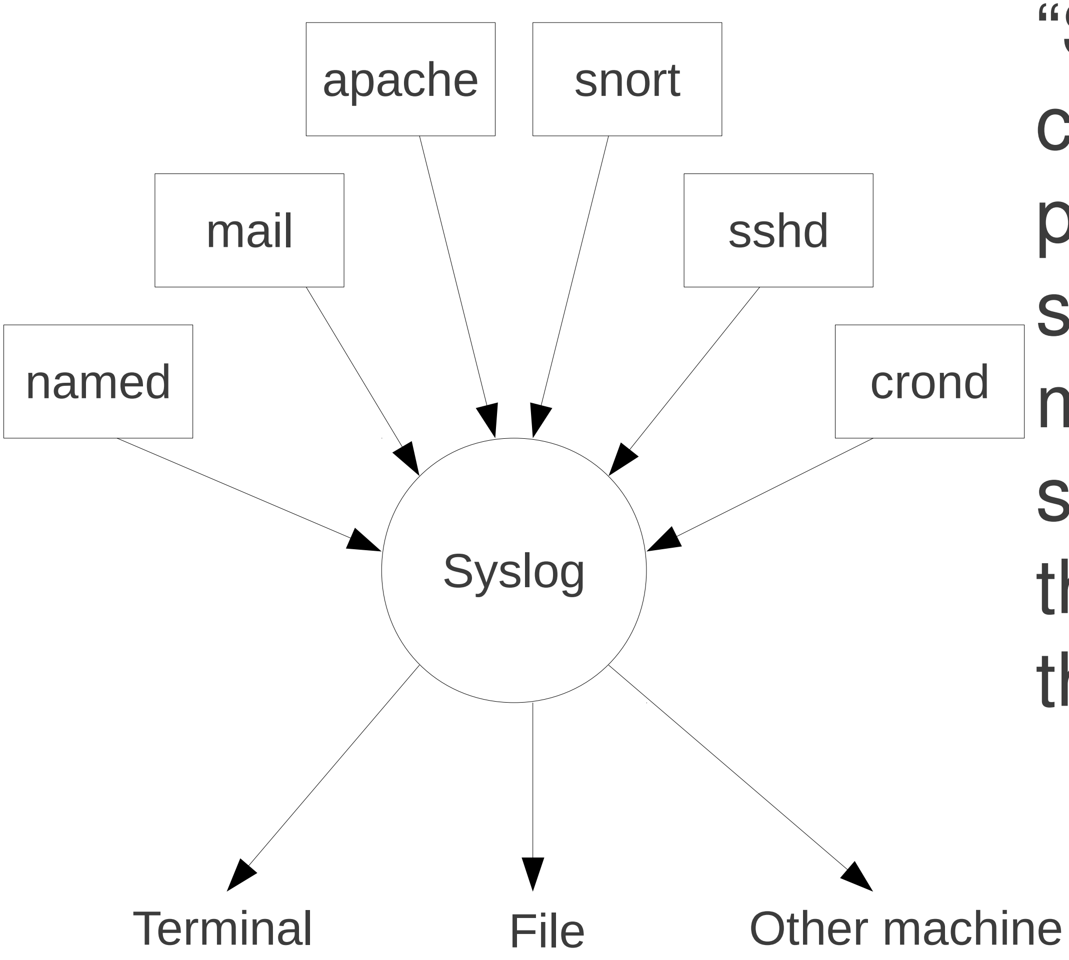
Presentation 1: Syslog-->Lttng
Presentation 2: Event Linking data structure

Naser Ezzati
Yannick Brosseau
Michel Dagenais

Department of Computer and Software Engineering

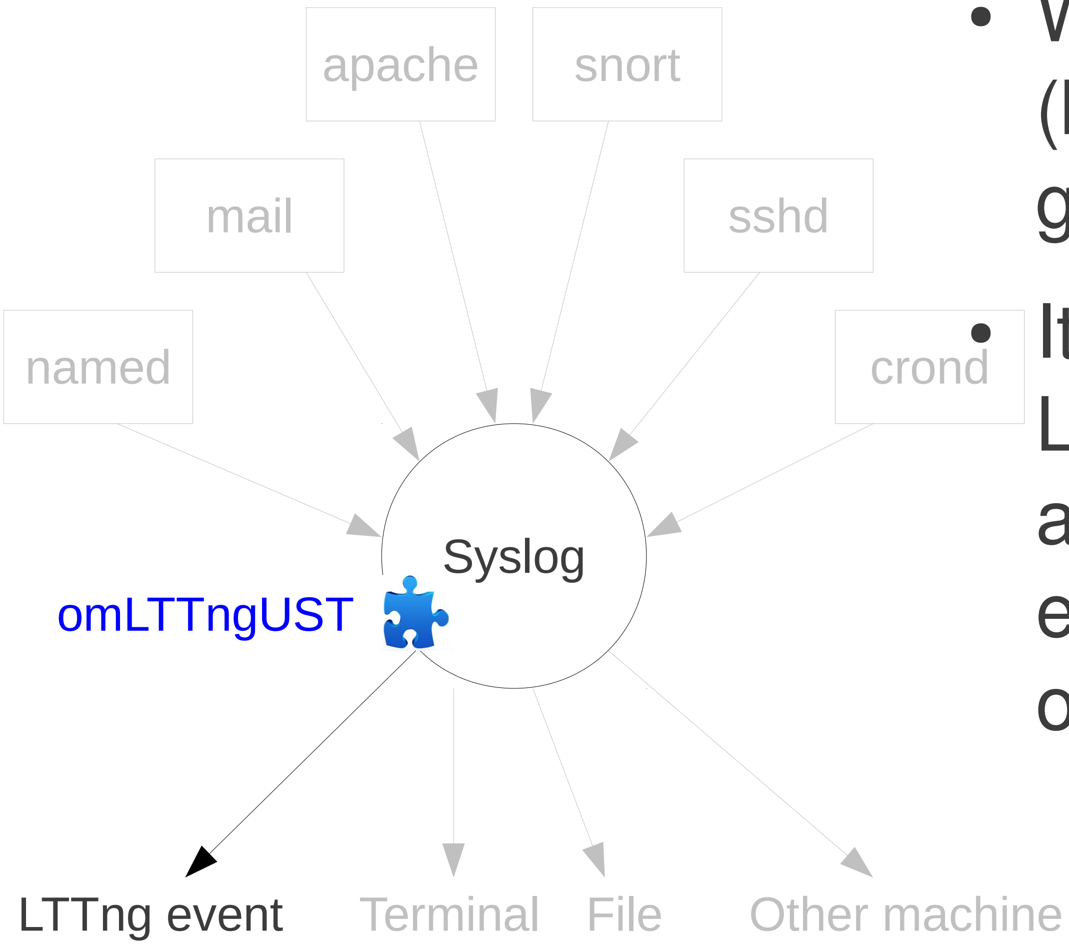
Dec 10, 2013
École Polytechnique, Montreal

Syslog



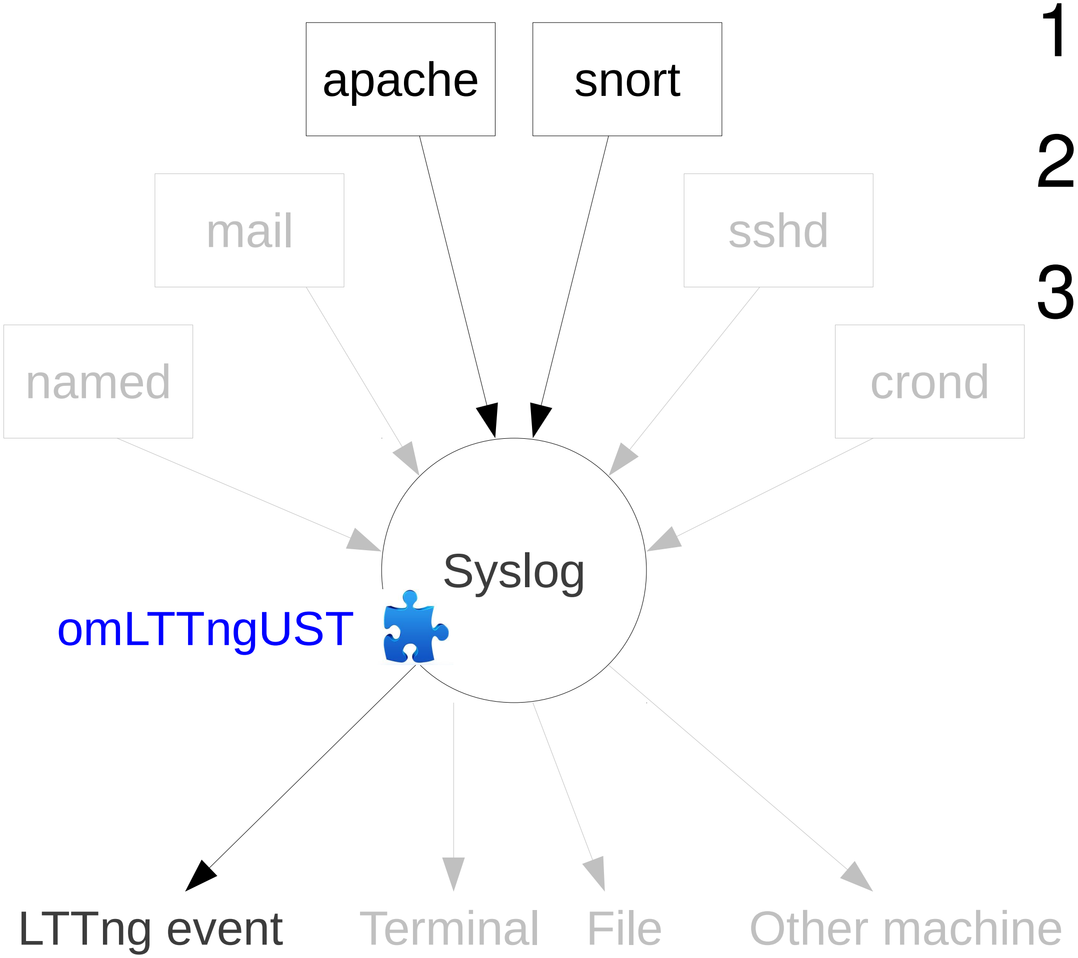
"Syslog is a standard for computer message logging. It permits separation of the software that generates messages from the system that stores them and the software that reports and analyzes them."

Syslog → Lttng



- We hooked Syslog Daemon (by adding 3 tracepoints) to generate LTTng UST events.
- It makes possible to gather LTTng trace events from any application generating syslog entries, without modifying the original application.

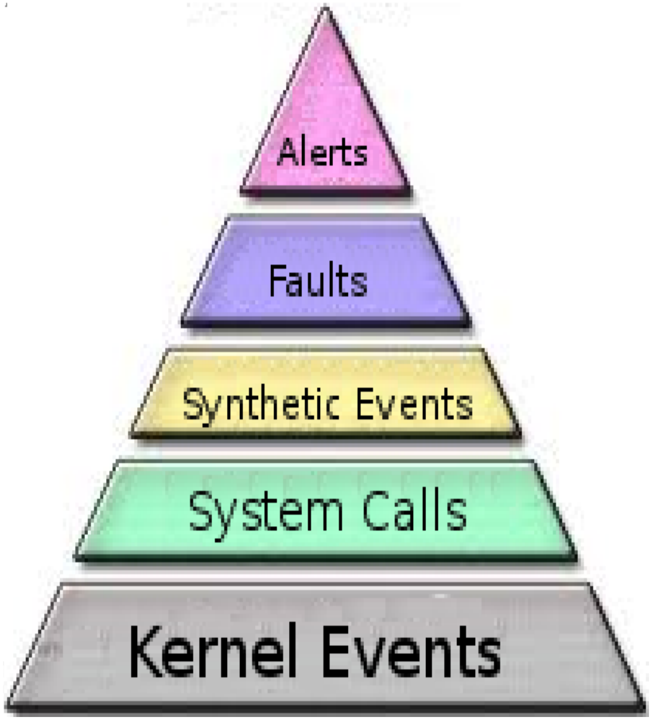
Demo



1. Syslog → LTTng
2. Snort → Syslog → LTTng
3. PHP → Syslog → LTTng

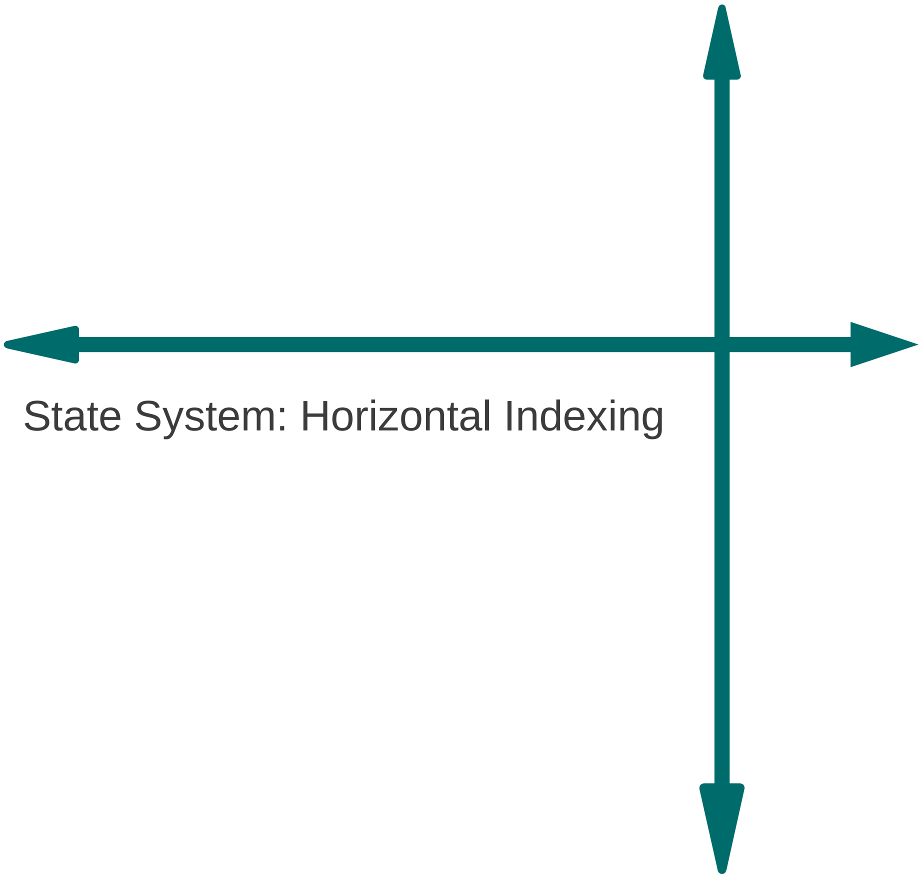
Event Linking data structure

Multi level information



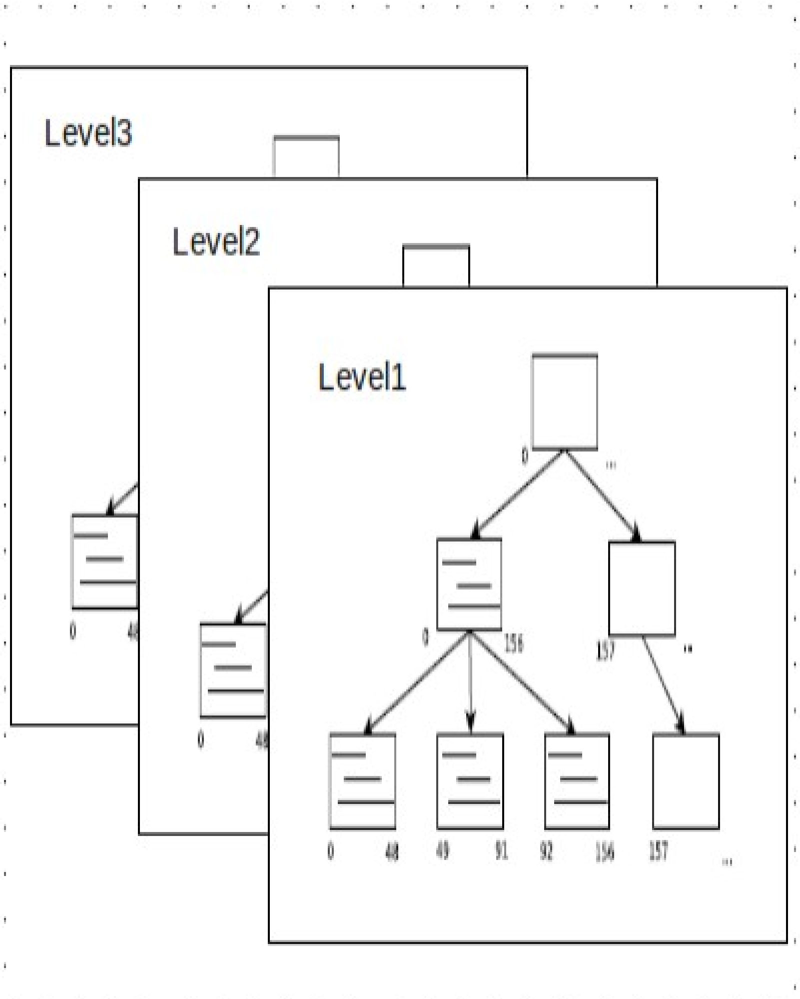
- We may have different layers of trace information
- Linking the different layers enables a multi-resolution analysis of the system under study.
- In this presentation, we discuss about the data structure.
    - And some real use-cases.

Integration with the State System



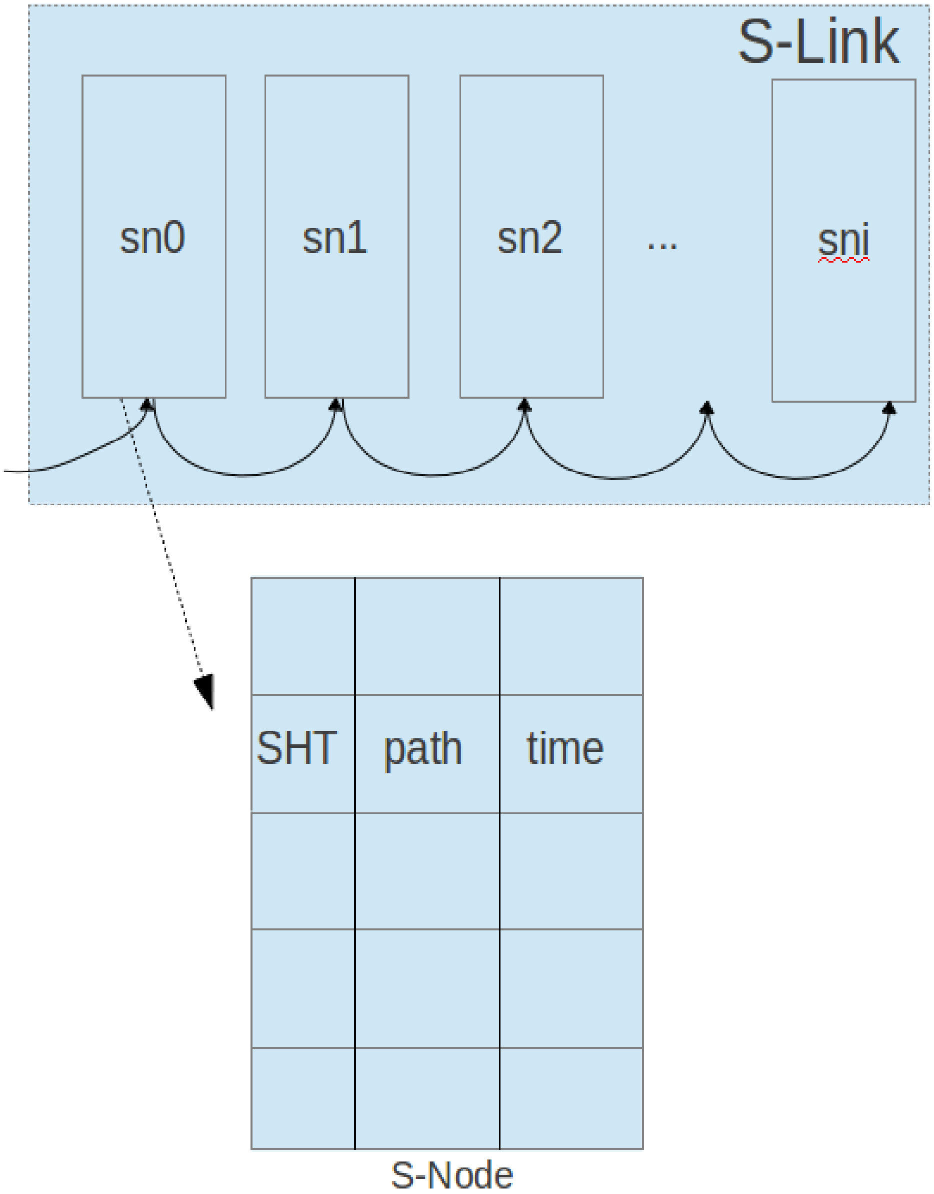
- State system is a more horizontal indexing structure.
  - It indexes/links the same-level states of an attribute.
- However, we are looking for a vertical indexing system.
  - To link the information from different layers.
- But the integration was a MUST.

S-Link data structure



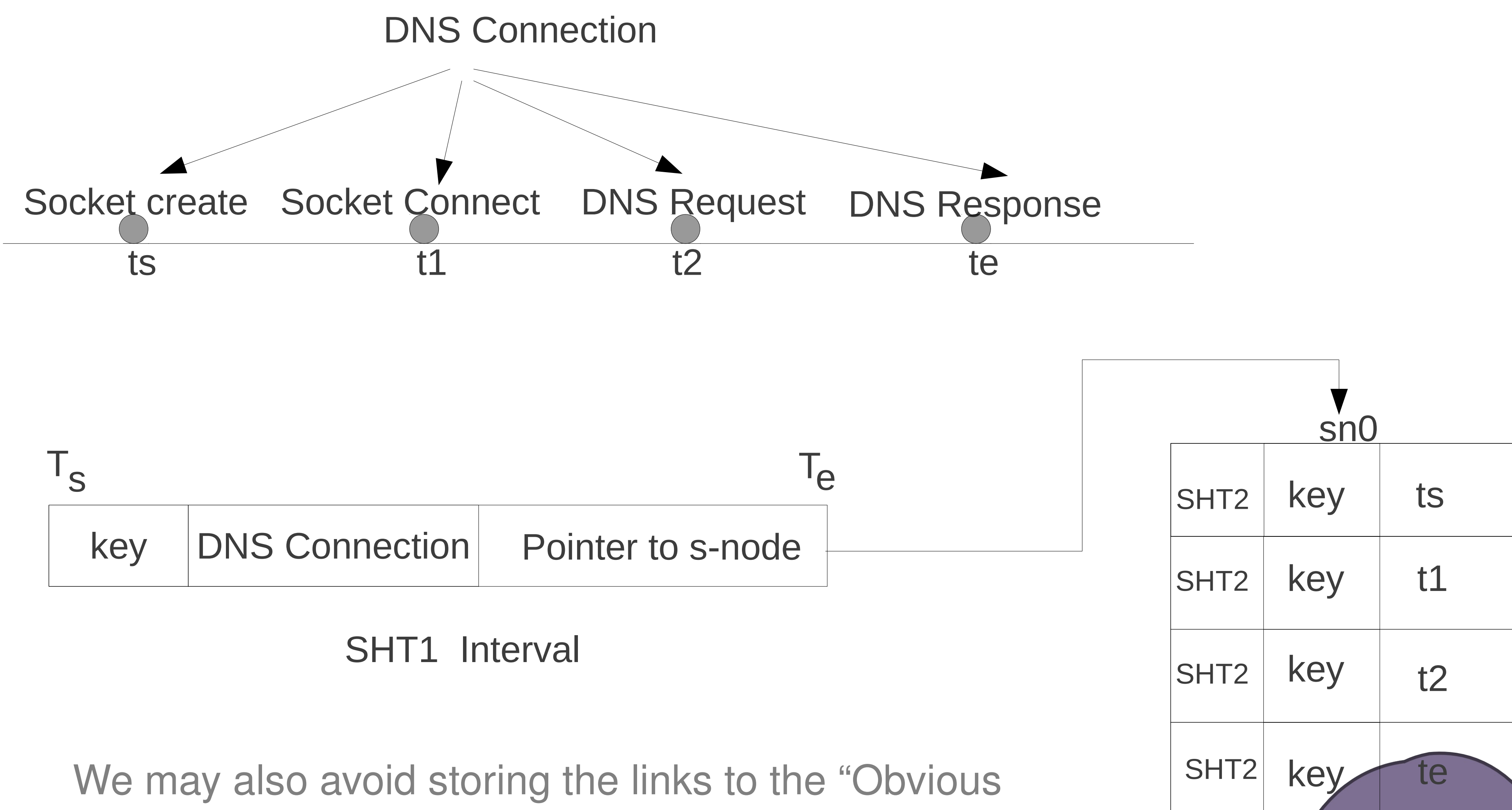
- First approach:
  - State system is used to store the abstract events
    - Each state system stores the events (intervals) of only one layer.
  - For each event (interval) we keep a pointer to structure named S-Link.

S-Link and S-node



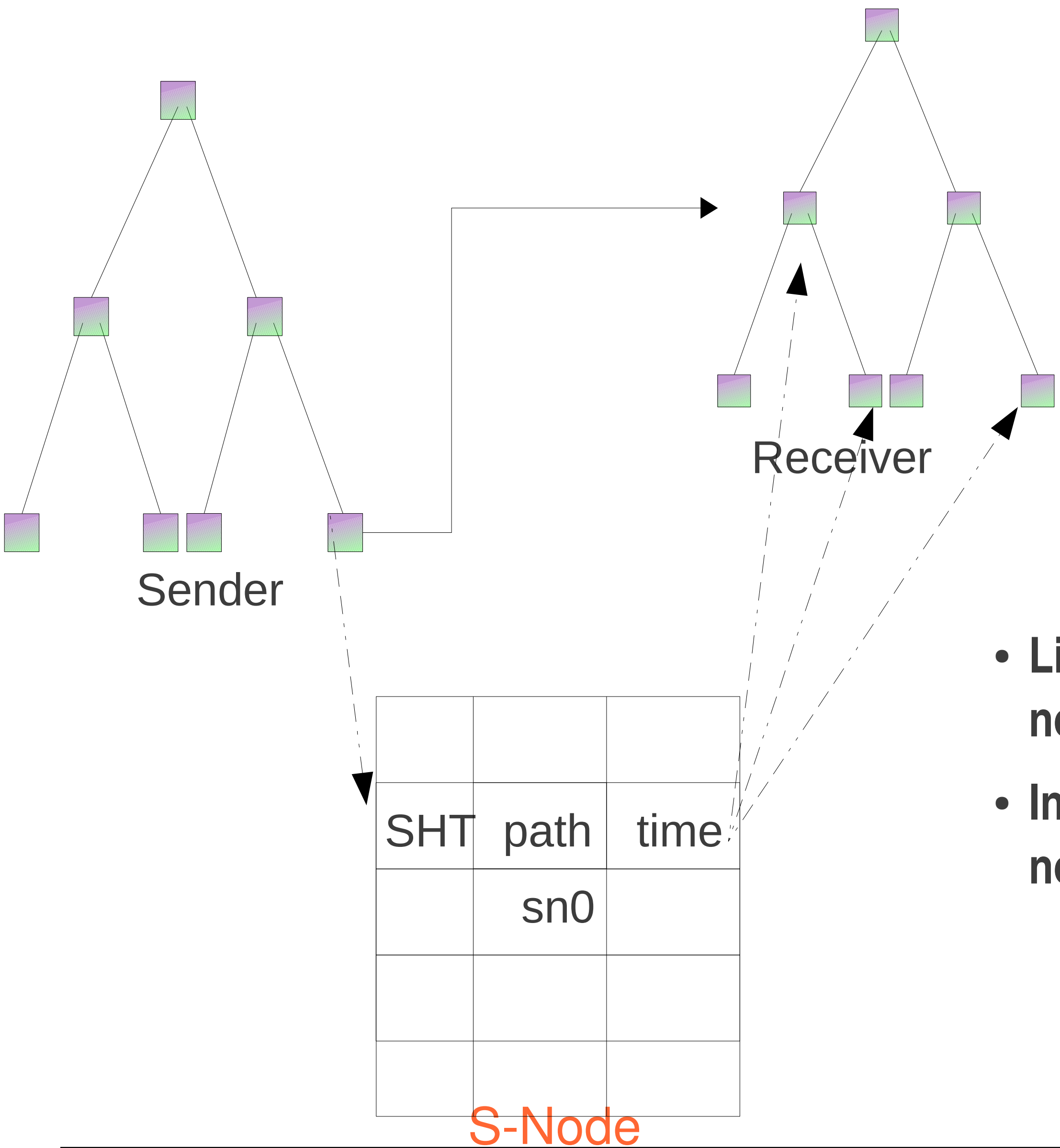
- S-link is an array of the same size s-nodes
- Each s-node has k entries.
- The k+1 th entry is a pointer to another s-node.
  - Using this mechanism any size of linking pointers is supported.

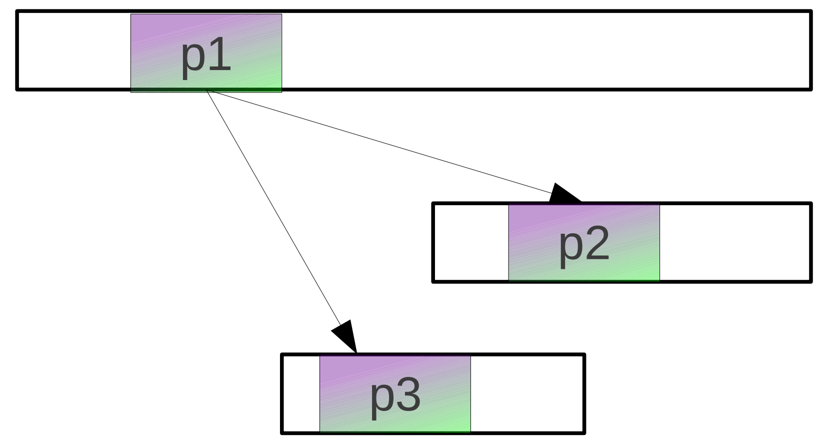
Example 1: Hierarchy of events



We may also avoid storing the links to the “Obvious members”.

Example2: Send & Receive



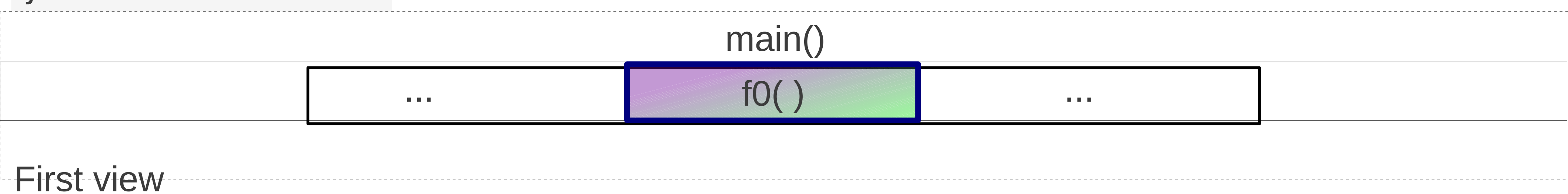


- **Link the corresponding sending and receiving nodes (processes, etc).**
- **In this approach, the corresponding receiving node(s) will be stored in the s-link structure.**

Example3: UST and Kernel Traces

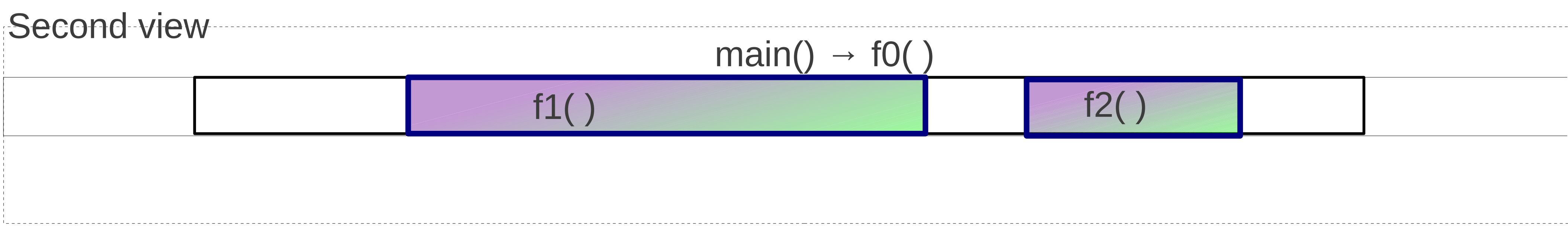
```
void main() {
  ...
  f0 ( );
  ...
}
void f0() {
  ...
  f1 ( );
   ...
  f2 ( );
  ...
}
```

Function calls and kernel traces



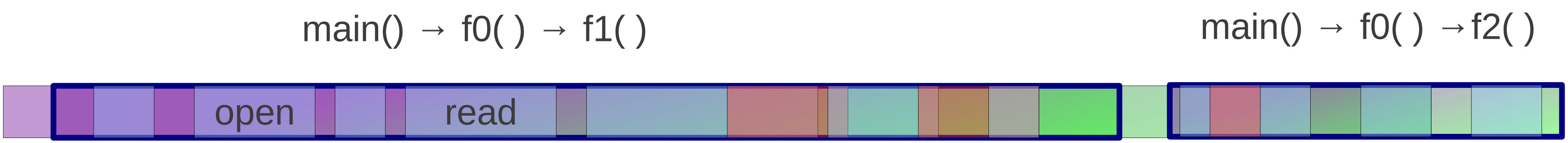
UST and Kernel Traces

```
Void f0() {
  ...
  f1 ( );
   ...
  f2 ( );
  ...
}
```



UST and Kernel Traces

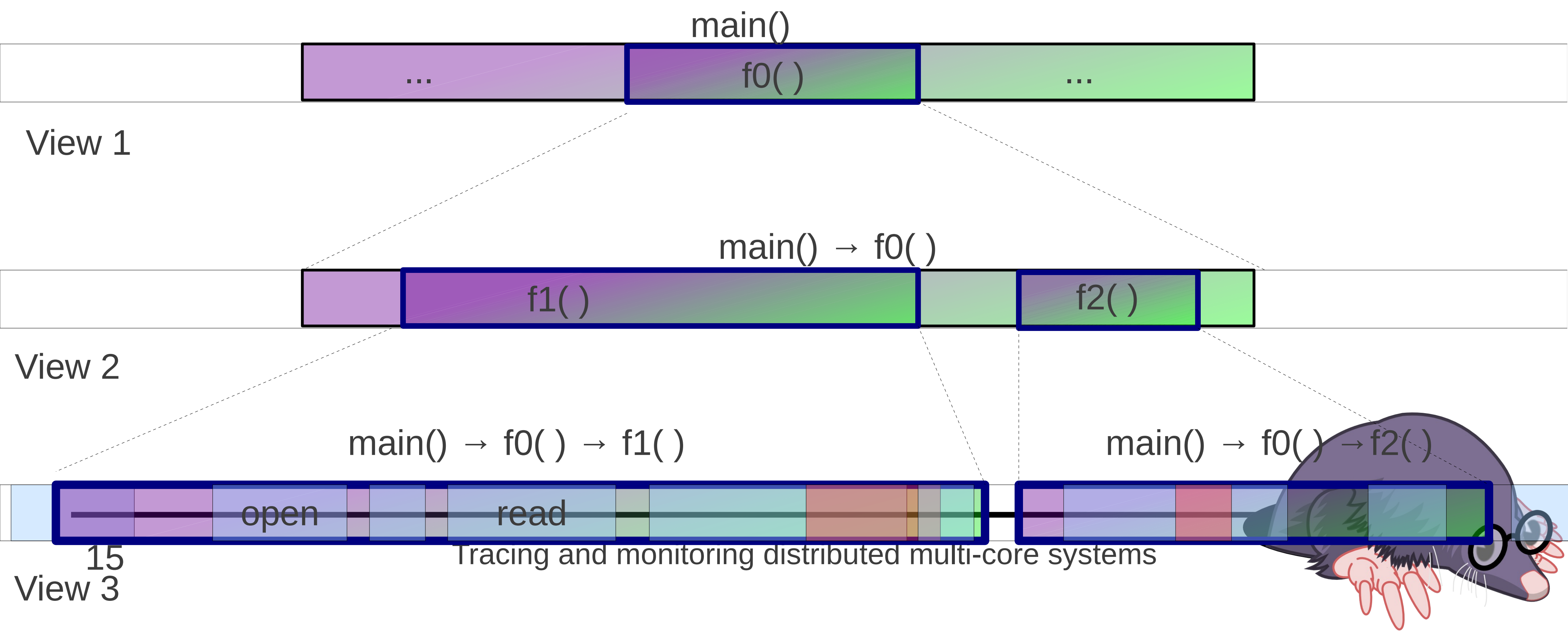
```
Void f0() {
  ...
  f1 ( );
   ...
  f2 ( );
  ...
}
```



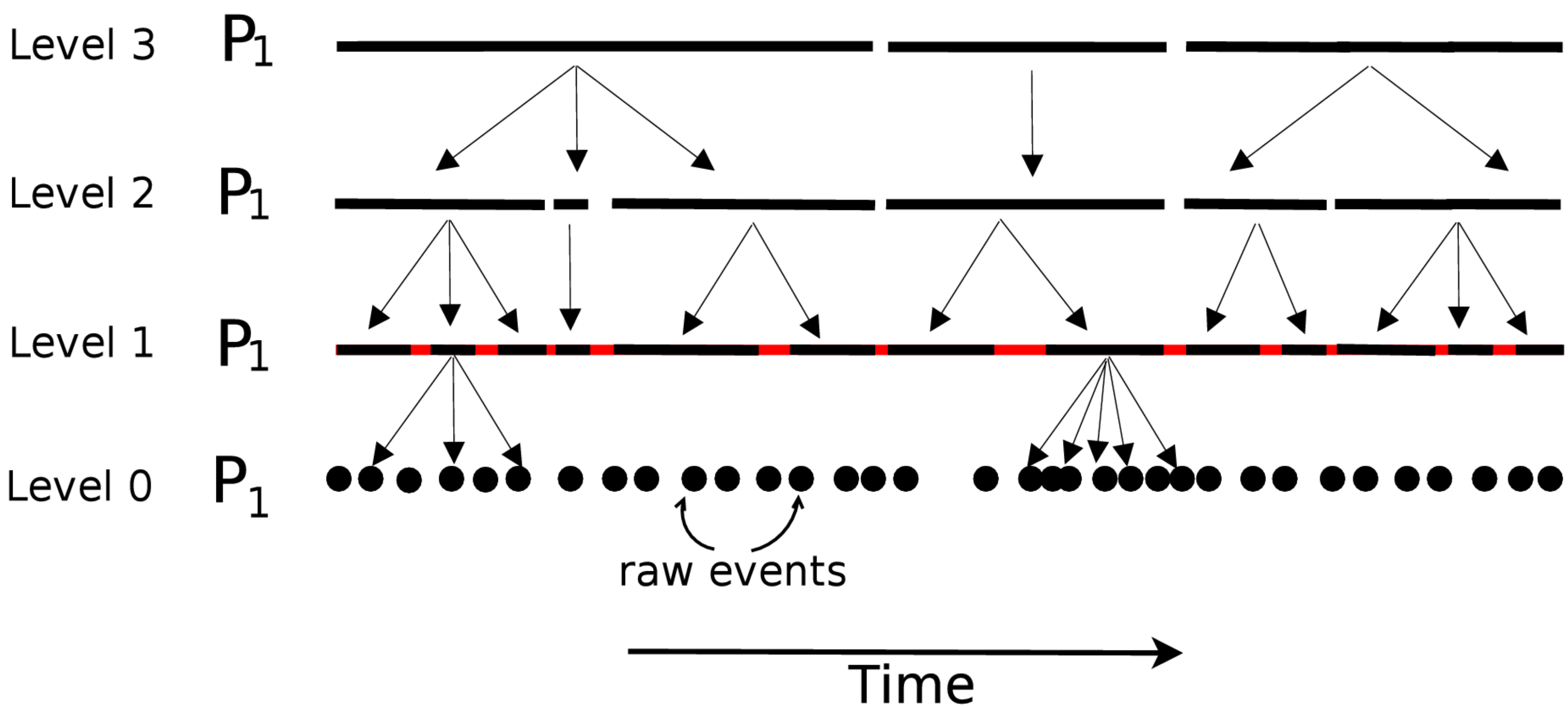
UST and Kernel Traces

```
Void f0() {
  ...
  f1 ( );
   ...
  f2 ( );
  ...
}
```

- One SHT for the each level
- Query the kernel SHT, retrieve the corresponding system calls and show them with the functions together.



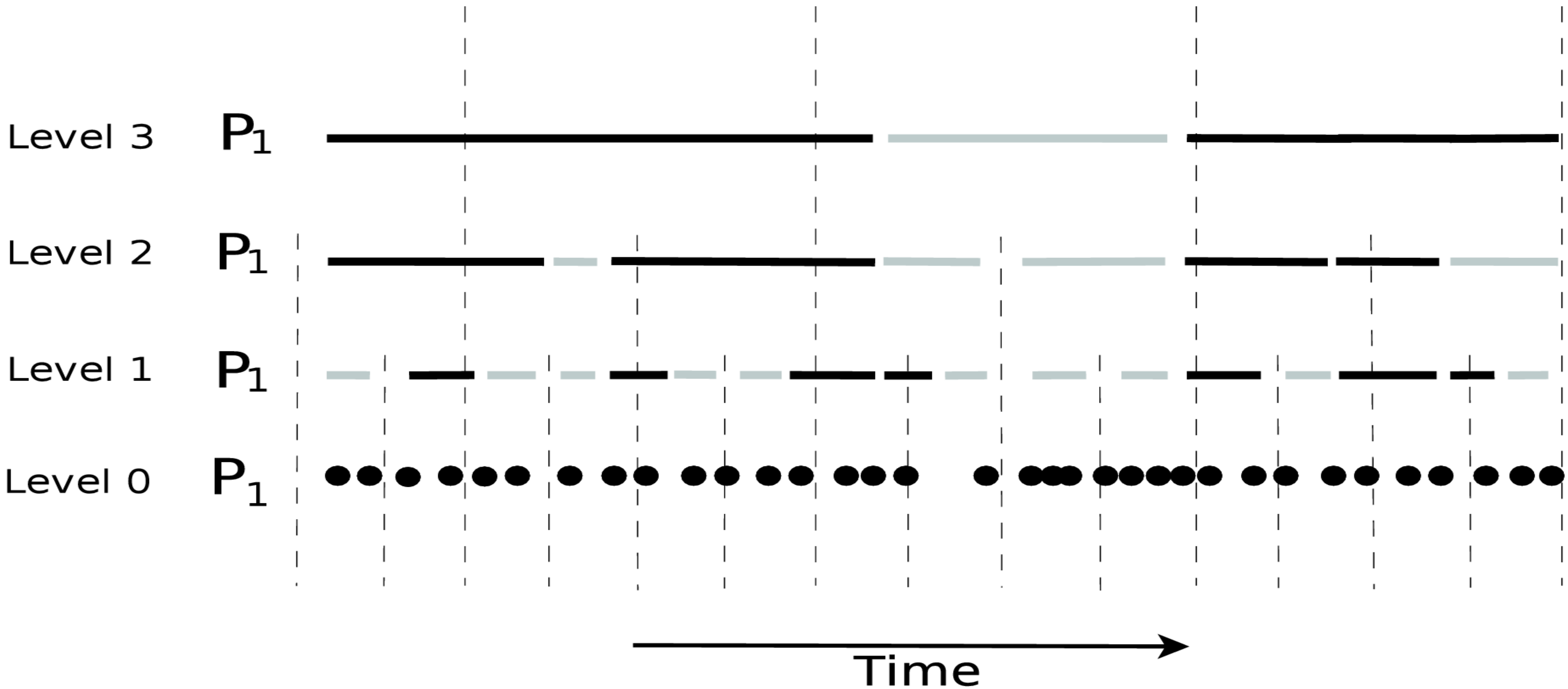
First Approach Overview



The first approach stores everything

- the events and the links.

Second Approach (Partial)



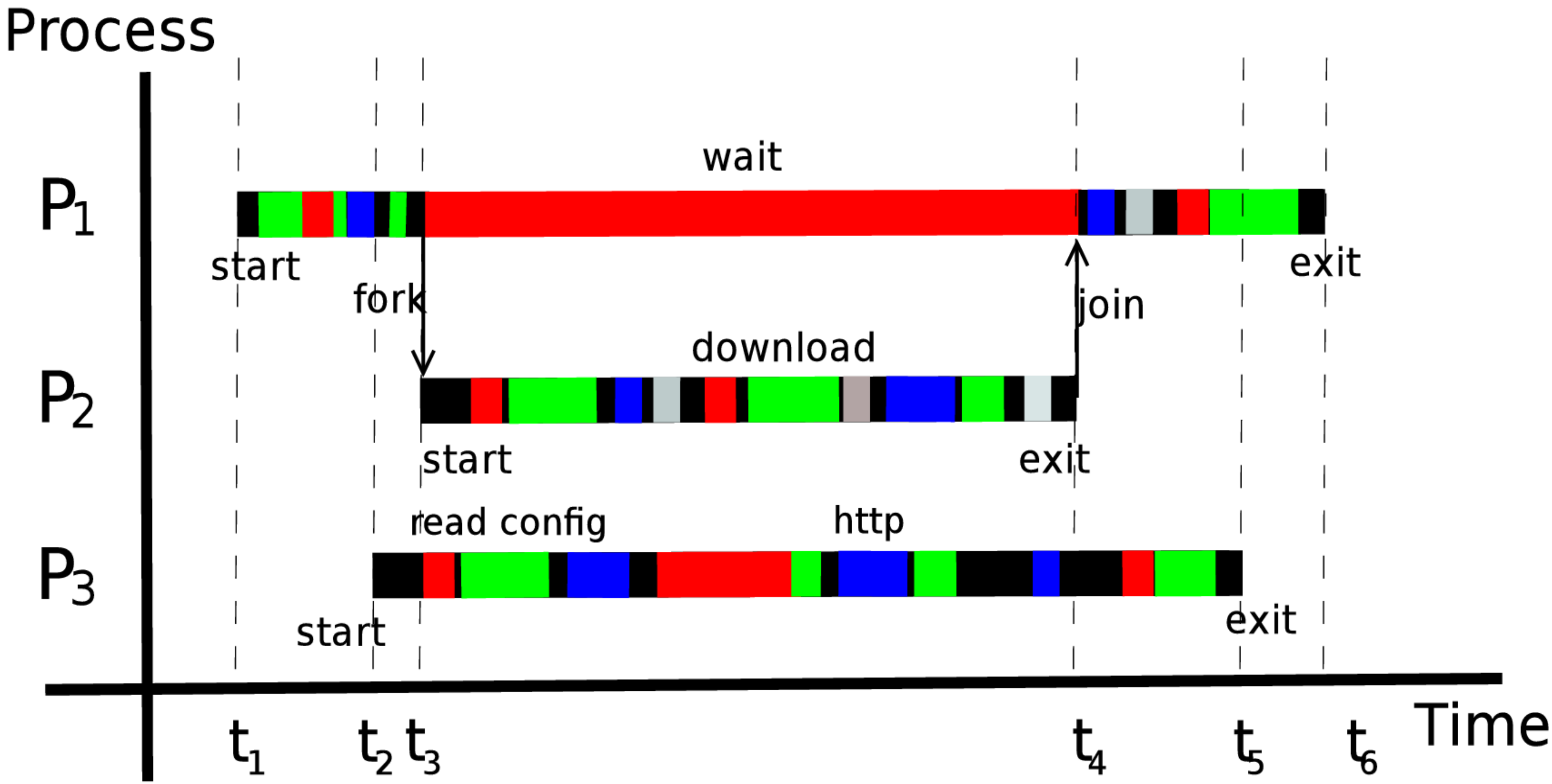
The second approach avoids storing everything!

- It only stores the <u>important</u> events.

Second Approach

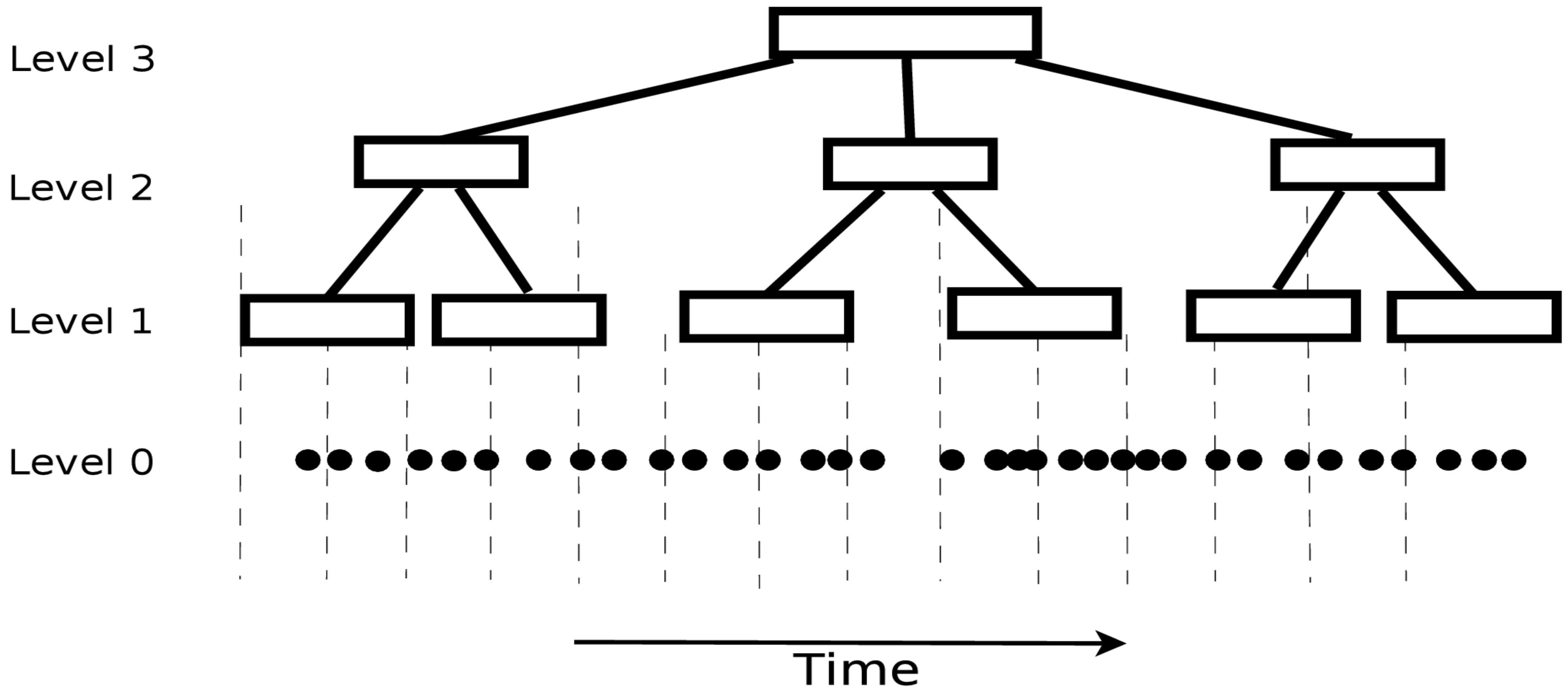
- The first approach stores all events (of the all levels) and also the links between them.
- Second approach, however, avoids storing all events and tries to re-generate them dynamically, on the fly, in the visualization phase.
  - It stores the events of the highest level completely, but for the intermediate levels, it only stores some (but enough) snapshots.
  - It supports two abstraction types: Data Abstraction (events) and Visual Abstraction (labels and colors)
    - It uses both the labels and colors to encode the view
  - It shows all levels in a single view using a zoom-able time-line view.

Highest Level (colors + labels)



Colors show the type of operations (File, Network, System, Wait, etc)

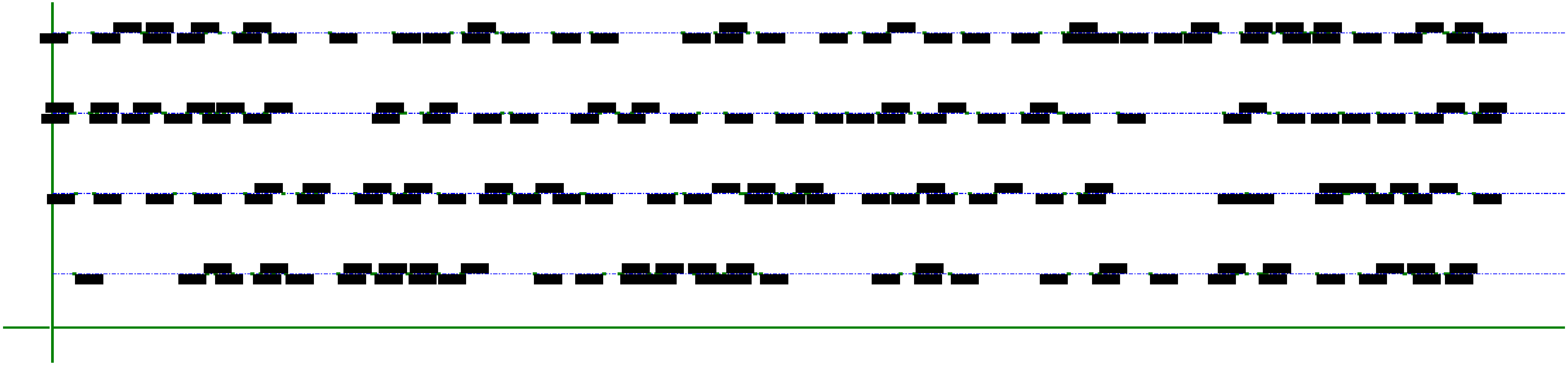
Structure



Each tree node stores one or more snapshots

Improvements

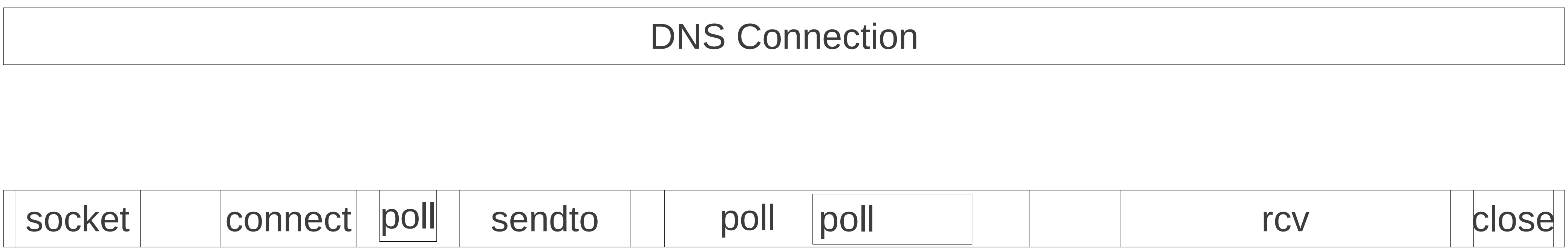
- The duration of a snapshot is set dynamically, based on the processing time required to generate the abstract events.
- At each snapshot, the important events from the previous checkpoint to the current one are stored.
- Dynamic trace abstraction is used to re-generate the events
- At each level, label placement techniques are used to show the best set of events (labels).
  - Priority rendering, dynamic aggregation, remove repetitive items, … .



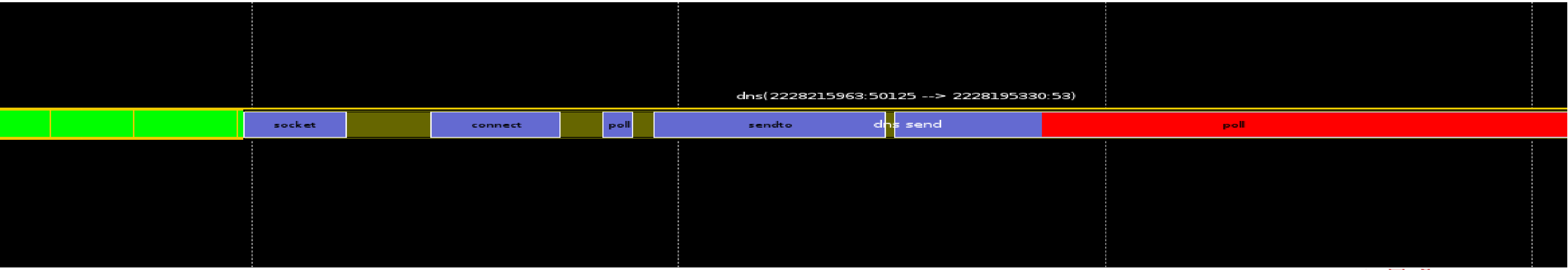
Algorithm To Fetch

- Traverse the tree to find the snapshots of the given query range.
  - output a list of labels of each level.
- Apply label placement techniques (remove duplications, perform dynamic aggregation, apply priority, etc.), and show the labels.

Example

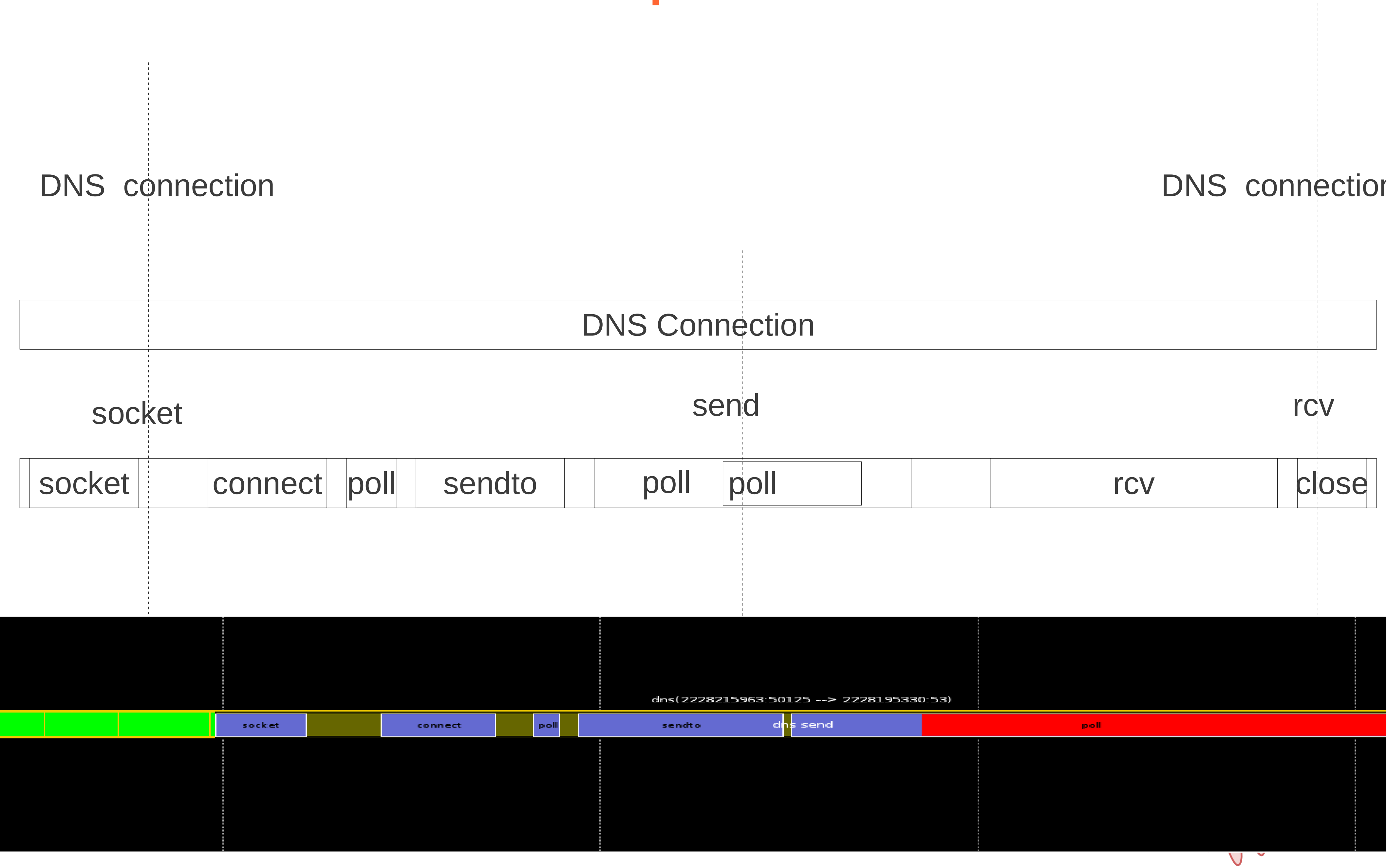


DNS Connection
socket
connect
poll
sendto
poll
poll
rcv
close



dns(2228215963:50125 --> 2228195330:53)
socket
connect
poll
sendto
dns send
poll

Snapshots



DNS connection
DNS connection
DNS Connection
socket
send
rcv
socket
connect
poll
sendto
poll
poll
rcv
close
dns(2228215963:50125 --> 2228195330:53)
socket
connect
poll
sendto
dns send
poll

Snapshots

DNS connectio

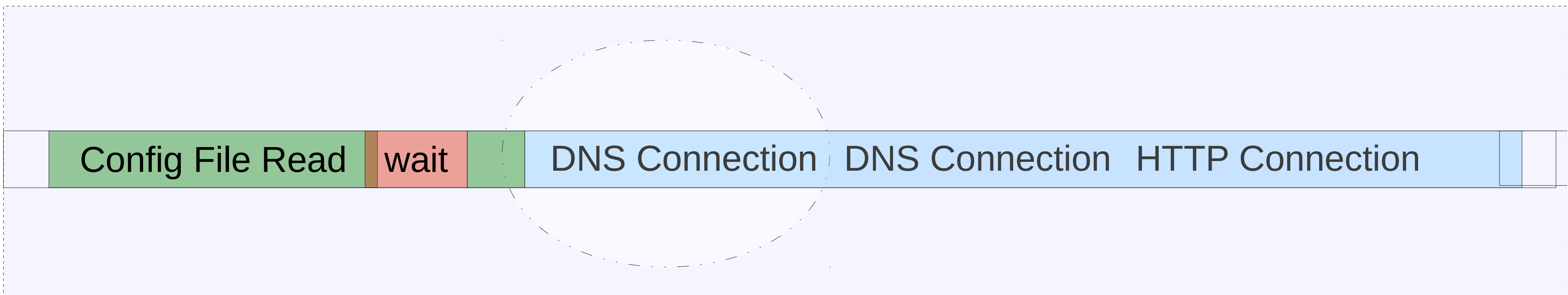
socket

send

rcv

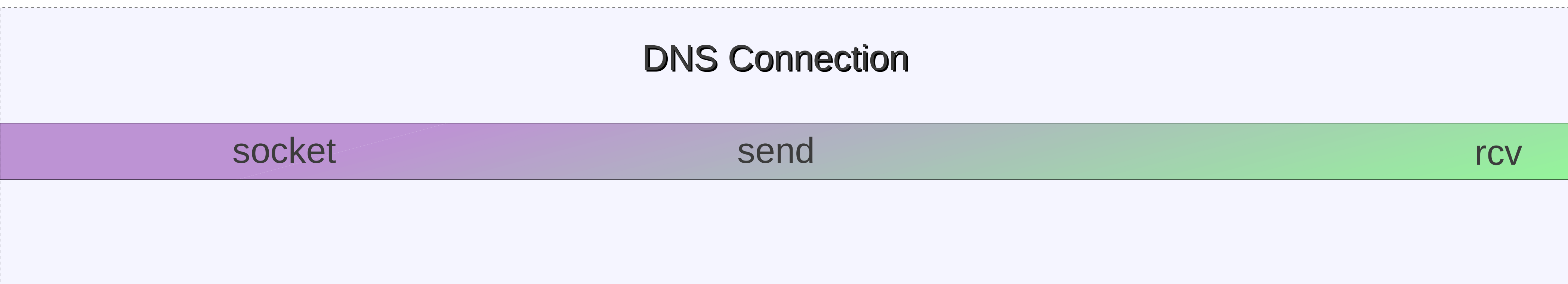
The View (Level 1)

generate this view from the snapshot structure



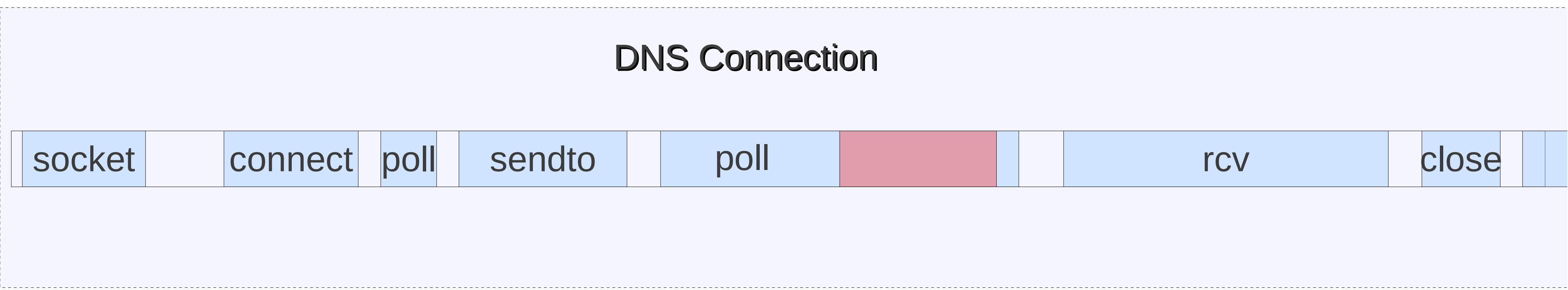
The View (Level 2)

generate this view from the snapshot structure



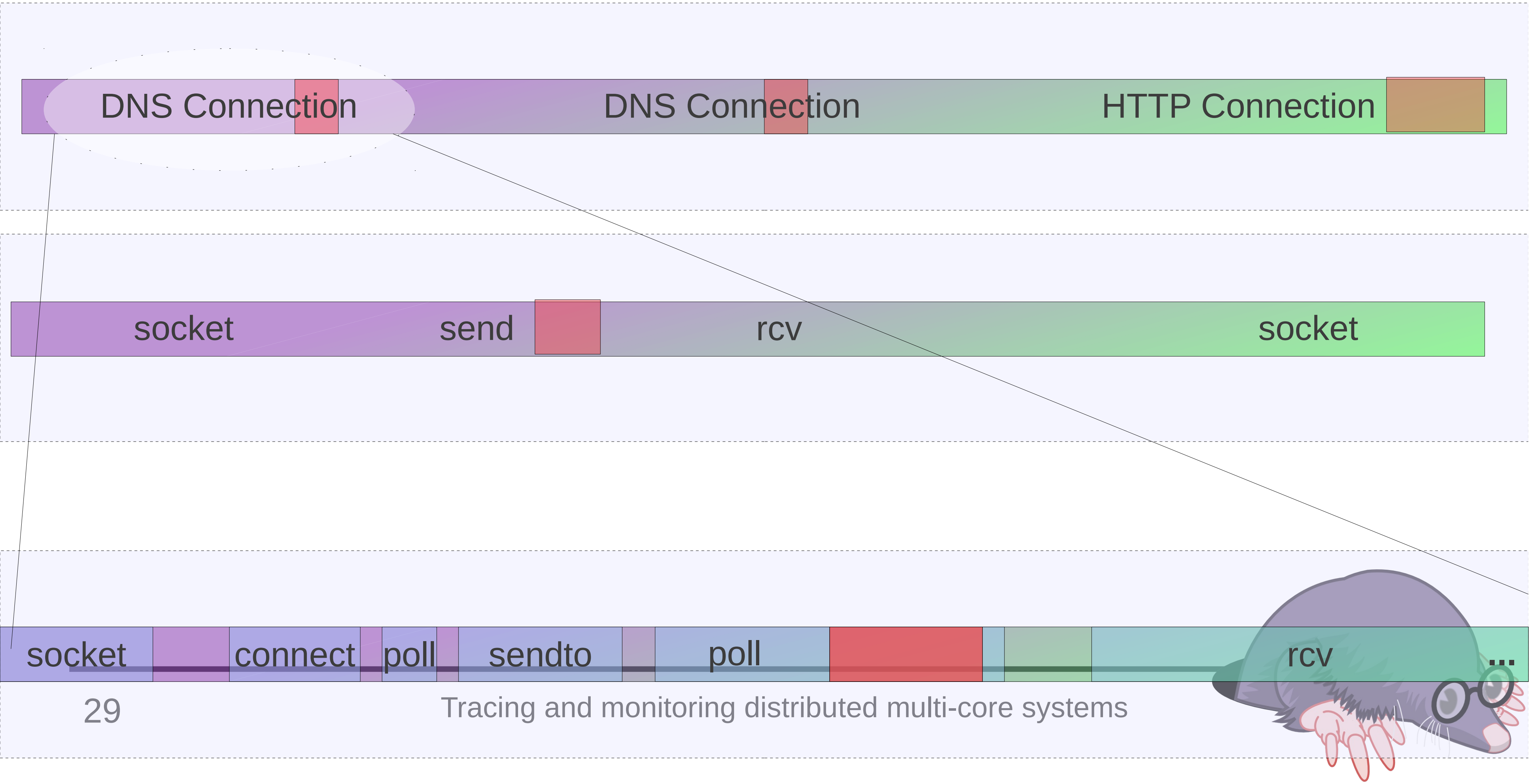
The View (Level 3)

generate this view from trace directly

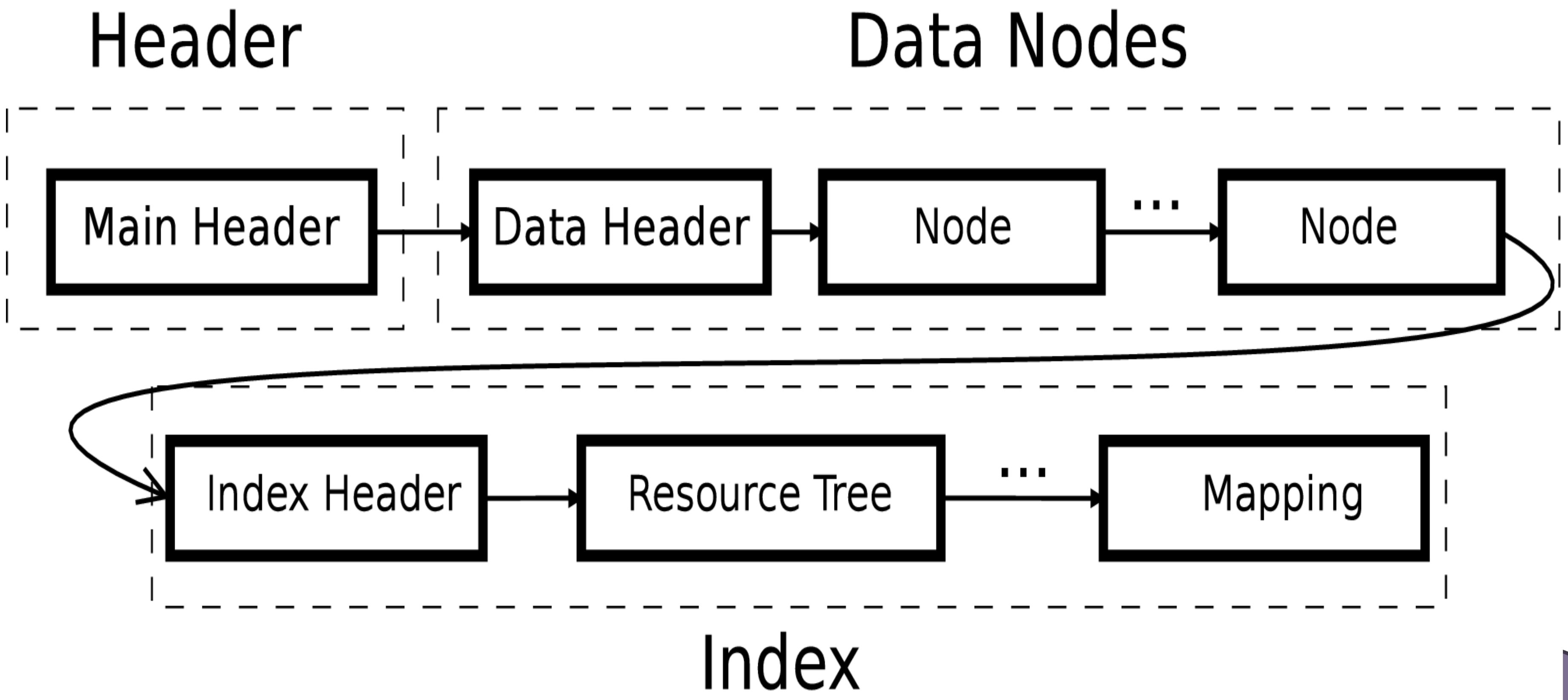


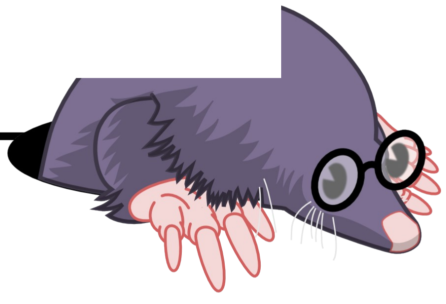
The View

generate these two views from the snapshot structure



Disk File Format





Summary

- A syslog plugin called omLTTngUST is presented.
- Linking data structure called S-Link data structure and some real use-cases are presented.
- Two approaches:
  - As a plugin to the State System:
    - that stores events and the links between them
  - A solution that does not store all events, and generates the abstract events dynamically in the visualization step, using some pre-stored important events.

Thank you.

Questions?

Demo