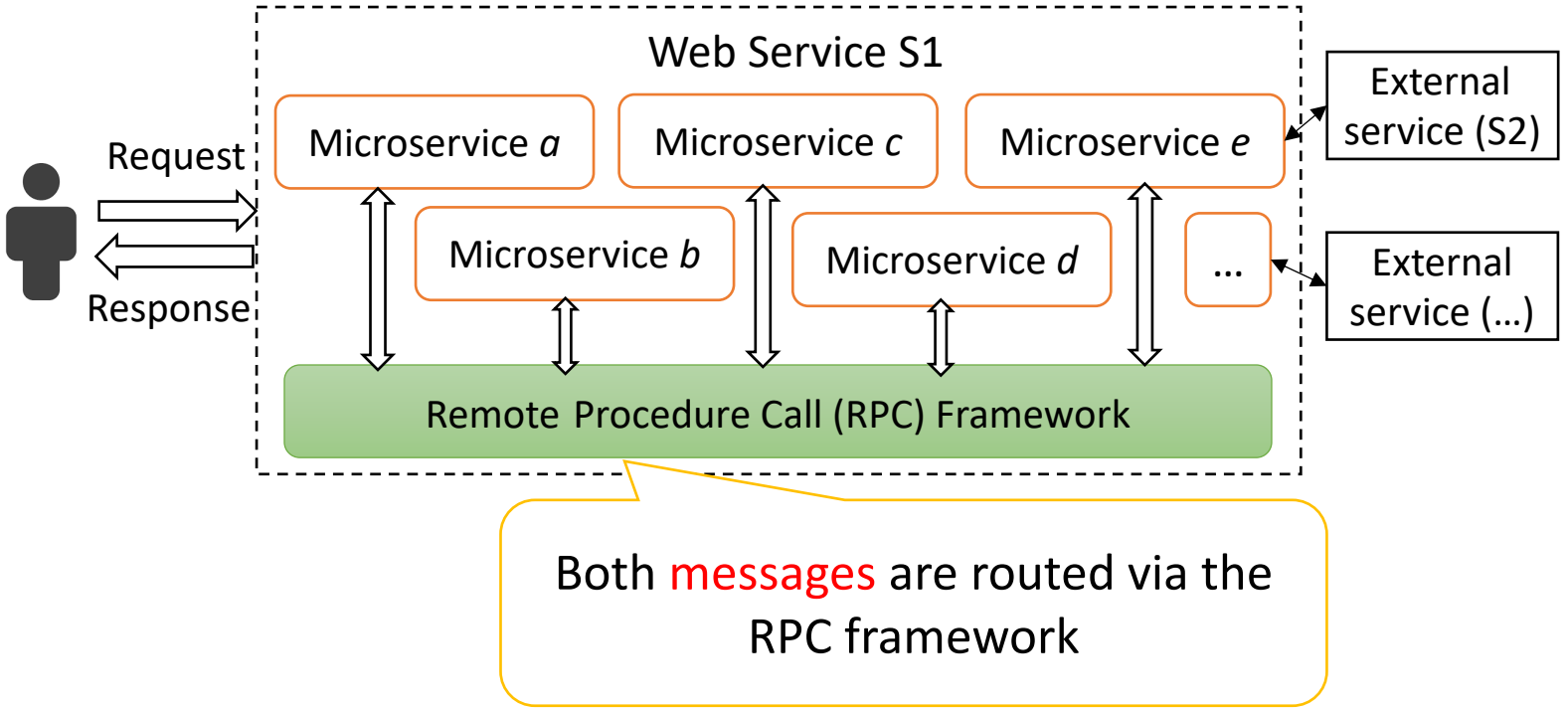


Microservice architecture

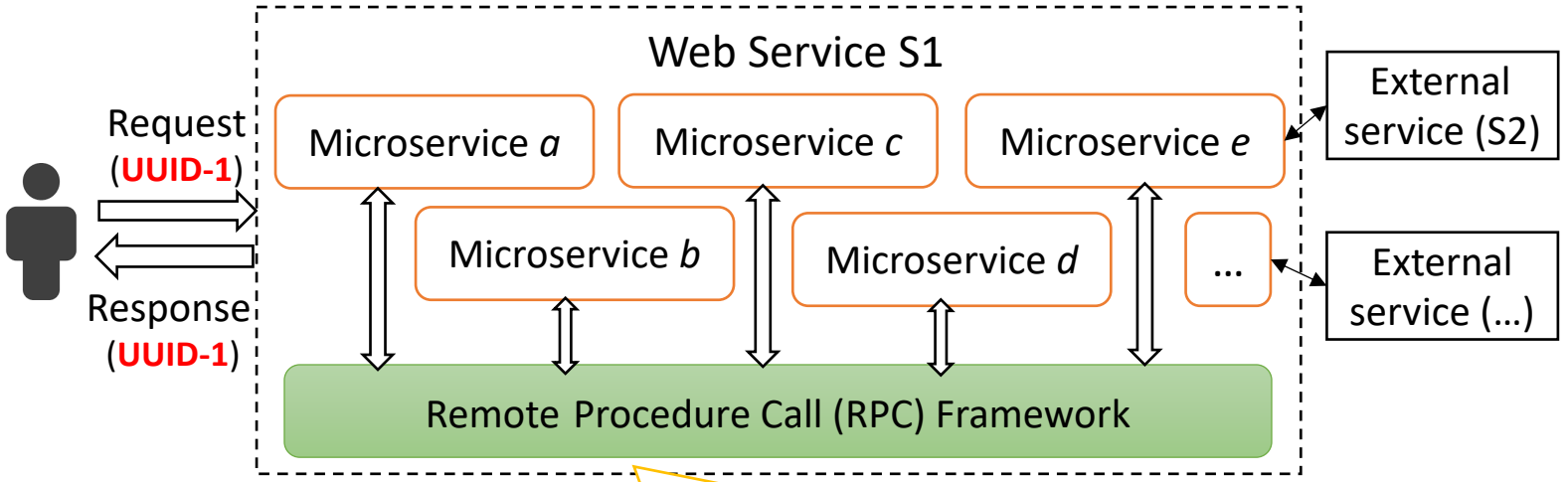


When microservice *m* calls microservices *n*, *m* sends a message to *n*

Microservice architecture



Tracing mechanism



The tracing mechanism can naturally record these messages with timestamps and UUIDs

Trace

- All messages with the same UUIDs constitute one **trace**

Message order	UUID	Sending time of (m → n) at m (msec)	Receiving time of (m → n) at n (msec)	Message (m→n)
①	UUID-1	-	1519747202138	call(start→a)
②	UUID-1	1519747202144	1519747202146	call(a→b)
③	UUID-1	1519747202149	1519747202151	call(b→c)
④	UUID-1	1519747202149	1519747202150	call(b→d)
⑤	UUID-1	1519747202155	1519747202156	*response*(c→b)
⑥	UUID-1	1519747202159	1519747202160	call(d→e)
⑦	UUID-1	1519747202188	1519747202190	*response*(e→d)
⑧	UUID-1	1519747202194	1519747202196	*response*(d→b)
⑨	UUID-1	1519747202253	1519747202256	call(b→e)
⑩	UUID-1	1519747202323	1519747202324	*response*(e→b)
⑪	UUID-1	1519747202355	1519747202356	*response*(b→a)
⑫	UUID-1	1519747202360	-	*response* (a→end)

Background

How to detect **trace anomalies**?

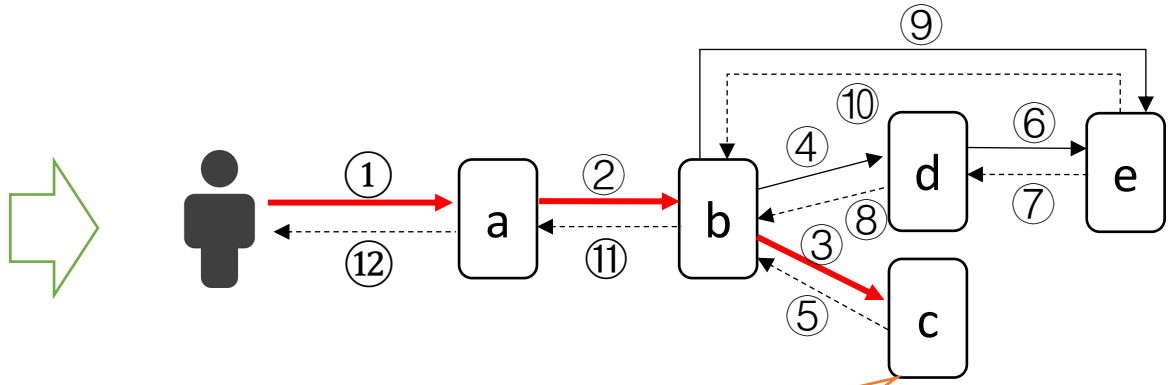
Response time

Message order	UUID	Sending time of (m→n) at m (msec)	Receiving time of (m→n) at n (msec)	Message (m→n)
①	UUID-1	-	1519747202138	call(start→a)
②	UUID-1	1519747202144	**1519747202146**	call(a→**b**)
③	UUID-1	1519747202149	1519747202151	call(b→c)
④	UUID-1	1519747202149	1519747202150	call(b→d)
⑤	UUID-1	1519747202155	1519747202156	*response*(c→b)
⑥	UUID-1	1519747202159	1519747202160	call(d→e)
⑦	UUID-1	1519747202188	1519747202190	*response*(e→d)
⑧	UUID-1	1519747202194	1519747202196	*response*(d→b)
⑨	UUID-1	1519747202253	1519747202256	call(b→e)
⑩	UUID-1	1519747202323	1519747202324	*response*(e→b)
⑪	UUID-1	**1519747202355**	1519747202356	*response*(**b**→a)
⑫	UUID-1	1519747202360	-	*response* (a→end)

Response time of microservice **b**:
1519747202355 – 1519747202146 = 209

Call path

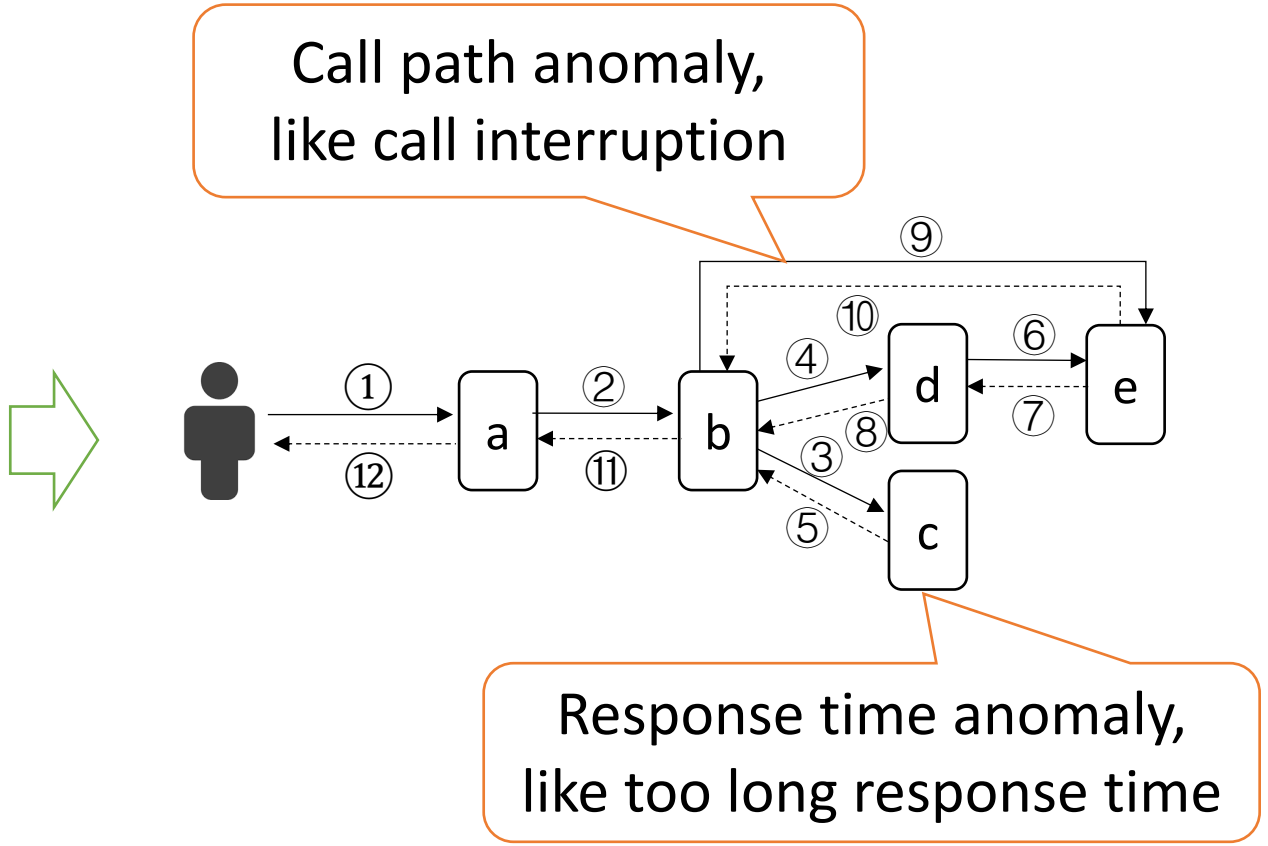
Message order	UUID	Sending time of (m→n) at m (msec)	Receiving time of (m→n) at n (msec)	Message (m→n)
①	UUID-1	-	1519747202138	call(start→a)
②	UUID-1	1519747202144	1519747202146	call(a→b)
③	UUID-1	1519747202149	1519747202151	call(b→c)
④	UUID-1	1519747202149	1519747202150	call(b→d)
⑤	UUID-1	1519747202155	1519747202156	*response*(c→b)
⑥	UUID-1	1519747202159	1519747202160	call(d→e)
⑦	UUID-1	1519747202188	1519747202190	*response*(e→d)
⑧	UUID-1	1519747202194	1519747202196	*response*(d→b)
⑨	UUID-1	1519747202253	1519747202256	call(b→e)
⑩	UUID-1	1519747202323	1519747202324	*response*(e→b)
⑪	UUID-1	1519747202355	1519747202356	*response*(b→a)
⑫	UUID-1	1519747202360	-	*response* (a→end)



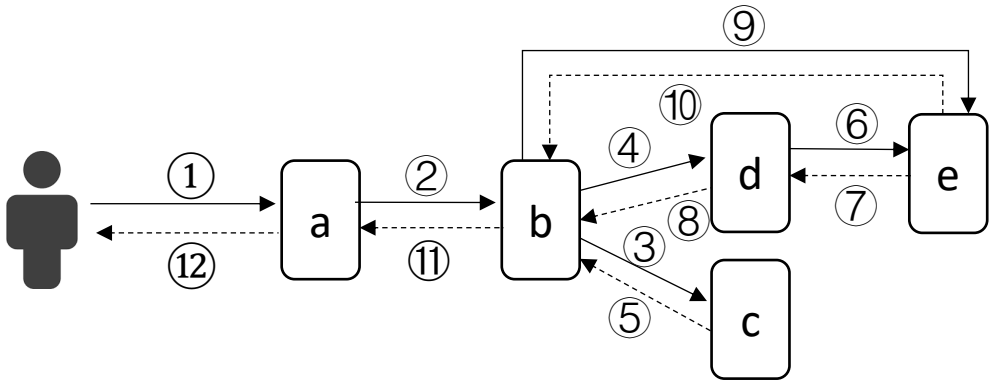
Call path of microservice c:
(start→a, a→b, b→c)

Trace anomaly

Message order	UUID	Sending time of (m→n) at m (msec)	Receiving time of (m→n) at n (msec)	Message (m→n)
①	UUID-1	-	1519747202138	call(start→a)
②	UUID-1	1519747202144	**1519747202146**	call(a→**b**)
③	UUID-1	1519747202149	1519747202151	call(b→c)
④	UUID-1	1519747202149	1519747202150	call(b→d)
⑤	UUID-1	1519747202155	1519747202156	*response*(c→b)
⑥	UUID-1	1519747202159	1519747202160	call(d→e)
⑦	UUID-1	1519747202188	1519747202190	*response*(e→d)
⑧	UUID-1	1519747202194	1519747202196	*response*(d→b)
⑨	UUID-1	1519747202253	1519747202256	call(b→e)
⑩	UUID-1	1519747202323	1519747202324	*response*(e→b)
⑪	UUID-1	**1519747202355**	1519747202356	*response*(**b**→a)
⑫	UUID-1	1519747202360	-	*response* (a→end)



Response time & call path

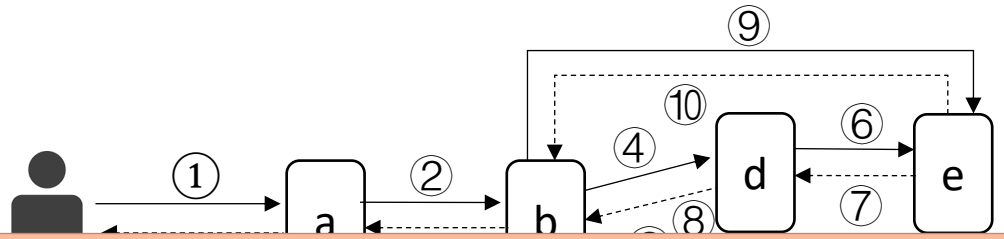


For a microservice, its response time is determined by both itself and its call path

Microservice s	Call path of microservice s (s, call path)	Response time of (s, call path) (msec)
a	(a, (start→a))	222
b	(b, (start→a, a→b))	209
c	(c, (start→a, a→b, b→c))	4
d	(d, (start→a, a→b, b→c, b→d))	44
e	**(e, (start→a, a→b, b→c, b→d, d→e))**	**28**
e	**(e, (start→a, a→b, b→c, b→d, d→e, b→e))**	**67**

Microservice *e* is invoked twice, with different response time

Response time & call path



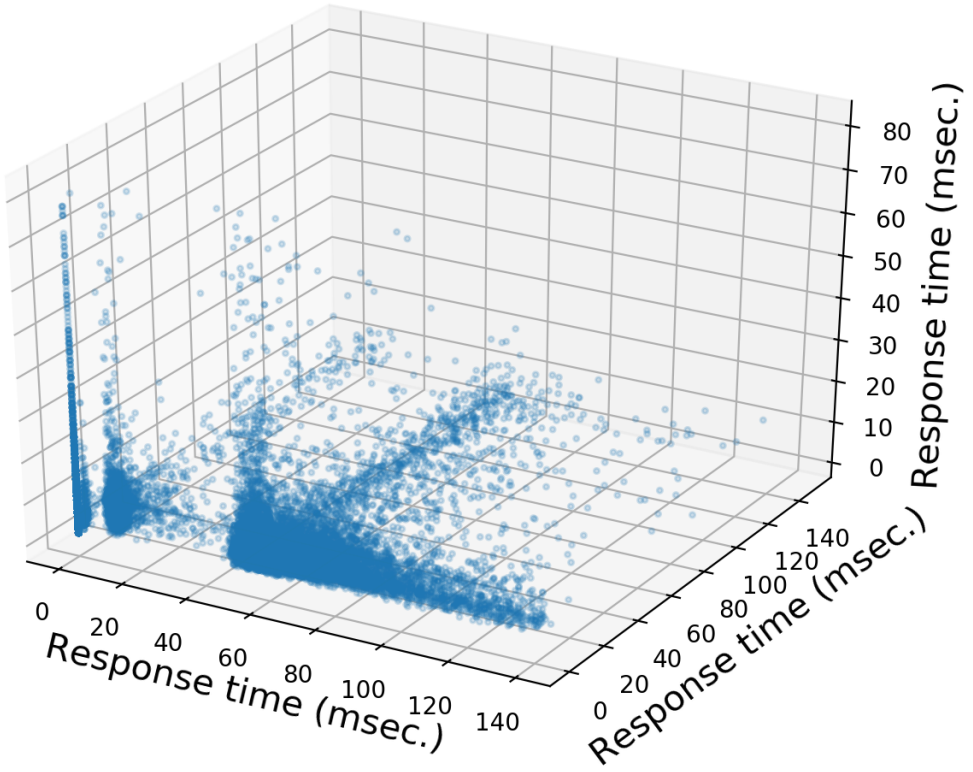
For a microservice, its response time is determined by both itself

This mandates that response times and call paths must be unified

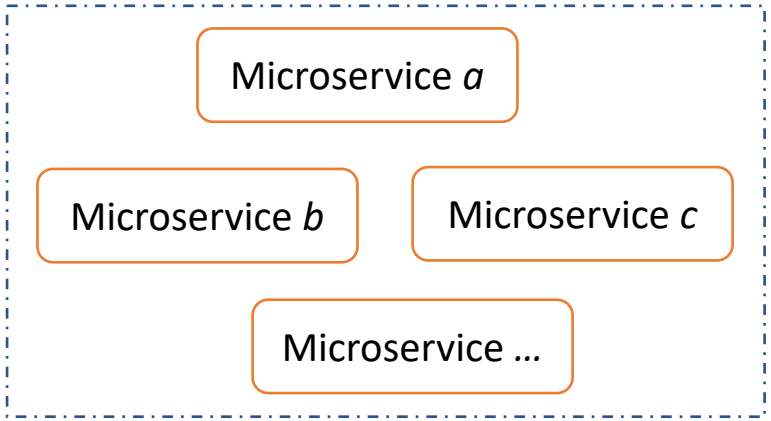
d	(d, (start→a, a→b, b→c, b→d))	44
e	(e, (start→a, a→b, b→c, b→d, d→e))	28
e	(e, (start→a, a→b, b→c, b→d, d→e, b→e))	67

The distribution of response time

- The distribution of traces' response time **without anomalies** from a small online service in a company, This service only contain **three microservices.**
- The three axes denote the three microservices' response time, respectively.
- Although there are only three microservices in this service, **the response time varies significantly without being anomalous**.



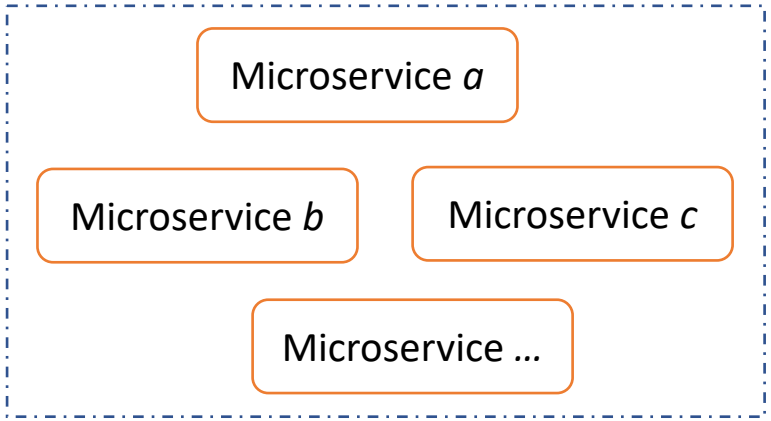
Trace anomaly detection



Hundreds of microservices

Complex call relationships

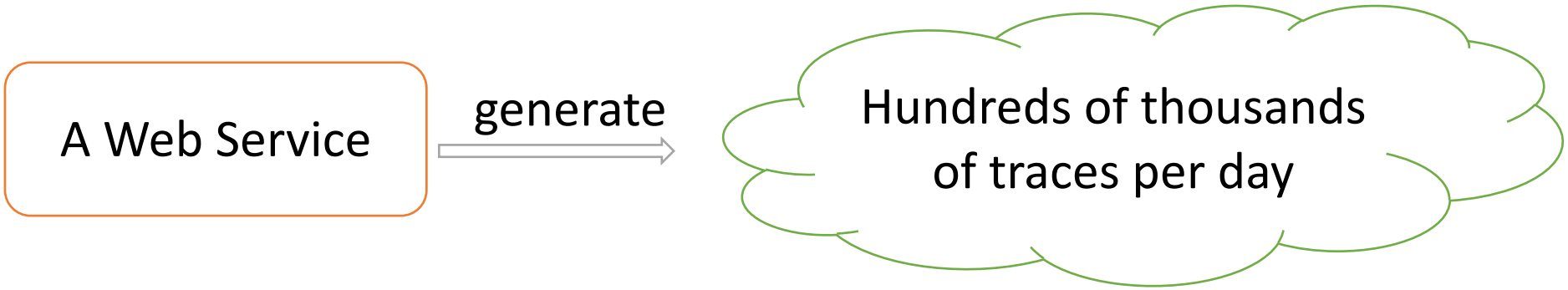
Trace anomaly detection



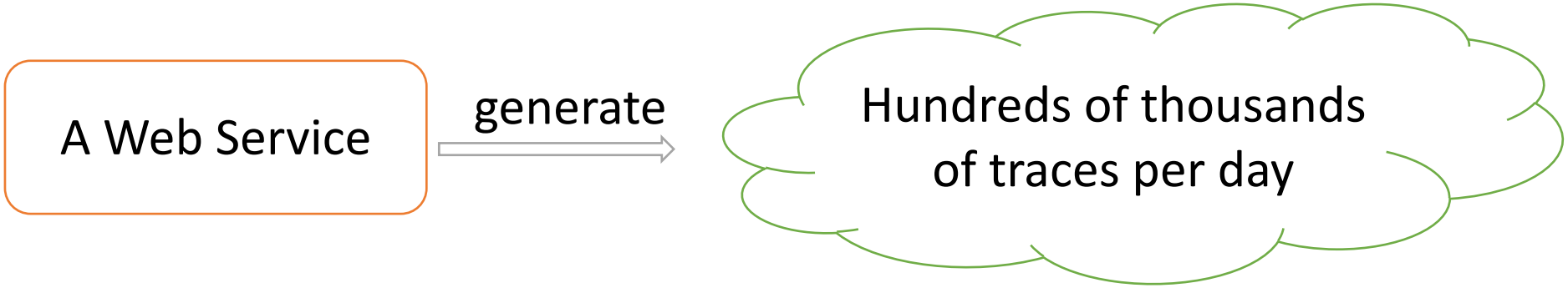
Hundreds of microservices

Complex call relationships

Core idea

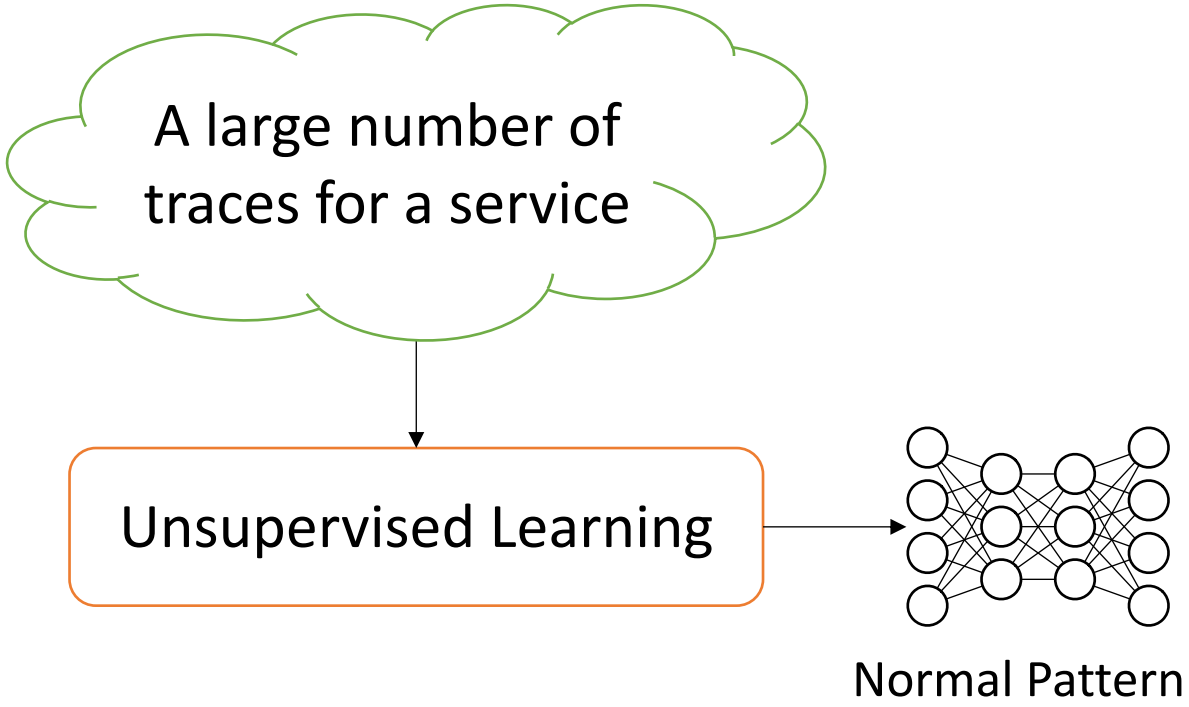

A Web Service
generate
Hundreds of thousands
of traces per day

Core idea

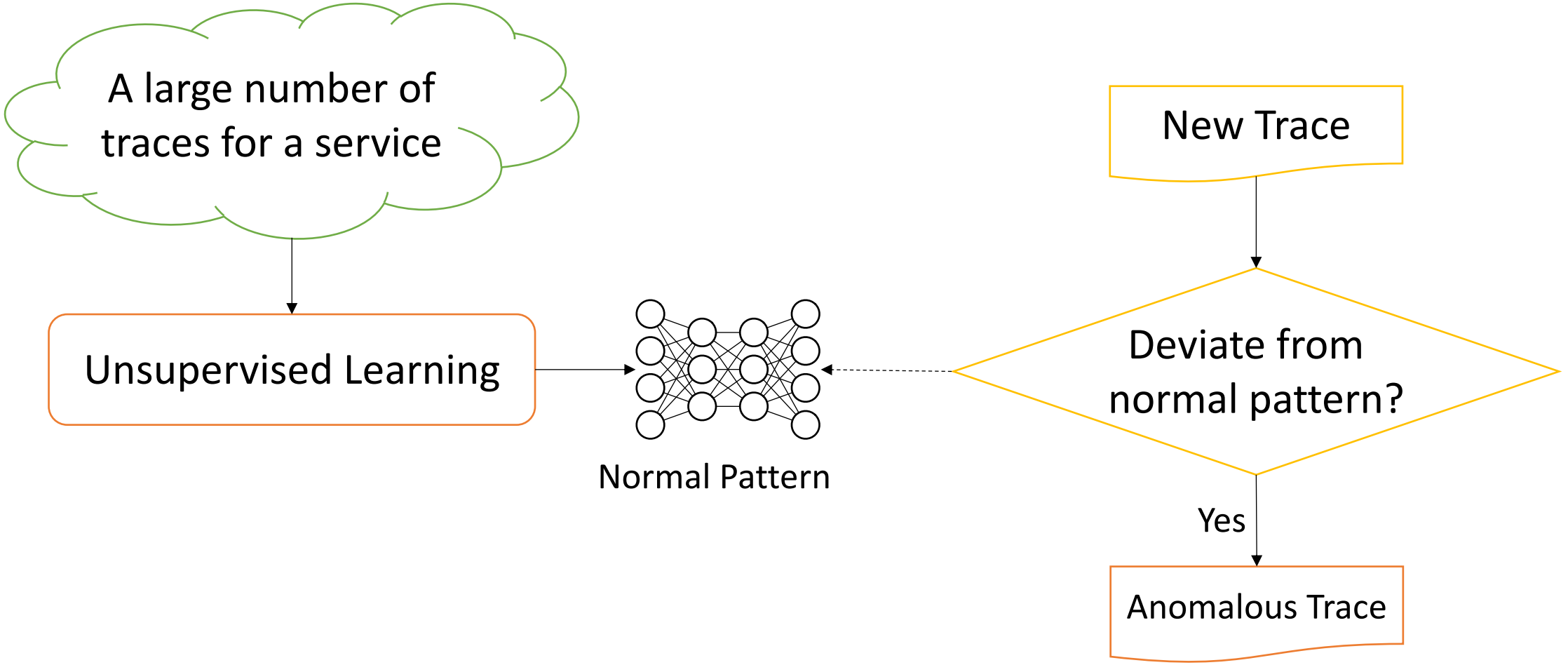


It is **infeasible** to accurately label the large number of traces for **supervised learning**

Core idea


A large number of traces for a service
Unsupervised Learning
Normal Pattern

Core idea


A large number of traces for a service
Unsupervised Learning
Normal Pattern
New Trace
Deviate from normal pattern?
Yes
Anomalous Trace

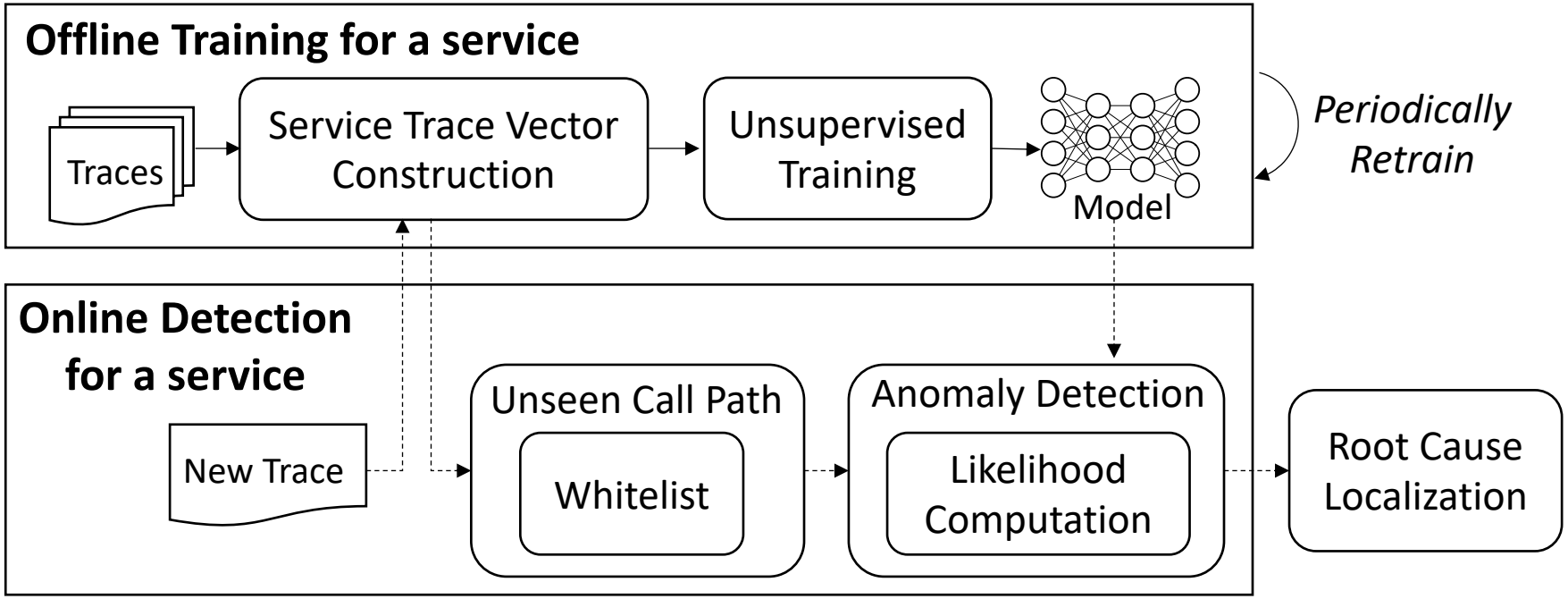
Background

Design

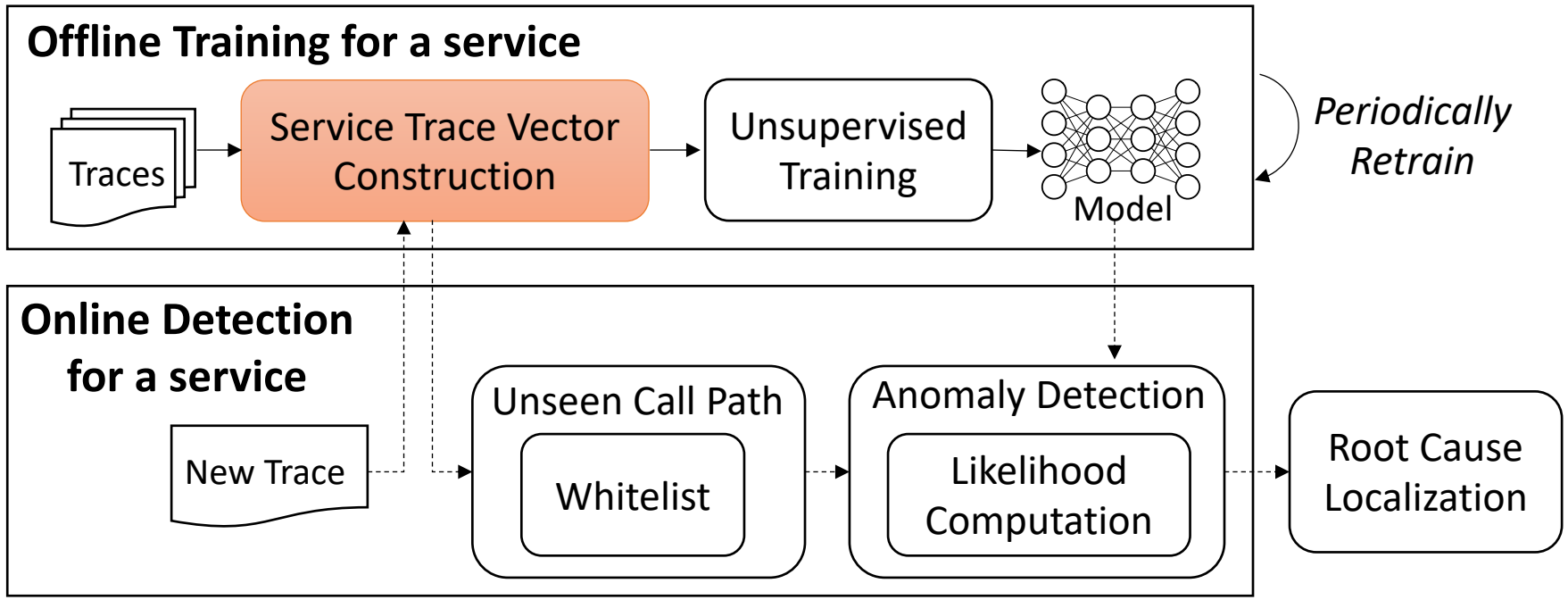
Evaluation

Conclusion

Design of TraceAnomaly

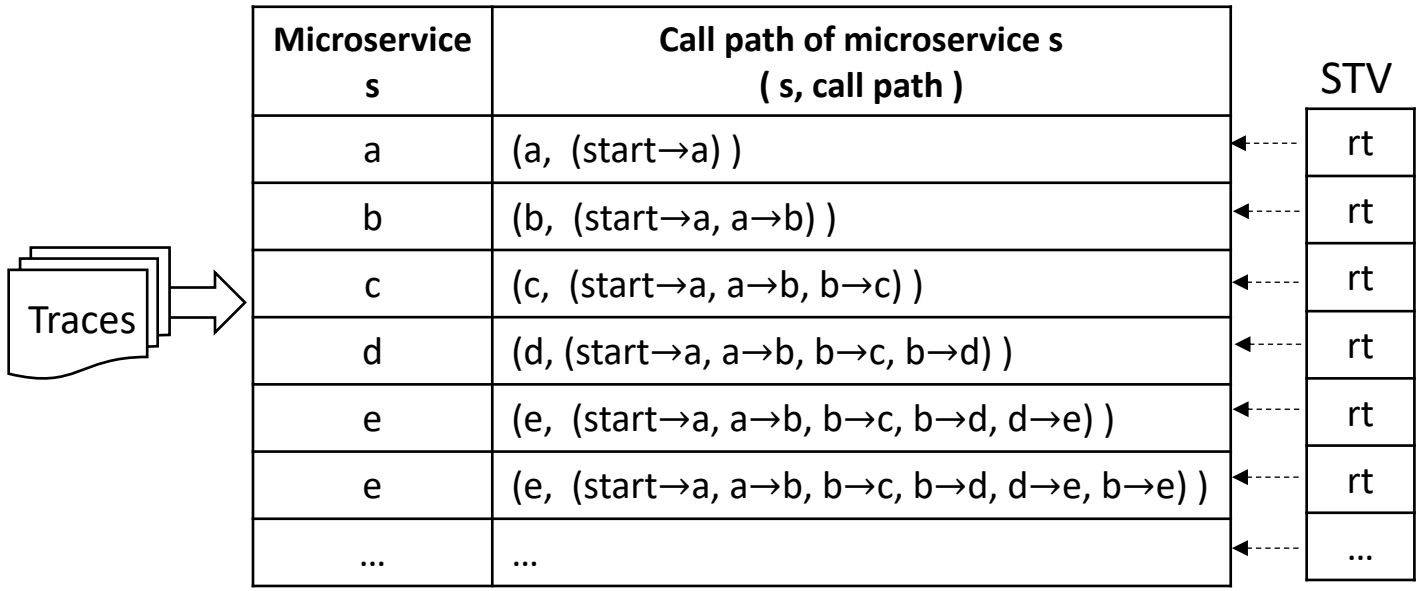


Design of TraceAnomaly



Service trace vector construction

- Unify response time and call paths of traces in an interpretable way
  - Encode the response time and call paths of a trace in a service into a STV (Service Trace Vector)

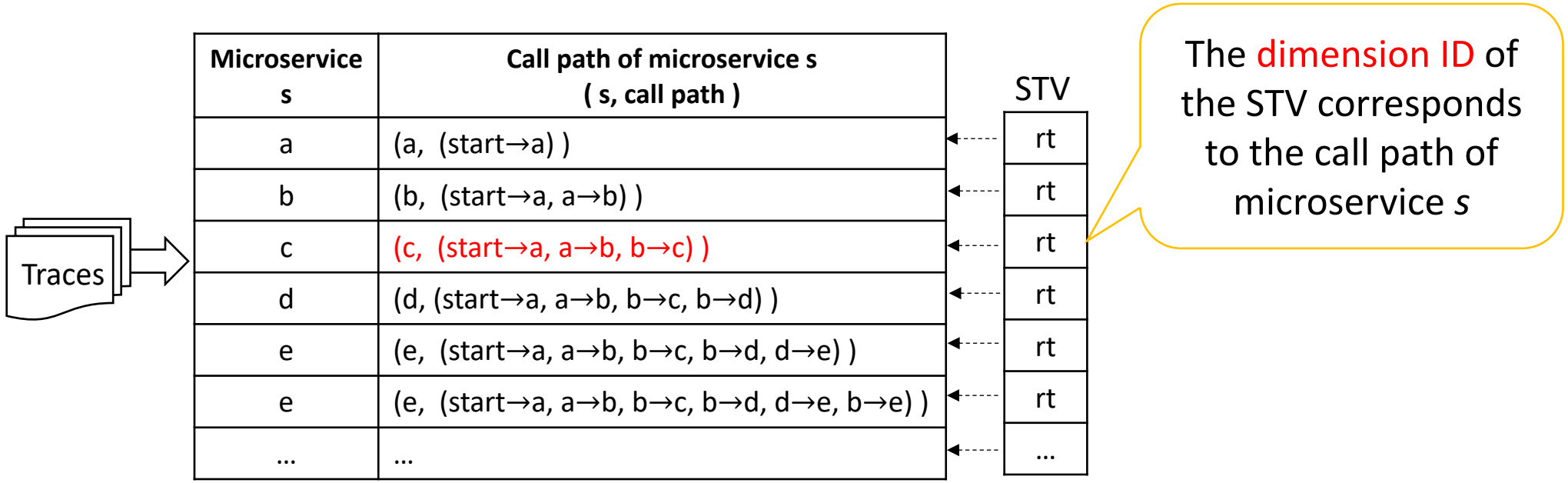


Microservice s	Call path of microservice s (s, call path)
a	(a, (start→a))
b	(b, (start→a, a→b))
c	(c, (start→a, a→b, b→c))
d	(d, (start→a, a→b, b→c, b→d))
e	(e, (start→a, a→b, b→c, b→d, d→e))
e	(e, (start→a, a→b, b→c, b→d, d→e, b→e))
…	…

STV
rt
rt
rt
rt
rt
rt
…

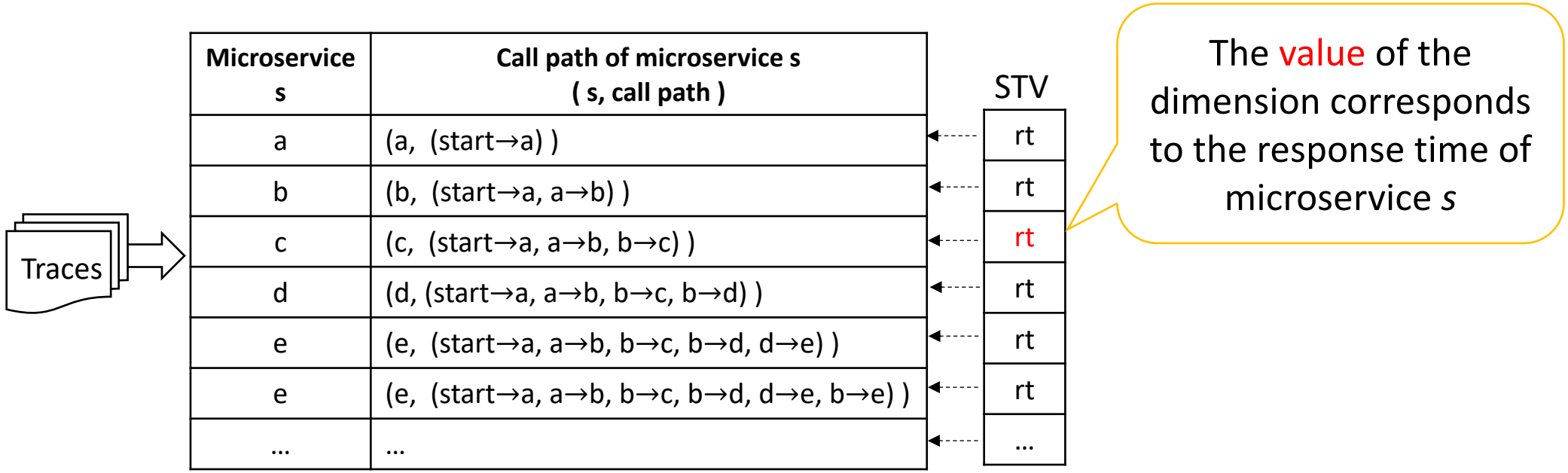
Service trace vector construction

- Unify response time and call paths of traces in an interpretable way
  - Encode the response time and call paths of a trace in a service into a STV (Service Trace Vector)

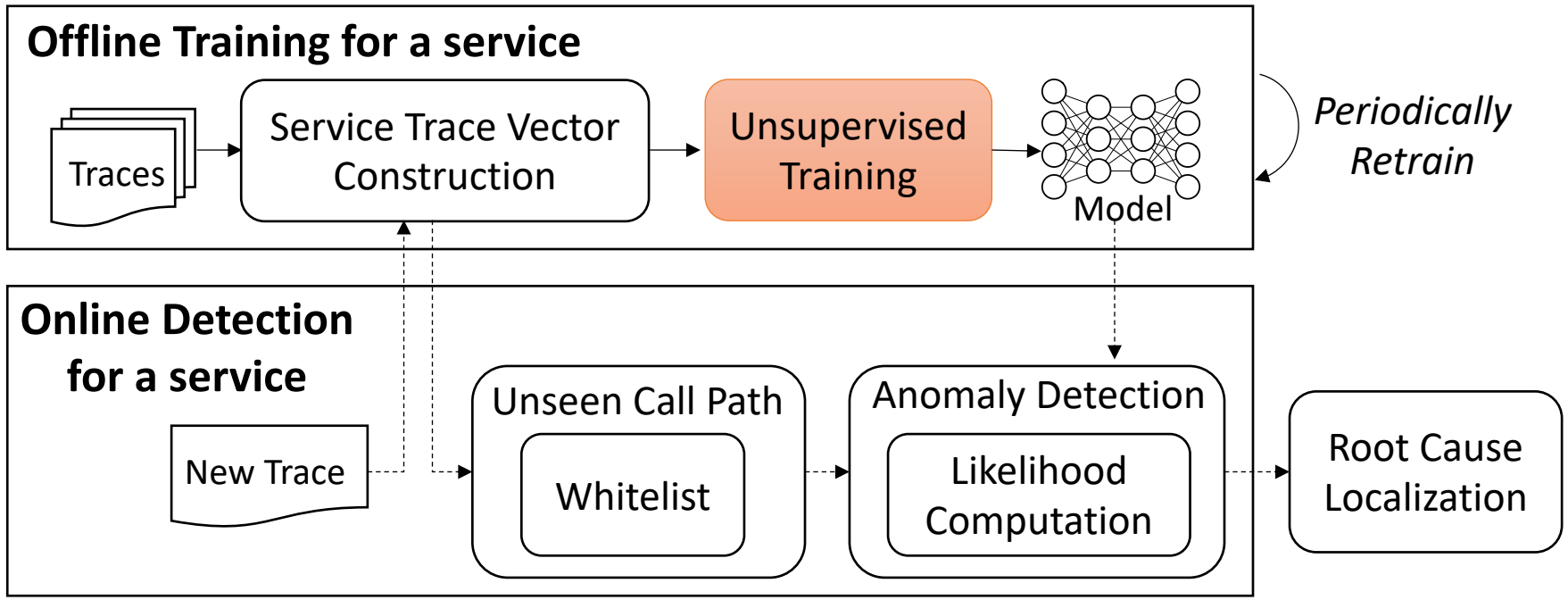


Service trace vector construction

- Unify response time and call paths of traces in an interpretable way
  - Encode the response time and call paths of a trace in a service into a STV (Service Trace Vector)

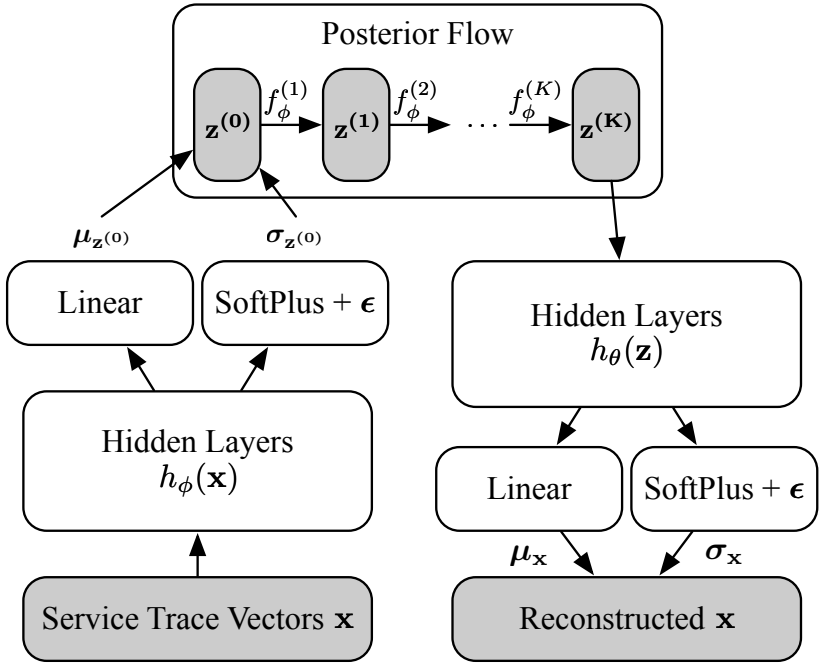


Design of TraceAnomaly



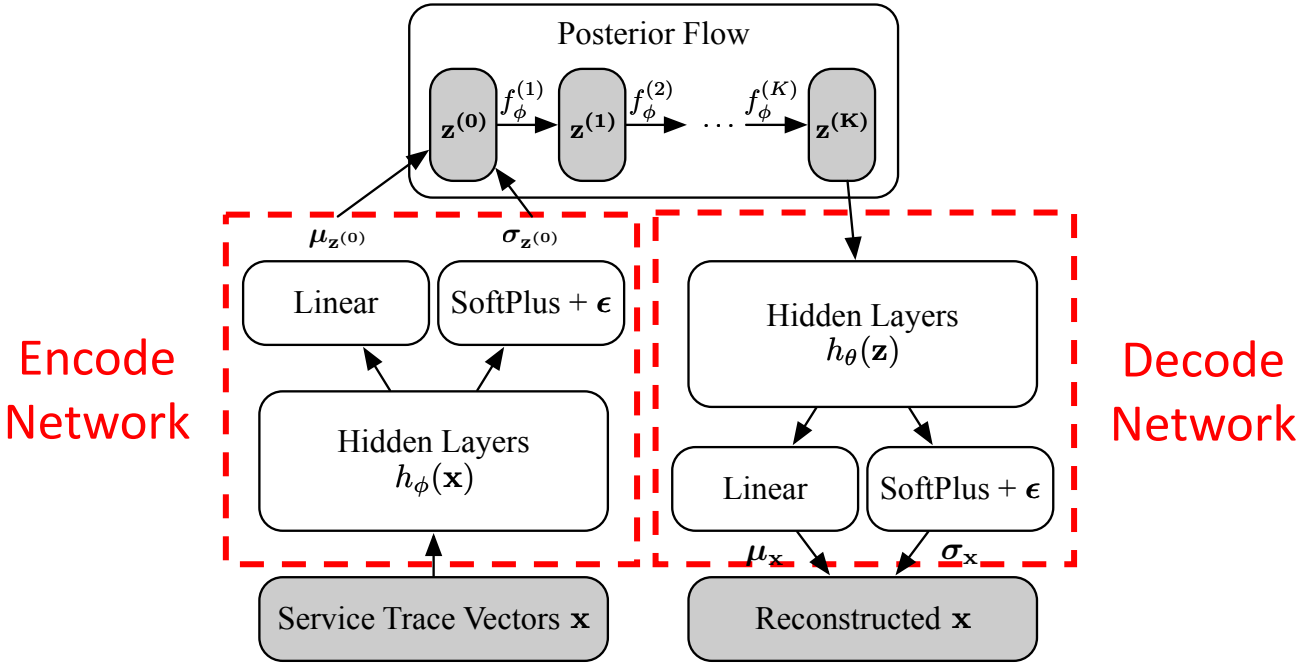
Unsupervised Learning

- The architecture of Bayesian Networks



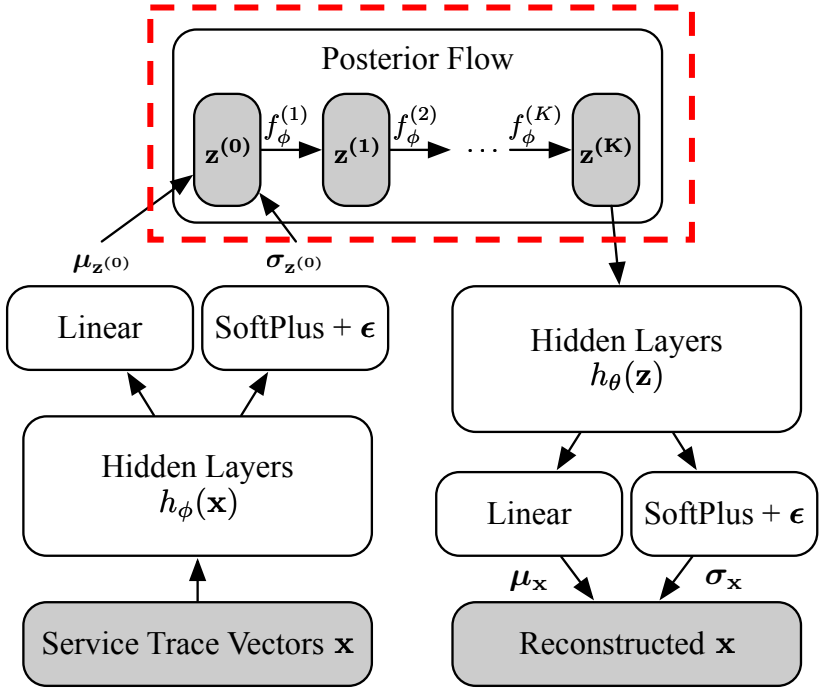
Unsupervised Learning

- The architecture of Bayesian Networks
  - Unsupervised learning through the encode-decode network

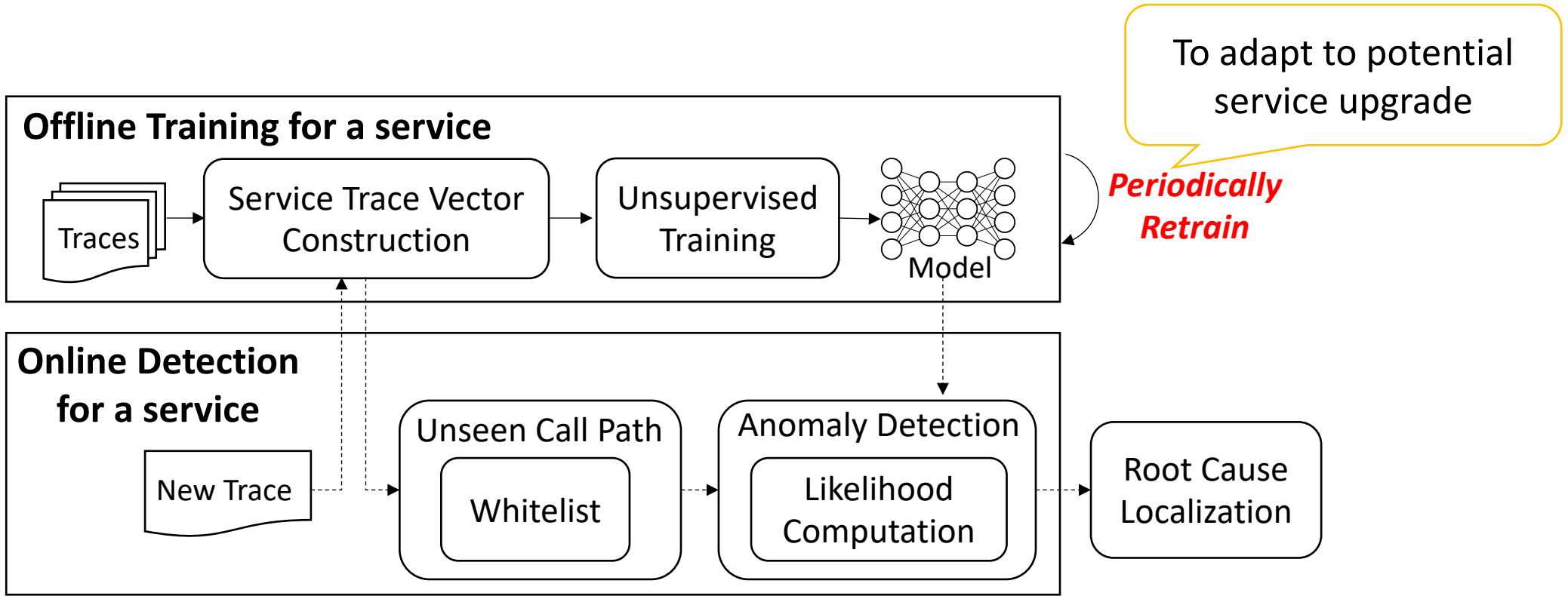


Unsupervised Learning

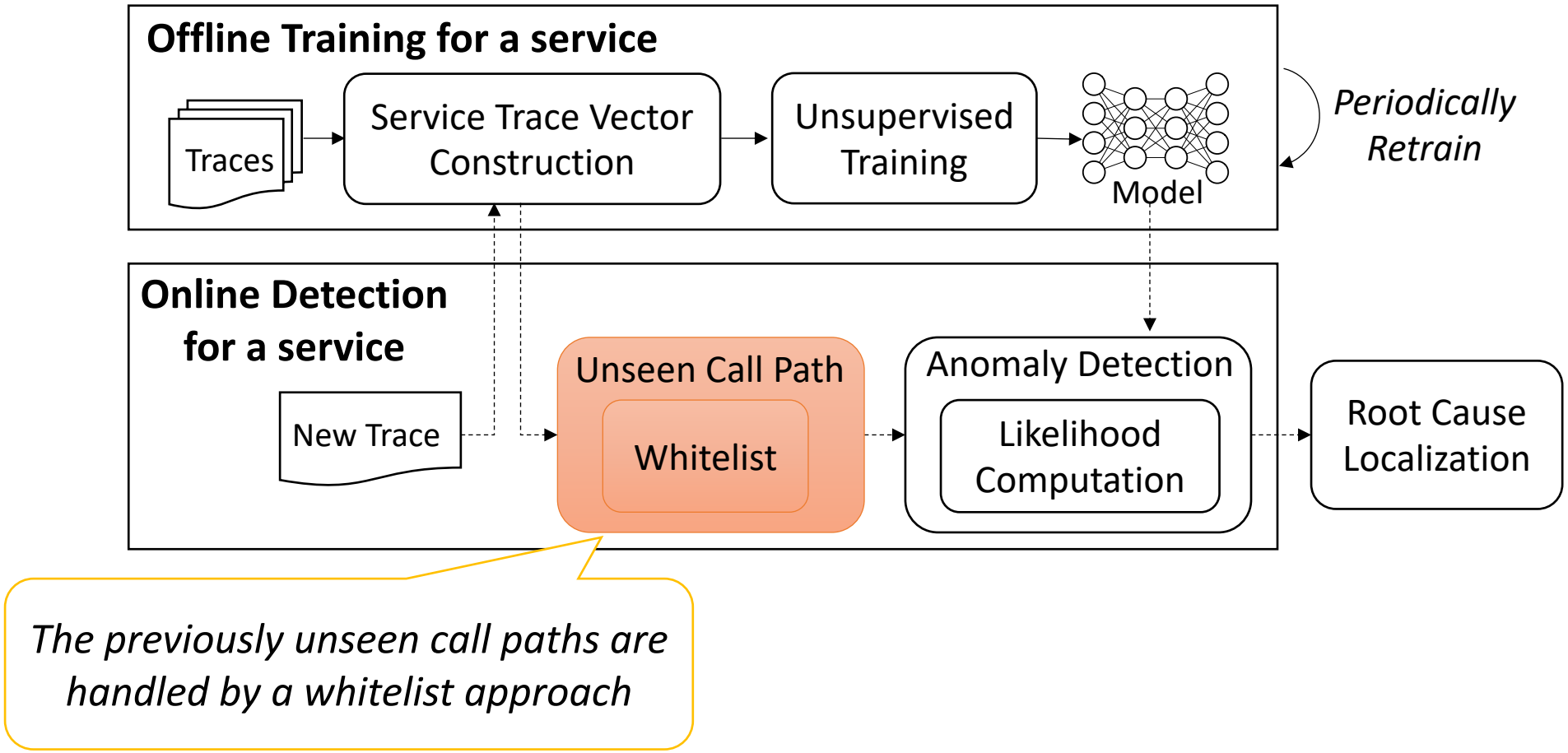
- The architecture of Bayesian Networks
  - Posterior Flow allows network to learn more complex patterns



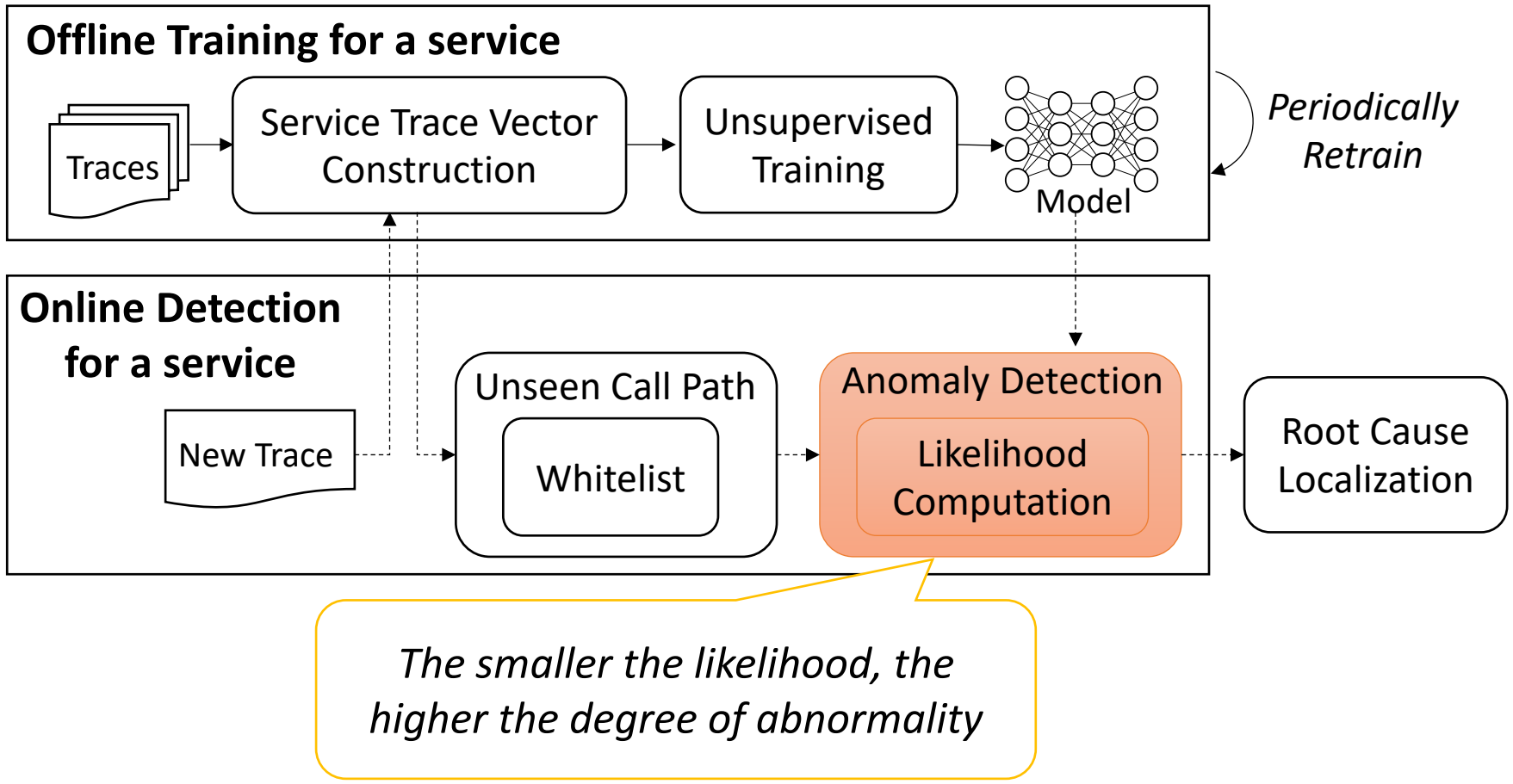
Design of TraceAnomaly



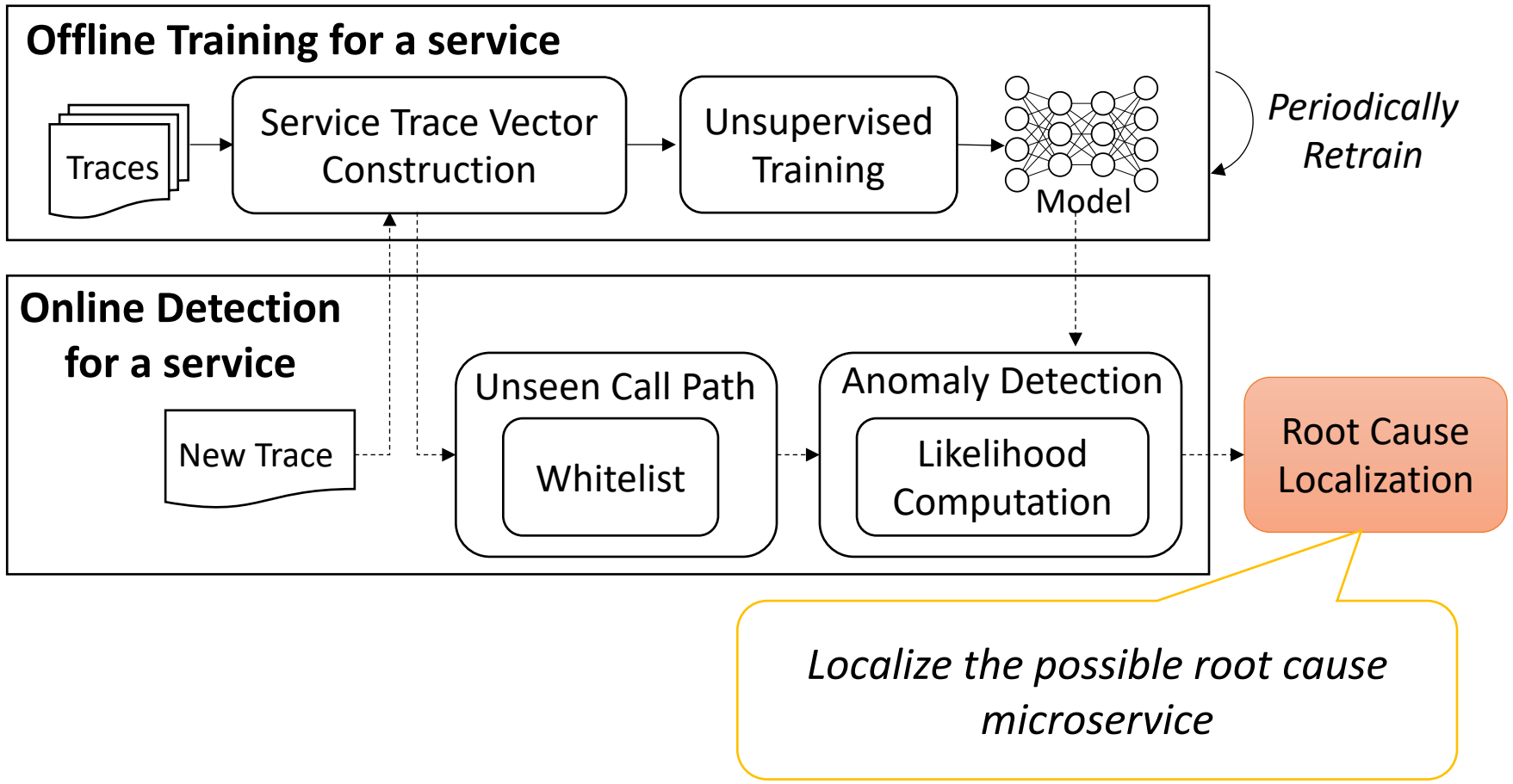
Design of TraceAnomaly



Design of TraceAnomaly



Design of TraceAnomaly



Background

Design

Evaluation

Conclusion

Evaluation of TraceAnomaly

- Testbed evaluation
- Real service evaluation

Evaluation of TraceAnomaly

- Testbed evaluation
  - The open source TrainTicket[1] testbed
- Real service evaluation

[1]. Zhou, Xiang, et al. "Fault Analysis and Debugging of Microservice Systems: Industrial Survey, Benchmark System, and Empirical Study." *IEEE Transactions on Software Engineering* 14.8 (2018): 1-1.

Testbed evaluation

TABLE II: Evaluation results of different approaches on a TrainTicket testbed which contains 41 microservices. The test set contains 30,356 normal test traces, 2,699 response time anomaly traces and 2380 invocation path anomaly traces.

	Overall		Response Time Anomaly		Invocation Path Anomaly		Training (minutes) for 24-hour traces	Test (seconds) for 4-hour traces
	Precision	Recall	Precision	Recall	Precision	Recall		
WFG-based [5]	0.76	0.92	0.65	0.96	0.96	0.87	0.06	0.4
DeepLog* [8]	0.52	0.71	0.65	0.96	0.34	0.42	306	78
CPD-based [7]	0.30	0.47	N/A	N/A	0.30	1.0	N/A	9
CFG-based [6]	0.70	0.49	0.70	0.94	N/A	N/A	N/A	0.1
AEVB [4]	0.17	0.52	0.17	0.98	N/A	N/A	6120	121
OmniAnomaly [42]	0.45	0.49	0.45	0.93	N/A	N/A	530	113
Multimodal LSTM [3]	0.60	0.96	N/A	0.94	N/A	0.97	9.5	109
TraceAnomaly	**0.98**	**0.97**	N/A	**0.94**	N/A	**0.99**	94	19

Testbed evaluation

TABLE II: Evaluation results of different approaches on a TrainTicket testbed which contains 41 microservices. The test set contains 30,356 normal test traces, 2,699 response time anomaly traces and 2380 invocation path anomaly traces.

	Overall		Response Time Anomaly		Invocation Path Anomaly		Training (minutes) for 24-hour traces	Test (seconds) for 4-hour traces
	Precision	Recall	Precision	Recall	Precision	Recall		
WFG-based [5]	0.76	0.92	0.65	0.96	0.96	0.87	0.06	0.4
DeepLog* [8]	0.52	0.71	0.65	0.96	0.34	0.42	306	78
CPD-based [7]	0.30	0.47	N/A	N/A	0.30	1.0	N/A	9
CFG-based [6]	0.70	0.49	0.70	0.94	N/A	N/A	N/A	0.1
AEVB [4]	0.17	0.52	0.17	0.98	N/A	N/A	6120	121
OmniAnomaly [42]	0.45	0.49	0.45	0.93	N/A	N/A	530	113
Multimodal LSTM [3]	0.60	0.96	N/A	0.94	N/A	0.97	9.5	109
TraceAnomaly	**0.98**	**0.97**	N/A	**0.94**	N/A	**0.99**	94	19

Outperforming other baseline approaches

Evaluation of TraceAnomaly

- Testbed evaluation
- Real service evaluation
  - Four large evaluation services from WeBank company

TABLE I: Details of the four large evaluation services from company *S*.

	No. of Microservices	Evaluation duration	Average No. of traces/day	No. of STV Dimensions	No. of call graph structures	No. of manually confirmed anomalous traces	Description (all for mobile users)
Service-1	344	5 days (Sun. - Thu.)	801,021	690	368	108	transaction query.
Service-2	61	4 days (Sun. - Wed.)	600,806	173	61	68	account opening.
Service-3	233	4 days (Wed. - Sat.)	502,408	508	302	81	repayment.
Service-4	113	4 days (Wed. - Sat.)	500,921	412	186	66	account balance query.

Real service evaluation

TABLE III: Online evaluation results of different approaches on four large online services which contain hundreds of microservices, whose statistics are shown in Table I.

	Service-1		Service-2		Service-3		Service-4		Overall (Union of 4 services)	
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
Hard-coded Rule	0.910	0.800	0.920	0.792	0.911	0.812	0.930	0.800	0.910	0.804
WFG-based [5]	0.020	0.500	0.012	0.323	0.050	0.410	0.032	0.300	0.031	0.386
DeepLog* [8]	0.270	0.680	0.241	0.560	0.320	0.643	0.302	0.601	0.290	0.628
CPD-based [7]	0.52	0.063	0.43	0.090	0.57	0.110	0.64	0.072	0.531	0.081
CFG-based [6]	0.170	0.610	0.250	0.570	0.102	0.503	0.180	0.630	0.164	0.562
TraceAnomaly	**0.980**	**1.000**	**0.982**	**1.000**	**0.981**	**1.000**	**0.973**	**1.000**	**0.981**	**1.000**

Real service evaluation

TABLE III: Online evaluation results of different approaches on four large online services which contain hundreds of microservices, whose statistics are shown in Table I.

	Service-1		Service-2		Service-3		Service-4		Overall (Union of 4 services)	
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
Hard-coded Rule	0.910	0.800	0.920	0.792	0.911	0.812	0.930	0.800	0.910	0.804
WFG-based [5]	0.020	0.500	0.012	0.323	0.050	0.410	0.032	0.300	0.031	0.386
DeepLog* [8]	0.270	0.680	0.241	0.560	0.320	0.643	0.302	0.601	0.290	0.628
CPD-based [7]	0.52	0.063	0.43	0.090	0.57	0.110	0.64	0.072	0.531	0.081
CFG-based [6]	0.170	0.610	0.250	0.570	0.102	0.503	0.180	0.630	0.164	0.562
TraceAnomaly	**0.980**	**1.000**	**0.982**	**1.000**	**0.981**	**1.000**	**0.973**	**1.000**	**0.981**	**1.000**

Outperforming other baseline approaches

Real service evaluation

TABLE III: Online evaluation results of different approaches on four large online services which contain hundreds of microservices, whose statistics are shown in Table I.

	Service-1		Service-2		Service-3		Service-4		Overall (Union of 4 services)	
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
Hard-coded Rule	0.910	0.800	0.920	0.792	0.911	0.812	0.930	0.800	0.910	0.804
WFG-based [5]	0.020	0.500	0.012	0.323	0.050	0.410	0.032	0.300	0.031	0.386
DeepLog* [8]	0.270	0.680	0.241	0.560	0.320	0.643	0.302	0.601	0.290	0.628
CPD-based [7]	0.52	0.063	0.43	0.090	0.57	0.110	0.64	0.072	0.531	0.081
CFG-based [6]	0.170	0.610	0.250	0.570	0.102	0.503	0.180	0.630	0.164	0.562
TraceAnomaly	**0.980**	**1.000**	**0.982**	**1.000**	**0.981**	**1.000**	**0.973**	**1.000**	**0.981**	**1.000**

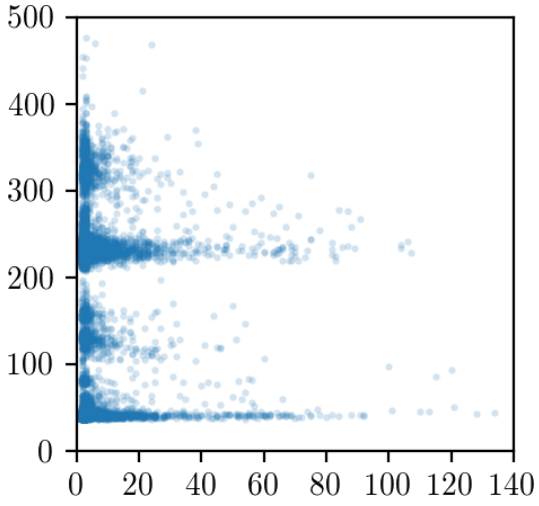
The precision is accurate, the recall might be biased towards 1 for our label approach

Real service evaluation

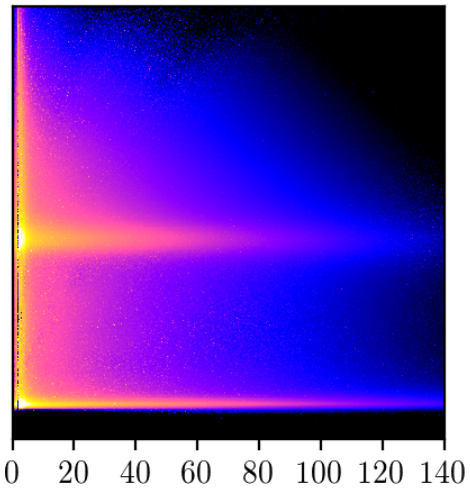
Label *Approach*

- It is infeasible to manually label hundreds of thousands of traces per day
- The union of all detected anomalous traces by all these baselines during the evaluation period was considered as the **candidate** anomaly trace set, and manually validated by two experienced operators separately.
- All the anomalous traces confirmed by both operators are labeled as **anomalous**, and labeled as **normal** otherwise.

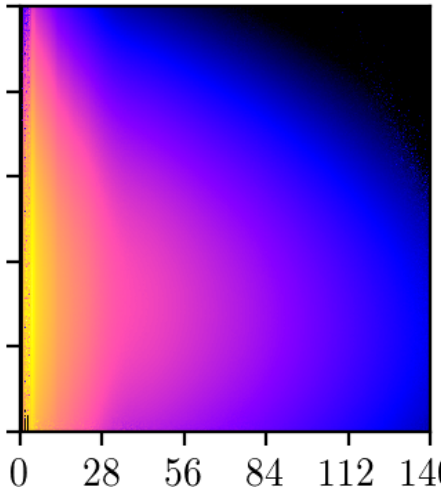
Algorithm analysis



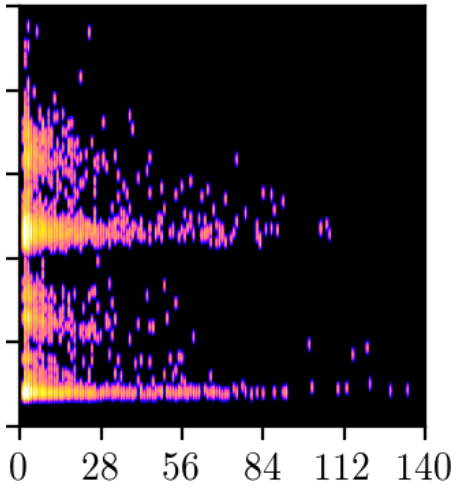
The distribution of traces' response time



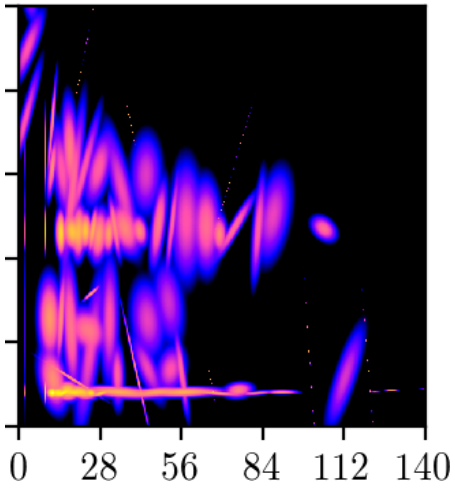
The learned distribution of **TraceAnomaly**



The learned distribution of **VAE**



The learned distribution of **KDE**



The learned distribution of **GMM**

Evaluation of TraceAnomaly

- Testbed evaluation
- Real service evaluation
- …

More details of the evaluation can be found in the paper

Background

Design

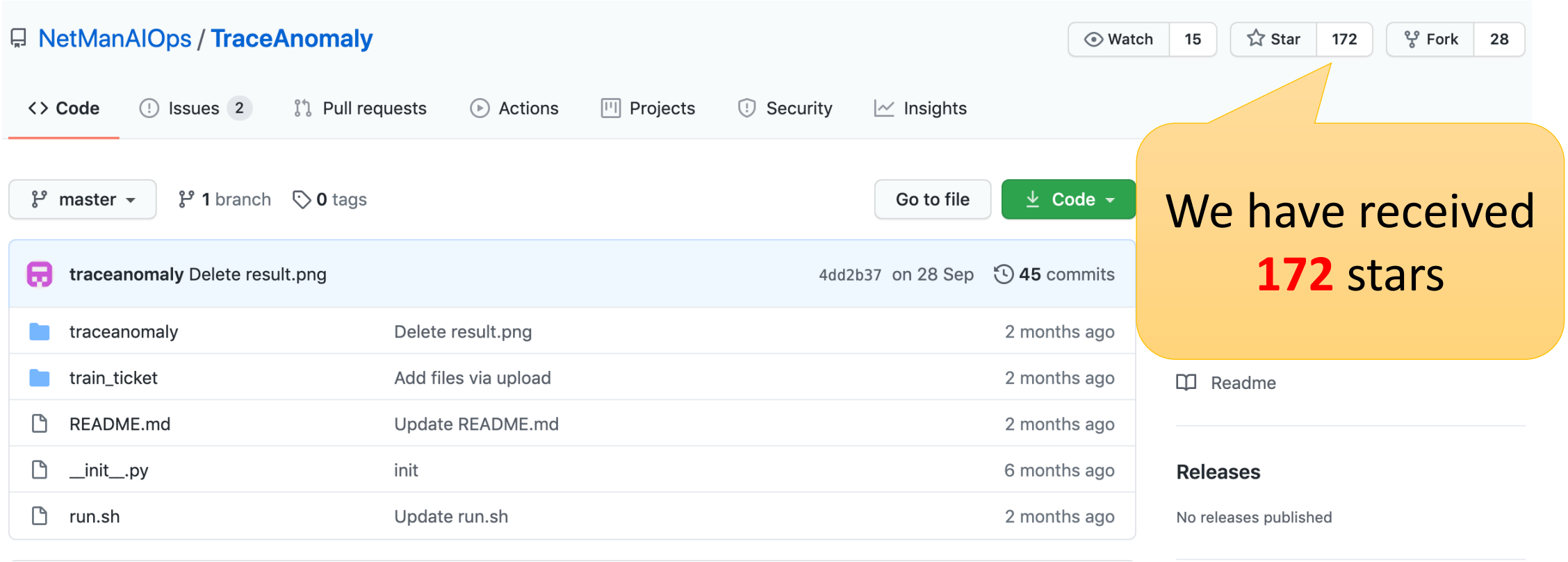
Evaluation

Conclusion

Conclusion

1. We propose STV to encode both the response time information and the call path information of traces
2. We propose an unsupervised deep learning algorithm for anomaly detection
3. Detailed evaluations on four large online services and a TrainTicket testbed show good performance
4. We propose a root cause localizing algorithm based on our designed STV
5. We have open-sourced the prototype of TraceAnoamly: https://github.com/NetManAIOps/TraceAnomaly

Open-sourced TraceAnomaly



Thank you!

Q&A

liuping15@mails.tsinghua.edu.cn
https://github.com/NetManAIOps/TraceAnomaly

ISSRE 2020