ZABBIX 2020 Conference BENELUX

Fighting false positives and notification floods

in Zabbix distributed environments

Nicola Mauri • Bitech SpA

BITECH
Information technology

www.bitech.it

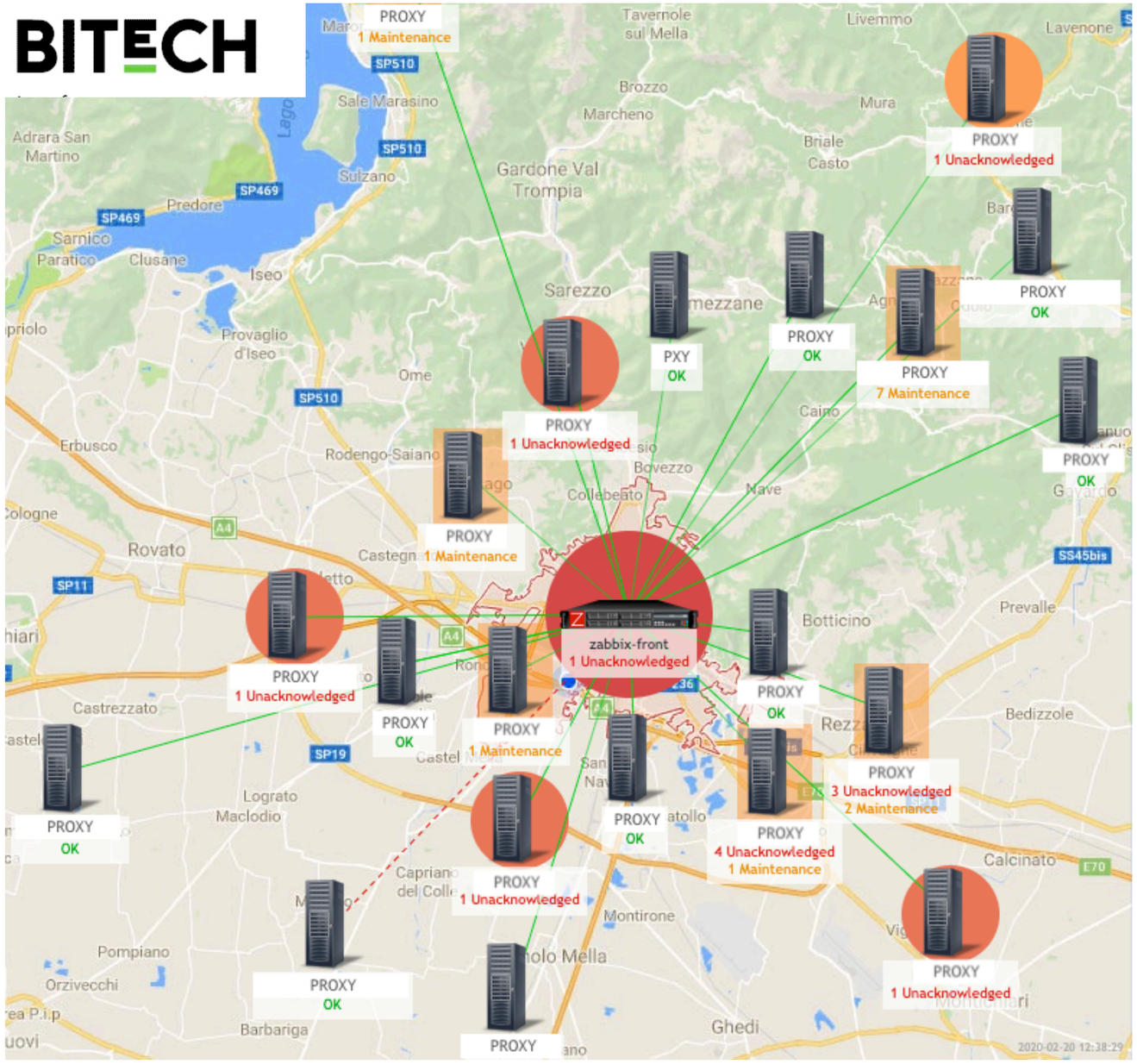
Nicola Mauri

Monitoring and Security Consultant

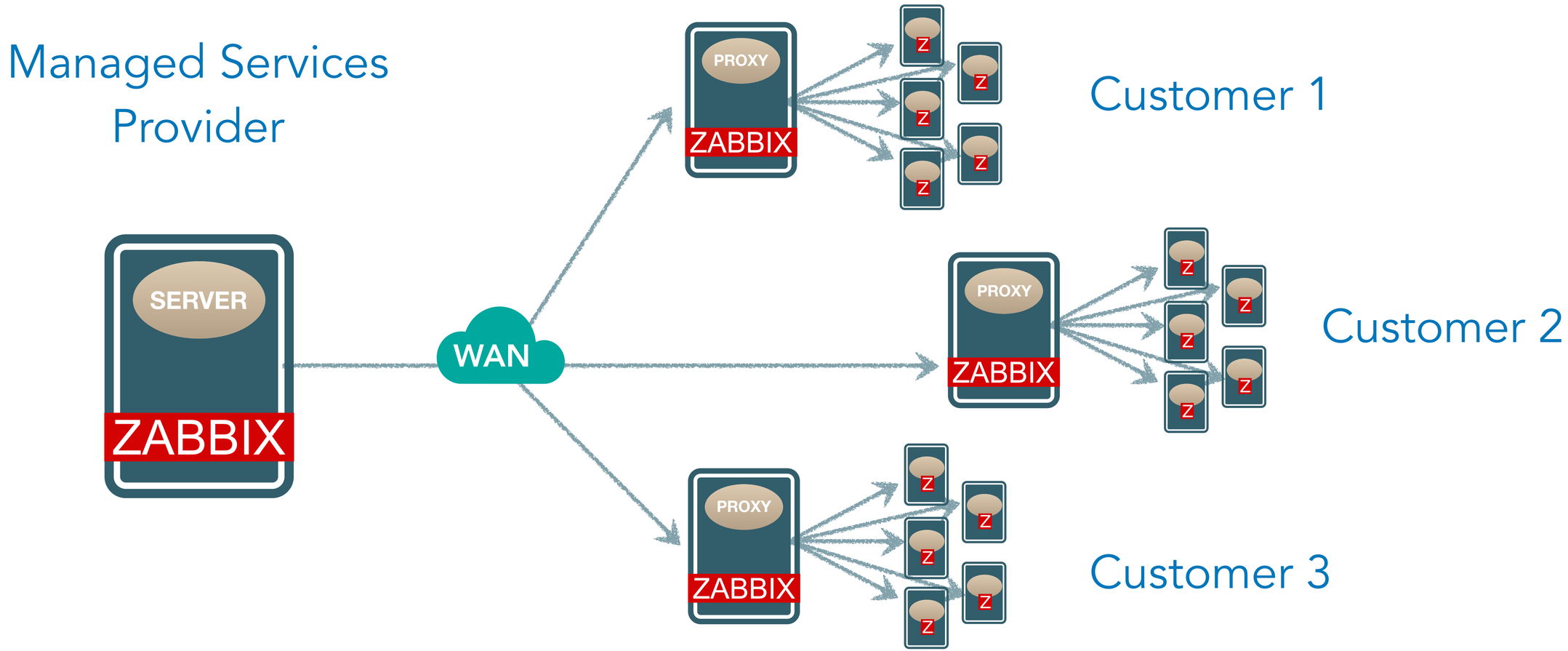
nm@nicolamauri.it

@nicolamauri

A typical MSP scenario



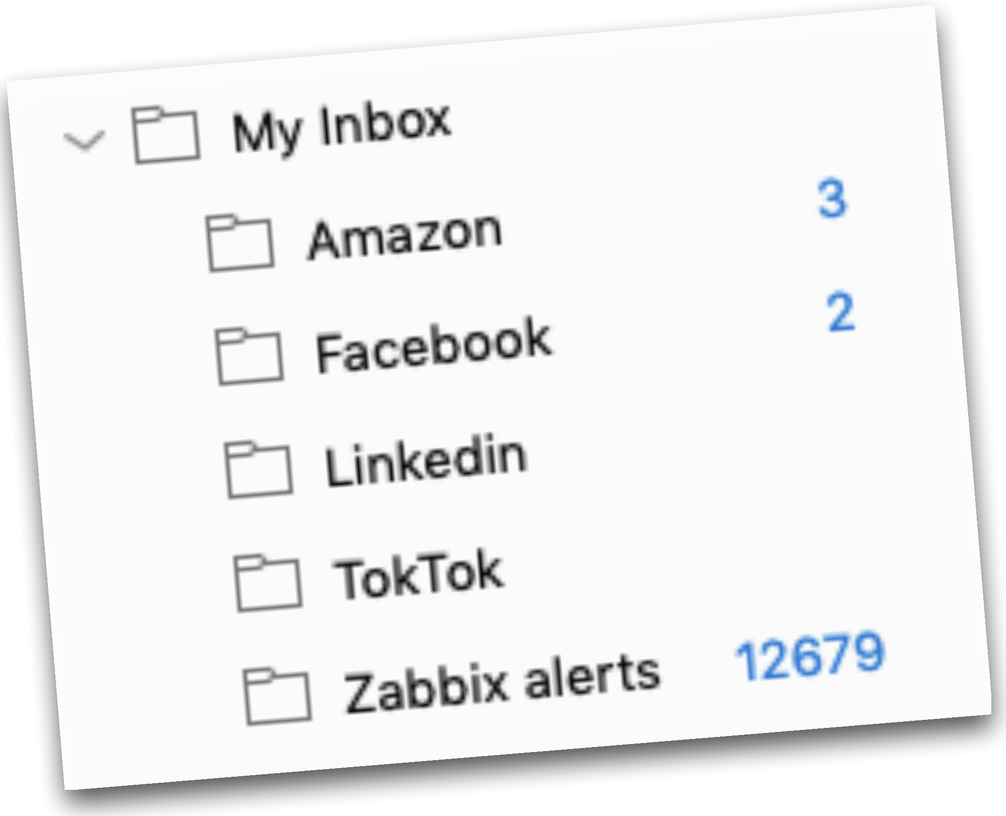
Our scenario



Zabbix in MSP scenarios

- Monitoring-as-a-Service (MaaS) model
- 1 Zabbix Server for several customers
- Active Proxies connecting through WAN
- Non-technical users receive alerts and need to take actions

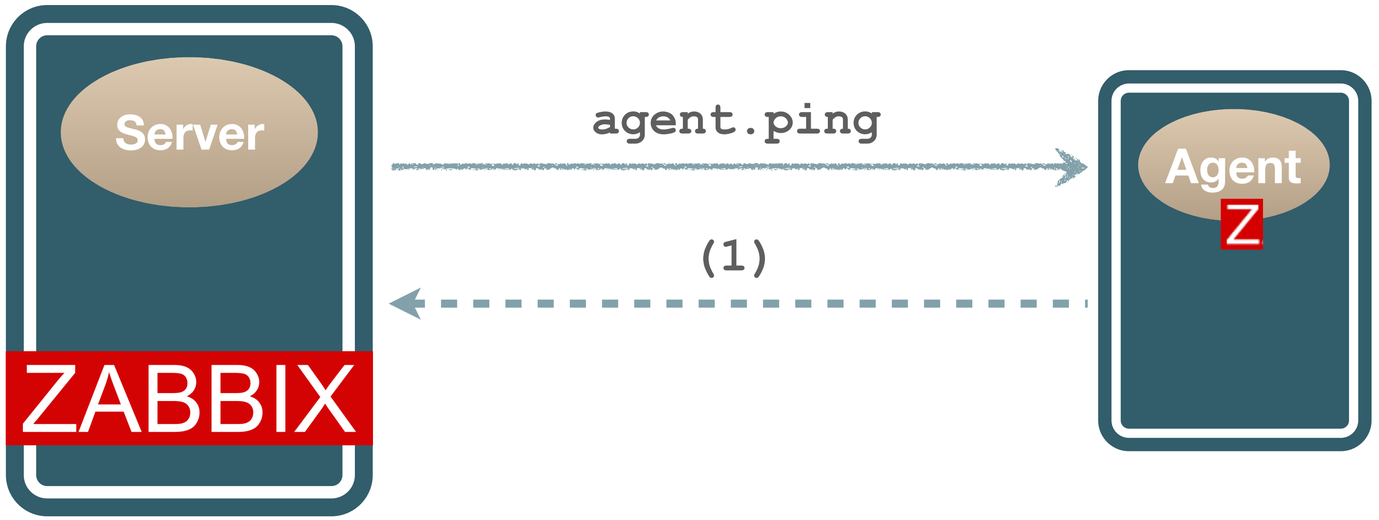
Why fighting them?



- **False positives** undermine monitoring system reputation among your users
- **Notification floods** make your monitoring system inefficient, often when you need it most

How do you determine if a server host is alive?

Agent monitoring: the default way



Parent triggers Template App Zabbix Agent

* Name	Zabbix agent on {HOST.NAME} is unreachable for 5 minutes
Severity	Not classified | Information | Warning | Average | **High** | Disaster
* Expression	{host1:agent.ping.nodata(5m)}=1

Agent monitoring: the default way

Host is down → Zabbix agent unreacheable

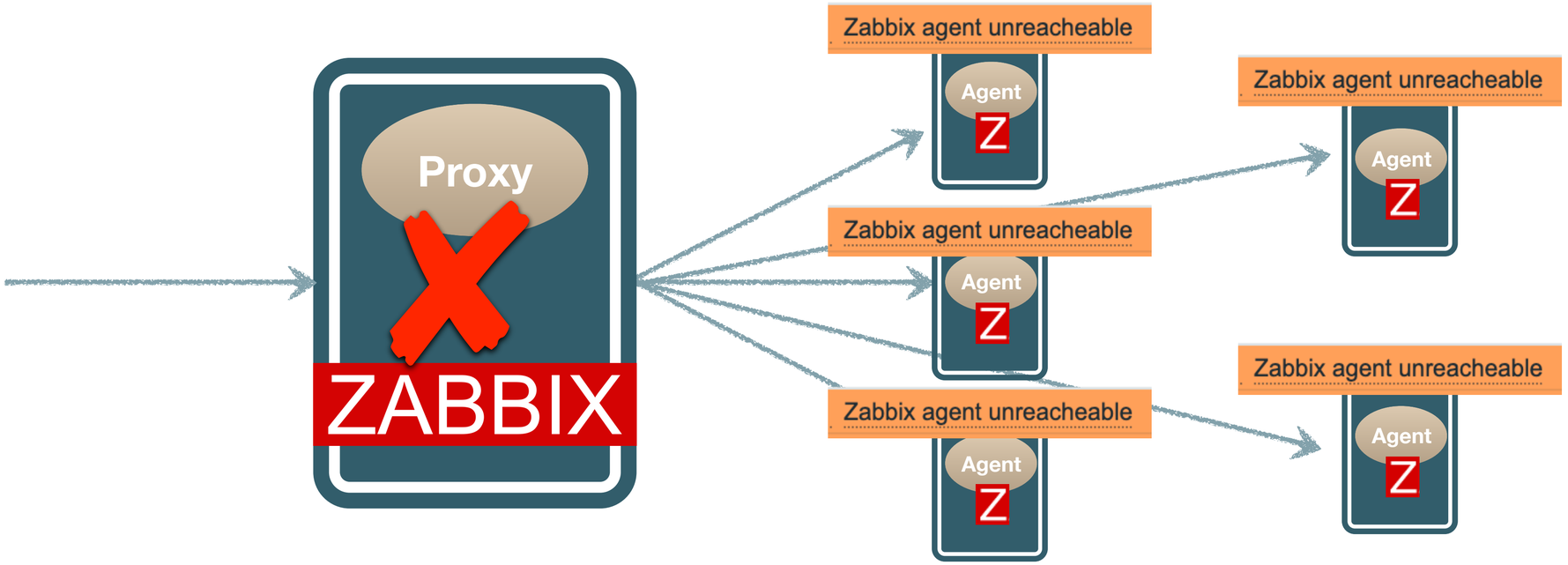
Host is up, agent is down → Zabbix agent unreacheable

Host is up, agent is unresponsive → Zabbix agent unreacheable

- Better than pinging:
  it catches a lot of problems with one single check.

- But it doesn't tell us which problem we have

Distributed environments



* Expression	{host1:agent.ping.nodata(5m)}=1

Distributed environments

agent.ping.nodata( )

also triggers upon non-agent problems:

- Zabbix Proxy is down
- Zabbix Server has connectivity issues
- Zabbix Server has performance issues

Solutions

- Triggers dependency?
- Implicit trigger dependency (ZBXNEXT-1891)
- Hosts dependency (Zabbix > 5.0)
- Event correlation?

Agent monitoring: a combined approach

Idea: using multiple checks to more precisely determine what's happening

Agent monitoring: a combined approach

Let's put all these information together:

ICMP	TCP 10050	agent.ping	Host status
✘	✘		Host is unreacheable
✔	✘		Agent service is down
✔	✔	nodata	System is overloaded or frozen
nodata	nodata	nodata	Unknown, probably not a host issue
✔	✔	✔	A great day for this agent
✘	✔	✔	Ok (a nasty network admin simply blocked our ICMPs)

Agent monitoring: a combined approach

Separate triggers allow sending notifications to the right people:

✘	✘		Host is unreacheable	Networking Team
✔	✘		Agent service is down	Monitoring Team
✔	✔	•	System is overloaded or frozen	System Admins
•	•	•	Unknown, probably not an agent issue	none
✔	✔	✔	A great day for this agent	–
✘	✔	✔	Ok, a nasty network admin simply blocked our ICMPs	–

1. “Host is unreacheable” trigger

ICMP	*TCP*	*agent.ping*	
✘	✘		Host is unreacheable

Trigger:

```
{host:icmpping.max(3m)}=0
and
{host:net.tcp.service[tcp,,10050].max(3m)}=0
```

On failures, `icmpping` and `net.tcp.service` return a '0' value, so you don’t need to use nodata().

2. “Agent service is down” trigger

ICMP	*TCP*	*agent.ping*	
✔	✘		Agent service is down (host and its services may be up)

Trigger expression:

```
{host:net.tcp.service[tcp,,10050].max(30m)}=0
```

Depends on trigger:

"Host is unreacheable"

3. “System is overloaded or freezed”

ICMP	*TCP*	*agent.ping*	
✔	✔	nodata	System is overloaded or frozen

- We can connect to the agent TCP port
  → *host must be up, agent must be up.*
- We do have data for icmp item
  → *the proxy/server chain is working.*
- We have no data from agent
  → *the agent is unresponsive: the system must be blocked or overloaded.*

3. “System is overloaded or freezed”

ICMP	*TCP*	*agent.ping*	
✔	✔	nodata	System is overloaded or frozen

Problem expression:

```
{host:net.tcp.service[tcp,,10050].min(1h)}=1
and {host:icmpping.nodata(3h)}=0
and {host:agent.ping.nodata(3h)}=1
```

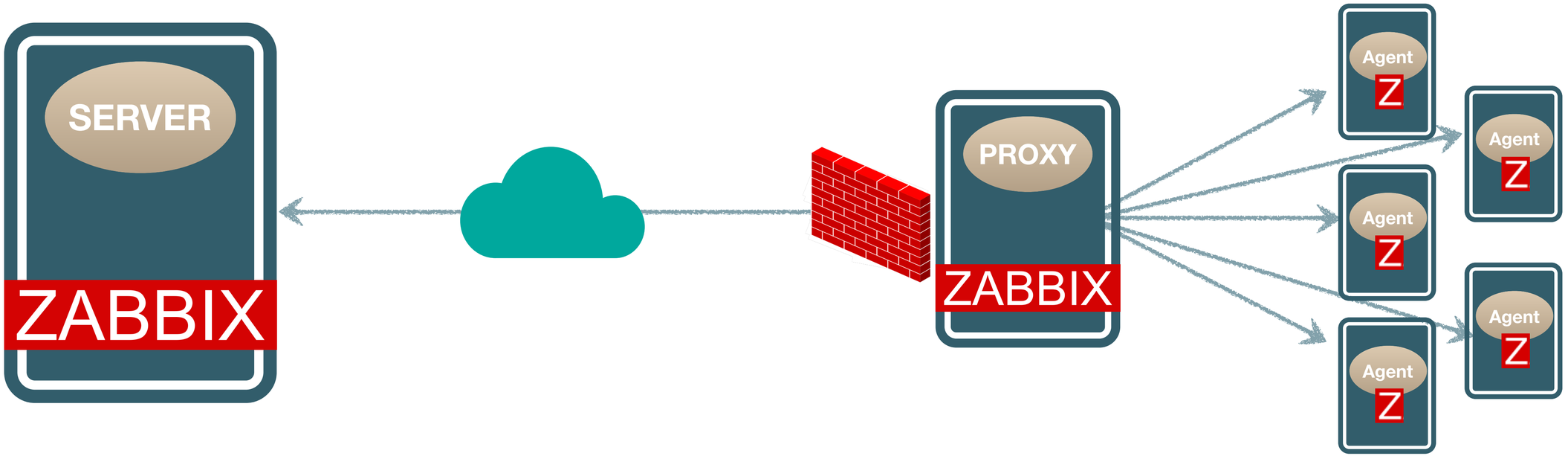
Recovery expression:

```
{host:agent.ping.count(30m,1)}>10
```

💡 Choose large time intervals to avoid flapping and some unobvious race conditions.

How do you detect Zabbix proxies failures?

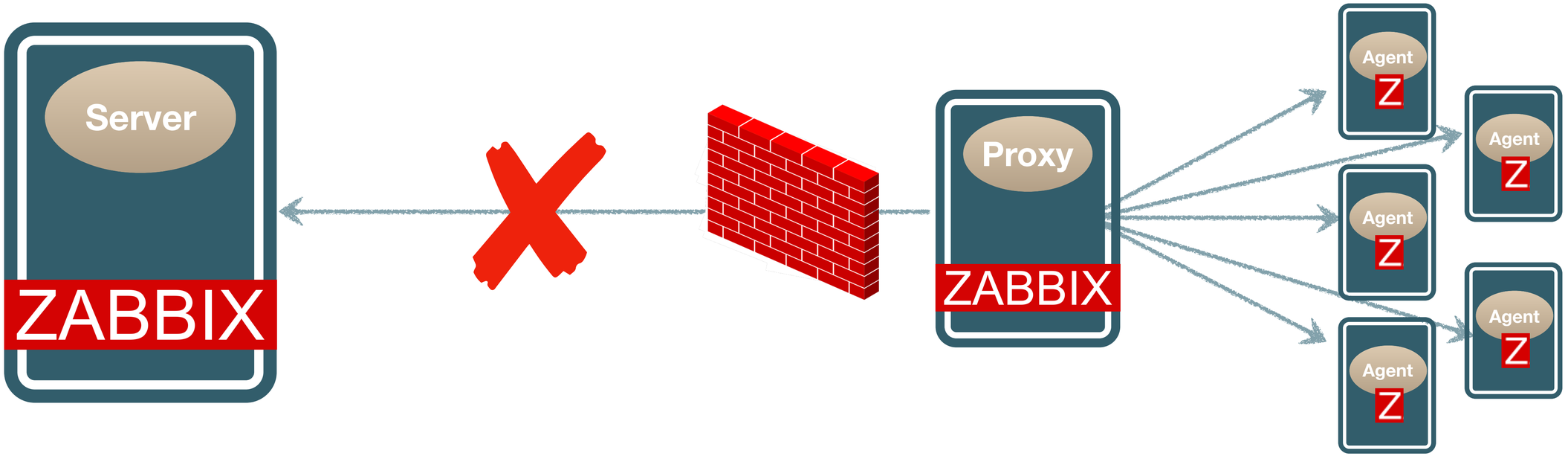
Detecting active proxies failures



A remote **active** proxy often sits behind a firewall and doesn't accept direct connections from Zabbix Server.

If a proxy is unreachable, we simply observe a lack of data.

Detecting active proxies failures



Internal item: `zabbix[proxy,{$PROXY_NAME},lastaccess]`

(not used in default templates)

“No connection from proxy” trigger

lastaccess	*Scenario*
>1 min	No connection from proxy

Trigger expression:

```
{proxy1:zabbix[proxy,{$PROXY_NAME},lastaccess].fuzzytime(1m)}=0
```

It could mean:

- Proxy host is down
- Proxy service is stopped
- Proxy site is unreacheable (network or power outage)

Remote proxies: best practices

1. Keep the proxy host itself monitored, using agent.
2. `lastaccess` requires proxy name parameter. Use macros and ensure that item gets evaluated, i.e.:

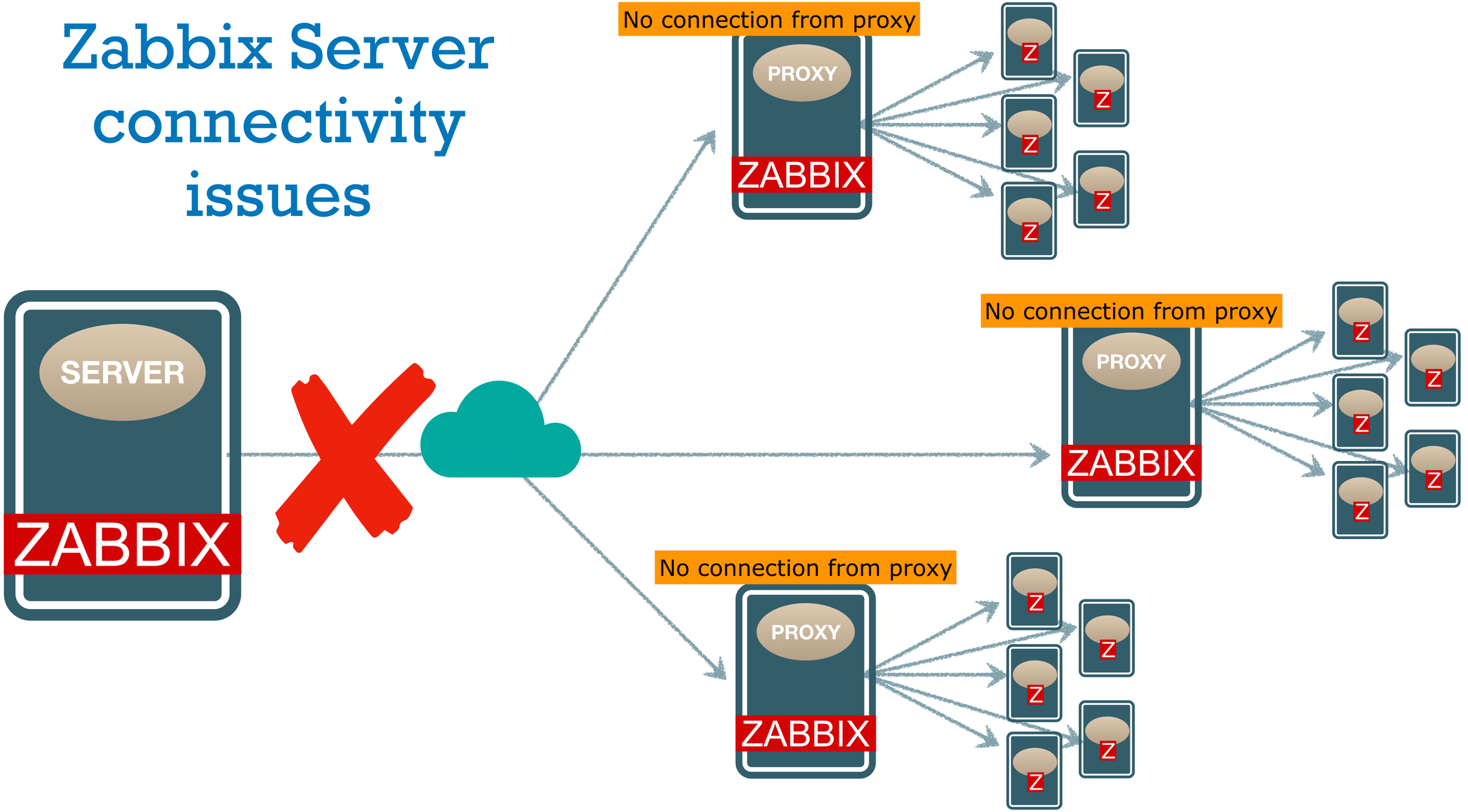
```
{myproxy1:zabbix[proxy,{$PROXY_NAME},lastaccess].nodata(24h)}=1
```

=> "Cannot determine proxy last access time"

3. Monitor proxy connectivity: ping its router from Zabbix Server, for rapid troubleshooting.
4. Use passive proxies, if security policies allow you.

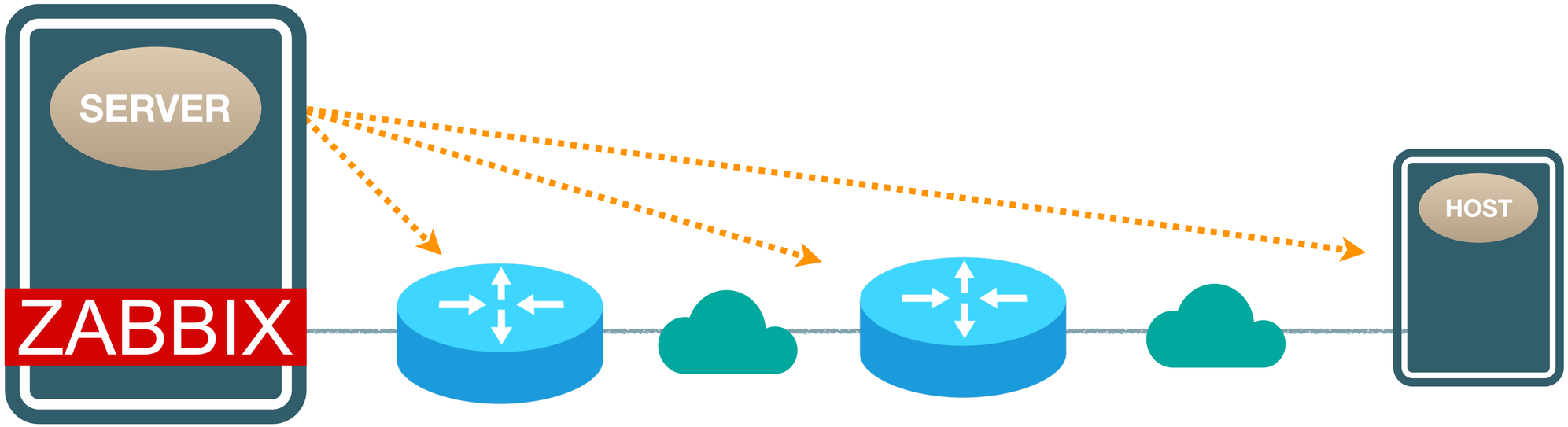
How do you handle Zabbix Server connectivity issues?

Zabbix Server connectivity issues



Zabbix Server connectivity issues

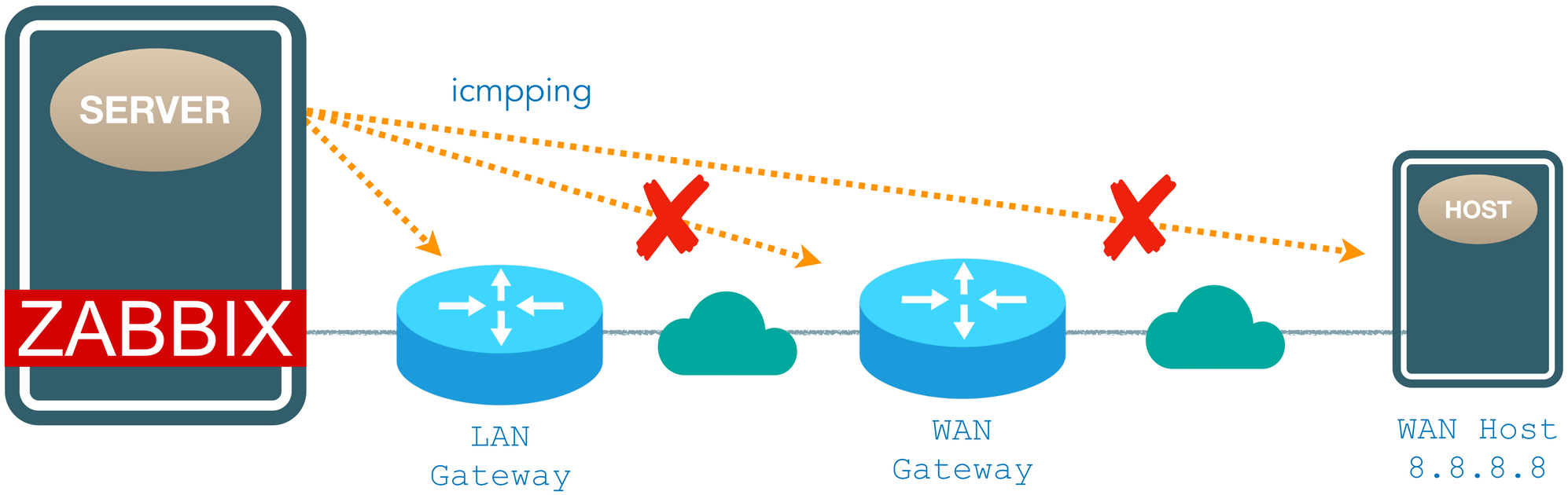
Idea: auto-detecting connectivity problems and preventing alerts to be fired



Zabbix Server connectivity issues

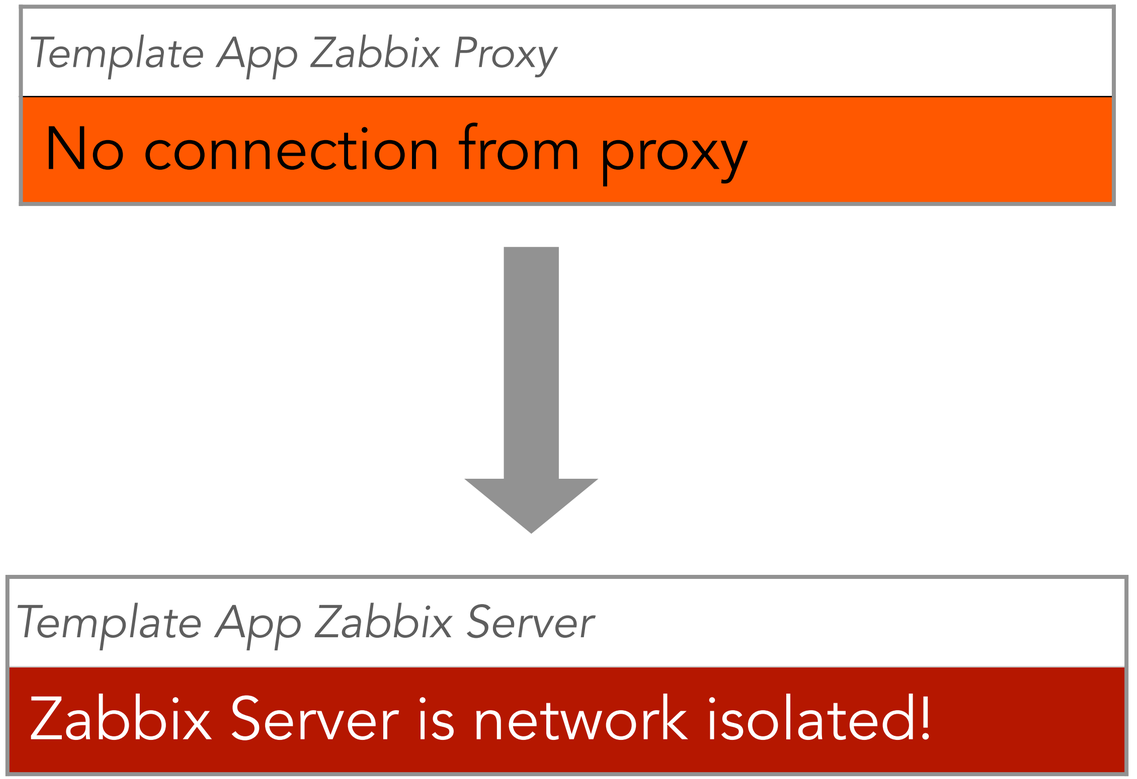
Zabbix Server is network isolated!

```
{zbxsrv:icmpping[{$WAN_GATEWAY}].last()}=0
and
{zbxsrv:icmpping[{$WAN_HOST}].last()}=0
```



Zabbix Server connectivity issues

Make proxy trigger dependent on this trigger in order to avoid misleading notifications.



(In theory an isolated server should not be able to send notifications. However mail notifications might be queued and sent later when connectivity is restored)

Conclusions

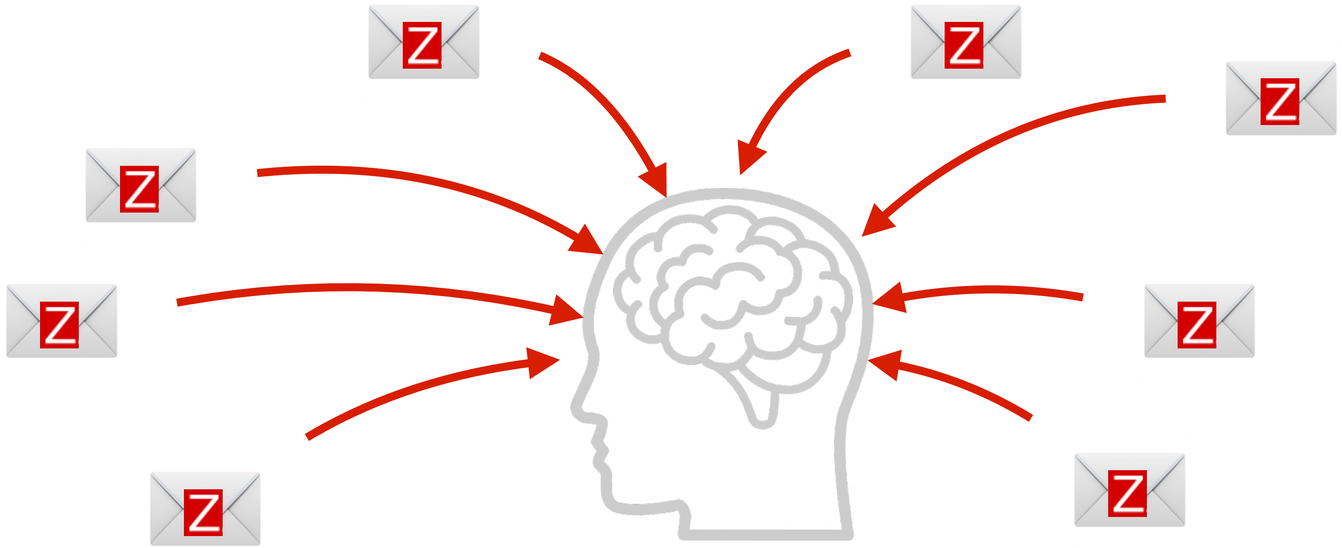
- Simple yet robust approach
- Tested on a wide range of deployments for 3+ years
- All logic stored into templates
  (no manual configuration or tuning needed)

Main benefits:

- More precise notifications
- Reduces false positives
- Prevents alert floods on distributed environments

Lessons learned in MSP environments

1. Do not overwhelm users



- PKI: Number of alerts /per user /per day
- Recommended (human-sustainable): **< 20**

2. Inform users about changes

- Users expect a **predictable and consistent behavior**
- Keep them informed about **changes** to monitoring service:
  - New hosts (auto-registration)
  - New discovered entities (LLD)
  - New triggers
  - New thresholds

3. Adopt a consistent severity classification

Applied thorough all your templates • Based on “objective” criteria • Associable to intervention priorities

For example:

CRITERIA DEFINITION		SEVERITY	EXPECTED RESPONSE TIME
Data loss	➡	**Disaster**	*Immediately (too late?)*
Service failure	➡	**High**	*Immediately*
Imminent service failure	➡	**Average**	*As soon as possible*
Unusual condition (no impact on services)	➡	**Warning**	*On daily/weekly basis*
Non-problem event to keep track	➡	**Information**	–
Internal monitoring event	➡	**Not classified**	–

BITECH
Information technology

Thank you!

Nicola Mauri

Monitoring and Security Consultant

nm@nicolamauri.it

@nicolamauri

www.bitech.it