2019/4/8-12
SAC2019

Real-Time Botnet Detection Using Nonnegative Tucker Decomposition

Hideaki Kanehara[1,2] Yuma Murakami[1] Jumpei Shimamura[3]
Takeshi Takahashi[2] Daisuke Inoue[2] Noboru Murata[1,2]

1: Waseda University
2: National institute of information and communications technology
3: Clwit Inc.

Outline

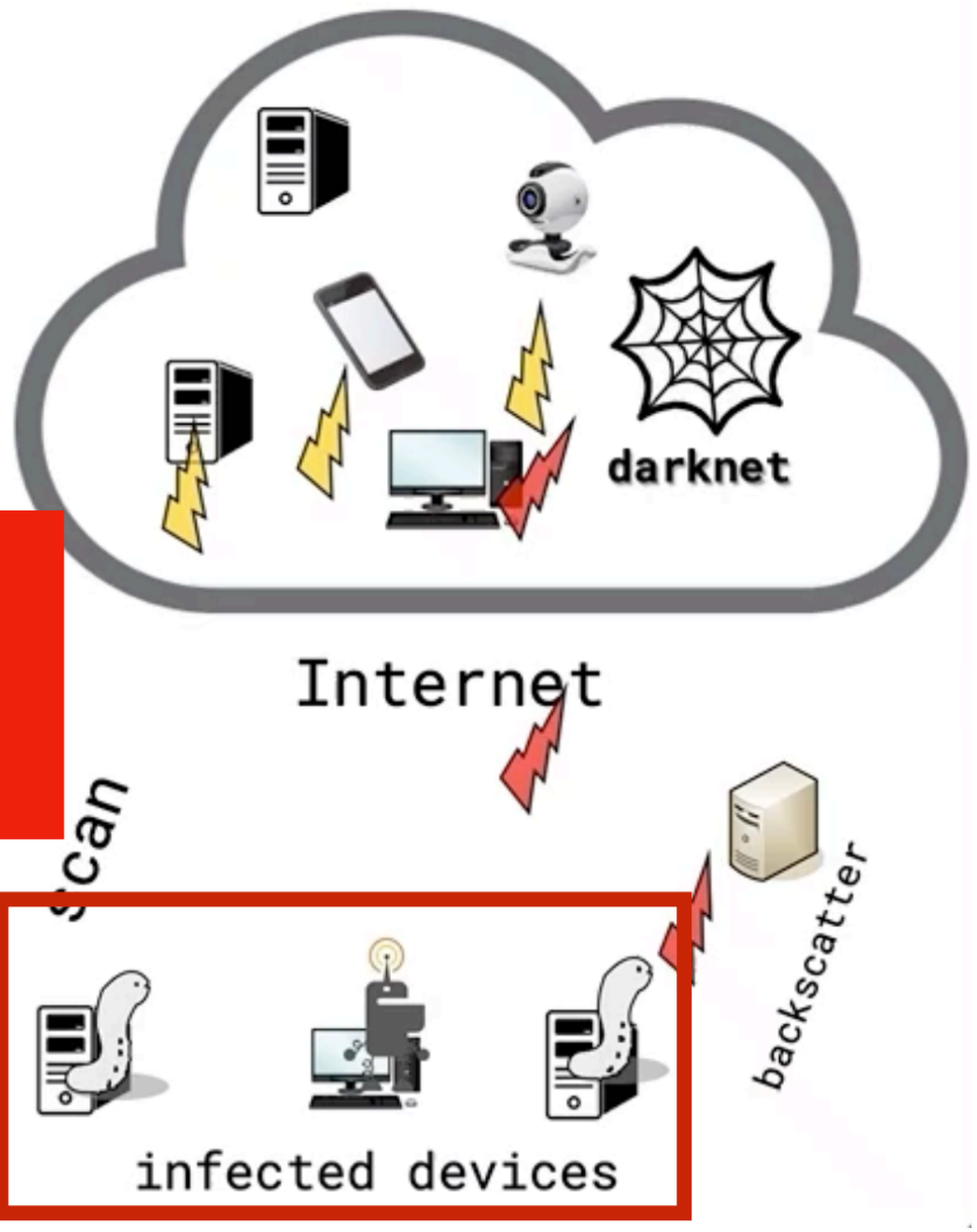
1. Background
2. Methodology
   - Factorization-based method
   - Real-time tensor factorization
   - Botnet detection using NTD
3. Experiment
   - Experimental setting
   - Result
     - NTD visualization
     - Comparison with the actual traffic
     - Related incident

Outline

1. Background
2. Methodology
   - Factorization-based method
   - Real-time tensor factorization
   - Botnet detection using NTD
3. Experiment
   - Experimental setting
   - Result
     - NTD visualization
     - Comparison with the actual traffic
     - Related incident

- early detection of cyber attacks is essential
  -> DDoS attacks are often performed by ***botnets***

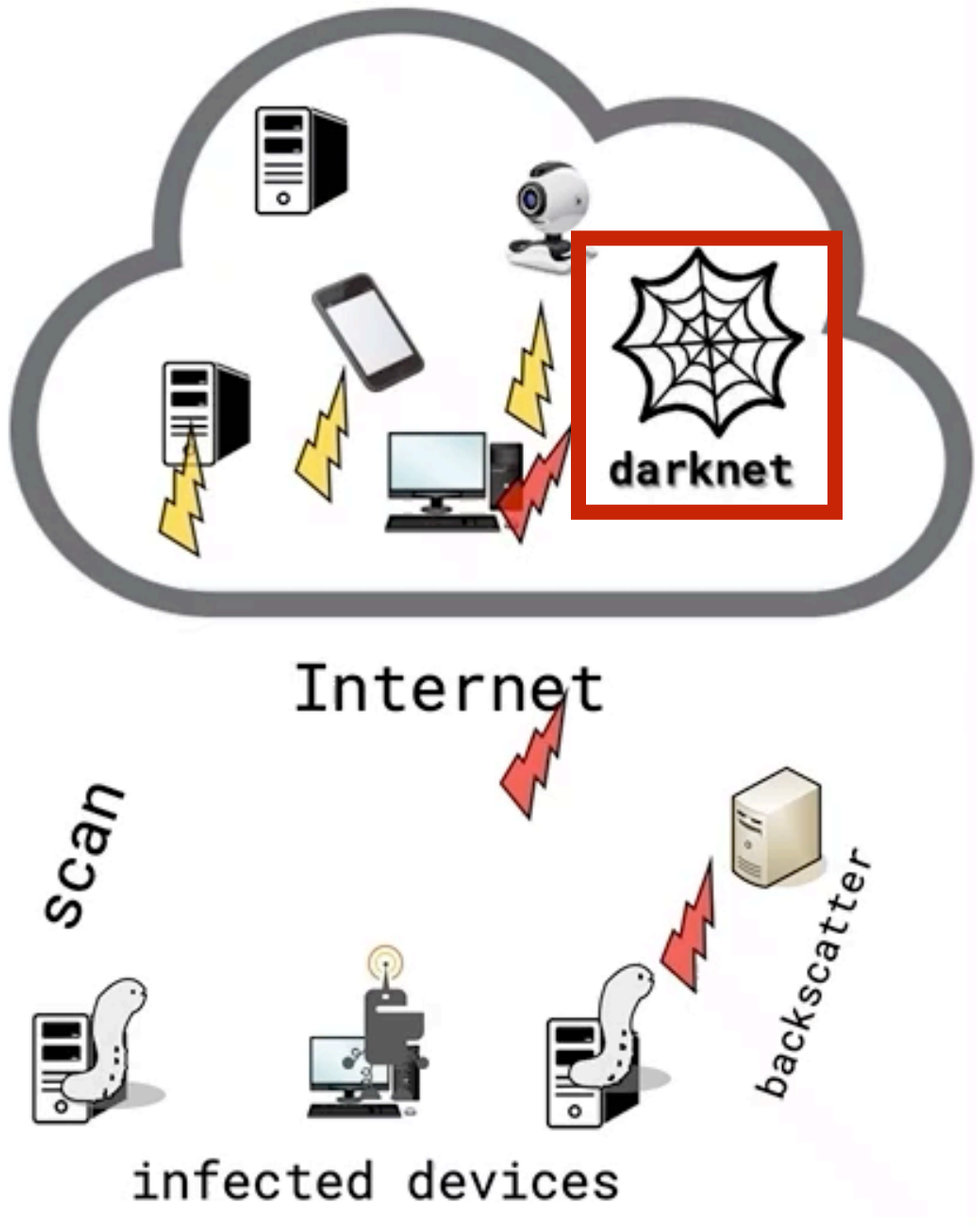
botnet: a group of infected devices that are remotely controlled by attackers



- early detection of cyber attacks is essential
  -> DDoS attacks are often performed by ***botnets***

darknet

- **unused IP address space**
  - but in reality, a lot of malicious packets arrive
    - a network scanning (for spreading malware infection)
- **tells us the malicious trend** of the wide area of the Internet without having to monitor the overall hosts



- An example of the darknet traffic

packet->

Timestamp	Src IP	Src port	Dst port	…
12:34:56	12.125.x.x	37721	25	…
12:34:56	252.156.x.x	52521	23	…
12:34:57	123.35.x.x	25162	8888	…
12:34:58	156.33.x.x	12732	3218	…
⋮	⋮	⋮	⋮	

Purpose

Cooperative behavior (botnet) detection

- An example of the darknet traffic

Timestamp	Src IP	Src port	Dst port	…
12:34:56	12.125.x.x	37721	25	…
12:34:56	252.156.x.x	52521	23	…
12:34:57	123.35.x.x	25162	8888	…
12:34:58	156.33.x.x	12732	3218	…
⋮	⋮	⋮	⋮	

Purpose

Cooperative behavior (botnet) detection

an activity of a **host group**
using almost the same **port numbers**
at almost the same **time/frequency**

- An example of the darknet traffic

Timestamp	Src IP	Src port	Dst port	…
12:34:56	12.125.x.x	37721	25	…
12:34:56	252.156.x.x	52521	23	…
12:34:57	123.35.x.x			
12:34:58	156.33.x.x			
⋮	⋮			

Also, we want to know …
- Where are they from (src IP)
- What is their aim (dst port)

Purpose

Cooperative behavior (botnet) detection

an activity of a **host group**
using specific **port numbers**
at almost the same **time/frequency**

Outline

1. Background
2. Methodology
   - Factorization-based method
   - Real-time tensor factorization
   - Botnet detection using NTD
3. Experiment
   - Experimental setting
   - Result
     - NTD visualization
     - Comparison with the actual traffic
     - Related incident

Overview of our proposed method

Data input stage

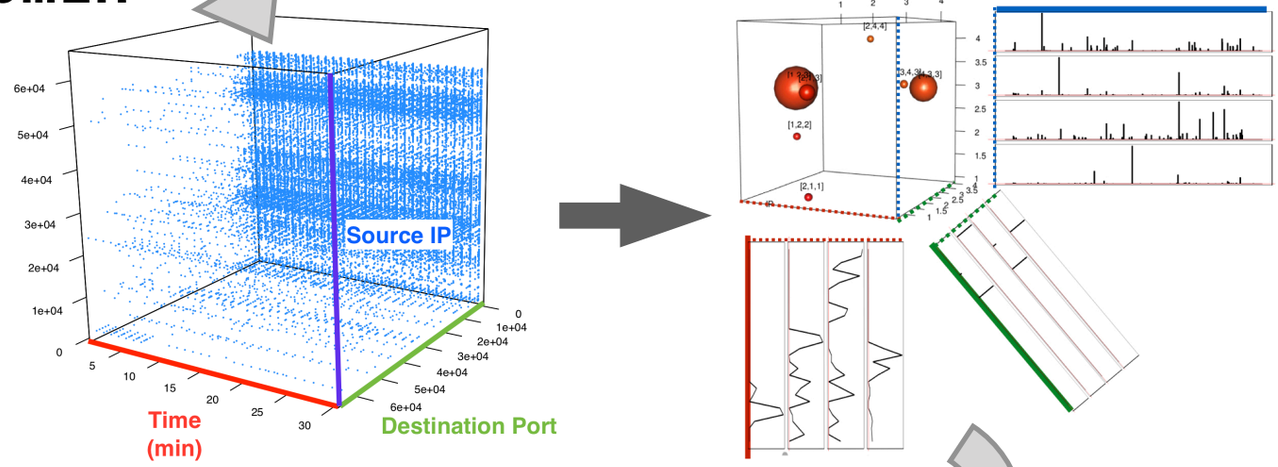
- Preprocessing darknet traffic

Timestamp	Src IP	Dst Port
12:34:56	12.125.x.x	25
12:34:56	252.156.x.x	23
⋮	⋮	⋮

Feature extraction stage

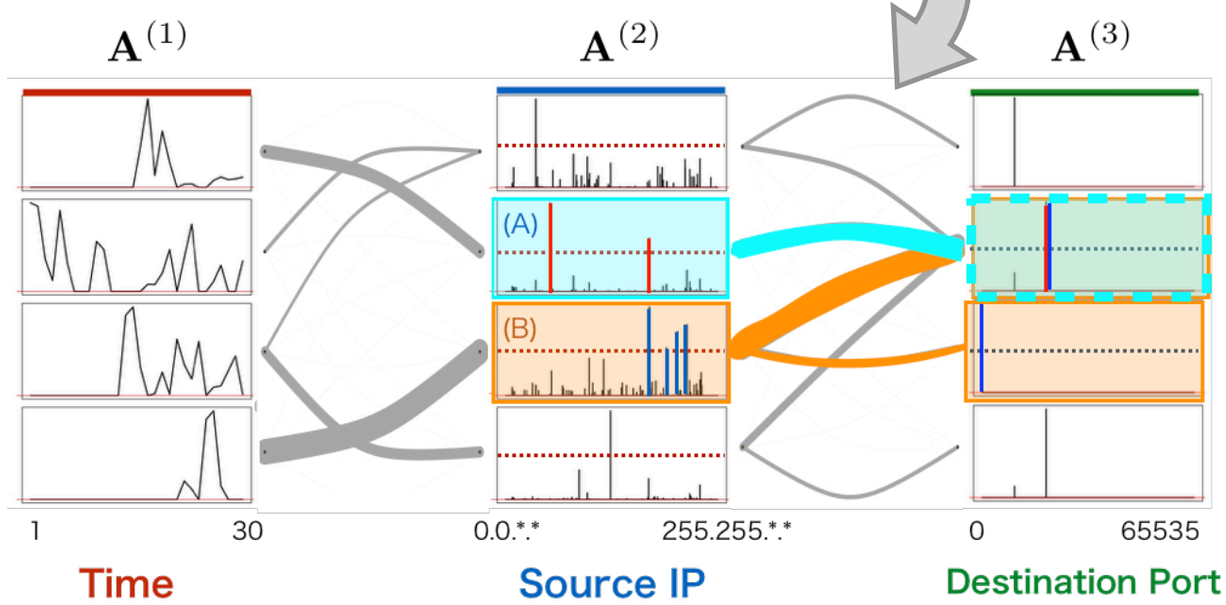
- Extracting frequent patterns
- Real-time Tensor factorization
  - LRA-NTD
  - FSTD

30min $30 \times 2^{16} \times 2^{16}$



Group activity detection stage

- thresholding to raise alerts
- If group activities exist, then trying to identify their src IPs and dst ports



Data input stage | Tensor

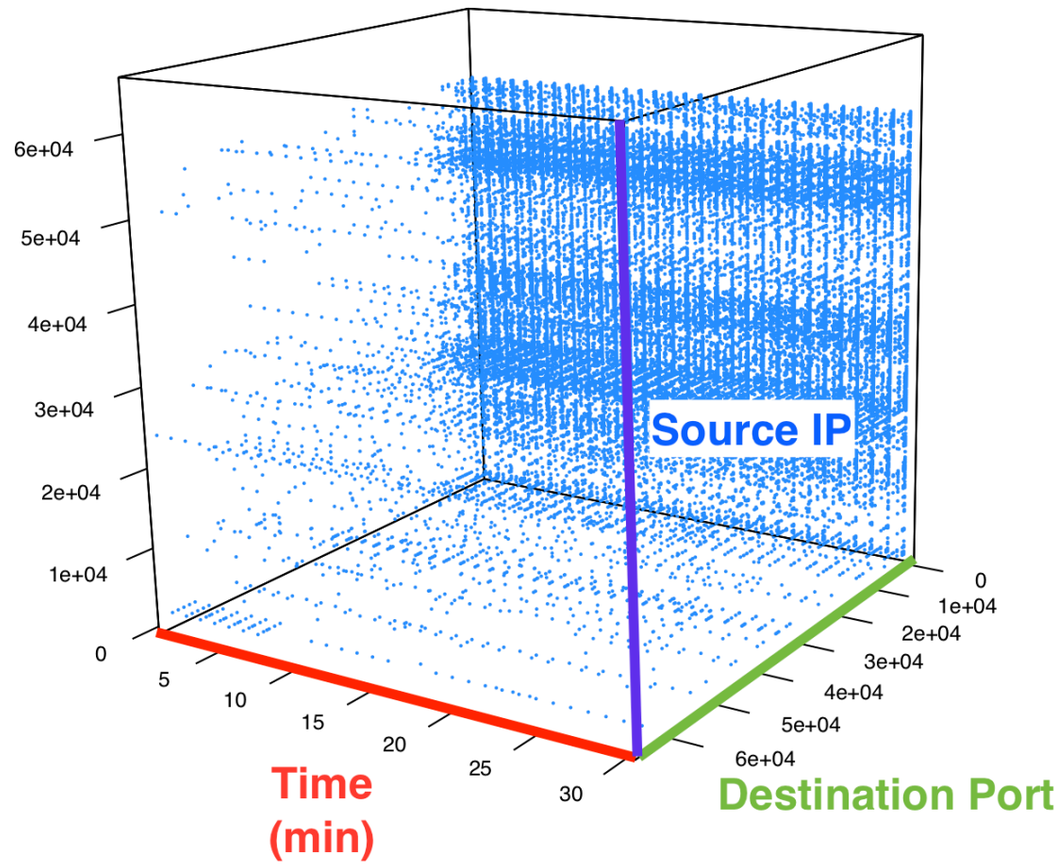
Purpose

Cooperative behavior (botnet) detection

Timestamp	Src IP	Dst port
12:34:56	12.125.x.x	25
12:34:56	252.156.x.x	23
12:34:57	123.35.x.x	8888
12:34:58	156.33.x.x	3218
⋮	⋮	⋮

Input data can be represented as a ***tensor*** **(multidimensional array)**
Timestamp • **Src IP** • **Dst Port**

$$30 \times 2^{16} \times 2^{16} \approx 10^{11}$$
$_{\text{min}}$ $_{\text{IP}}$ $_{\text{port}}$ **elements**

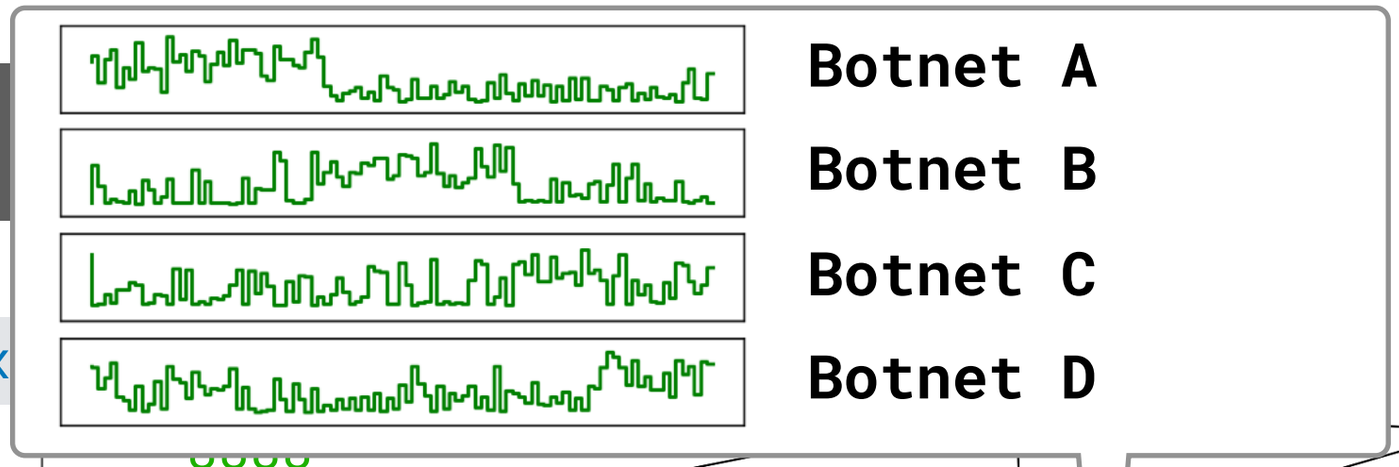


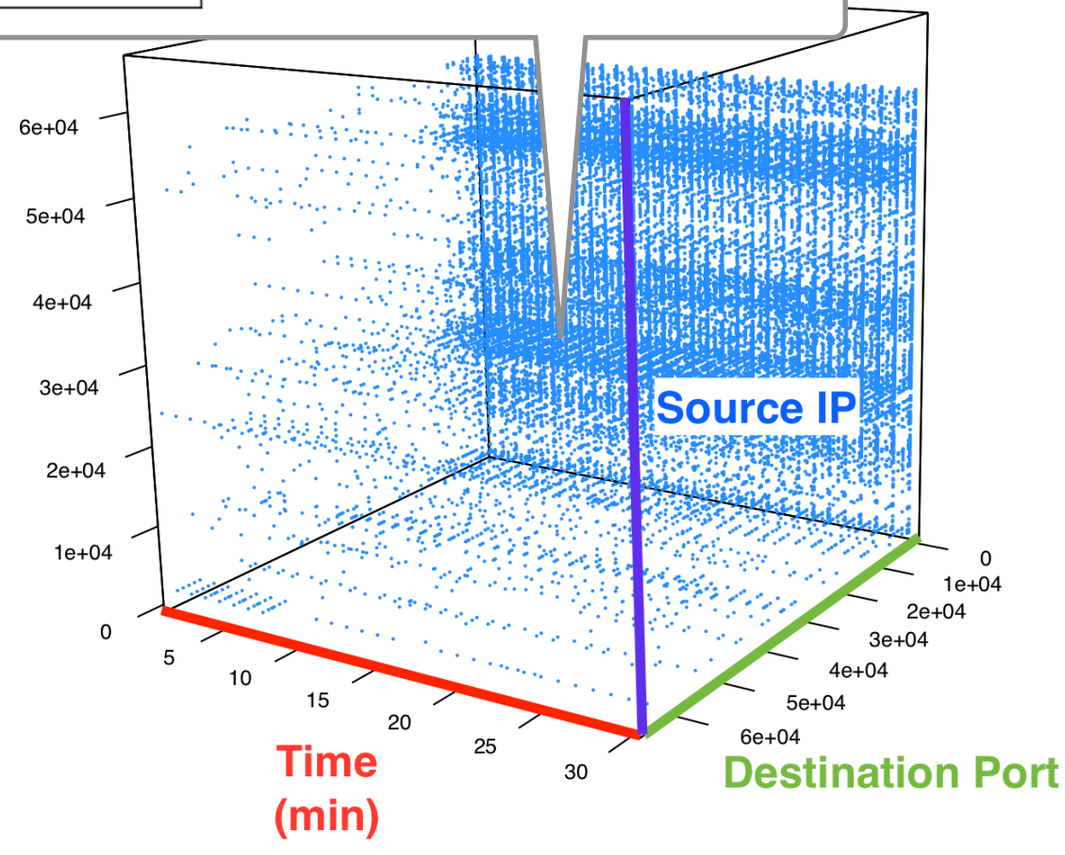
Data input stage | Tensor

Purpose

Cooperative behavior (botnet) detection

Timestamp	Src IP	
12:34:56	12.125.x.x	
12:34:56	252.156.x.x	
12:34:57	123.35.x.x	
12:34:58	156.33.x.x	3218
⋮	⋮	⋮





Input data can be represented as a ***tensor*** **(multidimensional array)**

Timestamp • **Src IP** • **Dst Port**

$$30_{\text{min}} \times 2^{16}_{\text{IP}} \times 2^{16}_{\text{port}} \approx 10^{11} \text{ elements}$$

Why factorization?

Simplify the problem:

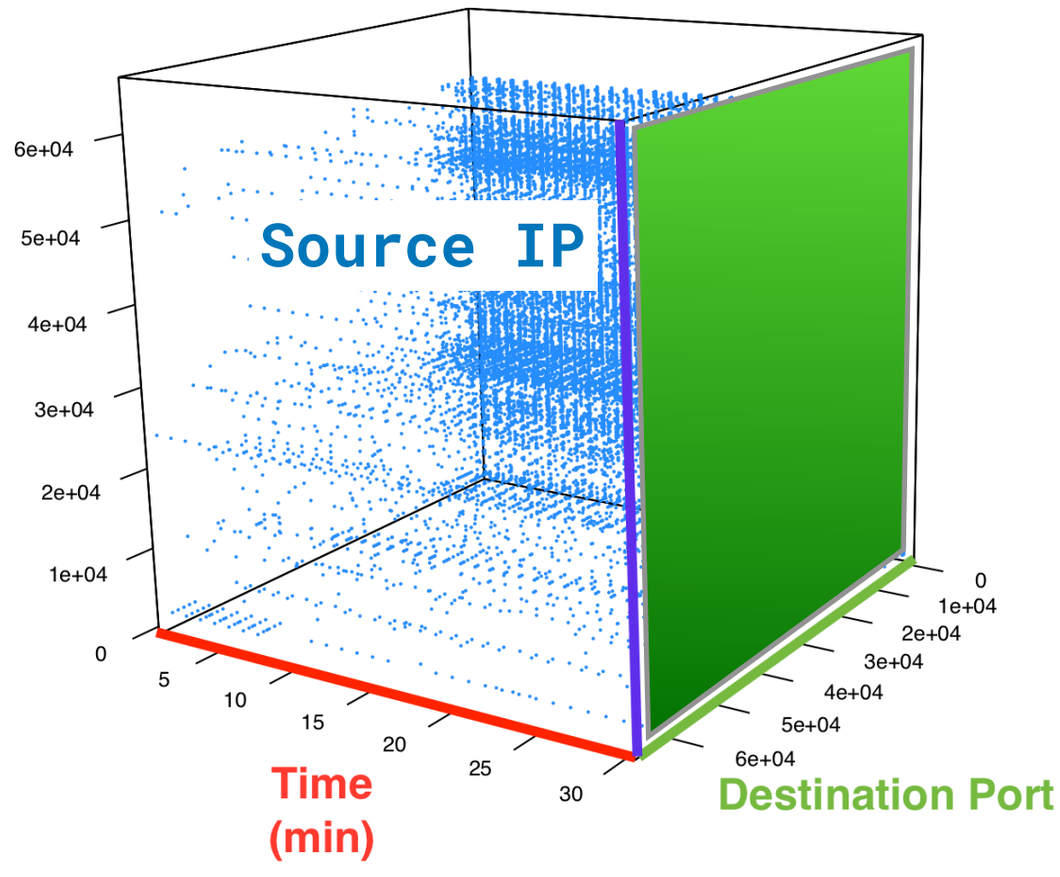
Grouping similar hosts from src IP and dst Port

Timestamp	Src IP	Dst Port
12:34:56	12.125.x.x	25
12:34:56	252.156.x.x	23
12:34:57	123.35.x.x	8888
12:34:58	156.33.x.x	3218
⋮	⋮	⋮

Input data can be represented as a matrix

Timestamp • Src IP • Dst Port

$$2^{16}_{\text{IP}} \times 2^{16}_{\text{port}}$$

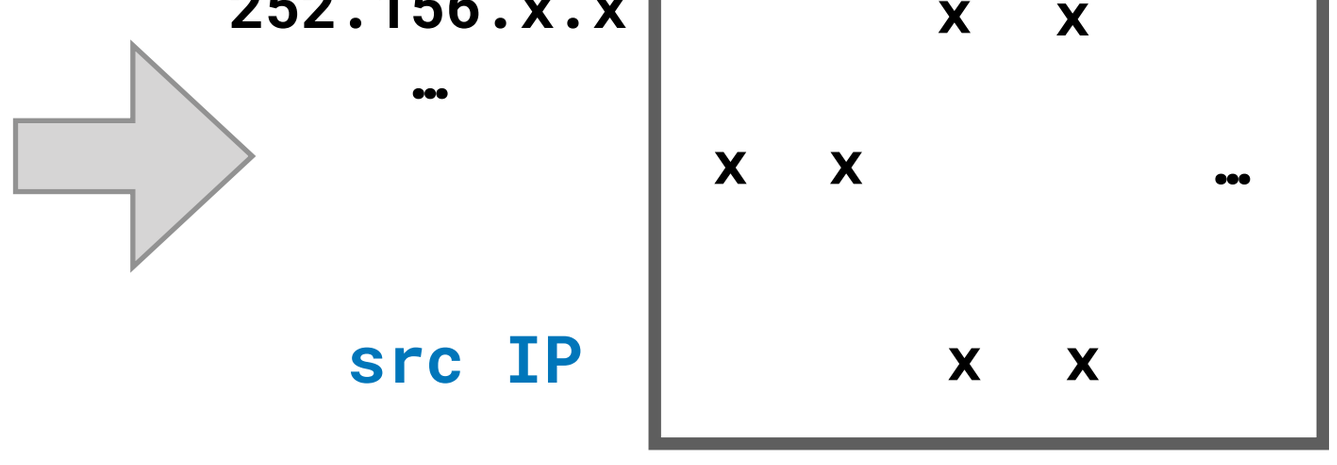


Why factorization?

Simplify the problem:

Grouping similar hosts from src IP and dst Port

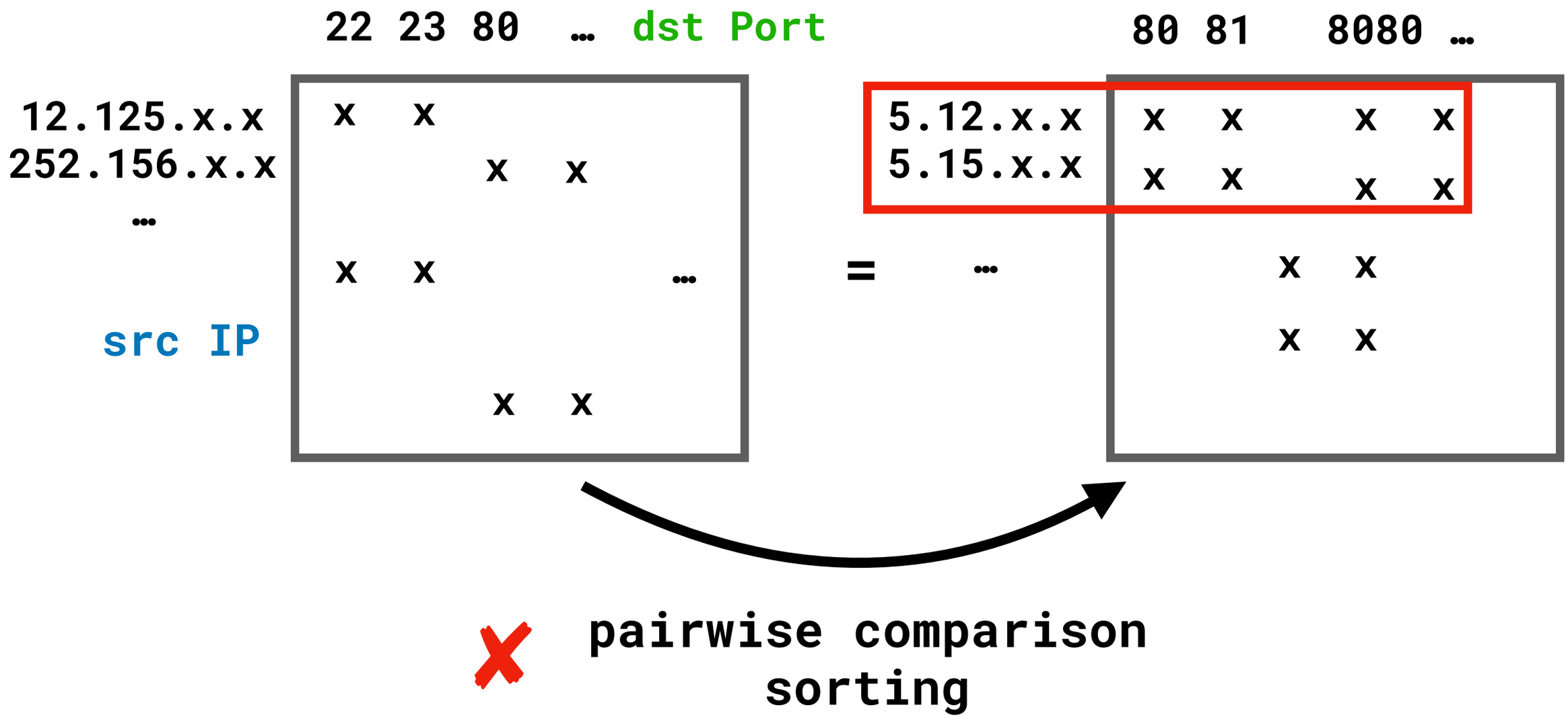
Src IP	Dst Port
12.125.x.x	22
12.125.x.x	23
252.156.x.x	80
252.156.x.x	8080
⋮	⋮



Why factorization?

Simplify the problem:

Grouping similar hosts from src IP and dst Port



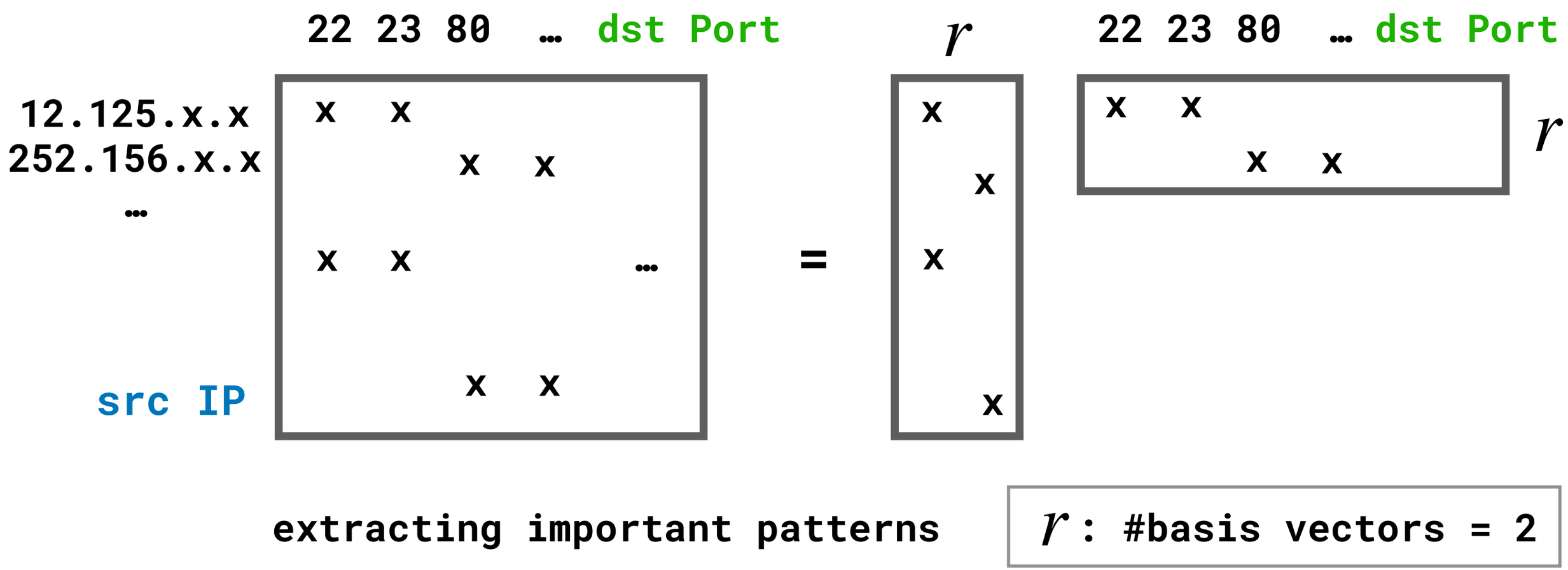
✗ pairwise comparison sorting

Why factorization?

Simplify the problem:

Grouping similar hosts from src IP and dst Port

-> One Solution: apply the matrix factorization



extracting important patterns

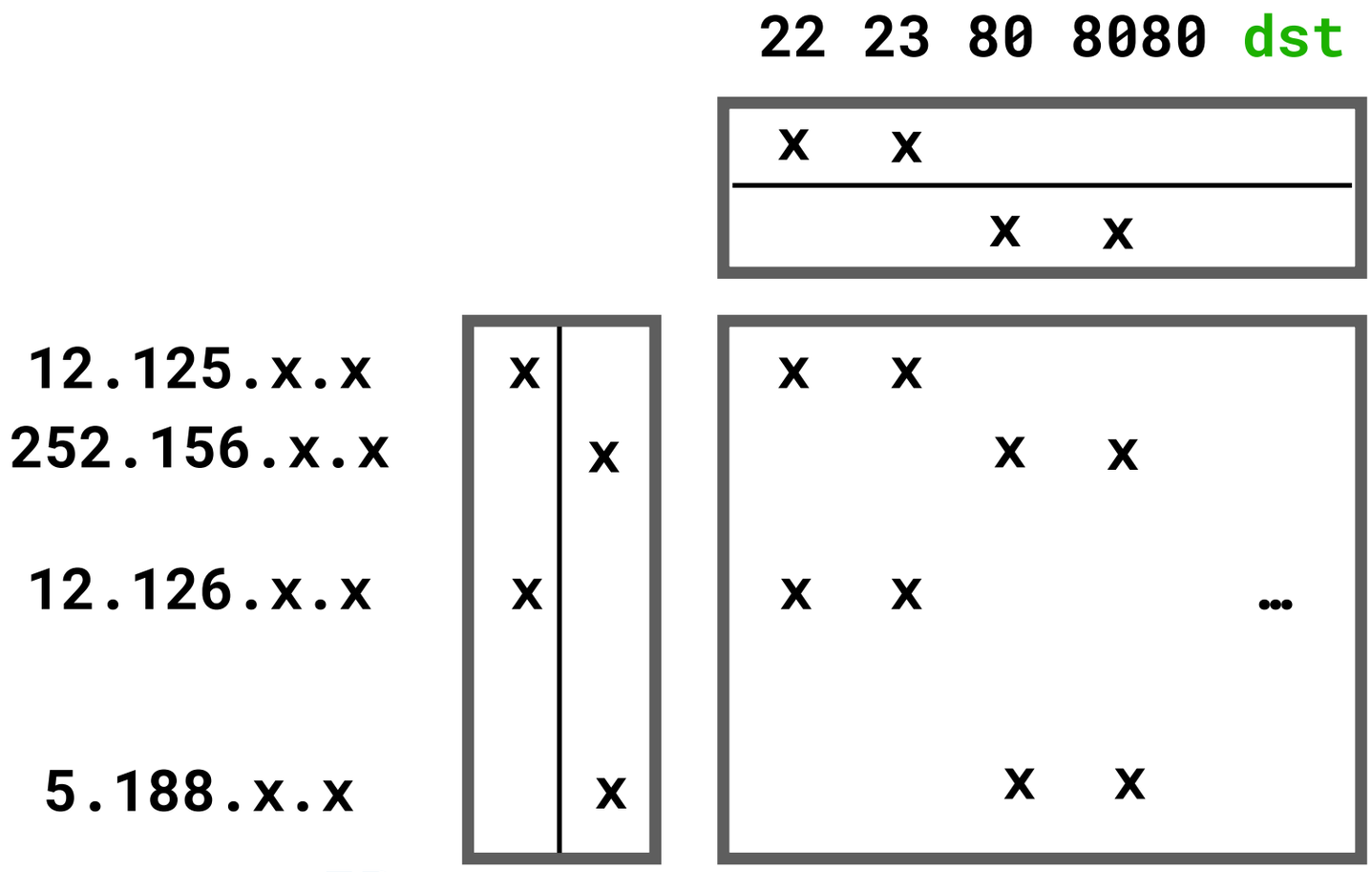
r : #basis vectors = 2

Why factorization?

Simplify the problem:

Grouping similar hosts from src IP and dst Port

-> One Solution: apply the matrix factorization

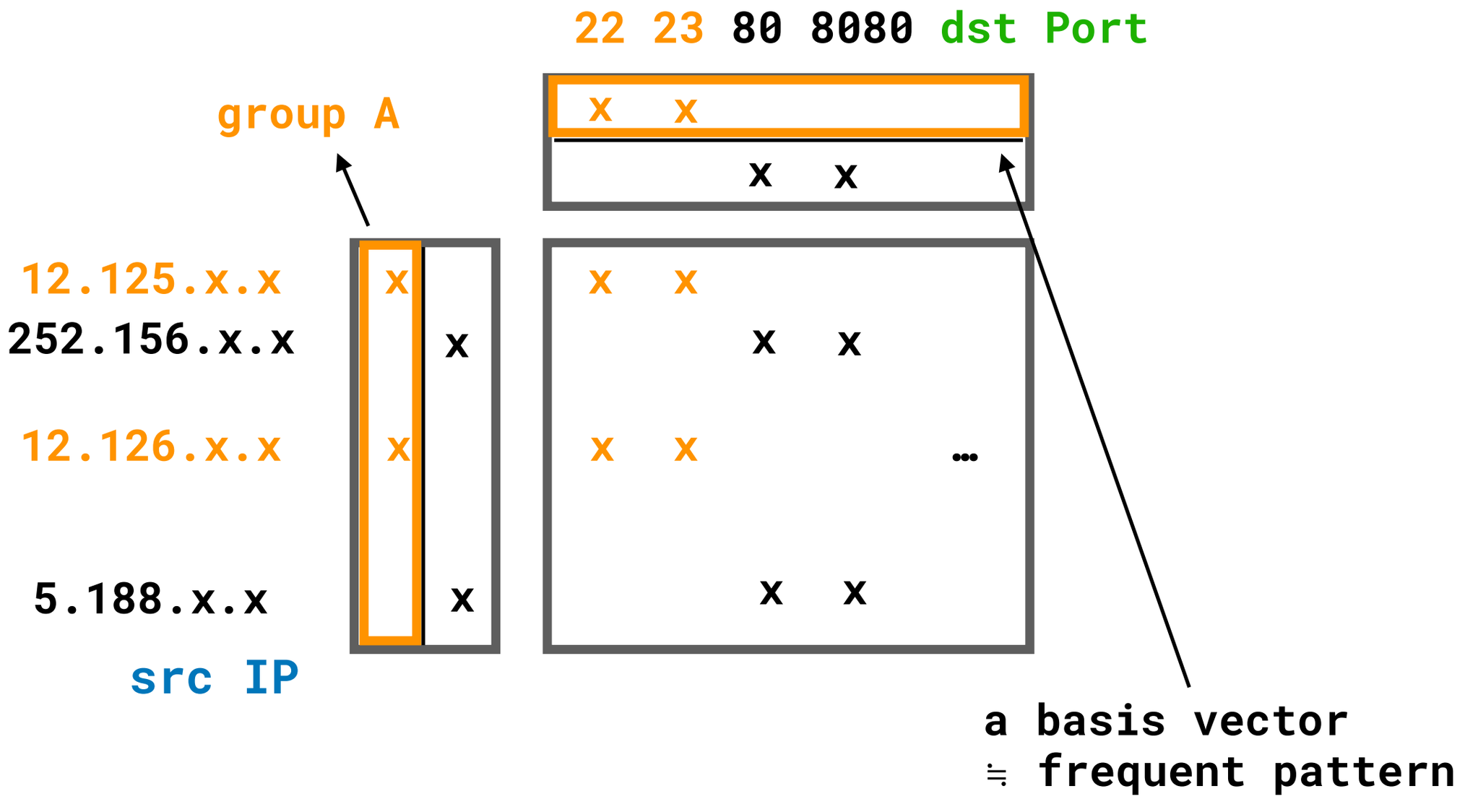


Why factorization?

Simplify the problem:

Grouping similar hosts from src IP and dst Port

-> One Solution: apply the matrix factorization

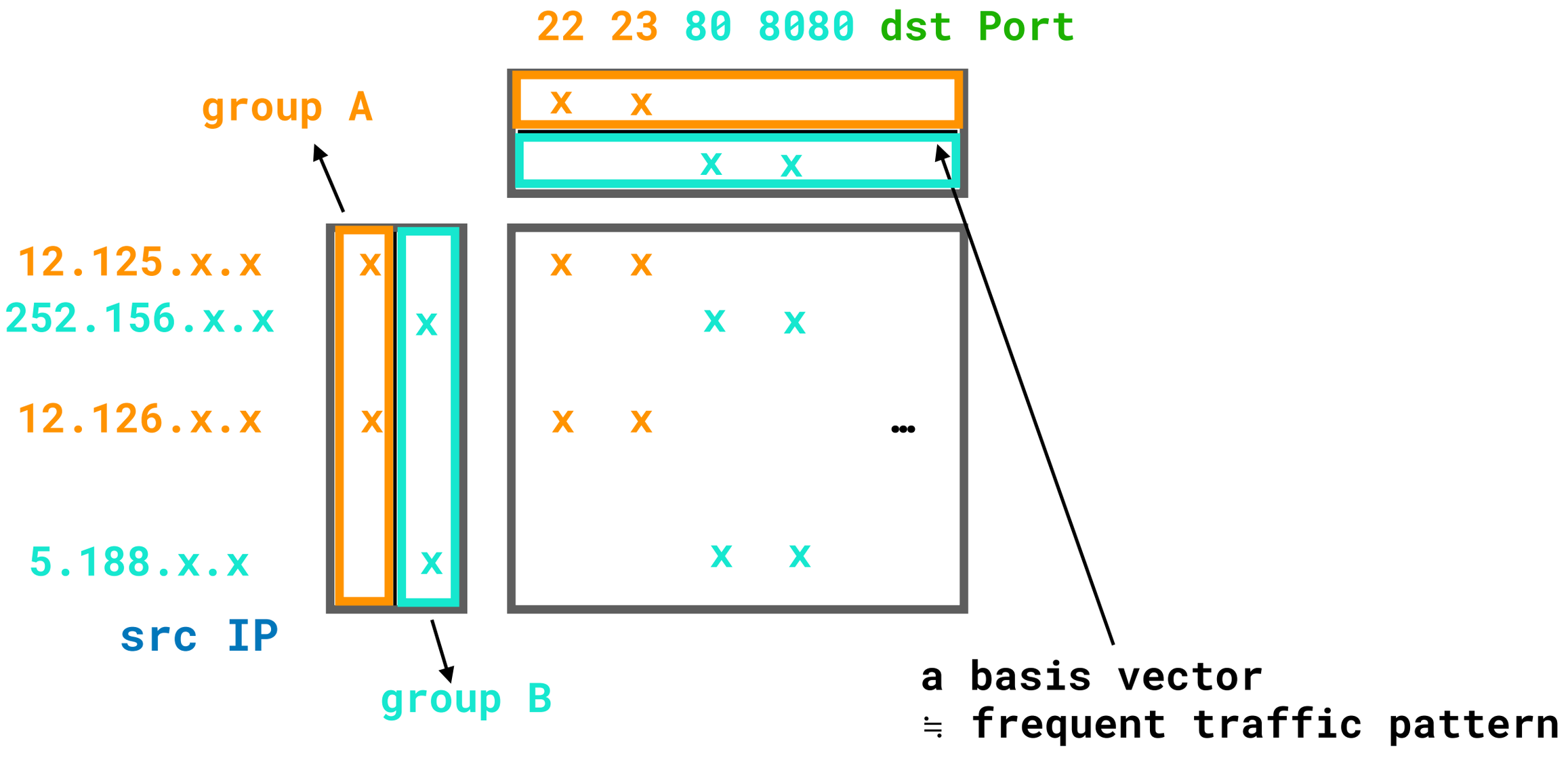


Why factorization?

Simplify the problem:

Grouping similar hosts from src IP and dst Port

-> One Solution: apply the matrix factorization

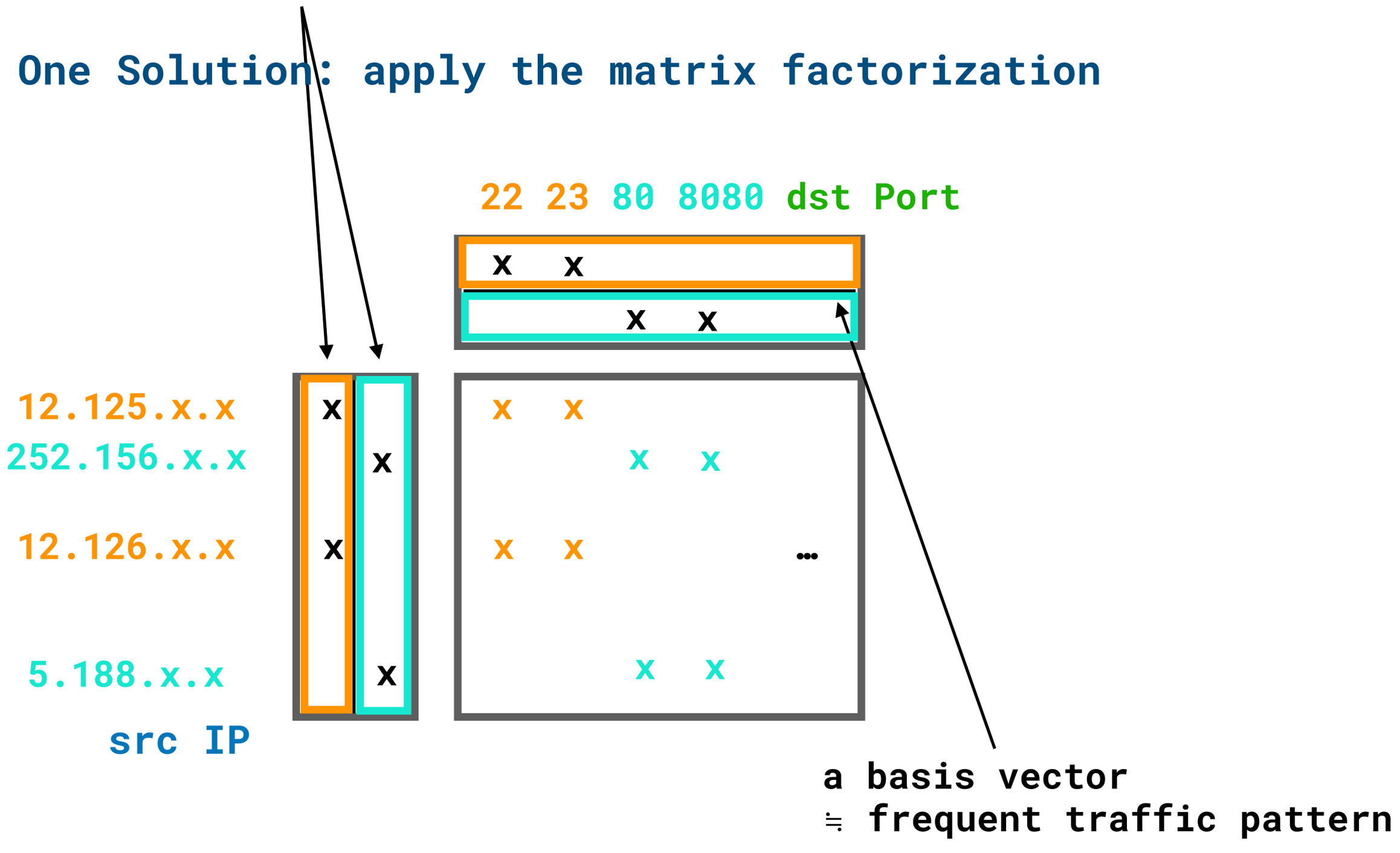


Why factorization?

Simplify the problem:

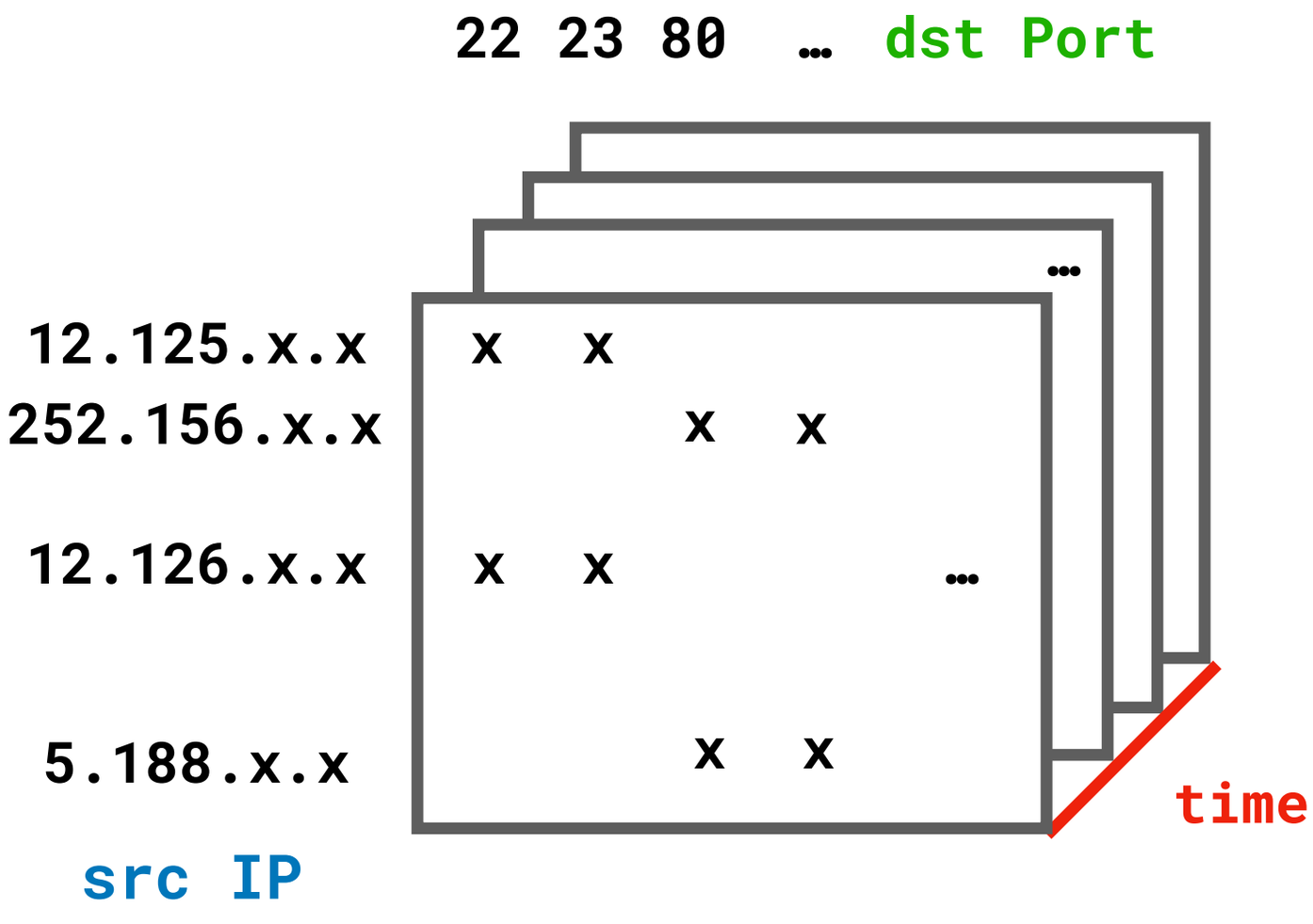
Grouping similar hosts from src IP and dst Port

-> One Solution: apply the matrix factorization



Tensor factorization

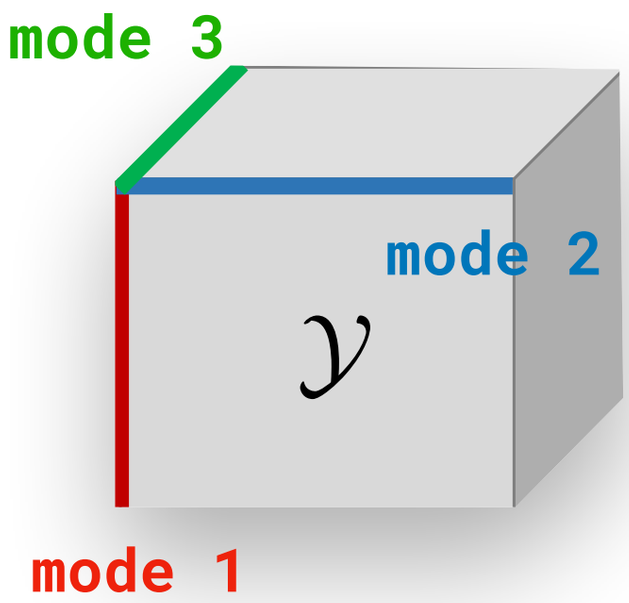
- Tensor factorization
  - higher-order extension of matrix factorization



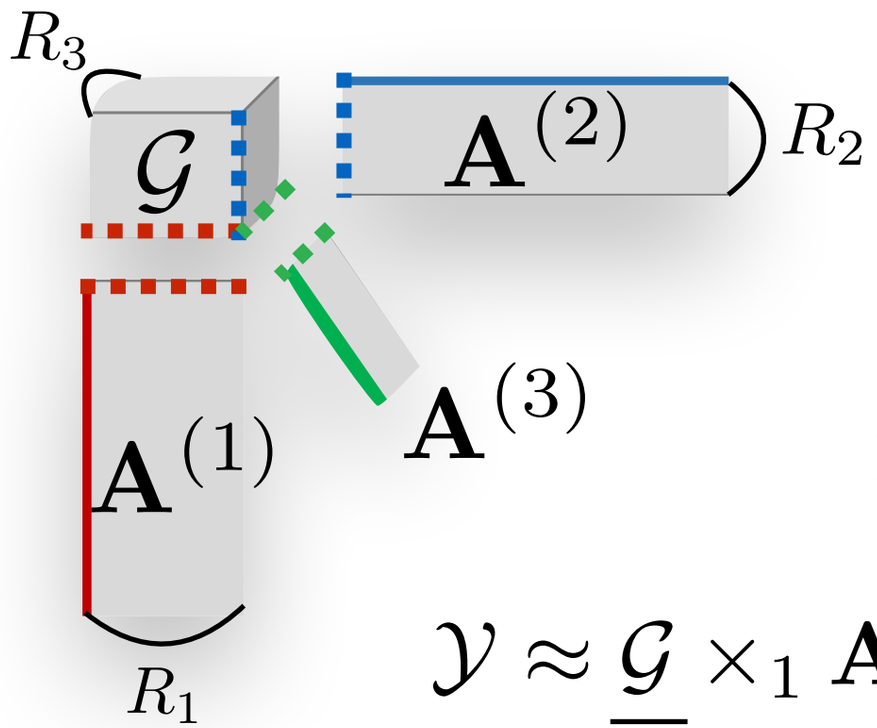
=

?

Tucker decomposition



$\approx$



$\times_n$: mode-n product

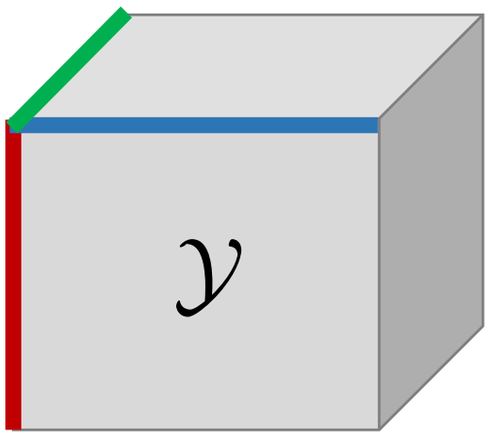
$$\mathcal{Y} \approx \underline{\mathcal{G}} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \mathbf{A}^{(3)}$$

weights

- $\mathbf{A}^{(n)}$: a factor matrix, set of frequent patterns
- $\mathcal{G}$: a core tensor
- R_n : #basis vectors of mode n
  - The larger, the better <=> computational cost

Nonnegativity constraint -> Nonnegative Tucker Decomposition (NTD)

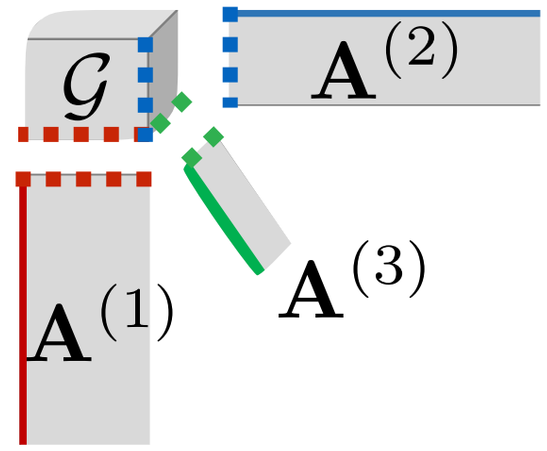
Real-time NTD implementation



NTD [Kim+,2007]

$$\min_{\mathcal{G},\mathbf{A}} \frac{1}{2}||\mathcal{Y} - \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \mathbf{A}^{(3)}||^2$$

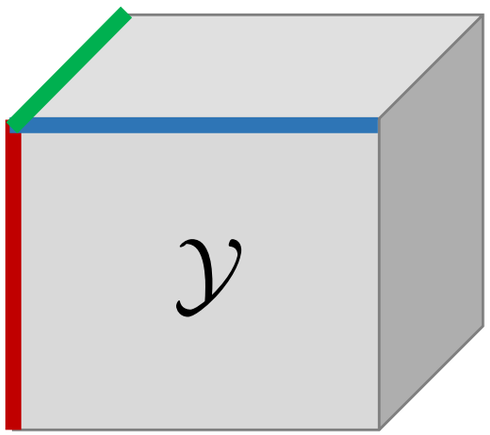
NTD



$$\mathbf{A}^{(n)} \leftarrow \mathbf{A}^{(n)} - \eta_{\mathbf{A}^{(n)}} \circledast \frac{\partial D}{\partial \mathbf{A}^{(n)}}$$

$$\mathcal{G} \leftarrow \mathcal{G} - \eta_{\mathcal{G}} \circledast \frac{\partial D}{\partial \mathcal{G}}$$

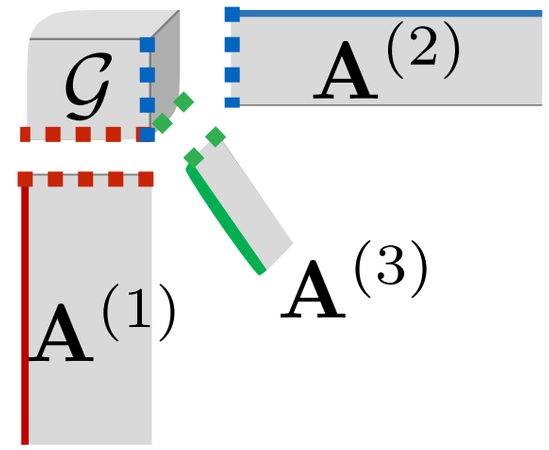
Real-time NTD implementation



NTD [Kim+,2007]

NTD

$$\min_{\mathcal{G},\mathbf{A}} \frac{1}{2}||\mathcal{Y} - \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \mathbf{A}^{(3)}||^2$$

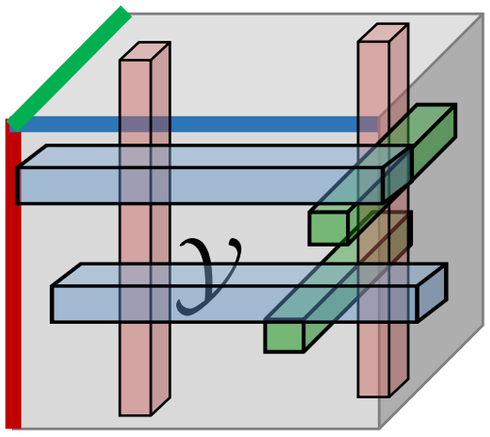


$$\mathcal{Y} \in \mathbb{R}^{30 \times 2^{16} \times 2^{16}}$$

$$\mathbf{A}^{(n)} \leftarrow \mathbf{A}^{(n)} - \eta_{\mathbf{A}^{(n)}} \circledast \frac{\partial D}{\partial \mathbf{A}^{(n)}}$$

$$\mathcal{G} \leftarrow \mathcal{G} - \eta_{\mathcal{G}} \circledast \frac{\partial D}{\partial \mathcal{G}}$$

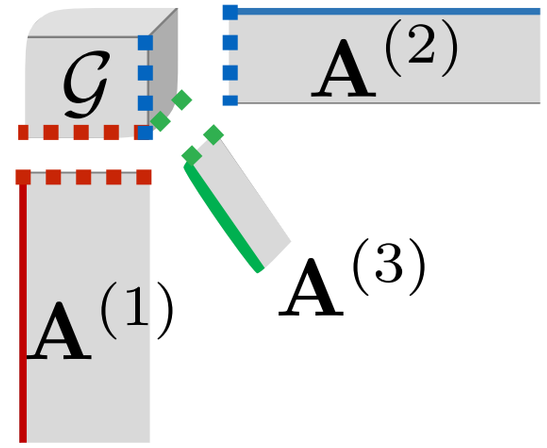
Real-time NTD implementation



NTD [Kim+,2007]

NTD

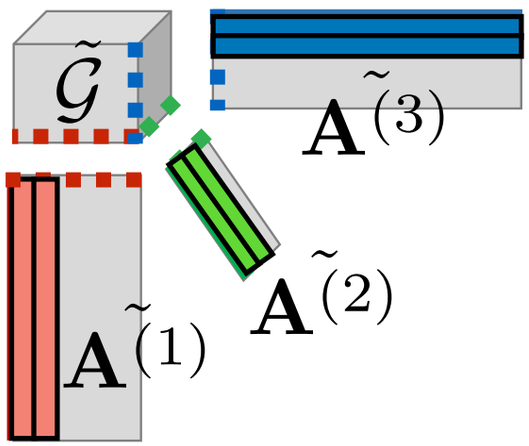
$$\min_{\mathcal{G},\mathbf{A}} \frac{1}{2}||\mathcal{Y} - \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \mathbf{A}^{(3)}||^2$$



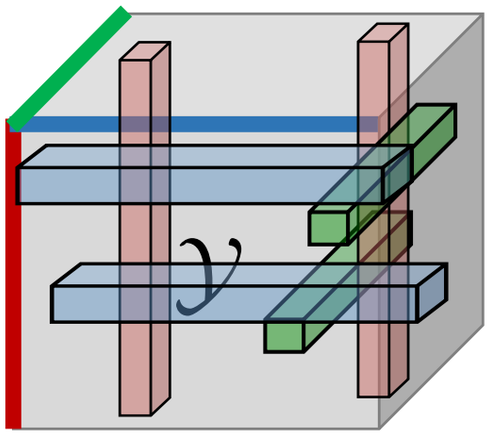
FSTD [Caiafa+,2010]

one of the fastest decomposition:
sampling the important *fibers*

Fiber Sampling Tensor Decomposition



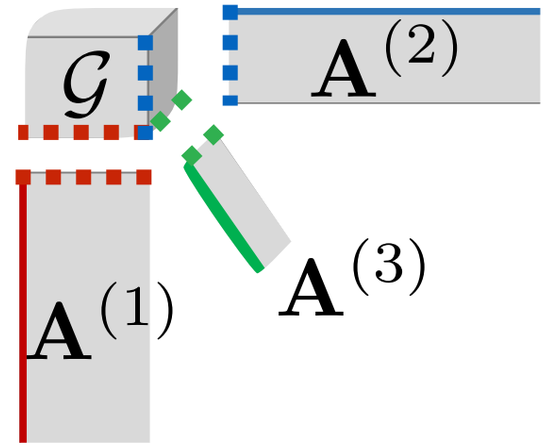
Real-time NTD implementation



NTD [Kim+,2007]

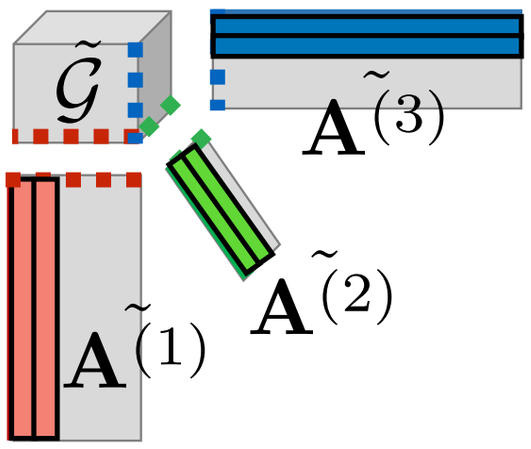
$$\min_{\mathcal{G},\mathbf{A}} \frac{1}{2}||\mathcal{Y} - \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \mathbf{A}^{(3)}||^2$$

NTD



FSTD [Caiafa+,2010]

one of the fastest decomposition:
sampling the important *fibers*



LRA-NTD [Zhou+,2015]

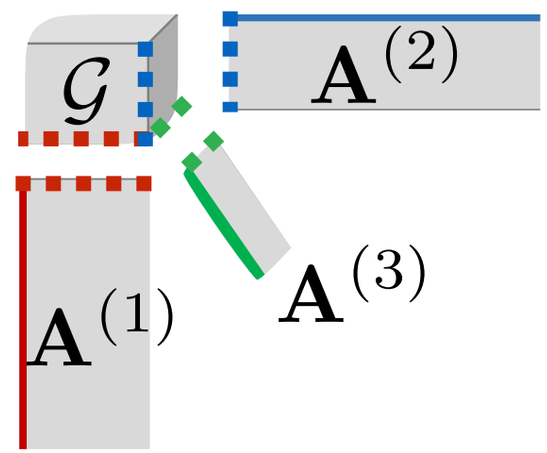
efficient NTD based
on two-step algorithm

$$\min_{\mathcal{G},\mathbf{A}} \frac{1}{2}||\tilde{\mathcal{Y}} - \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \mathbf{A}^{(3)}||^2$$

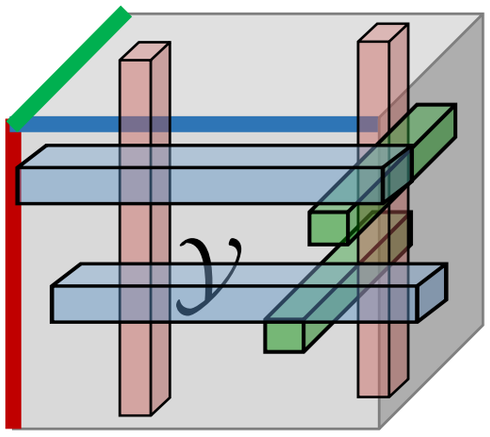
$$\tilde{\mathcal{Y}} = \tilde{\mathcal{G}} \times_1 \tilde{\mathbf{A}}^{(1)} \times_2 \tilde{\mathbf{A}}^{(2)} \times_3 \tilde{\mathbf{A}}^{(3)}$$

Low-Rank Approximation NTD

NTD



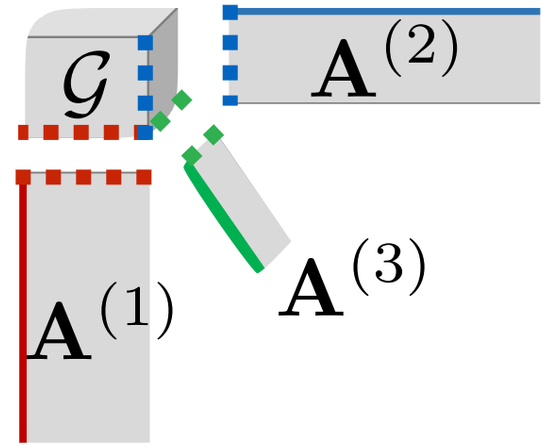
Real-time NTD implementation



NTD [Kim+,2007]

$$\min_{\mathcal{G},\mathbf{A}} \frac{1}{2}||\mathcal{Y} - \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \mathbf{A}^{(3)}||^2$$

NTD

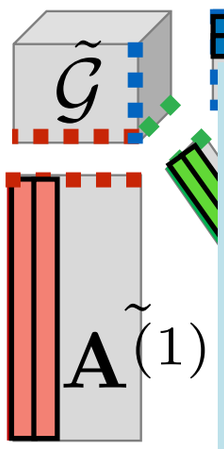


FSTD [Caiafa+,2010]

one of the fastest decomposition:
sampling the important *fibers*

LRA-NTD [Zhou+,2015]

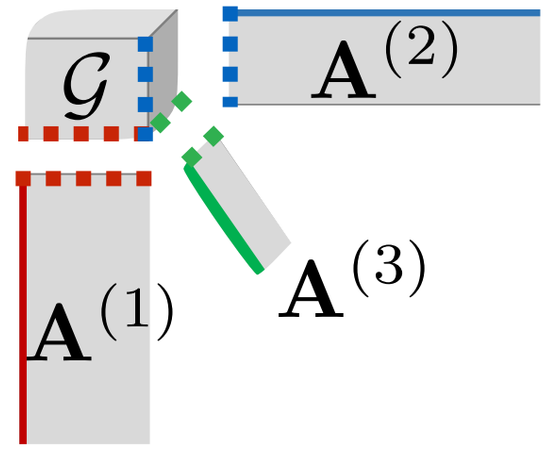
efficient NTD based



Our NTD implementation

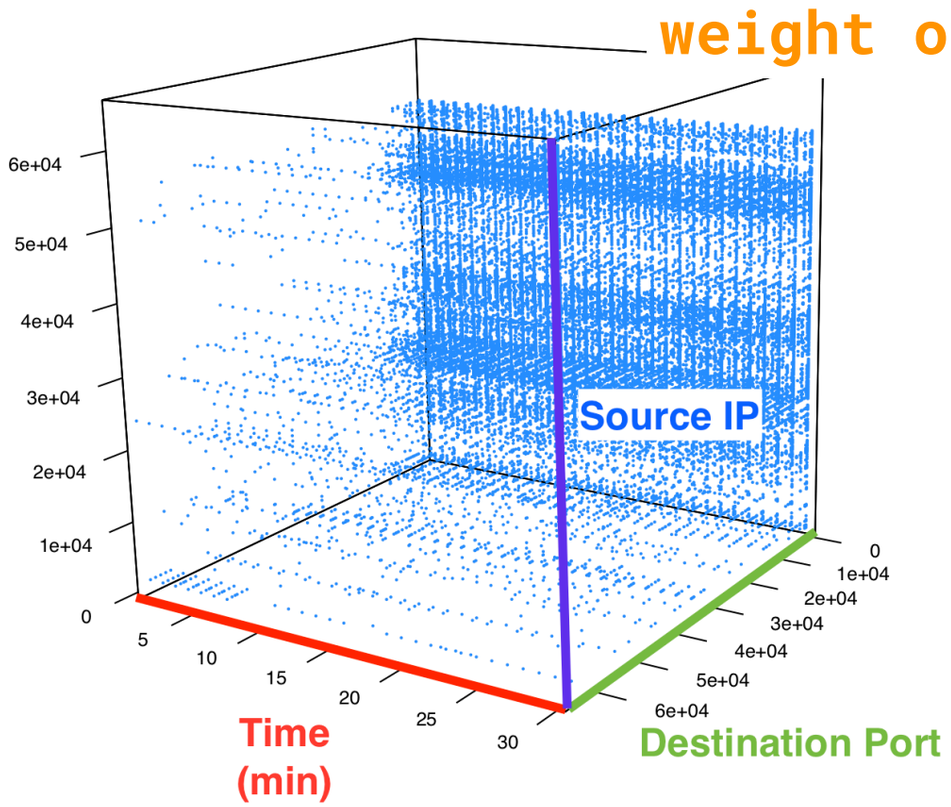
- computational time: 70-90s
- memory usage: 1.9-2.5G

NTD



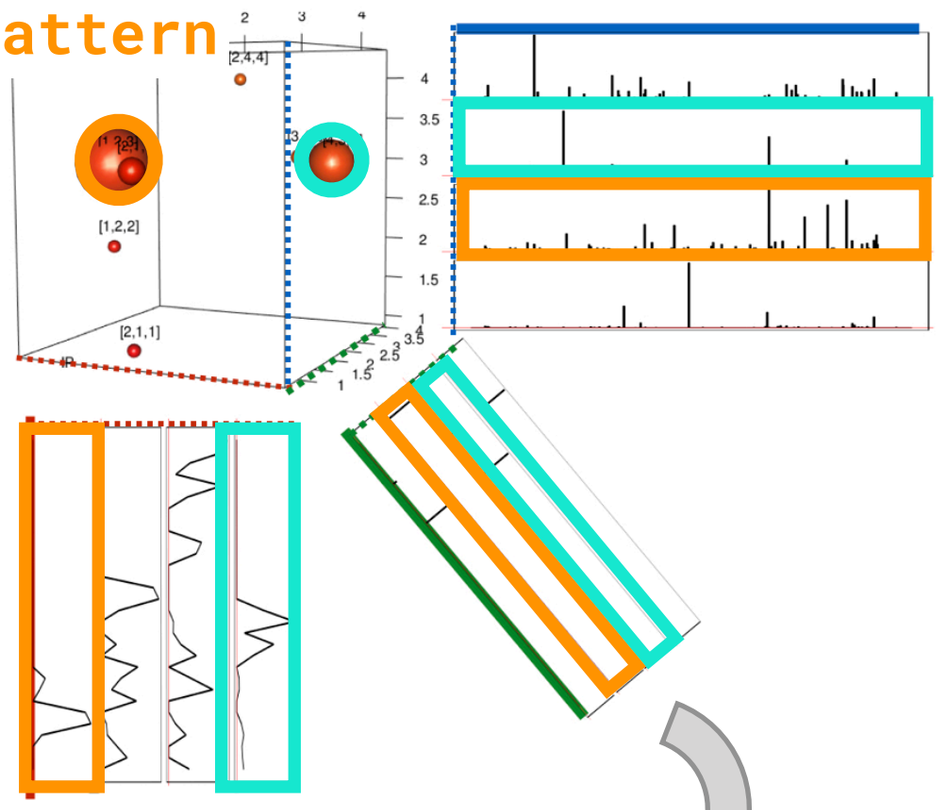
#basis vectors … NTD: $R_1 = R_2 = R_3 = 5$ FSTD: 25
CPU … Intel Xeon X5600 (2.8GHz)

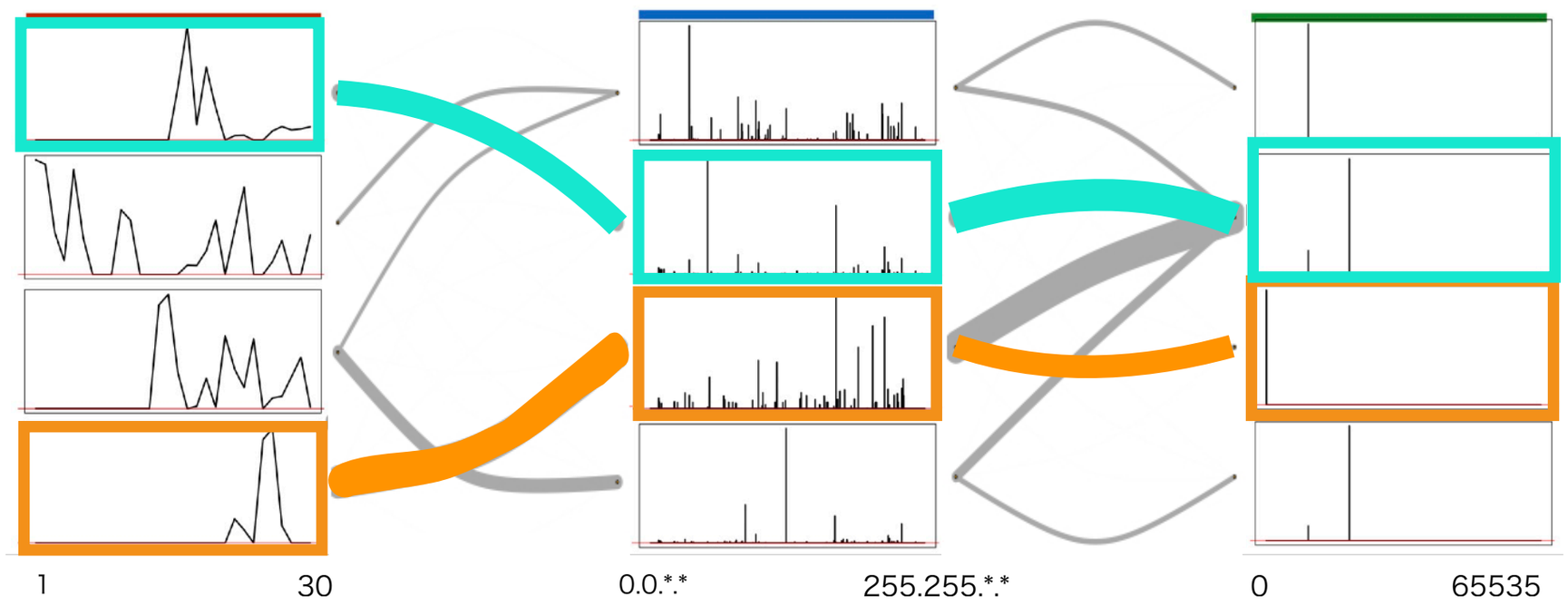
Feature extraction using NTD



≈

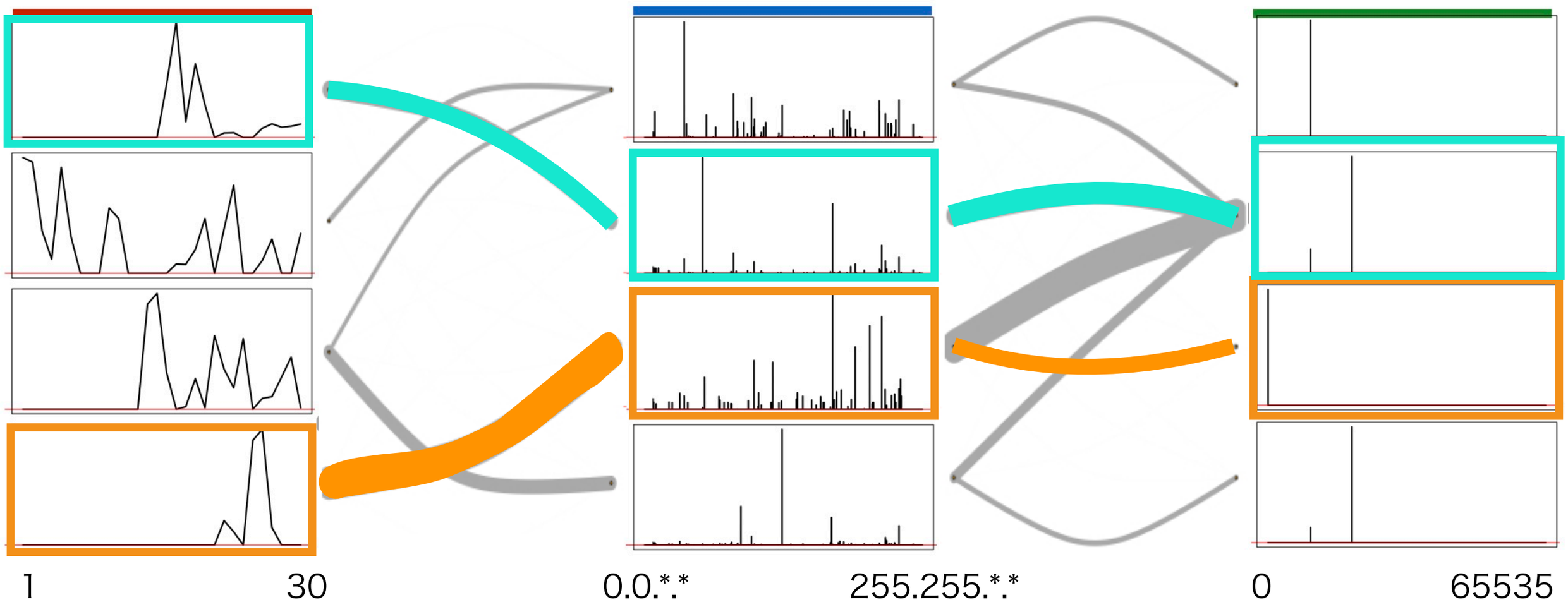
weight of each pattern





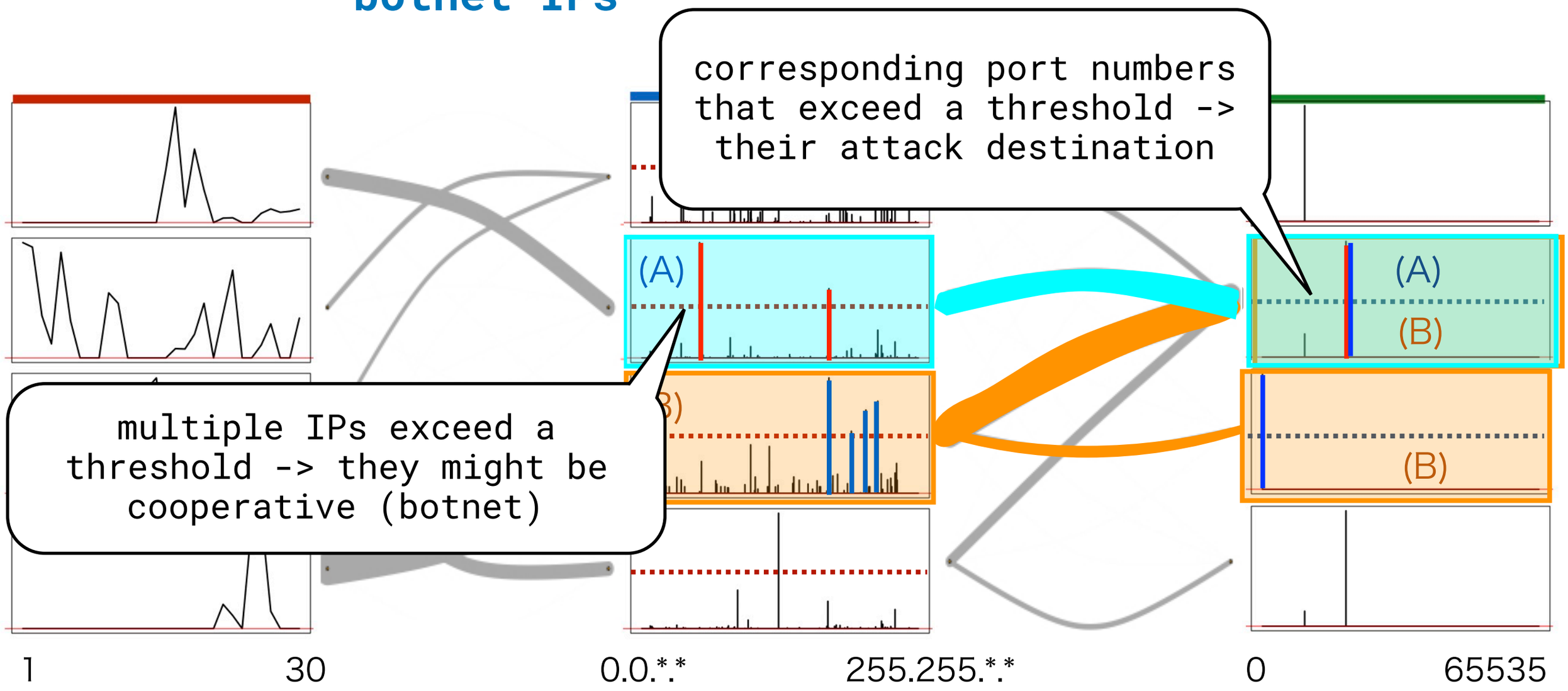
Botnet detection | NTD visualization

- results can be visualized like bipartite graphs
  - edge: core tensor values / node: basis vectors



Botnet detection | thresholding

- Identify src IPs/dst ports of coordinated group **botnet IPs**



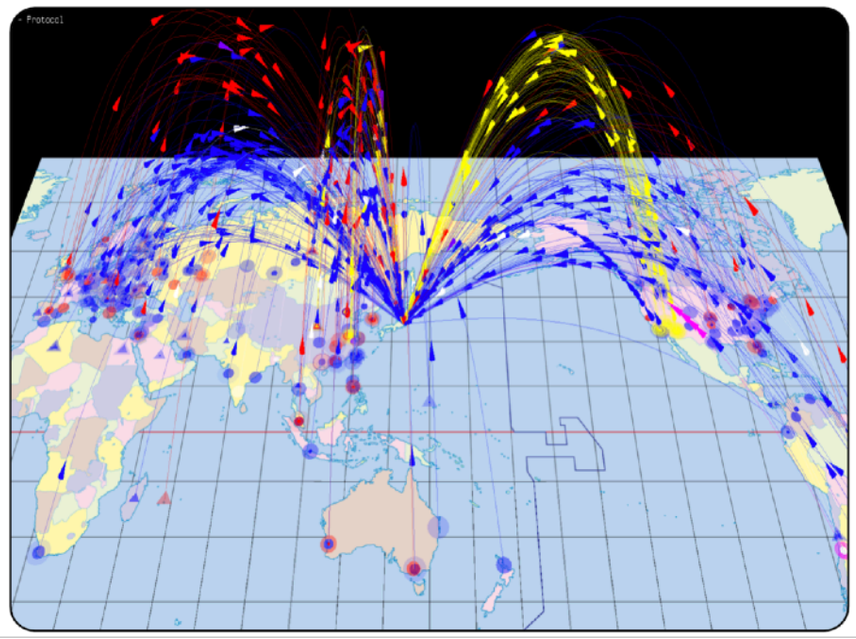
Outline

Experimental settings

- Input: darknet traffic (TCP/UDP) in different countries [1] (#IP addresses ≒ 35k)

- Output: src IPs, dst ports of coordinated groups
  - real-time detection: apply NTD every 30 min

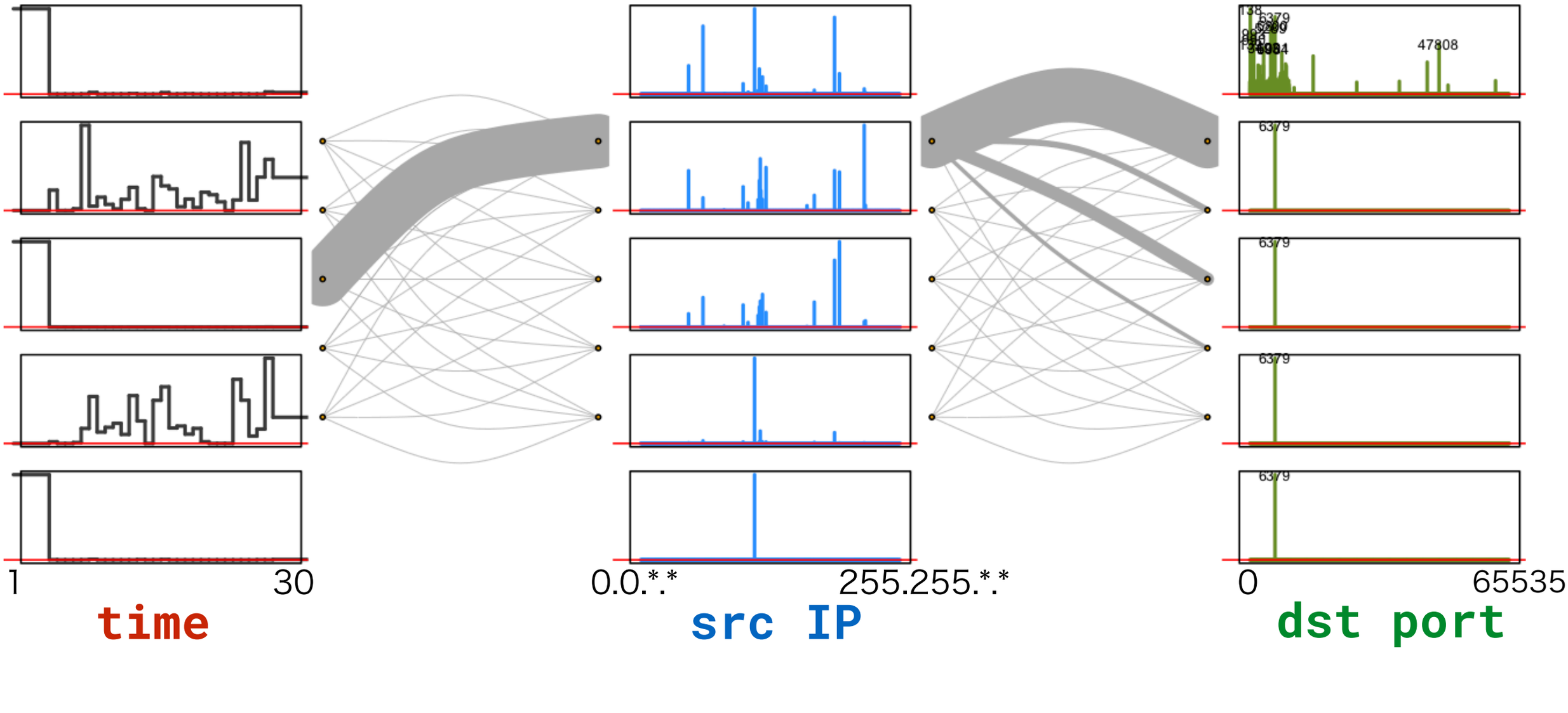
- We introduce one of the interesting result and evaluate qualitatively



[1] NICTER, Inoue et al., 2008

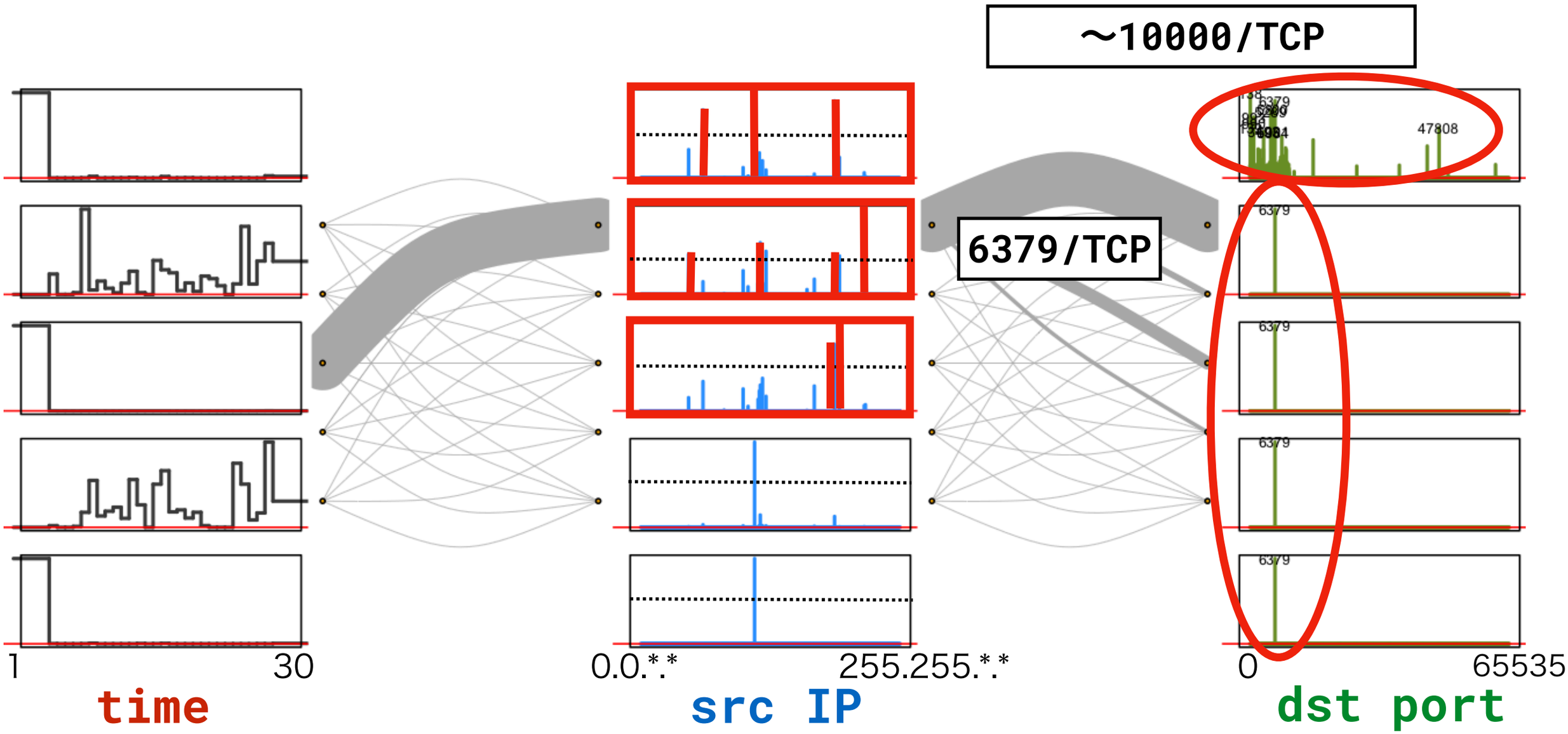
Result | NTD visualization

- factorized result of 5/9 5:30-6:00 TCP traffic



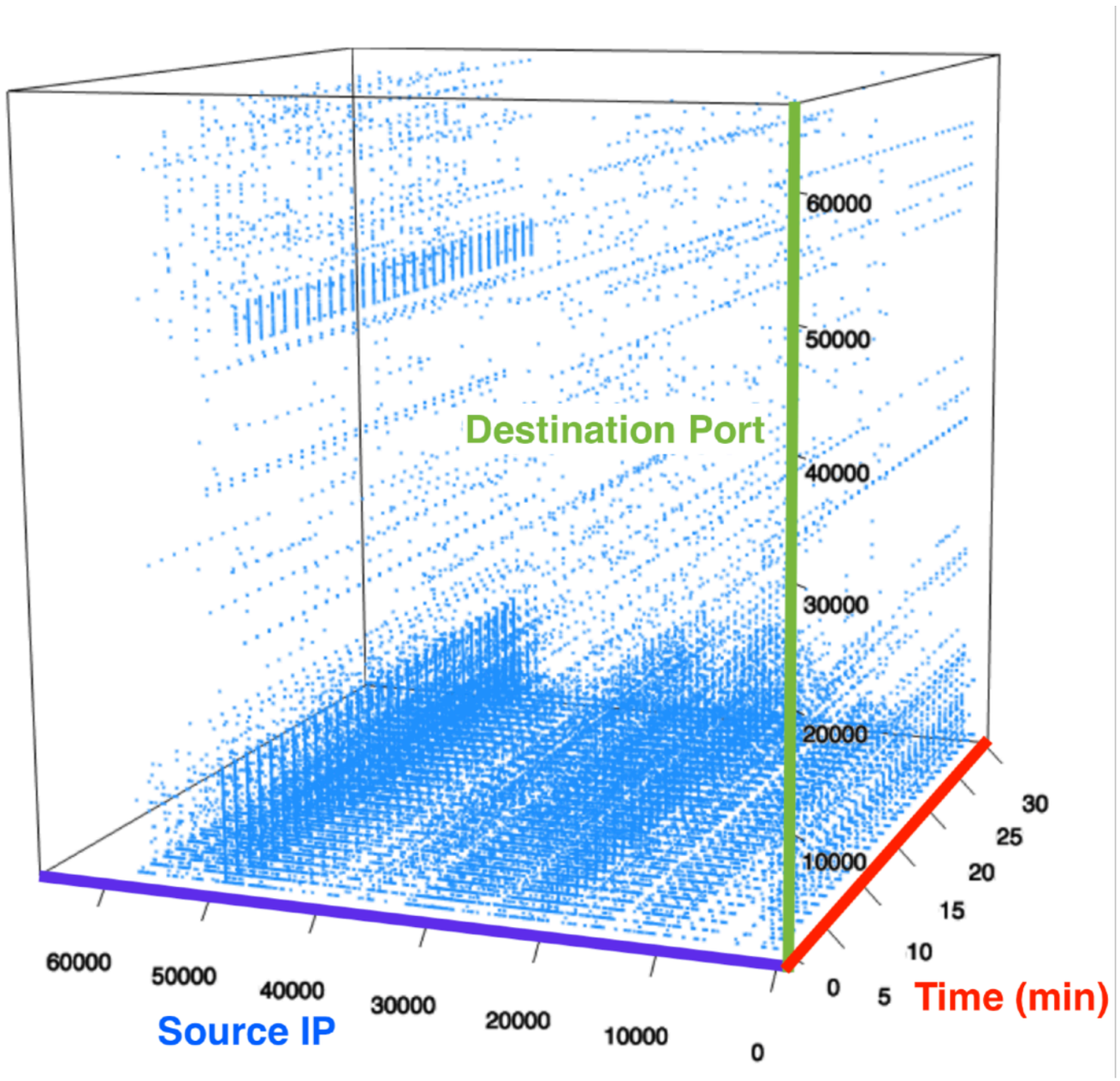
Result | NTD visualization

- factorized result of 5/9 5:30-6:00 TCP traffic
  - identifying the botnet IPs and their dst ports



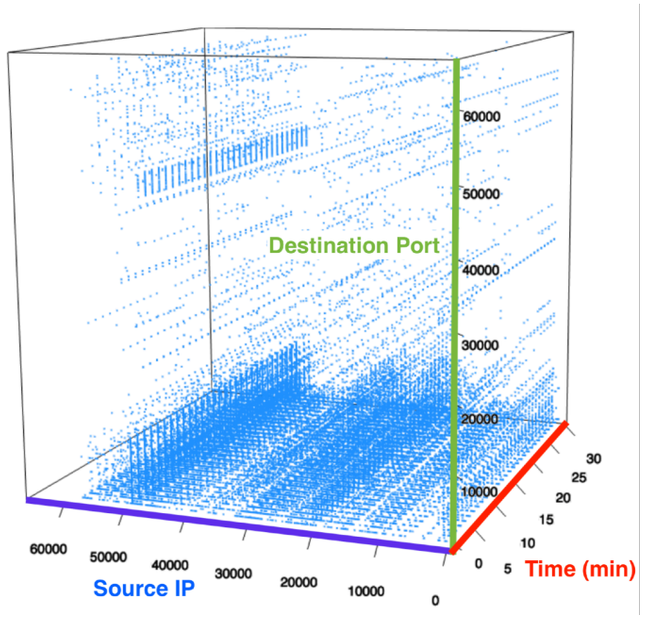
Result | Original traffic

- original darknet TCP traffic (5/9 5:30-6:00)



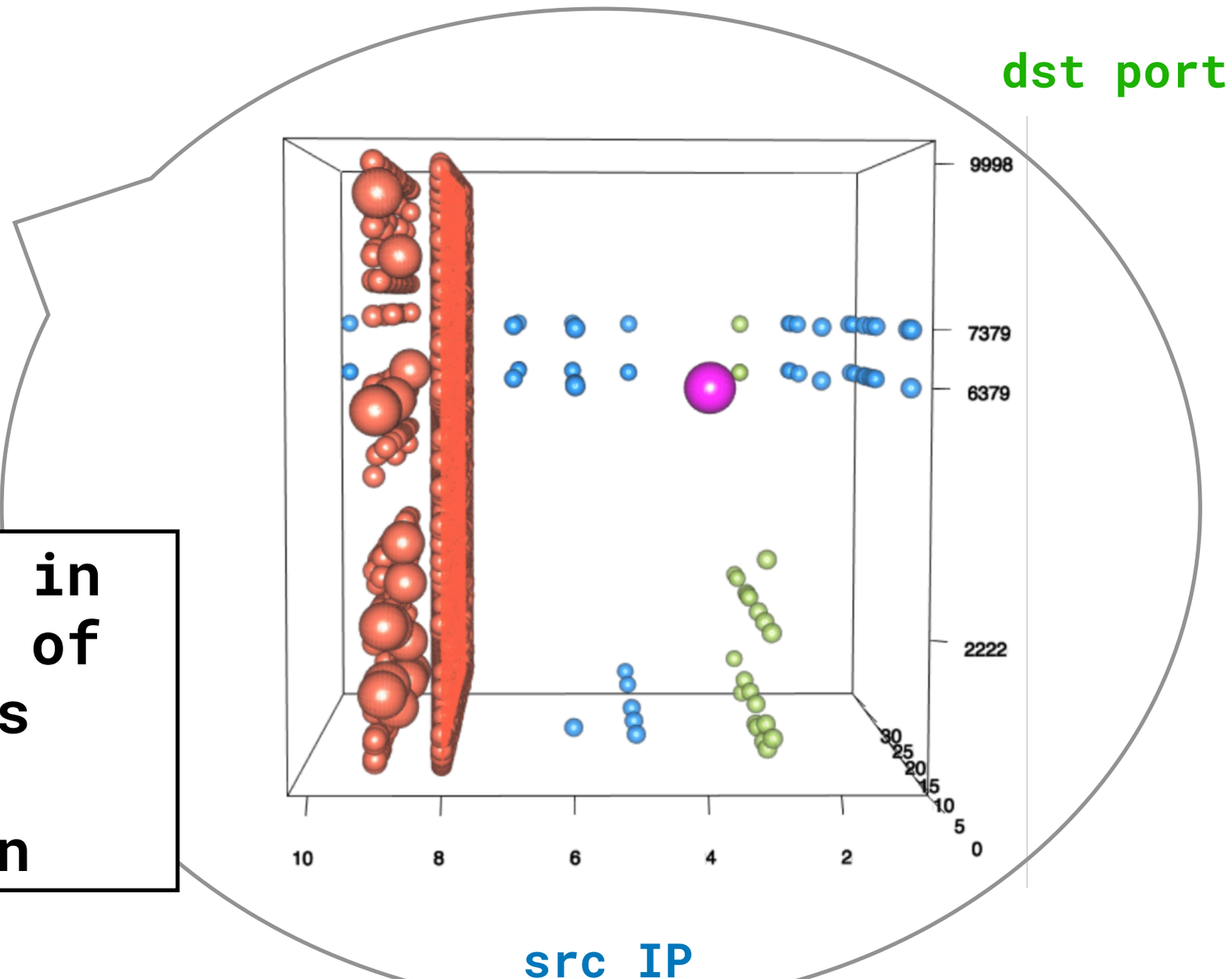
Result | Original traffic

- original darknet TCP traffic (5/9 5:30-6:00)
  - filtered by botnet IPs

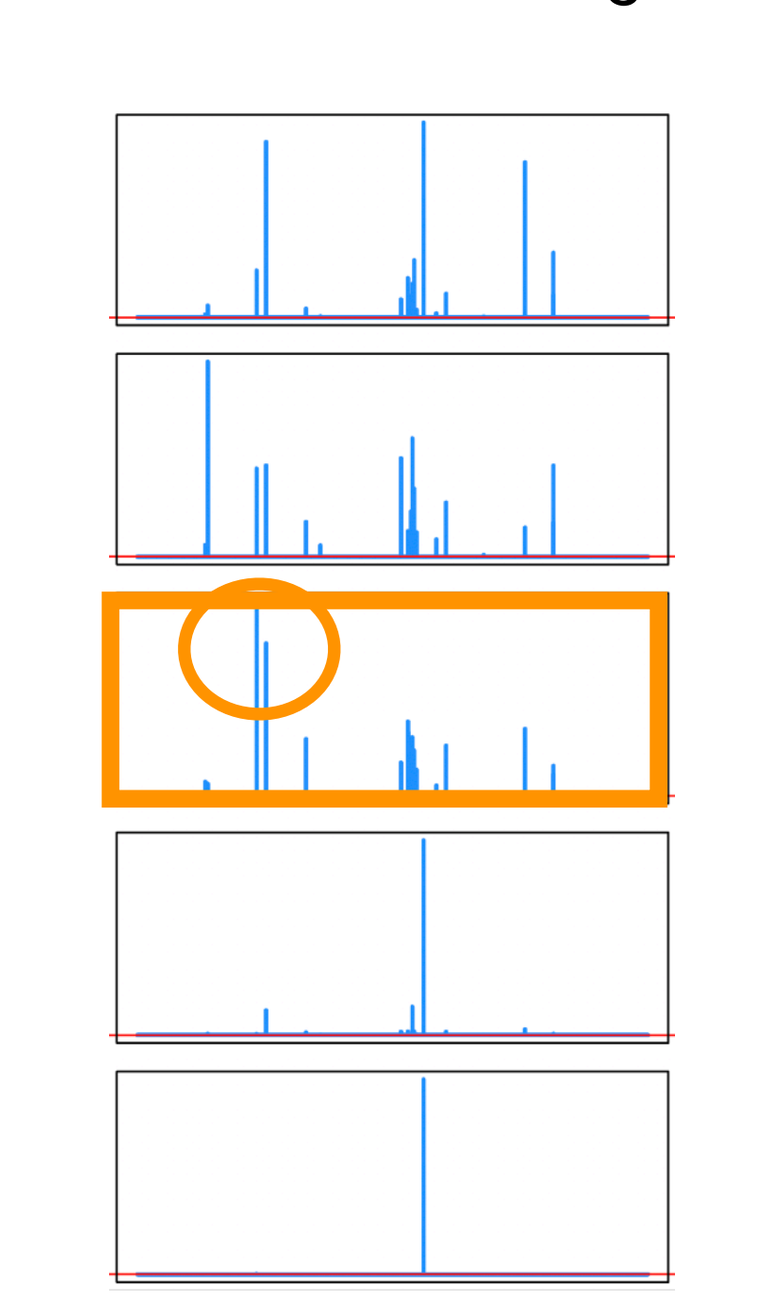


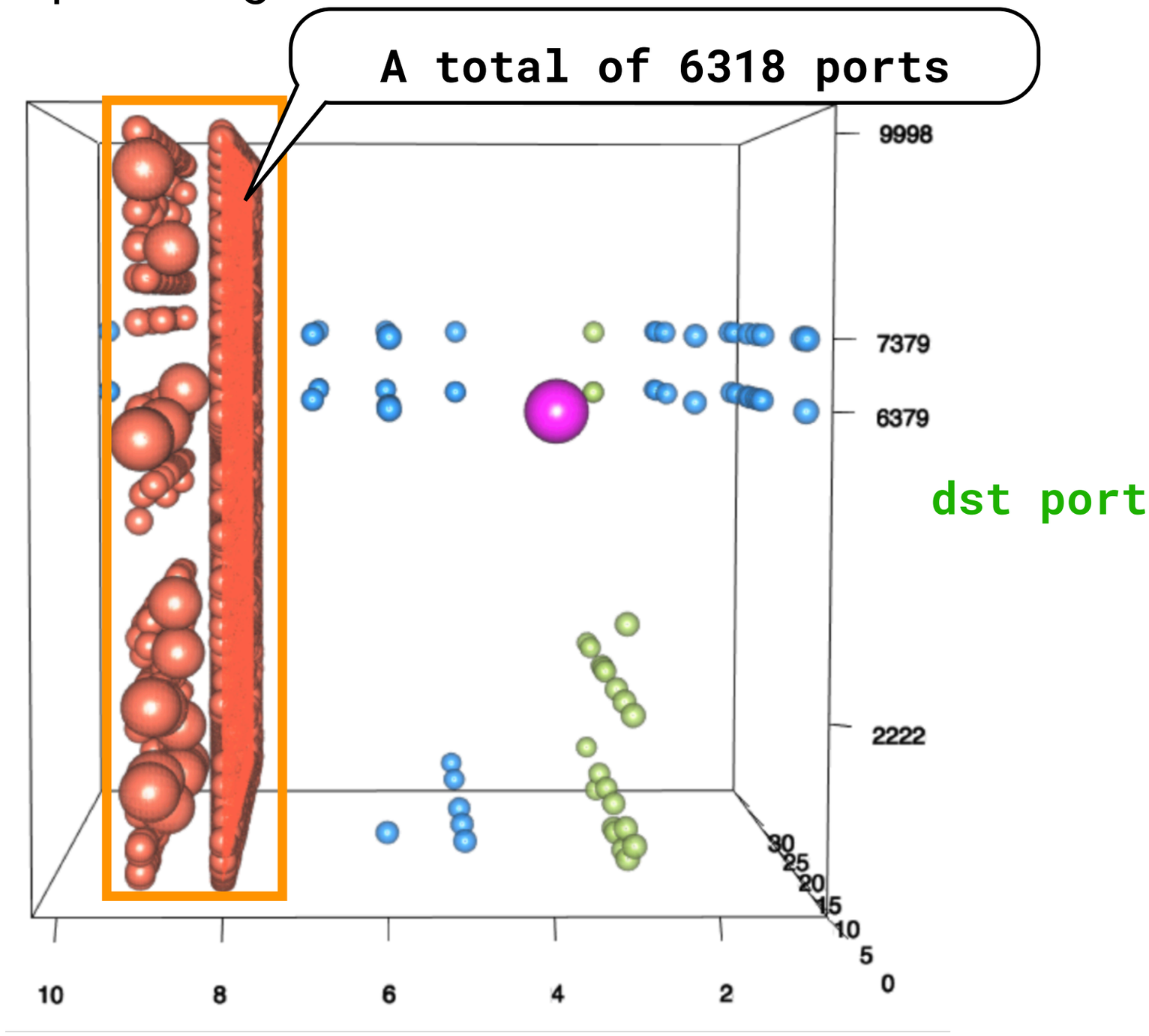
color: difference in the basis vectors of IP characteristics

size: #packets/min

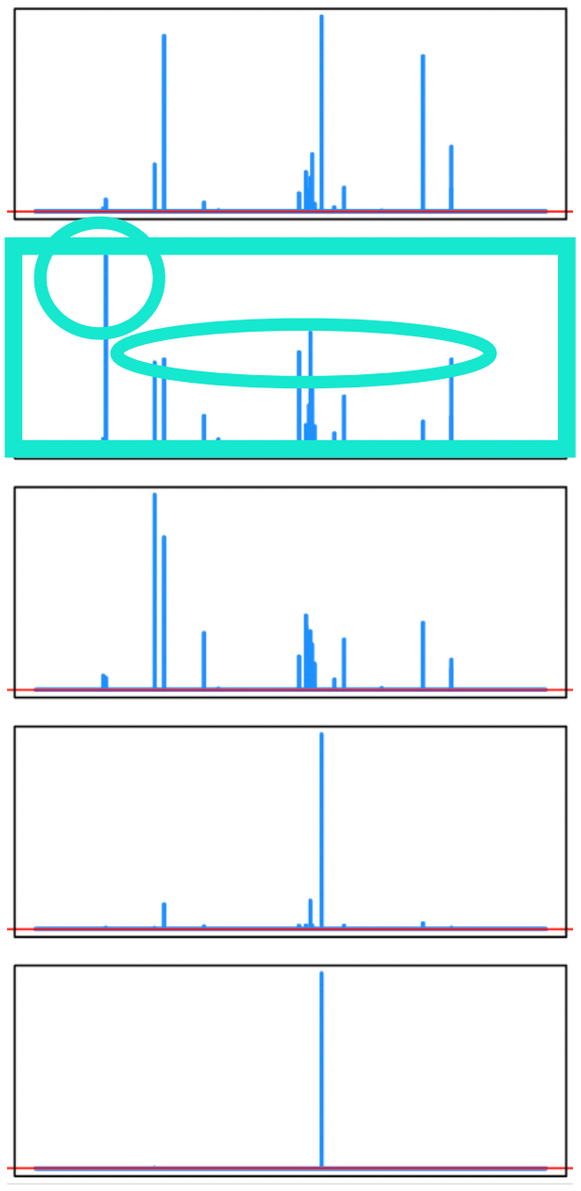


- malicious group A: probing attack



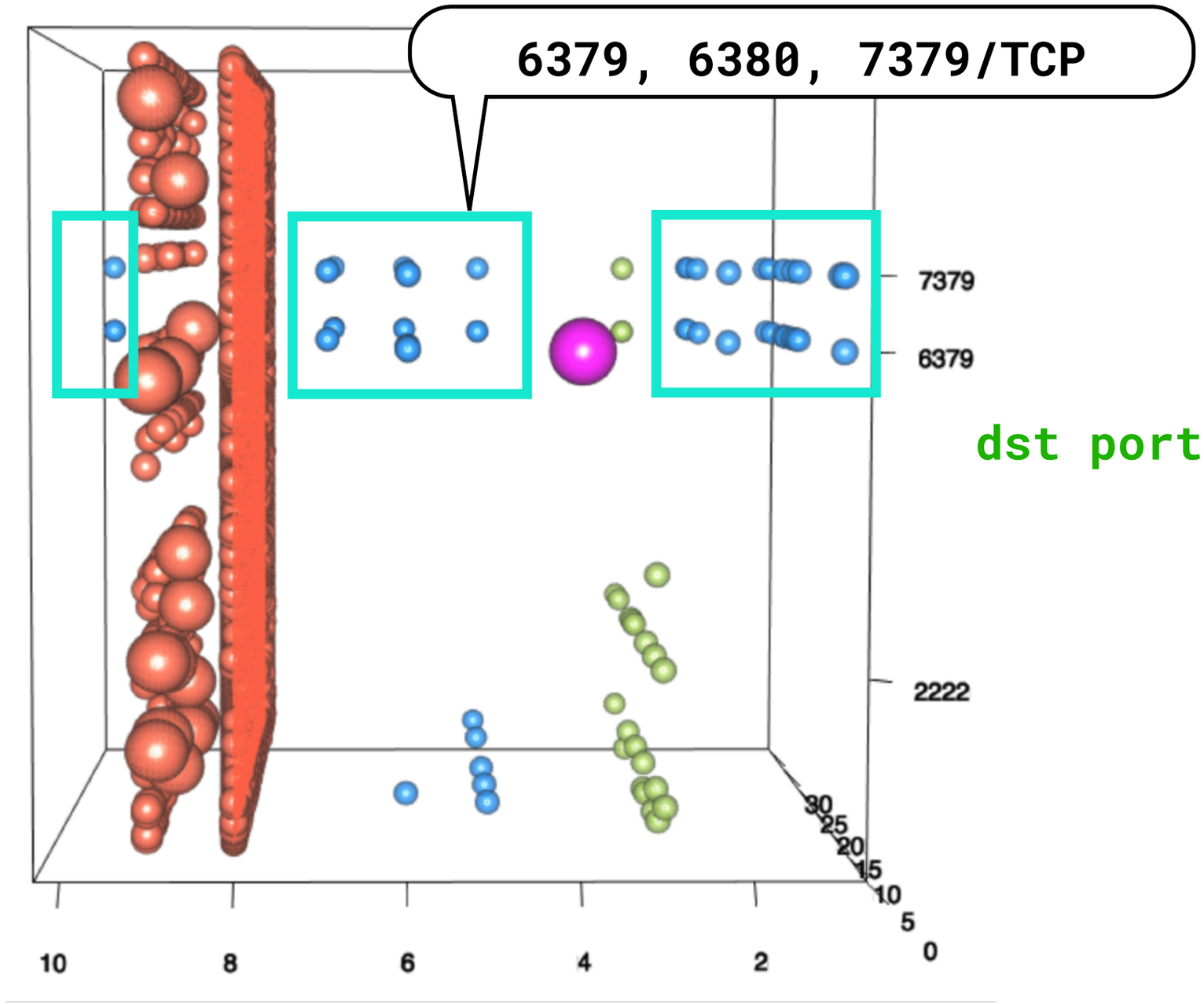


- malicious group B: exploiting some vulnerability?

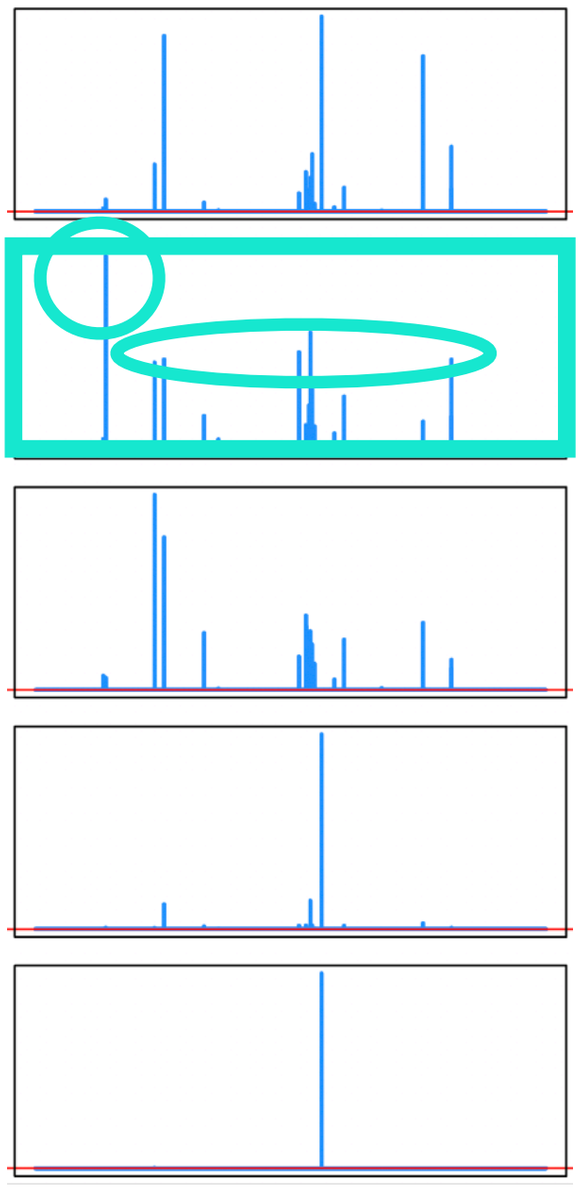


255.255.*.* 0.0.*.*

IP charasterics

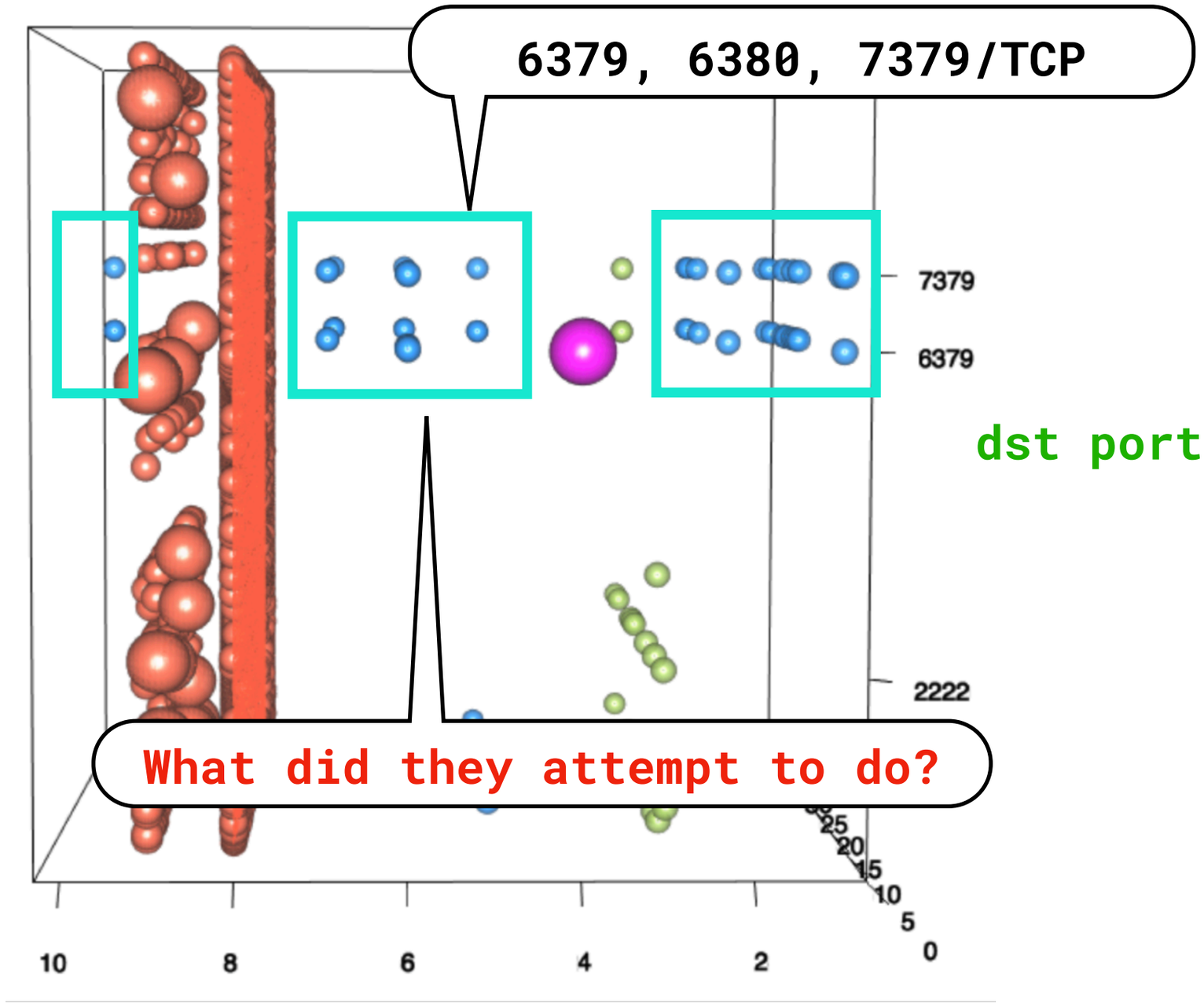


- malicious group B: exploiting some vulnerability?



255.255.*.* 0.0.*.*

IP charasterics



Result | Related incidents

- On March 8, the research blog announced the malware that abuse known vulnerabilities of *Redis* server (listens on the port 6379 by default)
  - try to find vulnerable Redis servers by Internet-wide scanning

Redis scan and infection

The script then launches another process named "*redisscan.sh*". The new process uses the *masscan* tool mentioned above to discover and infect publicly available Redis servers. It does so by creating a large list of IPs, **internal** and **external** and scanning port 6379 which is the default listening port of Redis.

```
#!/bin/bash
python rangeip.py
while read line
do
    masscan -p6379 $line --rate=20000 | tee -a masscan
    python order.py
    sh redisrun.sh
done < ip
```

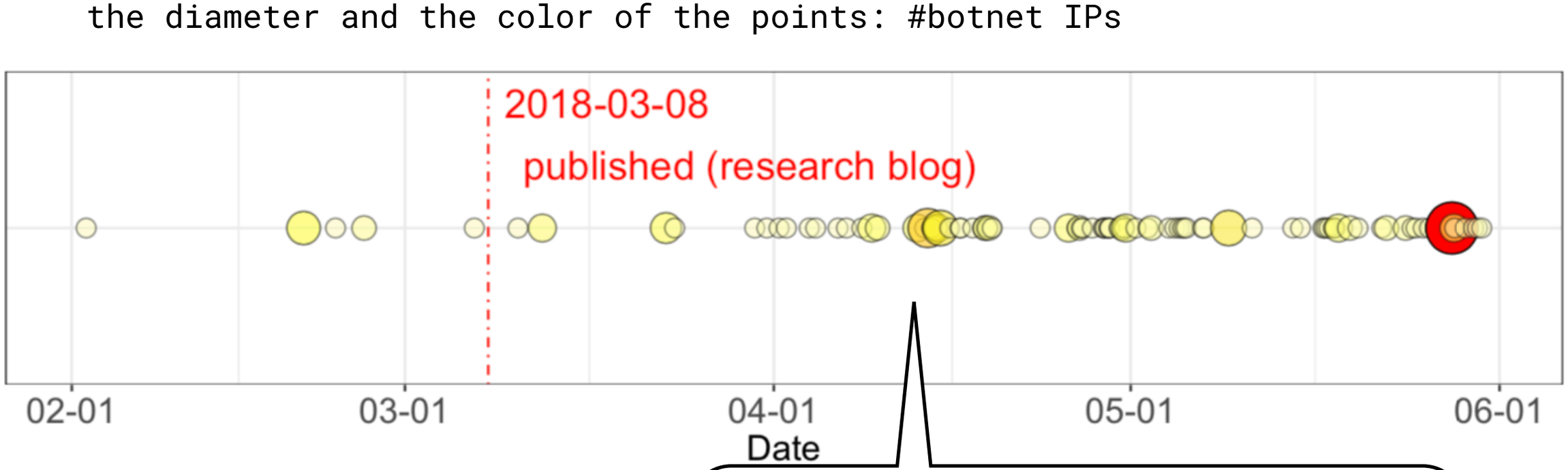
https://www.imperva.com/blog/2018/03/rediswannamine-new-redis-nsa-powered-cryptojacking-attack/

Result | Related incidents

- On March 8, the research blog announced the malware that abuse known vulnerabilities of Redis server (listens on the port 6379 by default)
  - try to find vulnerable Redis servers by Internet-wide scanning

summarized alerts on 6379/TCP

the diameter and the color of the points: #botnet IPs



Conclusion

- We proposed a novel botnet detection method from *darknet* traffic
  - Nonnegative Tucker decomposition (NTD):
    a powerful model for extracting co-occurrence patterns
    -> but requires too high computational cost
- Efficient NTD implementation enough to run in real-time
  - LRA-NTD
  - FSTD
- Demonstrated effectiveness by reviewing incidents

Future work

- Improve the NTD algorithm
  - Our approach is very fast, but loses much information
- Quantitative evaluation

APPENDIX