

# Preventing (Network) Time Travel with Chronos

Omer Deutsch, **Neta Rozen Schiff**, Danny Dolev, Michael Schapira



THE HEBREW  
UNIVERSITY  
OF JERUSALEM

# NTP Architecture

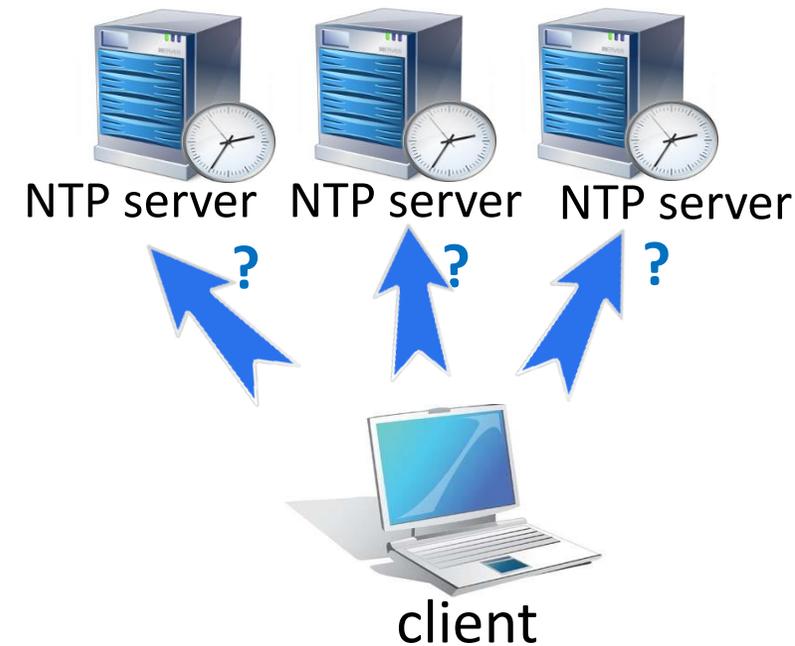
- NTP's client-server architecture consists of two main steps:

- Poll process:**

The NTP client gathers time samples from NTP servers

Poll process:

NTP queries



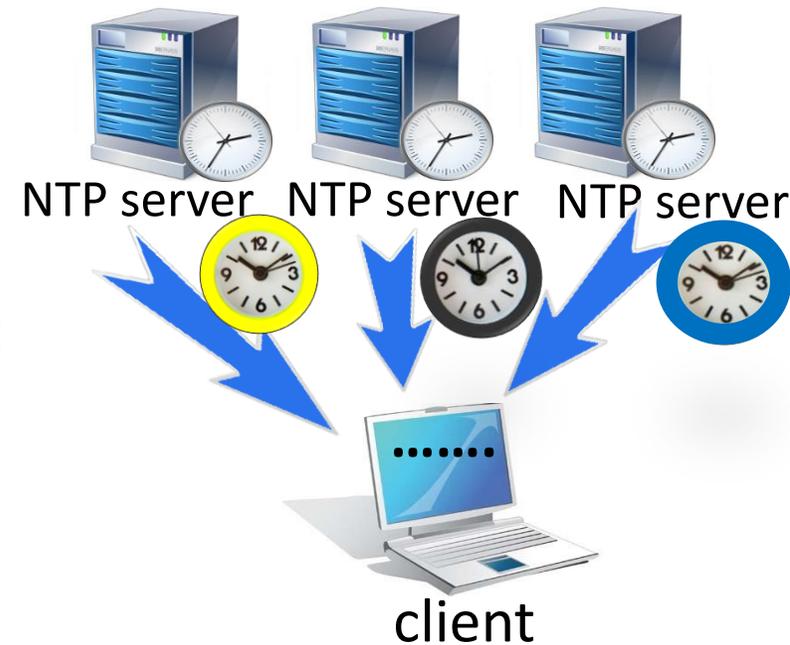
# NTP Architecture

- NTP's client-server architecture consists of two main steps:

- Poll process:**

The NTP client gathers time samples from NTP servers

**Poll process:**    **NTP responses:**



# NTP Architecture

- NTP's client-server architecture consists of two main steps:

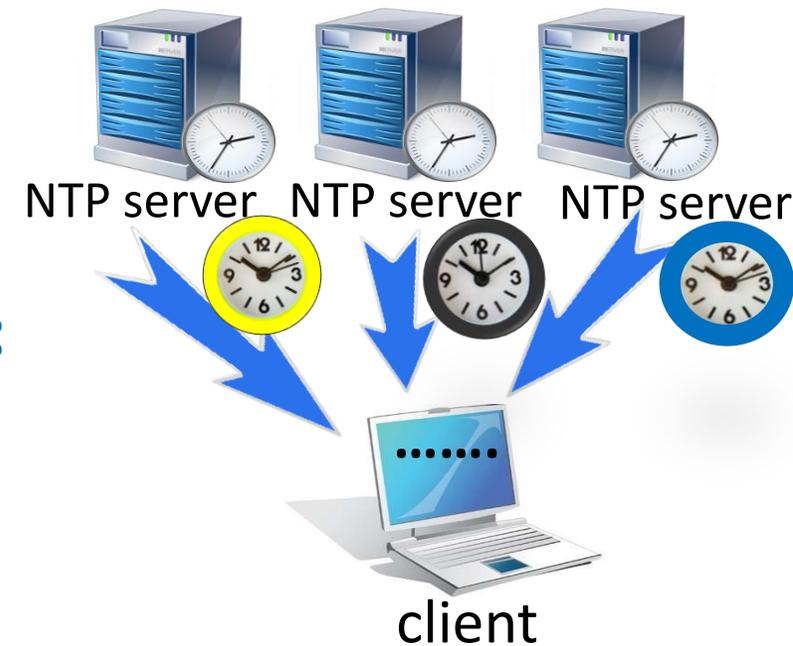
1. **Poll process:**

The NTP client gathers time samples from NTP servers

2. **Selection process:**

The “best” time samples are selected and are used to update the local clock

**Poll process:** NTP responses:  
**Selection process:**



# NTP Architecture

- NTP's client-server architecture consists of two main steps:

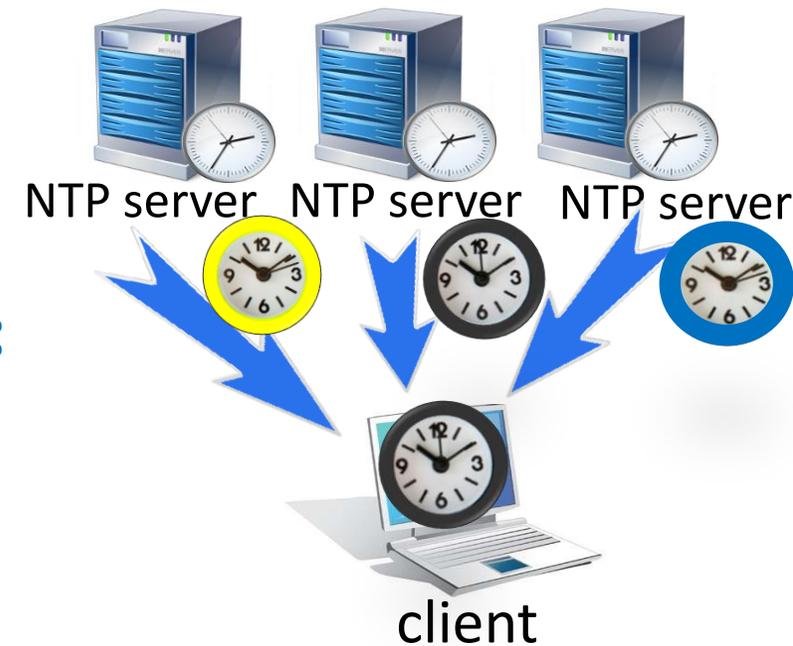
1. **Poll process:**

The NTP client gathers time samples from NTP servers

2. **Selection process:**

The “best” time samples are selected and are used to update the local clock

**Poll process:** NTP responses:  
**Selection process:**



# NTP Man-in-the-Middle (MitM) Attack

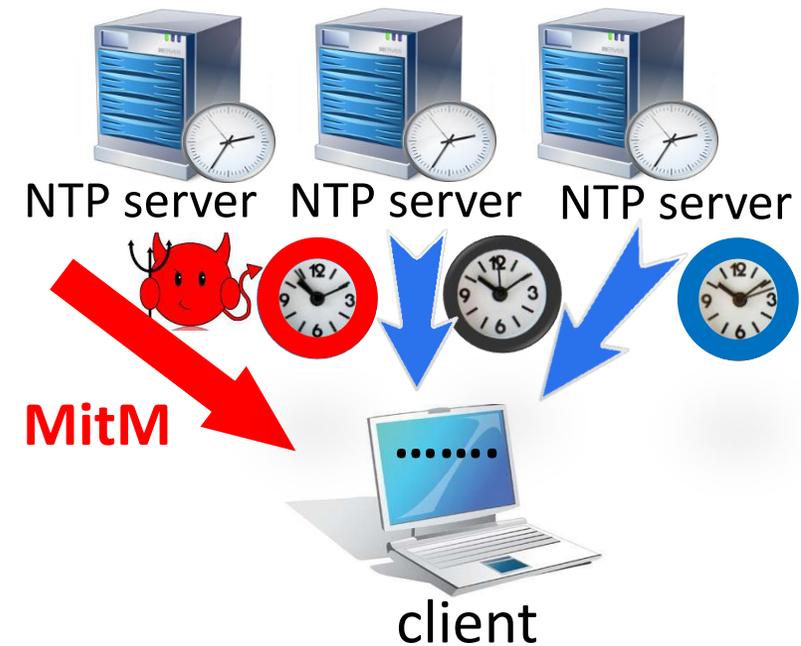
- NTP is highly vulnerable to time shifting attacks, especially by a MitM attacker
  - Can tamper with NTP responses



client

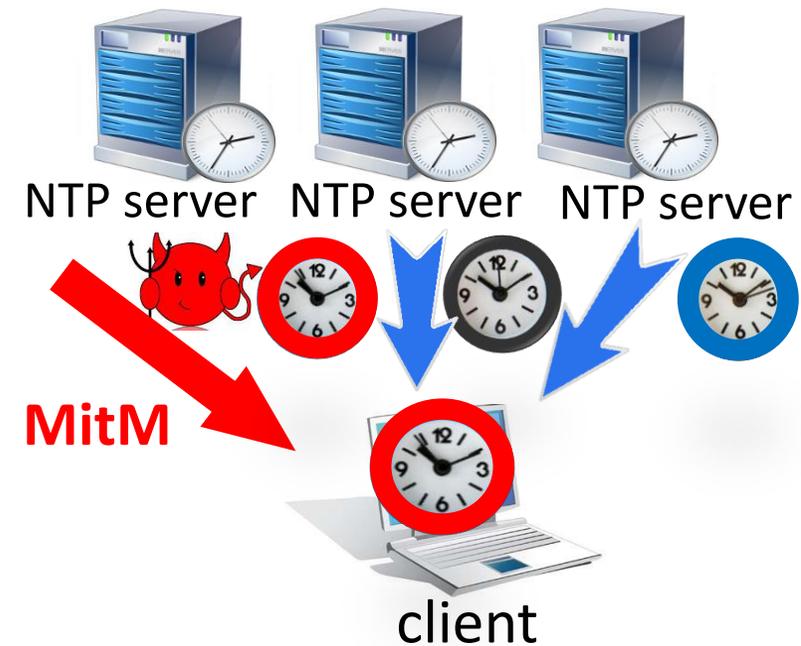
# NTP Man-in-the-Middle (MitM) Attack

- NTP is highly vulnerable to time shifting attacks, especially by a MitM attacker
  - Can tamper with NTP responses



# NTP Man-in-the-Middle (MitM) Attack

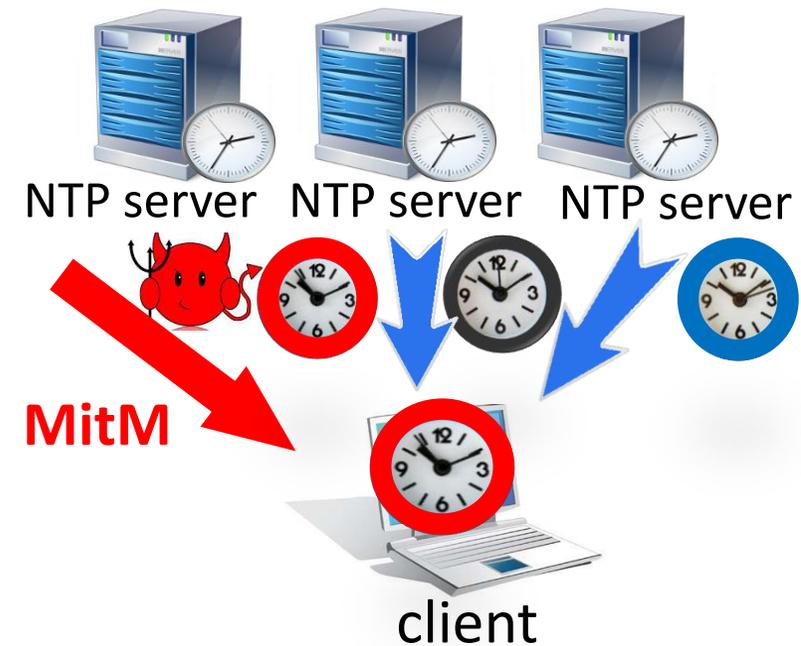
- NTP is highly vulnerable to time shifting attacks, especially by a MitM attacker
  - Can tamper with NTP responses



# NTP Man-in-the-Middle (MitM) Attack

- NTP is highly vulnerable to time shifting attacks, especially by a MitM attacker
  - Can tamper with NTP responses
  - Can impact local time at client simply by dropping and delaying packets to/from servers (**encryption and authentication are insufficient**)

- Previous studies consider MitM as “too strong for NTP”



# Why is NTP so Vulnerable to MitM?

- **NTP's poll process** relies on a small set of NTP servers (e.g., from pool.ntp.org), and this set is often DNS-cached (implementation property).

# Why is NTP so Vulnerable to MitM?

- **NTP's poll process** relies on a small set of NTP servers (e.g., from pool.ntp.org), and this set is often DNS-cached (implementation property).

Attacker only needs MitM capabilities with respect to few NTP servers

# Why is NTP so Vulnerable to MitM?

- **NTP's poll process** relies on a small set of NTP servers (e.g., from pool.ntp.org), and this set is often DNS-cached (implementation property).

Attacker only needs MitM capabilities with respect to few NTP servers

- **NTP's selection process** assumes that inaccurate sources are rare and fairly well-distributed around the UTC (the correct time)

# Why is NTP so Vulnerable to MitM?

- **NTP's poll process** relies on a small set of NTP servers (e.g., from pool.ntp.org), and this set is often DNS-cached (implementation property).

Attacker only needs MitM capabilities with respect to few NTP servers

- **NTP's selection process** assumes that inaccurate sources are rare and fairly well-distributed around the UTC (the correct time)

Powerful and sophisticated MitM attackers are beyond the scope of **traditional** threat models

# Chronos to the Rescue

The **Chronos NTP client** is designed to achieve the following:

- **Provable security** in the face of fairly powerful MitM attacks
  - negligible probability for successful timeshifting attacks
- **Backwards-compatibility**
  - no changes to NTP servers
  - limited software changes to client
- **Low computational and communication overhead**
  - query few NTP servers

# Threat Model

The attacker:

- Controls a large fraction of the NTP servers in the pool (say,  $\frac{1}{4}$ )
- Capable of both deciding the content of NTP responses and timing when responses arrive at the client
- Malicious

# Chronos Architecture

Chronos' design combines several ingredients:

- **Rely on many NTP servers**
  - Generate a large server pool (hundreds) per client
    - E.g., by repeatedly resolving NTP pool hostnames and storing returned IPs
  - Sets a very high threshold for a MitM attacker

# Chronos Architecture

Chronos' design combines several ingredients:

- **Rely on many NTP servers**
  - Generate a large server pool (hundreds) per client
    - E.g., by repeatedly resolving NTP pool hostnames and storing returned IPs
  - Sets a very high threshold for a MitM attacker
- **Query few servers**
  - Randomly query a small fraction of the servers in the pool (e.g., 10-20)
  - Avoids overloading NTP servers

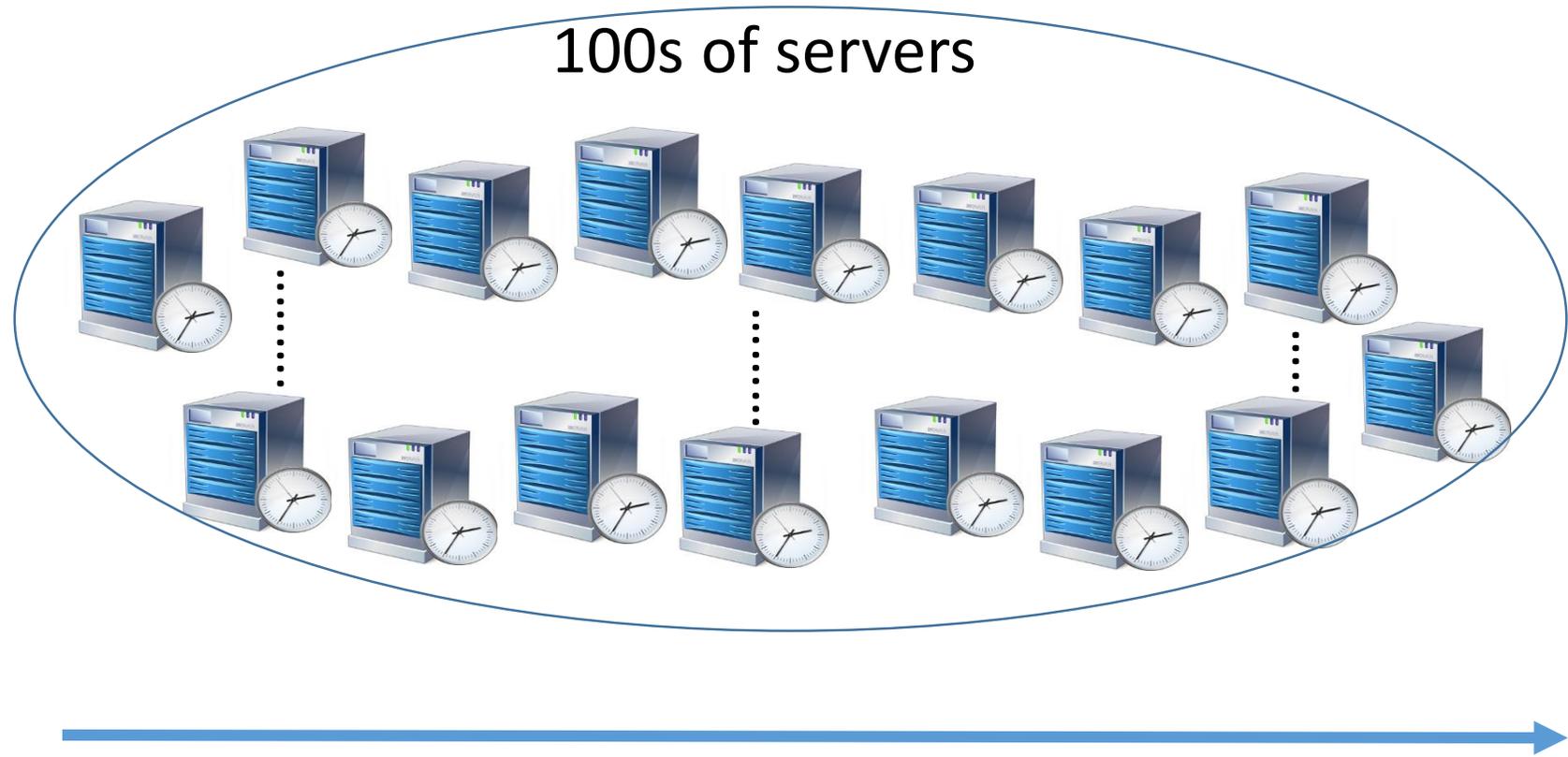
# Chronos Architecture

Chronos' design combines several ingredients:

- **Rely on many NTP servers**
  - Generate a large server pool (hundreds) per client
    - E.g., by repeatedly resolving NTP pool hostnames and storing returned IPs
  - Sets a very high threshold for a MitM attacker
- **Query few servers**
  - Randomly query a small fraction of the servers in the pool (e.g., 10-20)
  - Avoids overloading NTP servers
- **Smart filtering**
  - Remove outliers via a technique used in approximate agreement algorithms
  - Limit the MitM attacker's ability to contaminate the chosen time samples

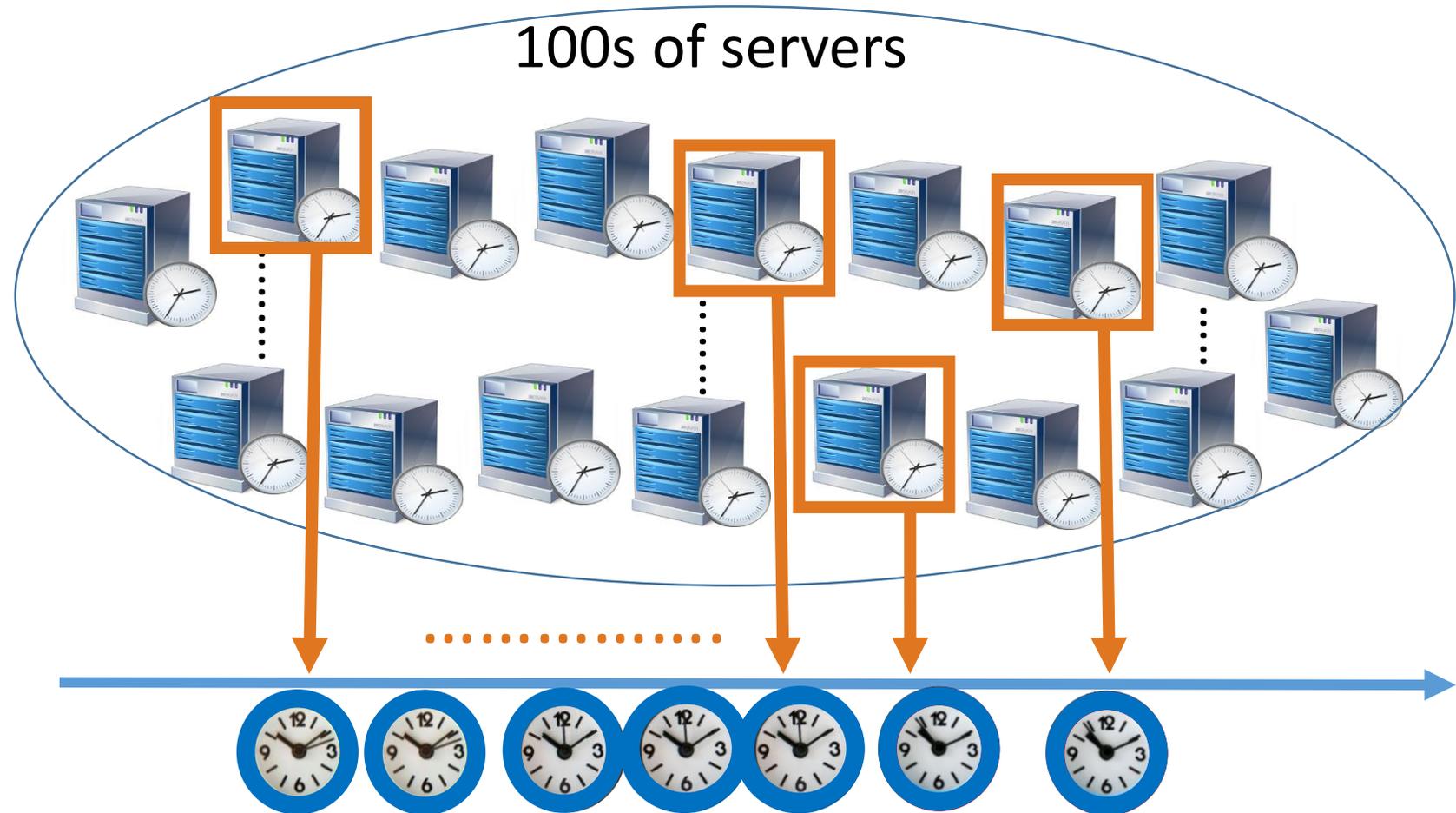
# Chronos' Time-Update Algorithm: Informal

- Query  $m$  (10s of) servers at random



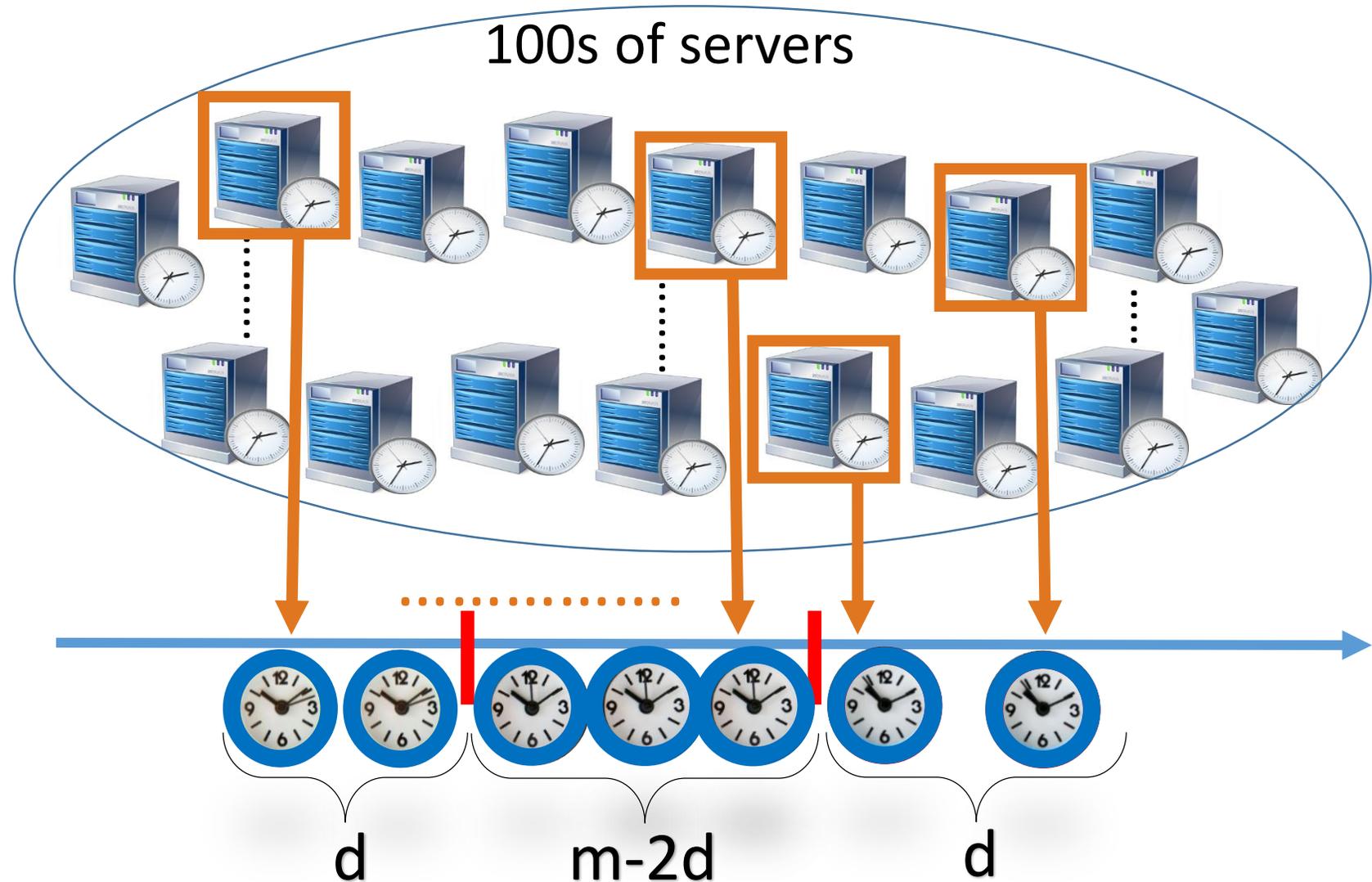
# Chronos' Time-Update Algorithm: Informal

- Query  $m$  (10s of) servers at random
- Order time samples from low to high



# Chronos' Time-Update Algorithm: Informal

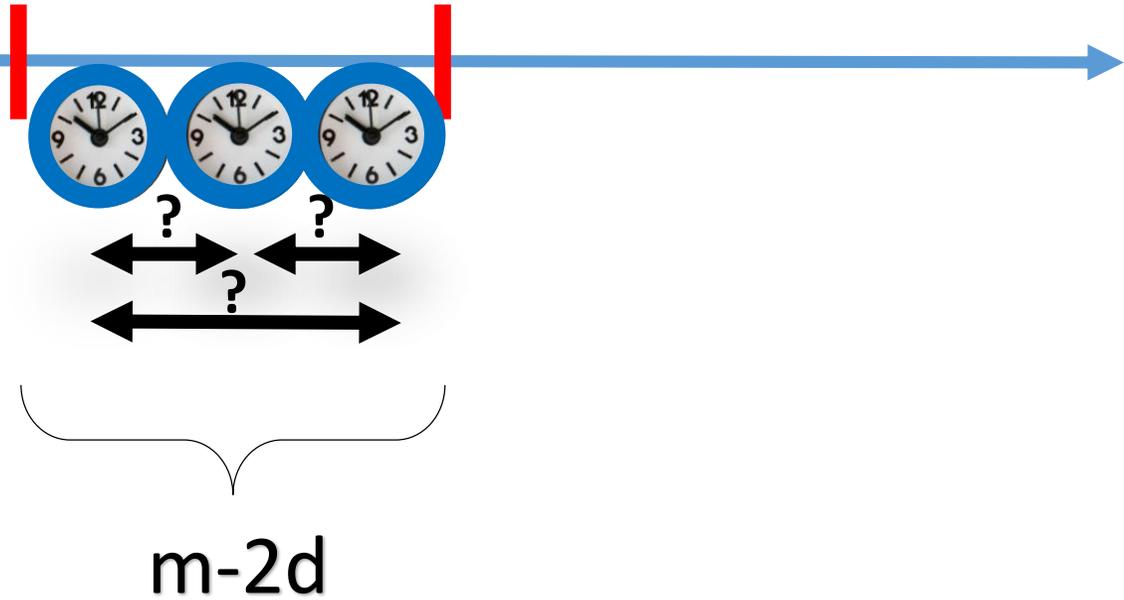
- Query  $m$  (10s of) servers at random
- Order time samples from low to high
- Remove the  $d$  lowest and highest time samples



# Chronos' Time-Update Algorithm: Informal

Check:

If (the remaining samples are close)



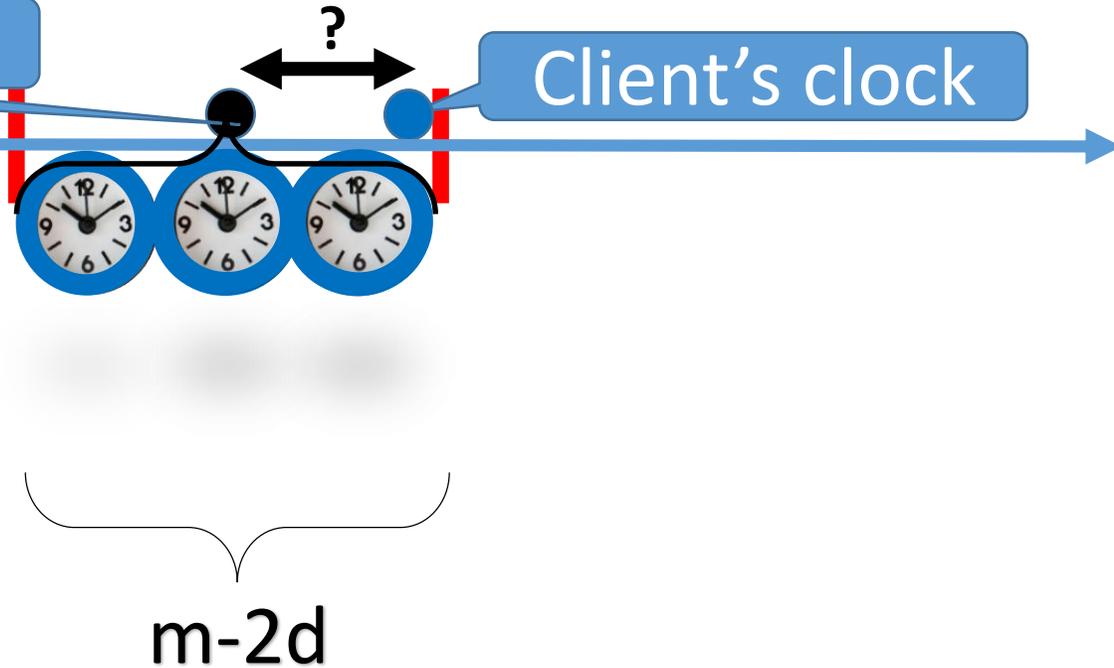
# Chronos' Time-Update Algorithm: Informal

Remaining samples' average

Client's clock

Check:

**If** (the remaining samples are close)  
**and** (average time close to local time)



# Chronos' Time-Update Algorithm: Informal

Remaining samples' average

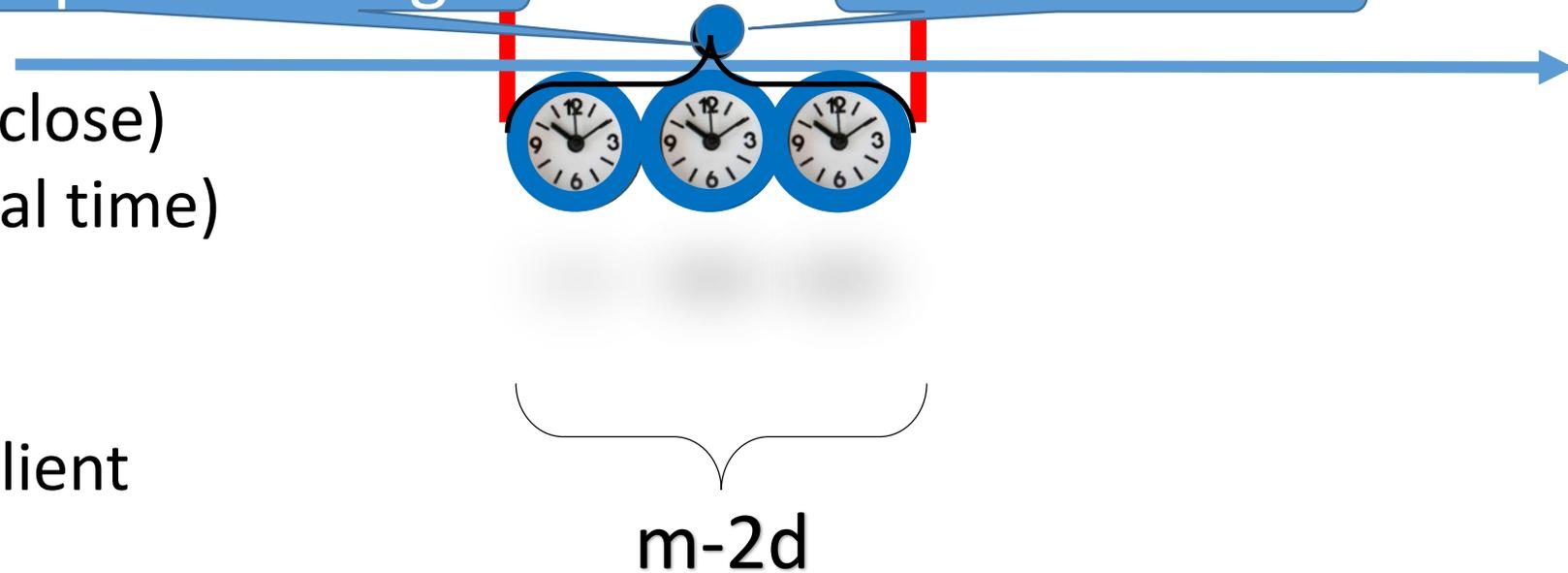
Client's clock

Check:

If (the remaining samples are close)  
and (average time close to local time)

• Then:

- Use average as the new client time



# Chronos' Time-Update Algorithm: Informal

Remaining samples' average

Client's clock

Check:

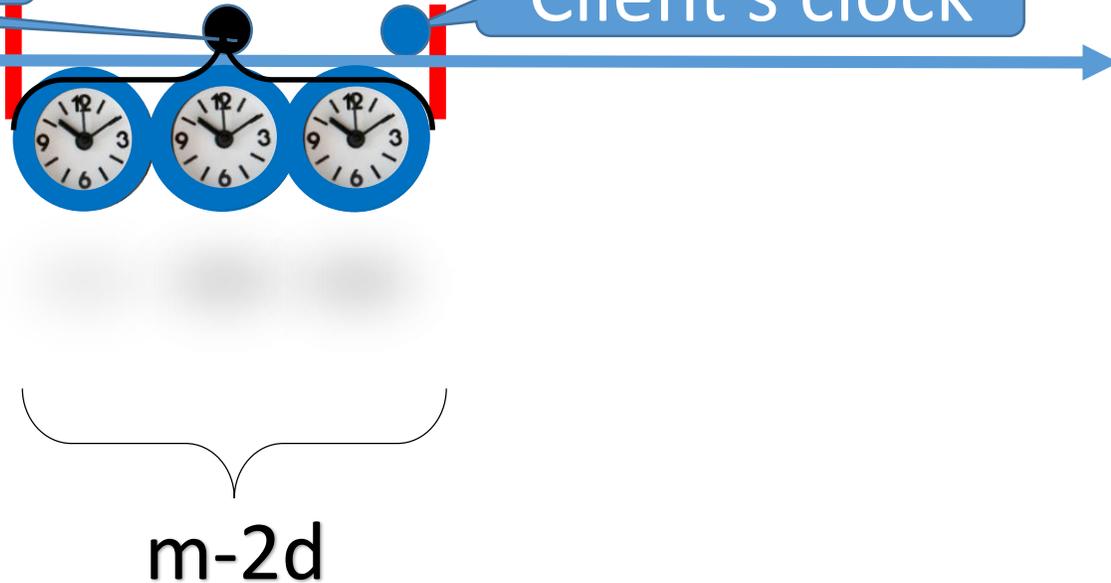
**If** (the remaining samples are close)  
**and** (average time close to local time)

• **Then:**

- Use average as the new client time

• **Else**

- Resample



# Chronos' Time-Update Algorithm: Informal

Check:

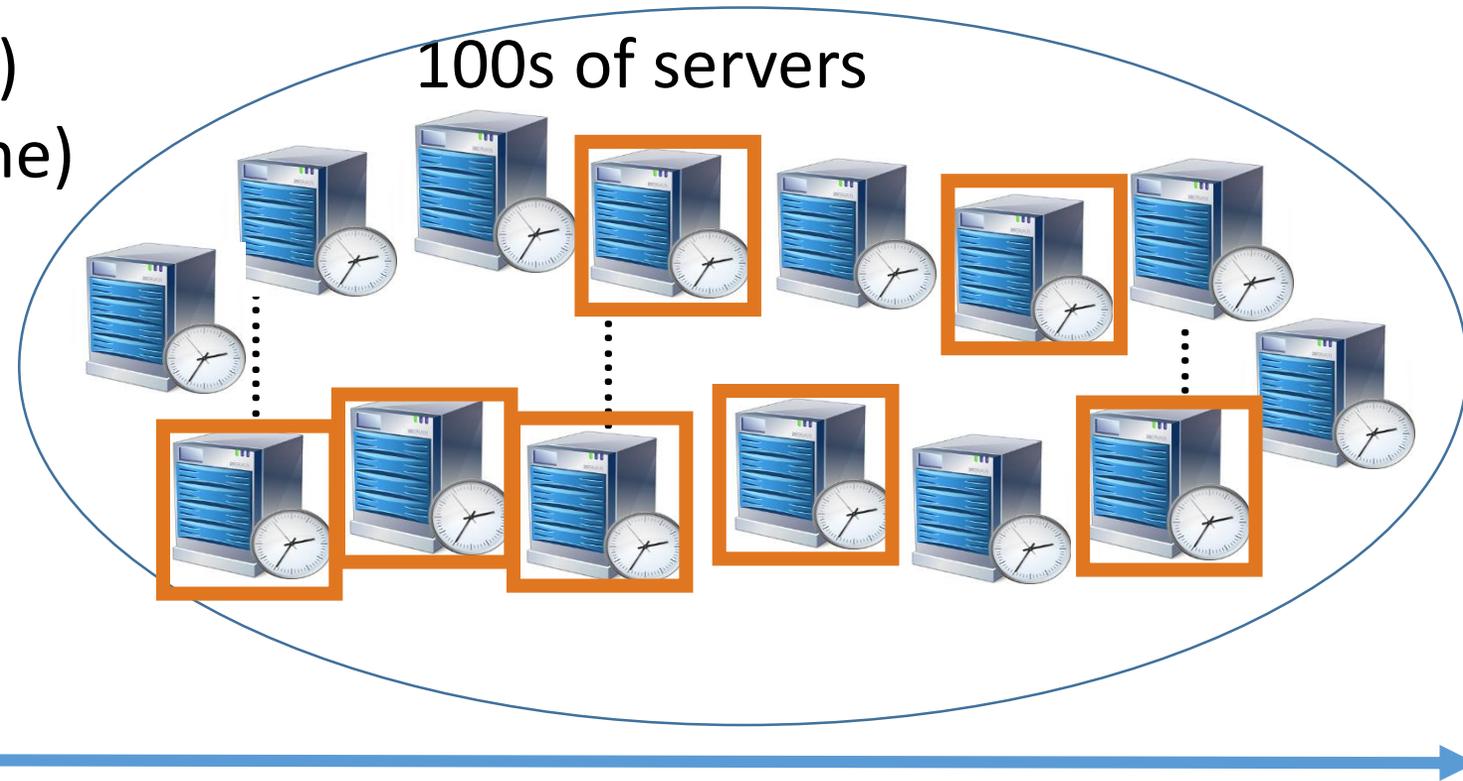
**If** (the remaining samples are close)  
**and** (average time close to local time)

• **Then:**

- Use average as the new client time

• **Else**

- Resample



# Chronos' Time-Update Algorithm: Informal

Check:

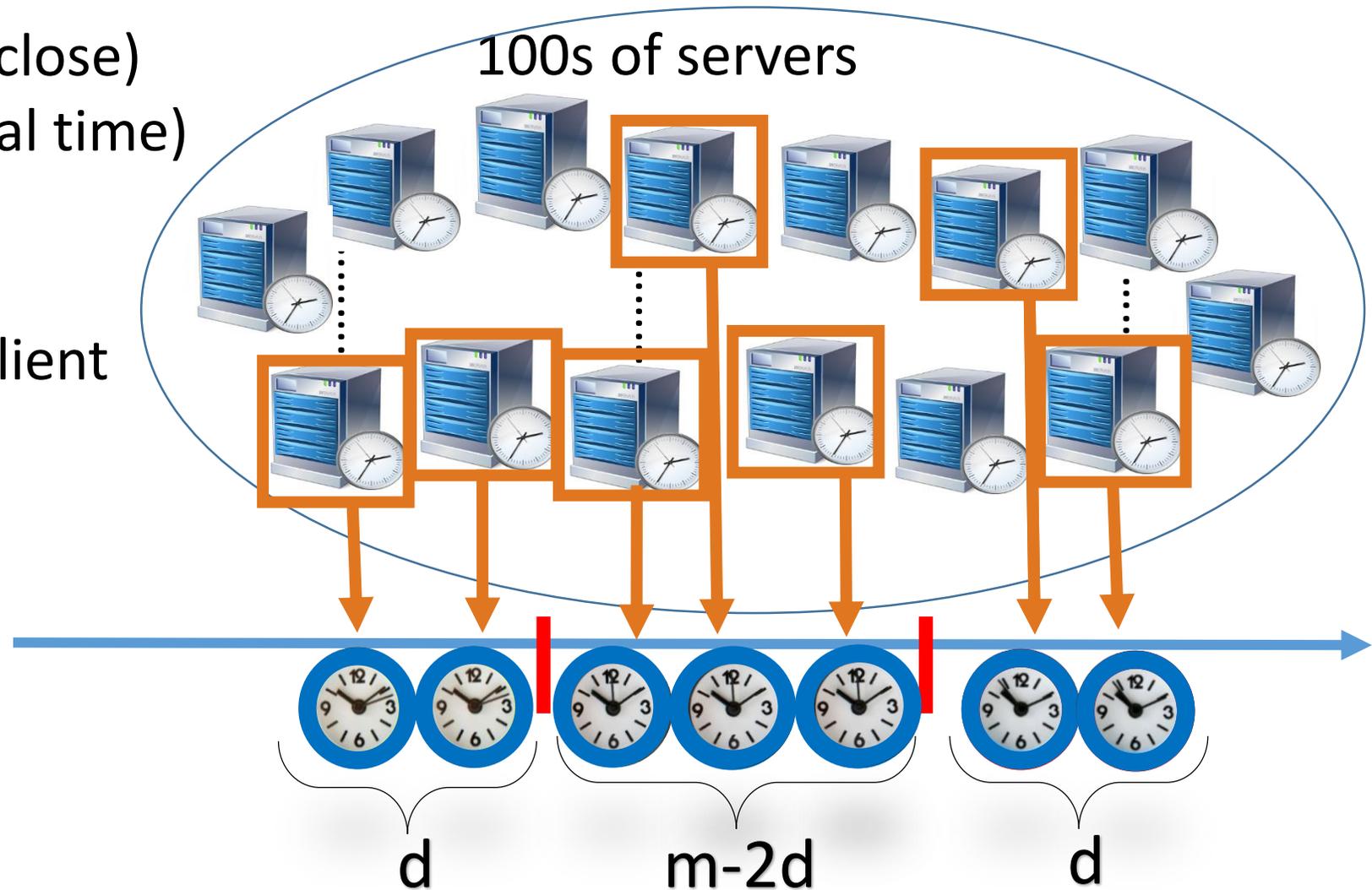
If (the remaining samples are close)  
and (average time close to local time)

• Then:

- Use average as the new client time

• Else

- Resample



# Chronos' Time-Update Algorithm: Informal

Check:

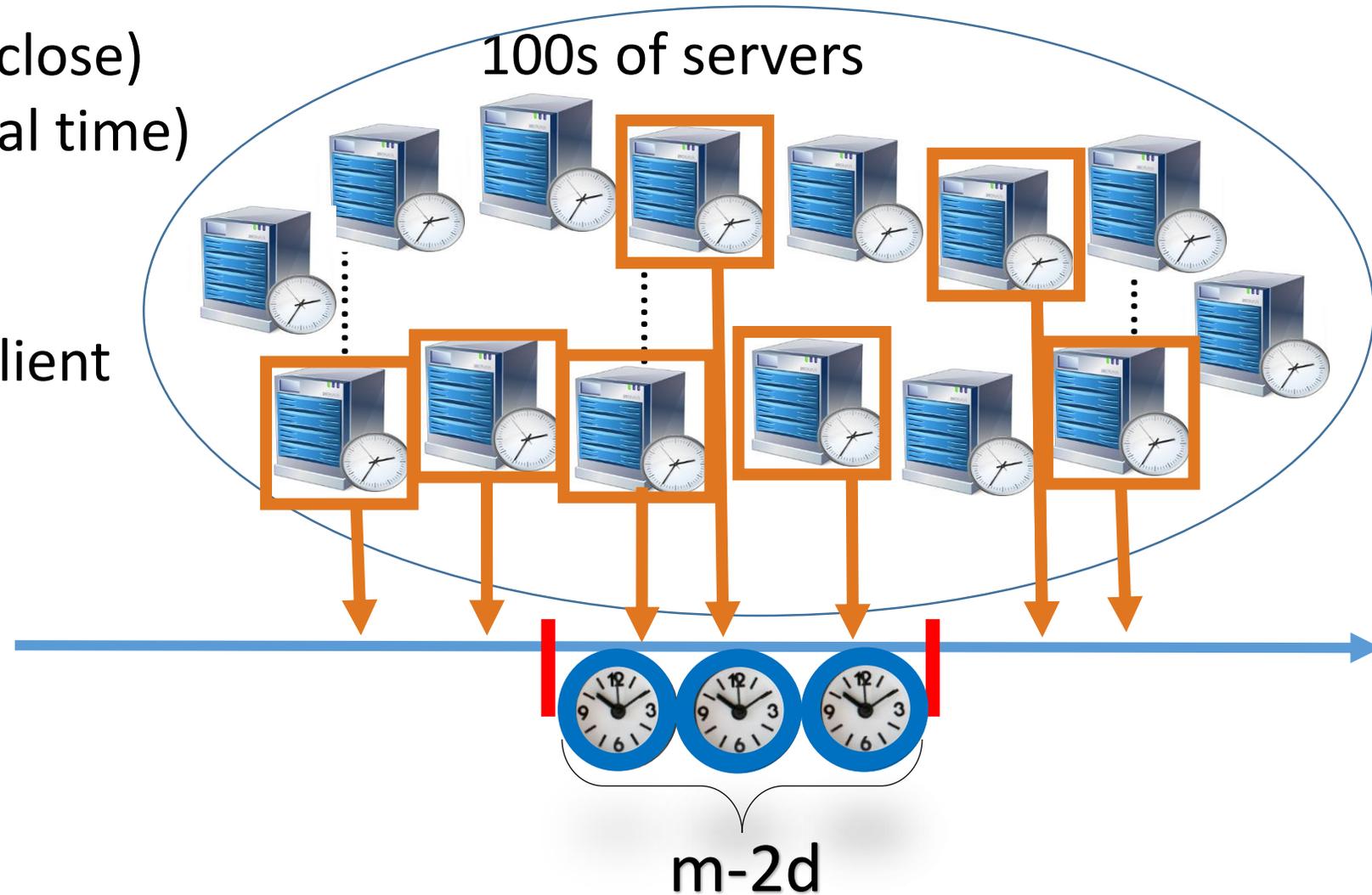
**If** (the remaining samples are close)  
**and** (average time close to local time)

• **Then:**

- Use average as the new client time

• **Else**

- Resample

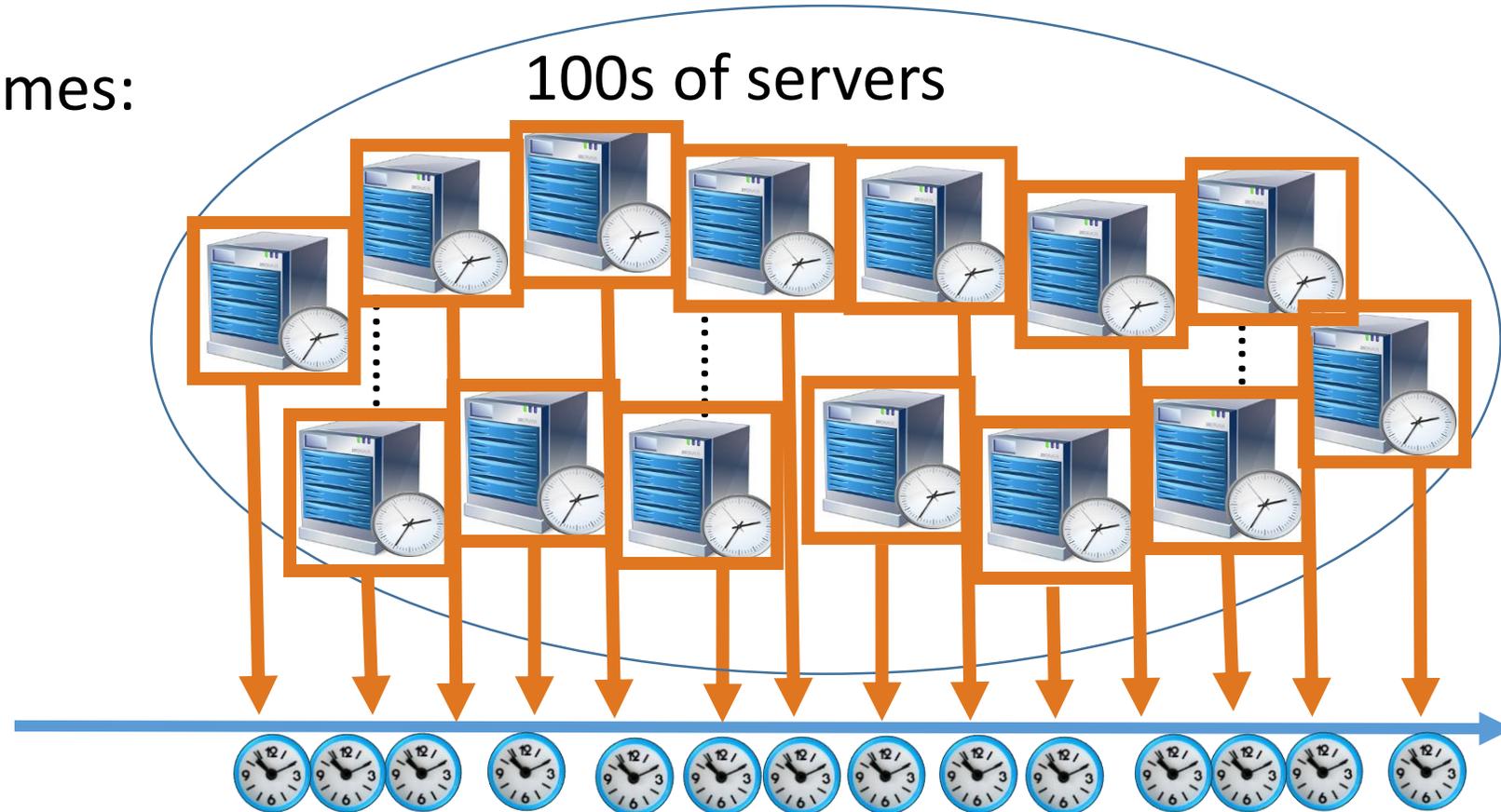


# Chronos' Time-Update Algorithm: Informal

if check & resample failed k times:

\\ **panic mode**

- Sample all servers

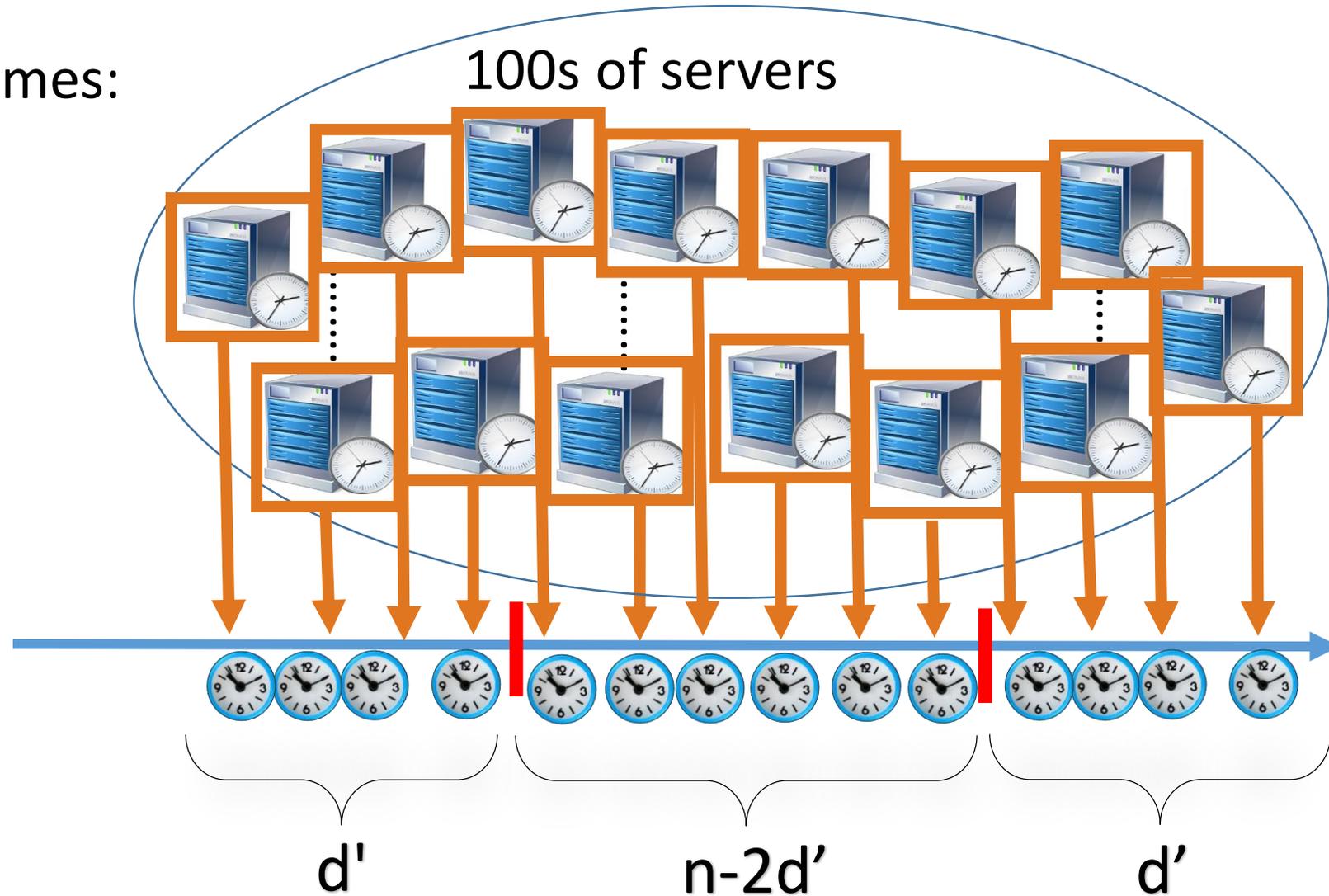


# Chronos' Time-Update Algorithm: Informal

if check & resample failed k times:

\\ **panic mode**

- Sample all servers
- Drop outliers

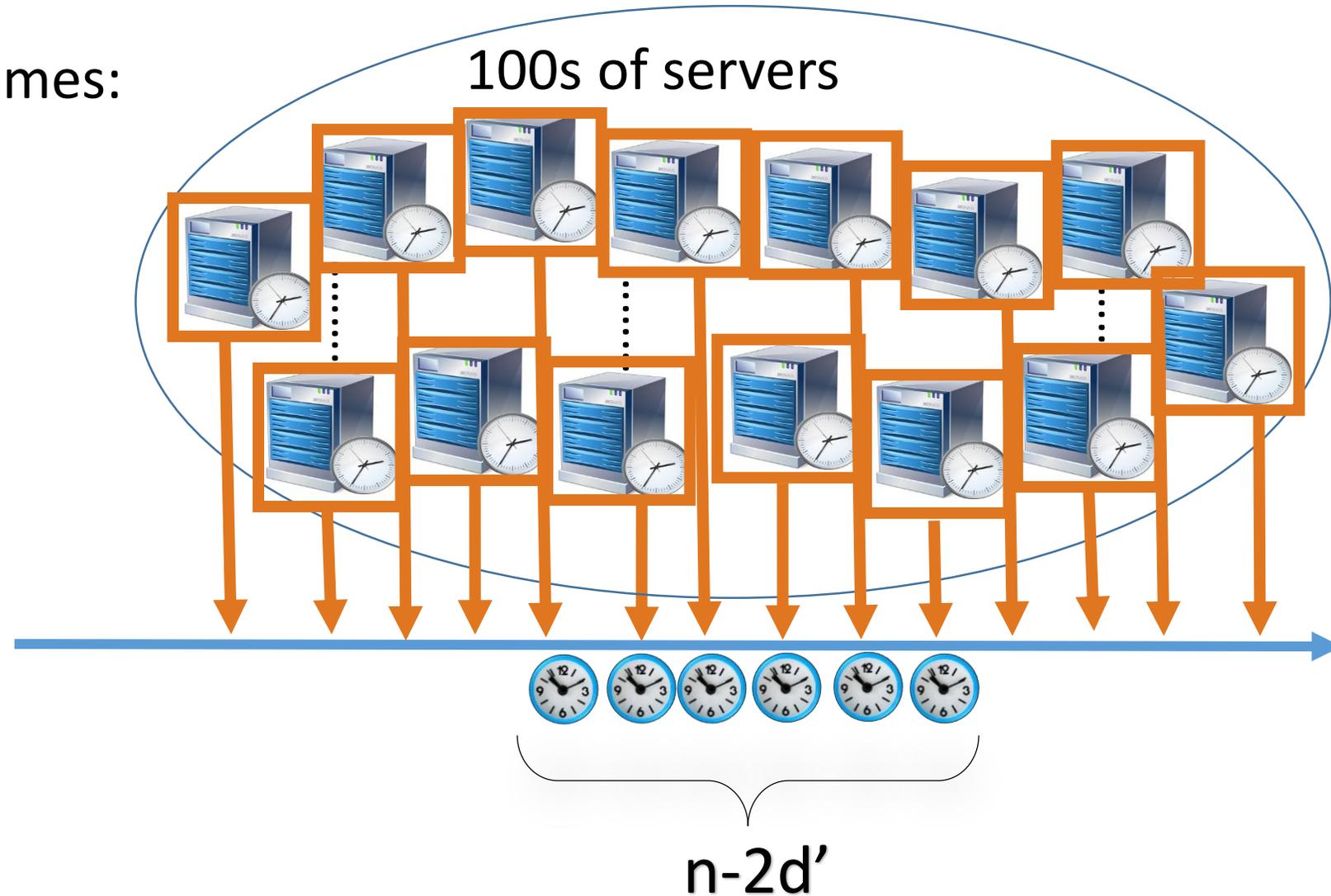


# Chronos' Time-Update Algorithm: Informal

if check & resample failed k times:

**panic mode**

- Sample all servers
- Drop outliers

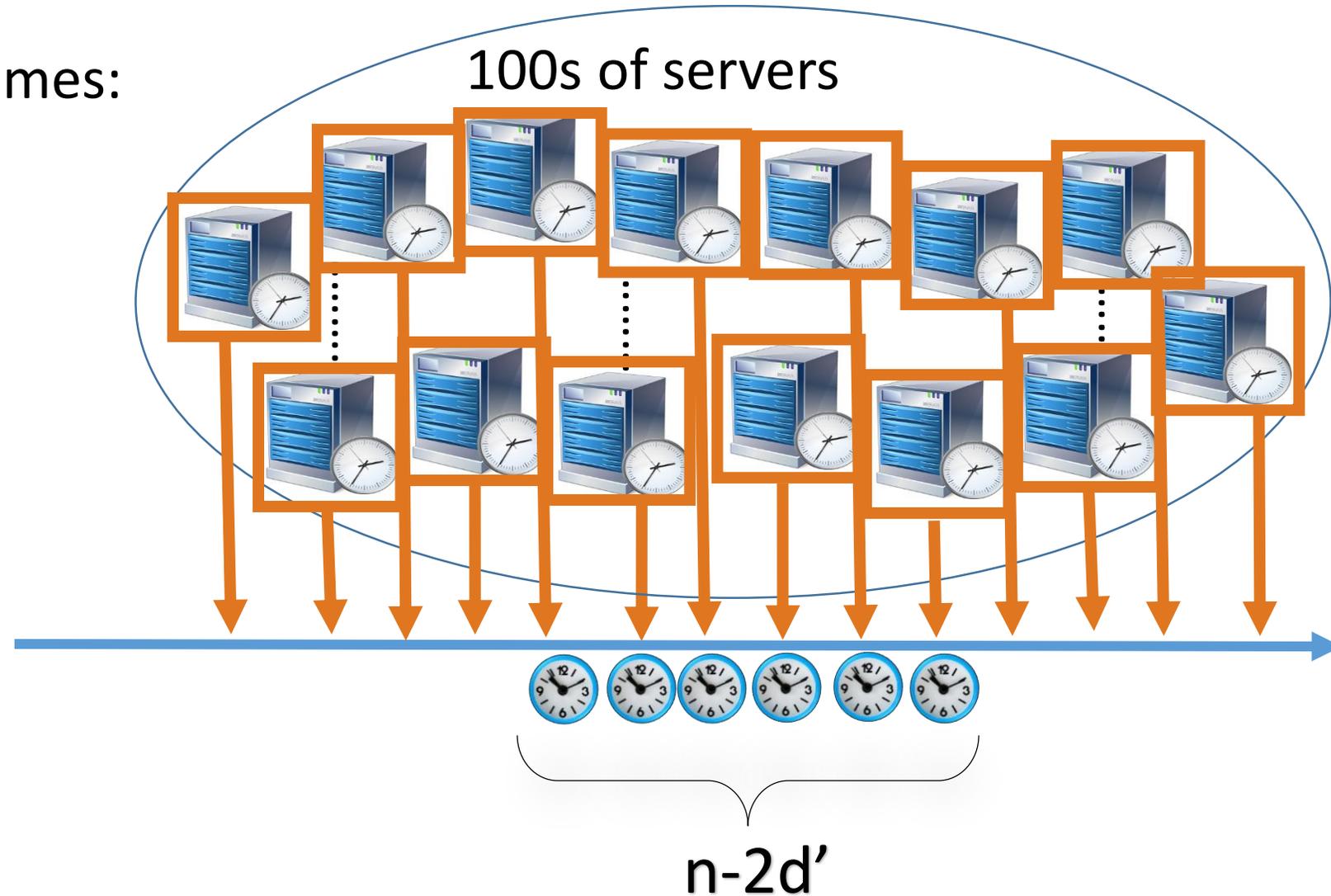


# Chronos' Time-Update Algorithm: Informal

if check & resample failed k times:

**panic mode**

- Sample all servers
- Drop outliers
- Use average as new client time

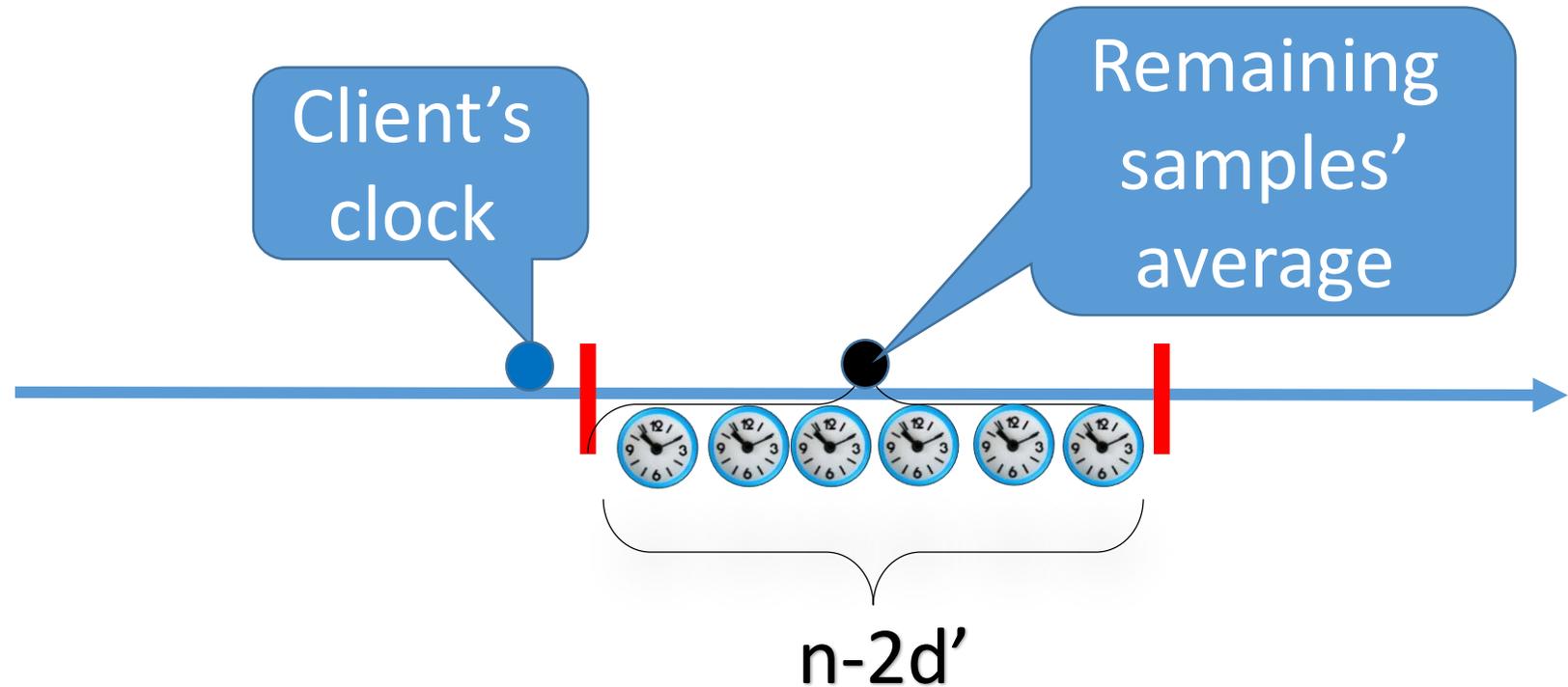


# Chronos' Time-Update Algorithm: Informal

if check & resample failed k times:

**panic mode**

- Sample all servers
- Drop outliers
- Use average as new client time

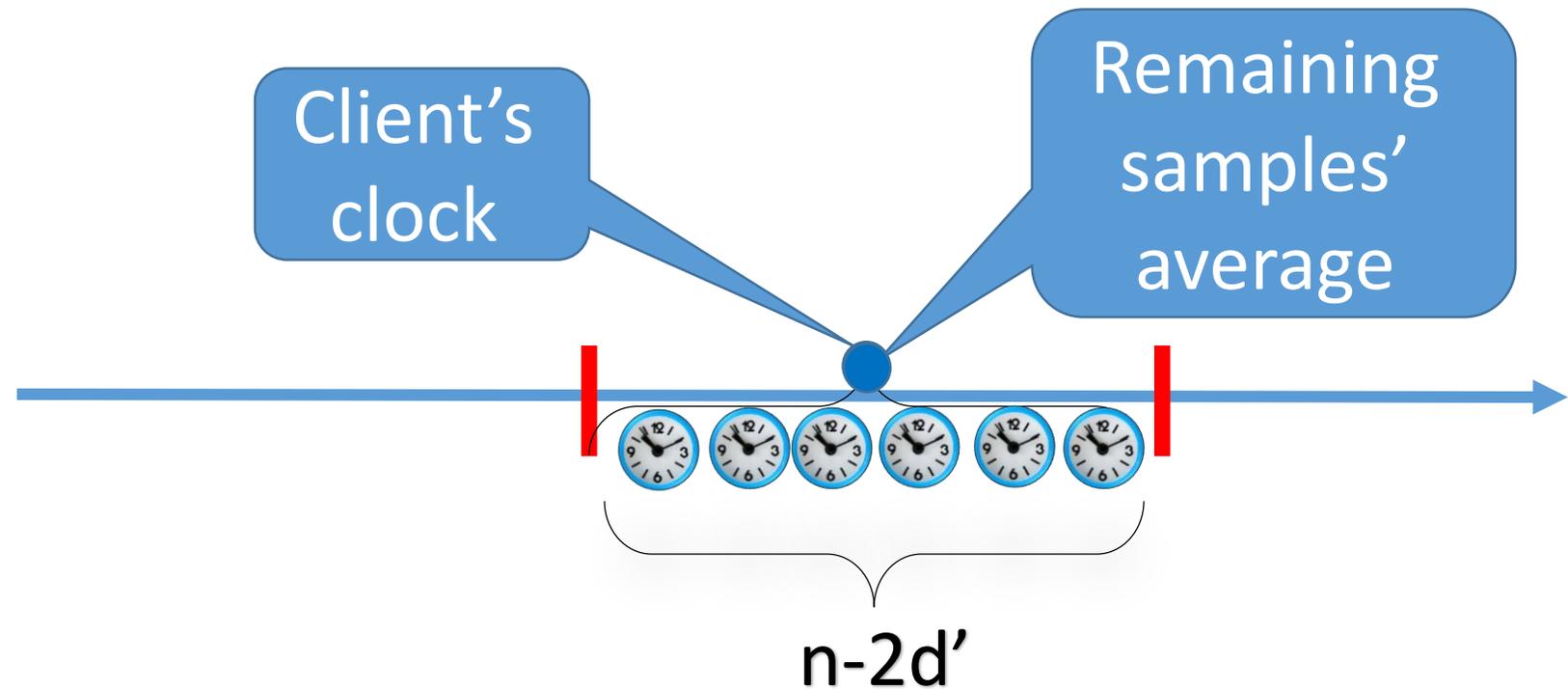


# Chronos' Time-Update Algorithm: Informal

if check & resample failed k times:

\\ **panic mode**

- Sample all servers
- Drop outliers
- Use average as new client time



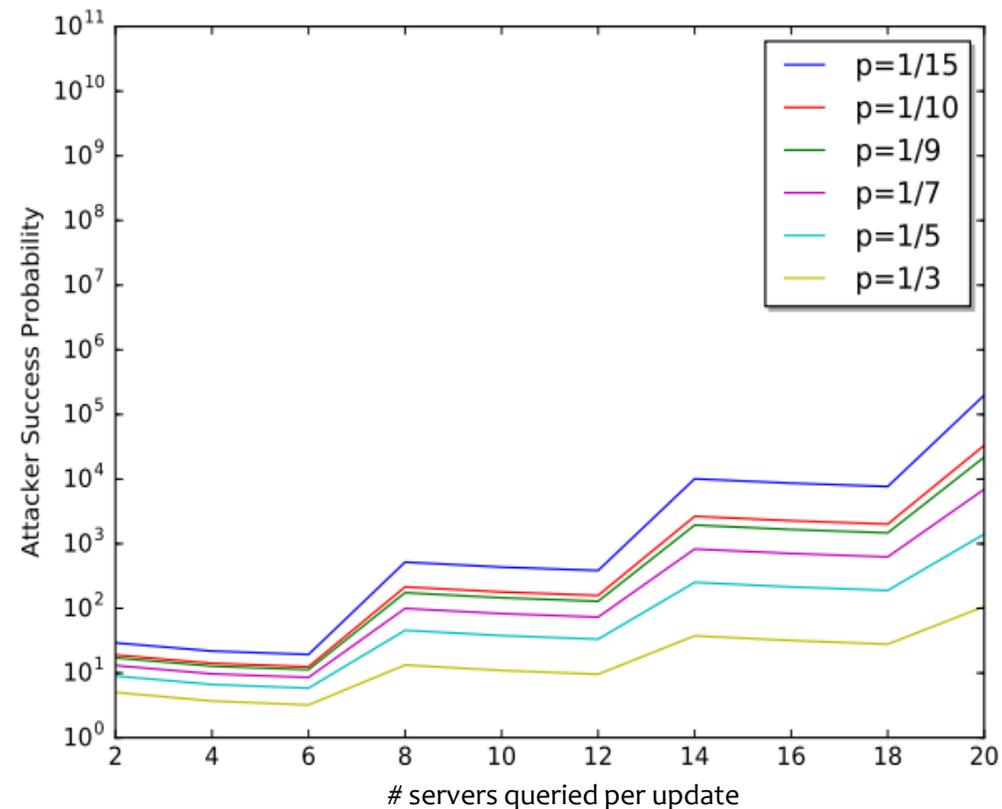
# Security Guarantees

Shifting time at a Chronos client by at least **100ms** from the UTC will take the attacker at least **22 years** in expectation

- ... when considering the following parameters:
  - Server pool of 500 servers, of whom 1/7 are controlled by an attacker
  - 15 servers queried once an hour
  - Good samples are within 25ms from UTC ( $\omega=25$ )
- These parameters are derived from experiments we performed on AWS servers in Europe and the US

# Chronos vs. Current NTP Clients

- Consider a pool of 500 servers, a  $p$ -fraction of which is controlled by an attacker.
- We compute the attacker's probability of successfully shifting the client's clock
  - for traditional NTP client
  - for Chronos NTP client
- We plot the ratio between these probabilities



# Security Guarantees: Intuition

Scenario 1:  $\#(\text{👼}) \leq d$      $\#(\text{👿}) \geq m-d$

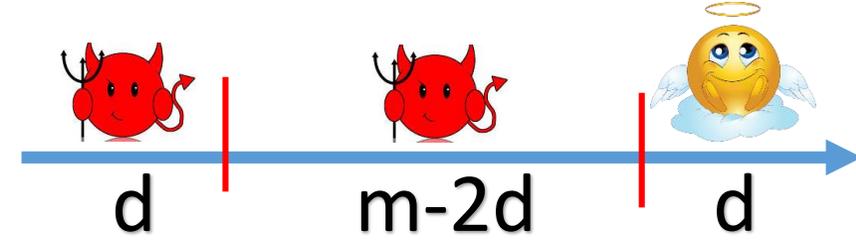
# Security Guarantees: Intuition

**Scenario 1:**  $\#(\text{👼}) \leq d$      $\#(\text{👿}) \geq m-d$

- Optimal attack strategy:

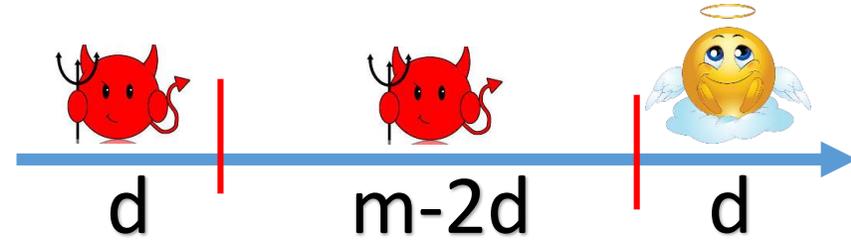
All malicious samples are lower than all good samples

(Or, all malicious samples are higher than all good samples)



# Security Guarantees: Intuition

**Scenario 1:**  $\#(\text{👼}) \leq d$      $\#(\text{👿}) \geq m-d$



- Optimal attack strategy:

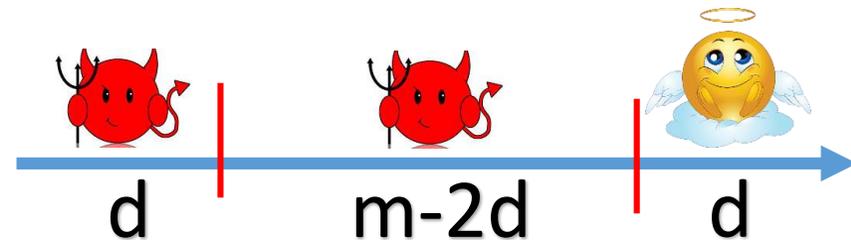
All malicious samples are lower than all good samples

(Or, all malicious samples are higher than all good samples)

- **Chronos enforces an upper bound of  $4\omega$  on the permissible shift from the local clock** (otherwise the server pool is re-sampled)

# Security Guarantees: Intuition

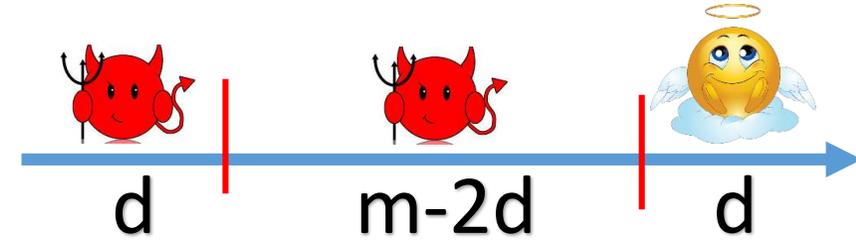
**Scenario 1:**  $\#(\text{👼}) \leq d$      $\#(\text{👿}) \geq m-d$



- Optimal attack strategy:  
All malicious samples are lower than all good samples  
(Or, all malicious samples are higher than all good samples)
- **Chronos enforces an upper bound of  $4\omega$  on the permissible shift from the local clock** (otherwise the server pool is re-sampled)
- **The probability that  $\#(\text{👿}) \geq m-d$  is extremely low** (see paper for detailed analysis)  
The probability of repeated shift is negligible.

# Security Guarantees: Intuition

**Scenario 1:**  $\#(\text{👼}) \leq d$      $\#(\text{👿}) \geq m-d$



- Optimal attack strategy:  
All malicious samples are lower than all good samples  
(Or, all malicious samples are higher than all good samples)
- **Chronos enforces an upper bound of  $4\omega$  on the permissible shift from the local clock** (otherwise the server pool is re-sampled)
- **The probability that  $\#(\text{👿}) \geq m-d$  is extremely low** (see paper for detailed analysis)  
The probability of repeated shift is negligible.

**Consequently, a significant time shift is practically infeasible**

# Security Guarantees: Intuition

Scenario 2:  $\#(\text{👼}) > d$      $\#(\text{👿}) < m-d$

# Security Guarantees: Intuition

**Scenario 2:**  $\#(\text{👼}) > d$      $\#(\text{👿}) < m-d$

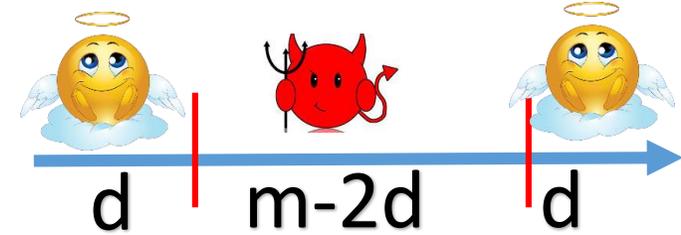
• **Option 1:** Only malicious samples remain

➤ Assumption: every good sample at most  $\omega$ -far from UTC

➤ At least one good sample on each side

→ All remaining samples are between two good samples

→ All remaining samples are at most  $\omega$ -away from UTC



# Security Guarantees: Intuition

**Scenario 2:**  $\#(\text{👼}) > d$      $\#(\text{👿}) < m-d$

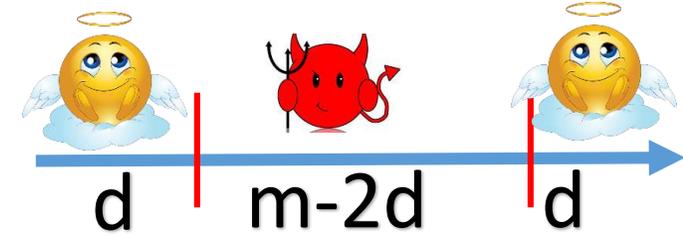
• **Option I:** Only malicious samples remain

➤ Assumption: every good sample at most  $\omega$ -far from UTC

➤ At least one good sample on each side

→ All remaining samples are between two good samples

→ All remaining samples are at most  $\omega$ -away from UTC

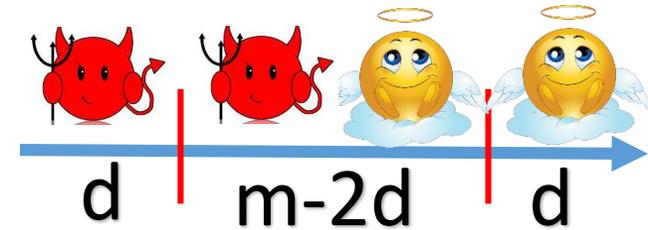


• **Option II:** At least one good sample remains

➤ Enforced: Remaining samples within the same  $2\omega$ -interval

➤ Remaining malicious samples are within  $2\omega$  from a good sample

→ Remaining malicious samples are at most  $3\omega$ -away from UTC



# Security Guarantees: Intuition

**Scenario 2:**  $\#(\text{👼}) > d$      $\#(\text{👹}) < m-d$

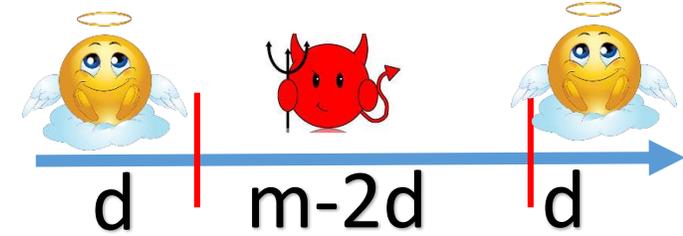
• **Option I:** Only malicious samples remain

➤ Assumption: every good sample at most  $\omega$ -far from UTC

➤ At least one good sample on each side

→ All remaining samples are between two good samples

→ All remaining samples are at most  $\omega$ -away from UTC

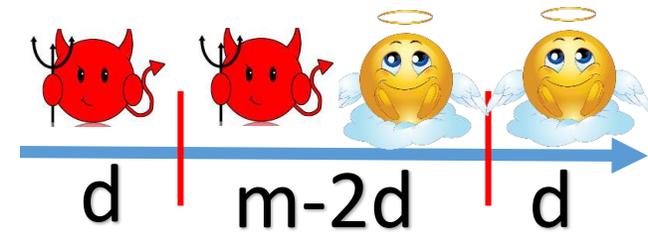


• **Option II:** At least one good sample remains

➤ Enforced: Remaining samples within the same  $2\omega$ -interval

➤ Remaining malicious samples are within  $2\omega$  from a good sample

→ Remaining malicious samples are at most  $3\omega$ -away from UTC



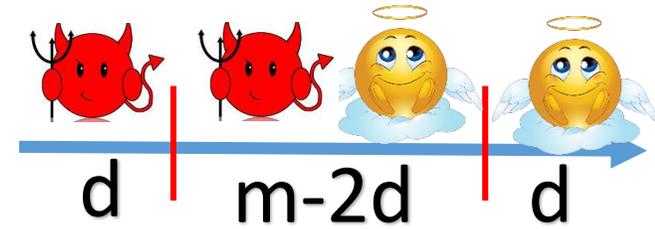
**Hence, these attack strategies are ineffective**

# Can Chronos be exploited for DoS attacks?

- Chronos repeatedly enters Panic Mode.
- Optimal attack strategy requires that attacker repeatedly succeed in accomplishing  $\#(\text{👿}) > d$      $\#(\text{👼}) < m-d$ 
  - At least one malicious sample remains
  - Malicious sample violates condition that all remaining samples be clustered
  - This leads to resampling (until Panic Threshold is exceeded).

# Can Chronos be exploited for DoS attacks?

- Chronos repeatedly enters Panic Mode.

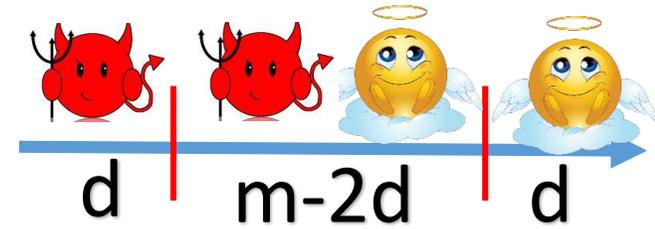


- Optimal attack strategy requires that attacker repeatedly succeed in accomplishing  $\#(\text{devil}) > d$   $\#(\text{angel}) < m-d$

- At least one malicious sample remains
- Malicious sample violates condition that all remaining samples be clustered
- This leads to resampling (until Panic Threshold is exceeded).

# Can Chronos be exploited for DoS attacks?

- Chronos repeatedly enters Panic Mode.



- Optimal attack strategy requires that attacker repeatedly succeed in accomplishing  $\#(\text{devil}) > d$   $\#(\text{angel}) < m-d$ 
  - At least one malicious sample remains
  - Malicious sample violates condition that all remaining samples be clustered
  - This leads to resampling (until Panic Threshold is exceeded).

Even for low Panic Threshold ( $k=3$ ), probability of success is negligible (will take attacker decades to force Panic Mode)

# Observations and Extensions

- When the pool of available servers is small (say, 3), using Chronos's sampling scheme on the entire server pool ( $n=m$ ), yields meaningful deterministic security guarantees.
- Important implications for PTP security

# Chronos Vs. Current NTP Architecture

	Current NTP	Chronos
preprocessing		Collect NTP server addresses to form a (large) server pool
Poll process	Send queries to several NTP servers from an externally provided list	M (e.g., tens) servers randomly chosen from the (large) pool
Selection process	<ul style="list-style-type: none"><li>• Apply Marzullo's algorithm to identify a majority of samples that (approximately) agree on the time</li><li>• Take average of remaining samples</li><li>• If new time far from current time → update. Else → nothing</li></ul>	<ul style="list-style-type: none"><li>• Remove d lowest and d highest time samples</li><li>• Verify remaining samples are clustered, else → resample</li><li>• Take average of remaining samples</li><li>• If new time close to current time → update. Else → resample</li></ul>

# Conclusion

- NTP is very vulnerable to time-shifting attacks by MitM attackers
  - Not designed to protect against strategic man-in-the-middle attacks
  - Attacker who controls a few servers/sessions can shift client's time
- We presented the **Chronos NTP client**
  - Provable security in the face of powerful and sophisticated MitM attackers
  - Backwards-compatibility with legacy NTP (software changes to client only)
  - Low computational and communication overhead

# Future Research

- Tighter security bounds?
- Weighing servers according to reputation?
- Benefits of server-side changes?
- Extensions to other time-synchronization protocols (e.g., PTP)?

# Thank You



See full paper (@NDSS'18):

[http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2018/02/ndss2018\\_02A-2\\_Deutsch\\_paper.pdf](http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2018/02/ndss2018_02A-2_Deutsch_paper.pdf)