1 million concurrent calls and 20k Calls per second with OpenSIPS

Vlad Paiu

OpenSIPS Project Developer

OpenSIPS Solutions

Scalability Perspectives

- Simple Logic
- Less I/O
- High Traffic
- SBCs, Trunking, Wholesale, Load-Balancers

Scalability Perspectives

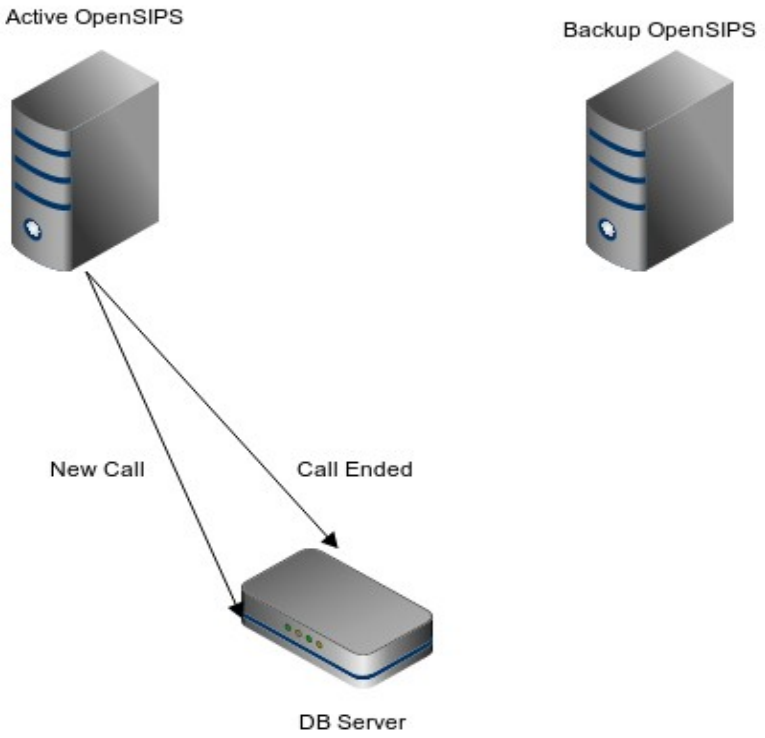
- **Concurrent Calls**
  - Memory Intensive
- **Calls Per Second**
  - CPU Intensive

- Dialog Module
  - CDR Accounting
  - Call Limiting
  - Dialog Profiling
  - Call Statistics

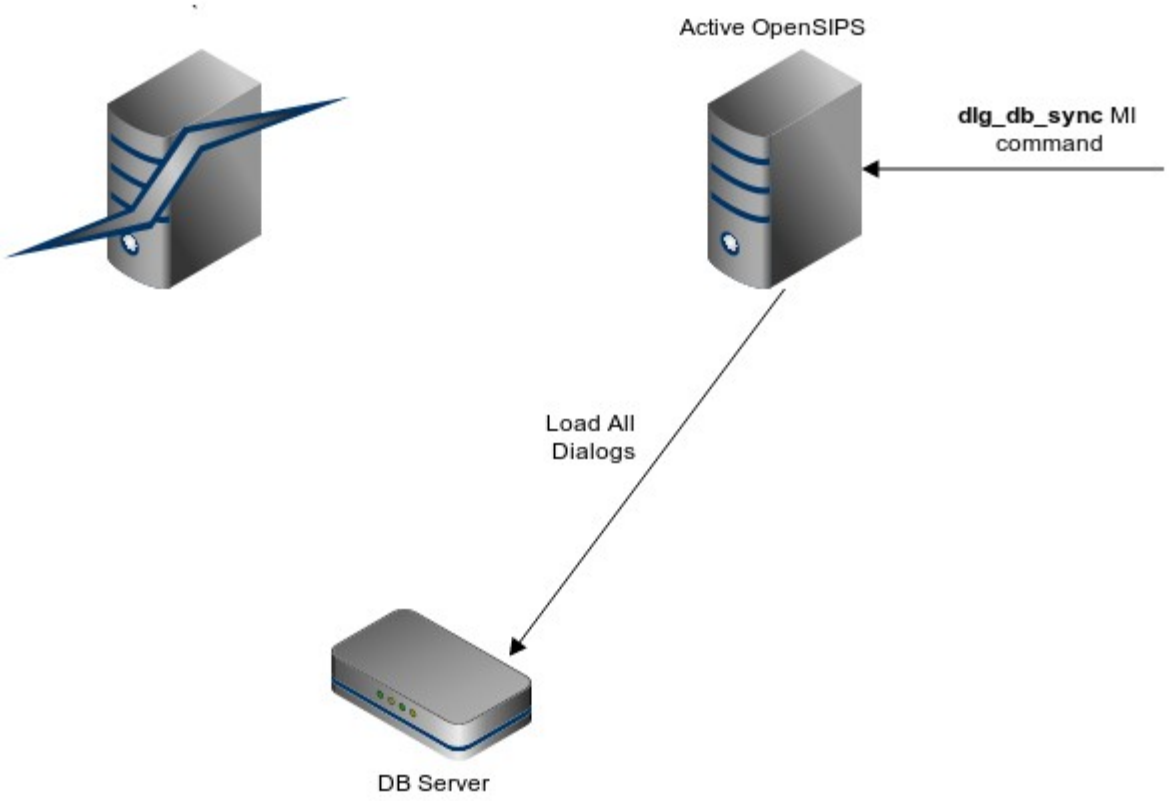
- Highly scalable on it's own, but...

- We need(ed) a DB to store dialogs
  - Persistency
  - Failover

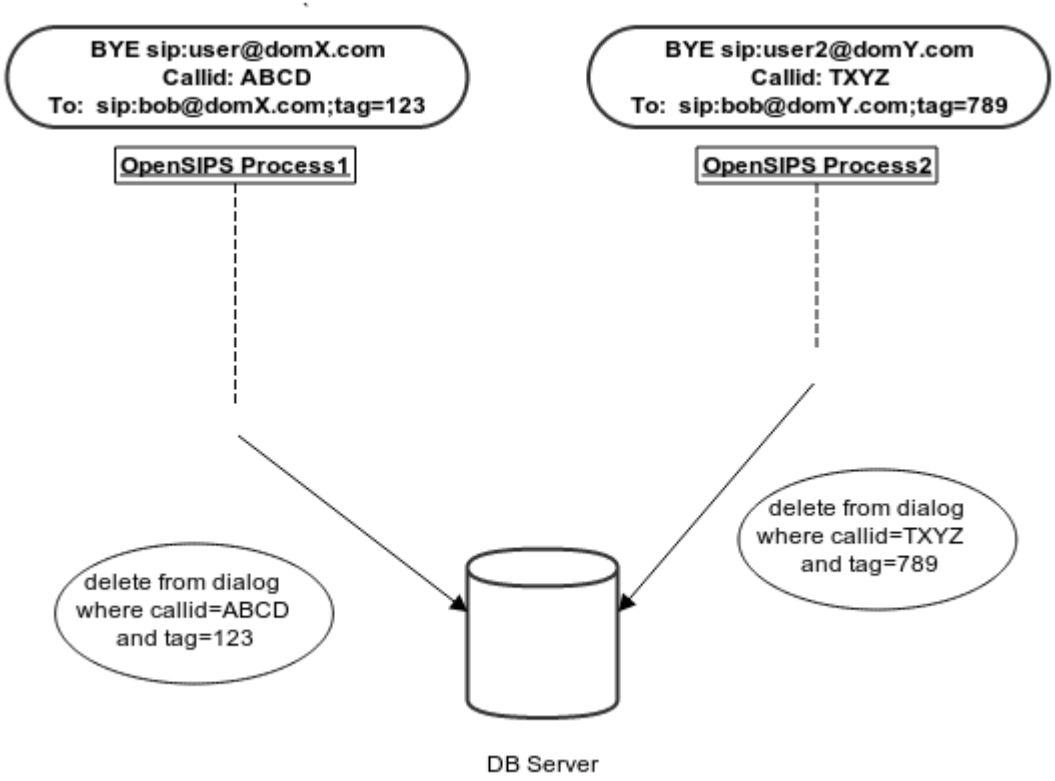
1.8 brought dlg_db_sync MI command



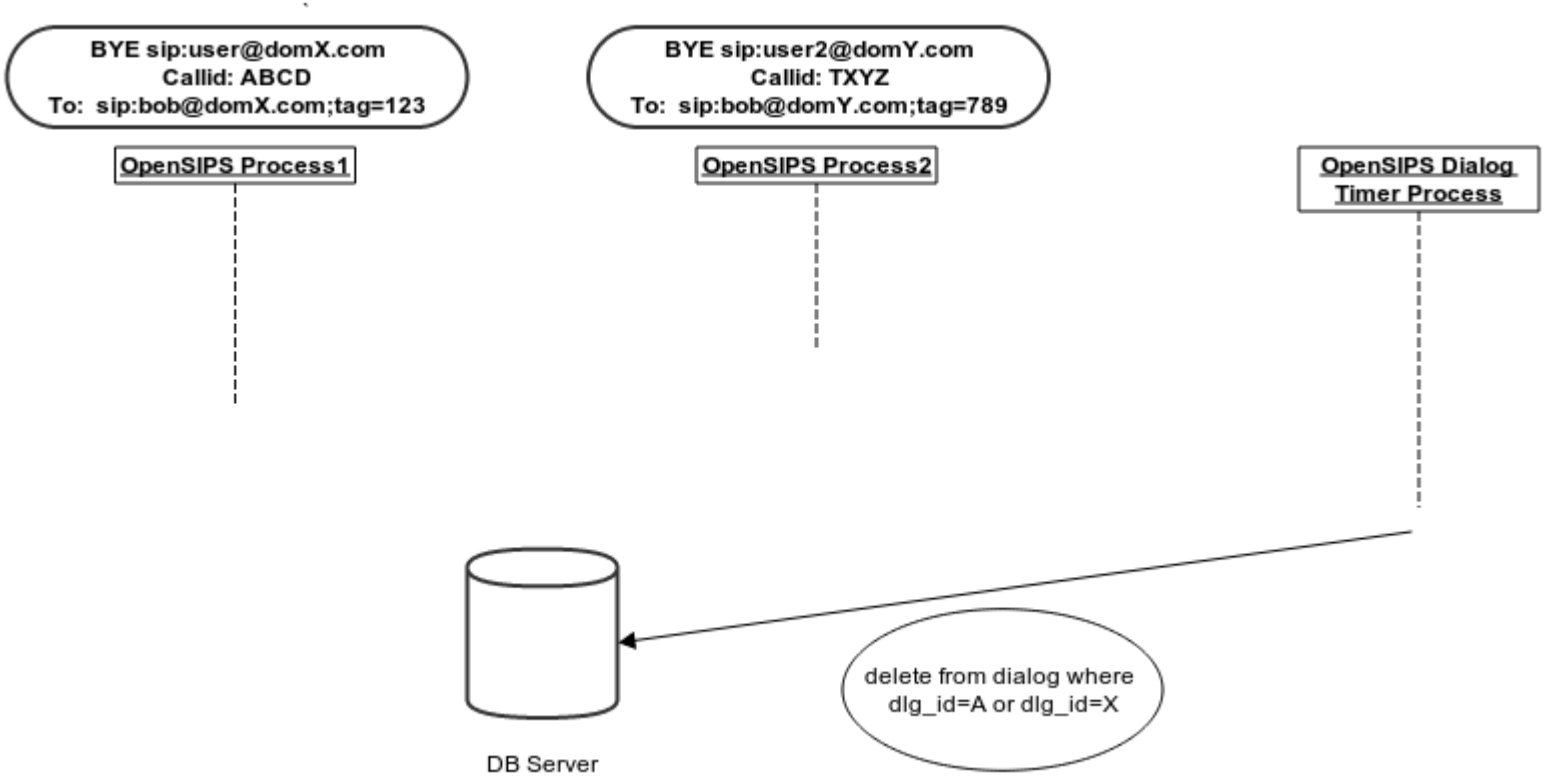
Persistency & Failover



1.8 Dialog Termination



1.9 Dialog Termination



Concurrent Calls Performance

```
query_buffer_size=100
modparam("dialog", "timer_bulk_del_no", 100)
modparam("dialog", "own_timer_process", 1)
```

- But what If we have 1 Million concurrent calls and our server fails ??

- The backup will kick in , issue dlg_db_sync and try to load 1 Million rows from the DB...
- It will take a LONG time
- By the time we've loaded everything, most likely those dialogs would have already ended

Concurrent Calls Performance

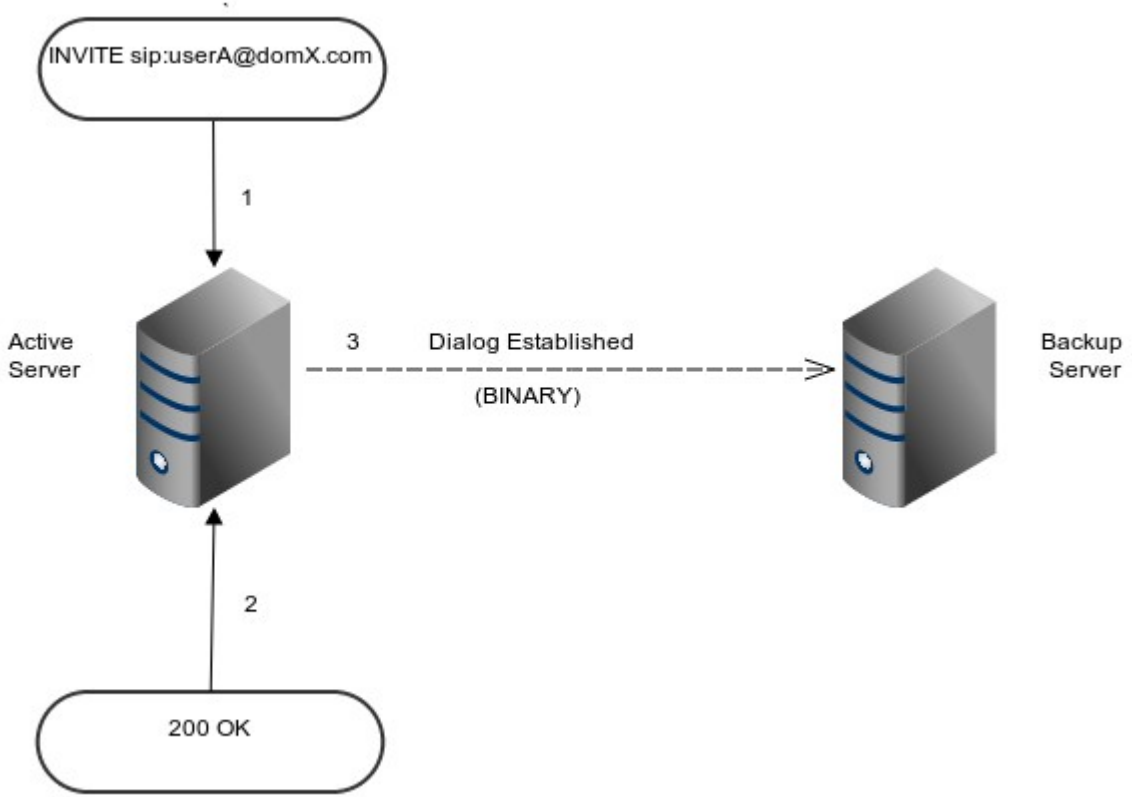
- For large deployments , we NEED real-time dialog replication
  - For fast fail-over

1.10 Binary Internal Interface

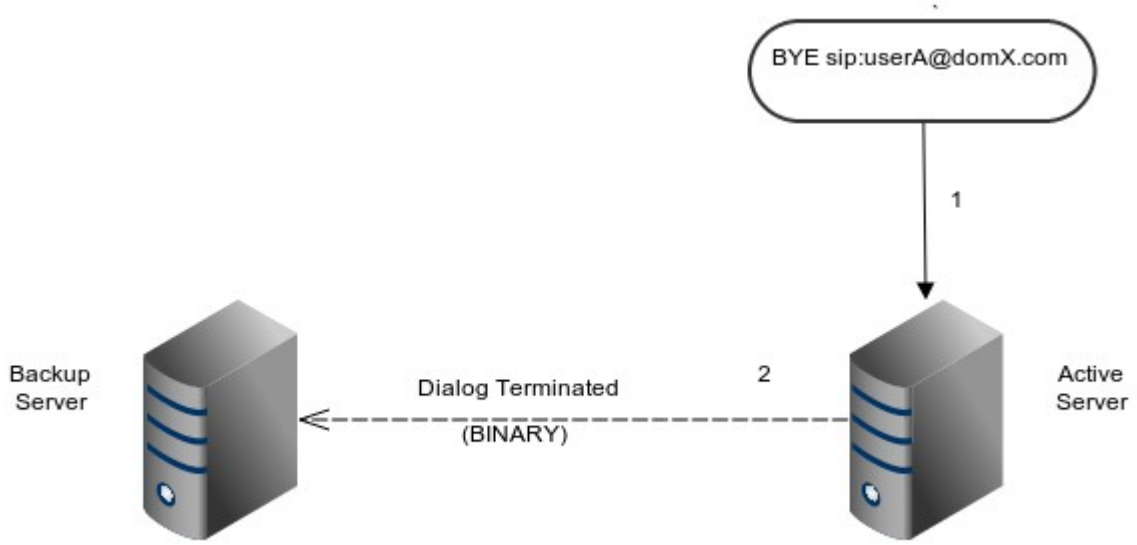
- **Fast and Efficient communication channel between OpenSIPS instance**

- **To be used for real-time data replication**
  - **Dialog state**
  - **Registrations**
  - **Transactions**

1.10 Binary Internal Interface



1.10 Binary Internal Interface



- bin_listen = 10.0.0.150:5062
- bin_children = 5

- modparam("dialog", "accept_replicated_dialogs", 1)

- modparam("dialog", "replicate_dialogs_to", "10.0.0.150:5062")

- Backup servers are kept in perfect sync
- The dialog module can now be massively scalable, no longer being limited by the DB back-end

- Highly CPU Intensive
- OpenSIPS Architecture is great at this
- Highly Protocol Dependent

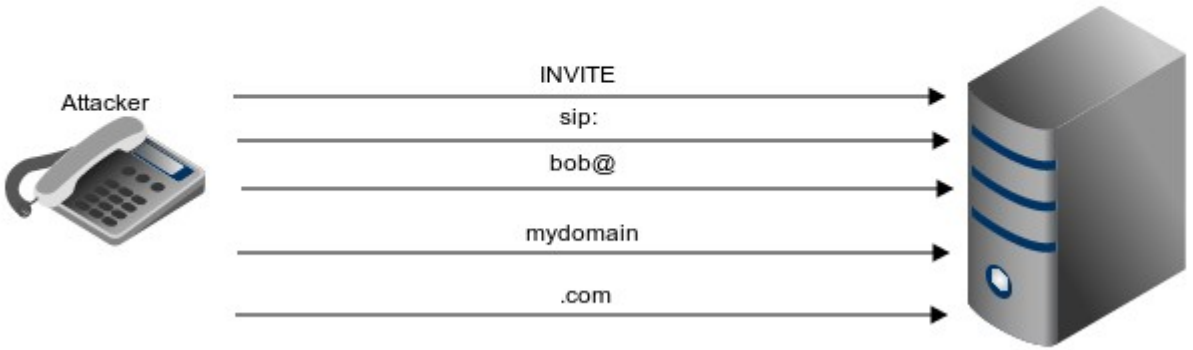
- UDP is easily scalable

- Starting with 1.8, you can have different number of workers for each UDP interface
  - listen=udp:127.0.0.1:5060 use_children 5

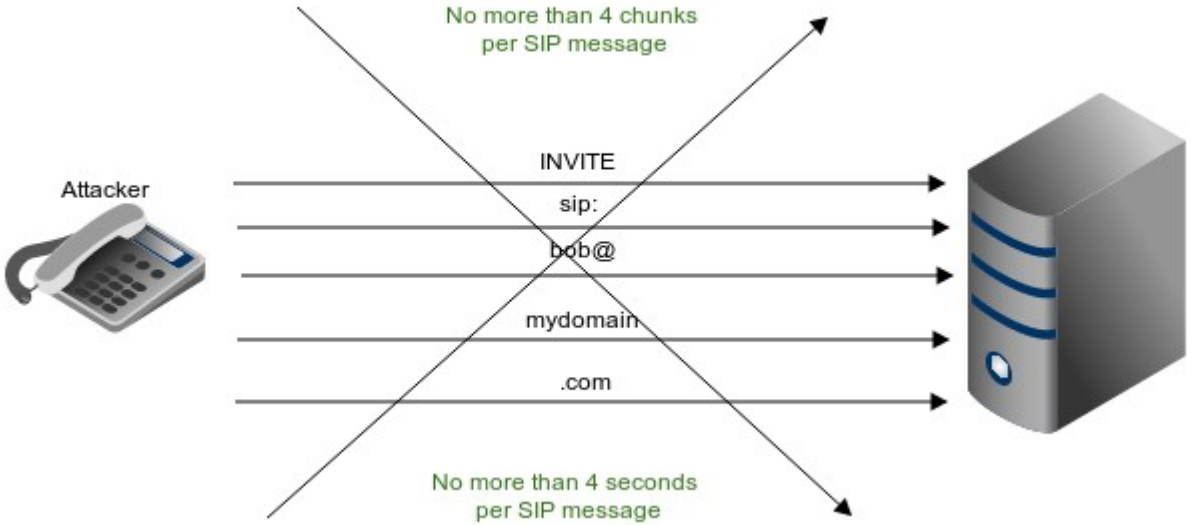
- TCP is harder to scale
- Once you increase the number of connections, you can reach various limitations, even in the TCP kernel stack

- 1.10 has many TCP improvements
- TCP reading is faster, uses less memory
  - 64 KB less per connection
  - Save ~ 6GB per 100 000 TCP connections
- TCP reading is more robust
  - No longer vulnerable to TCP fragmentation attacks (DOS)

CPS Scalability - TCP



CPS Scalability - TCP



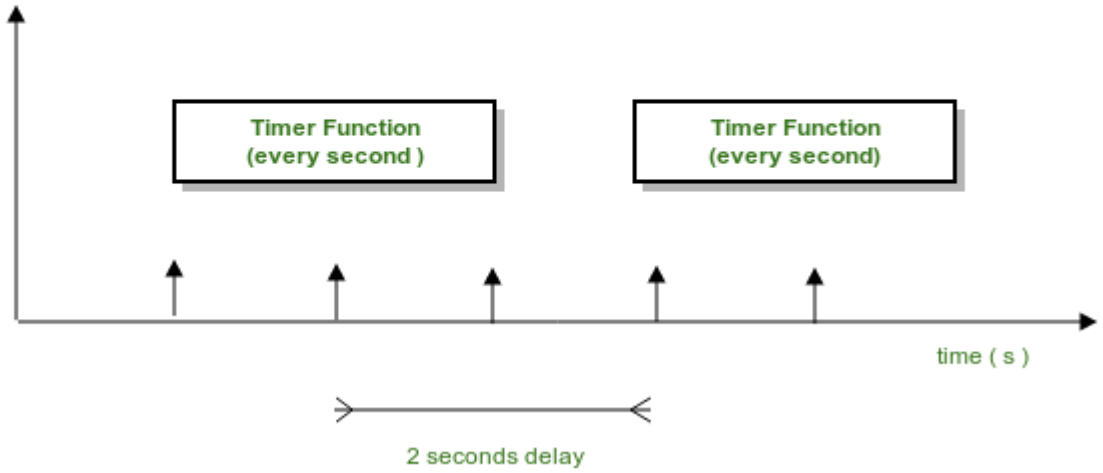
- 1.10 has asynchronous TCP
- No longer have OpenSIPS blocking for TCP connect or write operations

- If you want High CPS, you should employ caching as much as possible
  - Local Caching Engine
  - Memcached
  - Redis
- Integrated DNS caching
  - Via the dns_cache module

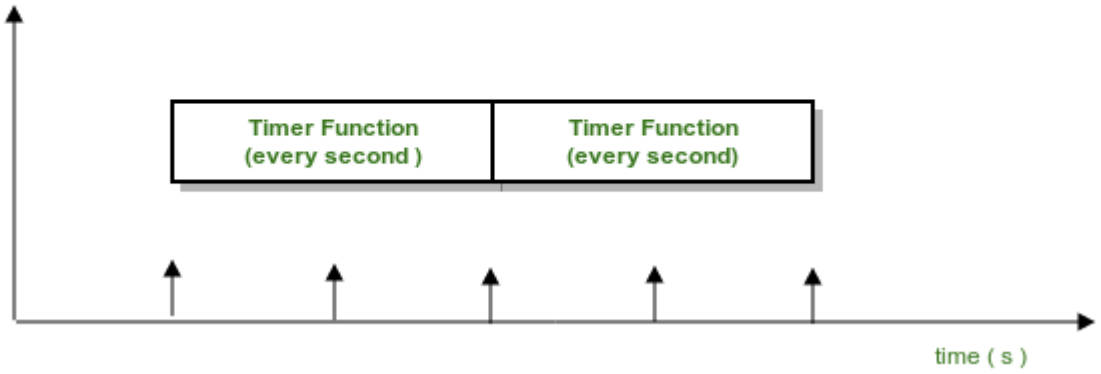
CPS Scalability – Time Drifting

- When having large CPS traffic, the OpenSIPS timer processes can get overloaded
- Timer drifting can have serious consequences
- Starting with 1.9 , OpenSIPS has auto-adaptive timers

CPS Scalability – Time Drifting



CPS Scalability – Time Drifting



- Critical to have a good monitoring of the system

- OpenSIPS exposes various tools for this
  - Load Statistics
  - Thresholds for various OPS
  - Pike
  - Ratelimit

- OpenSIPS is a highly scalable SIP server
- Ideal for high traffic deployment types
  - SBCs
  - Load Balancers
  - Trunking

Thank you for your attention
You can find out more at www.opensips.org
vladpaiu@opensips.org

Questions are welcome