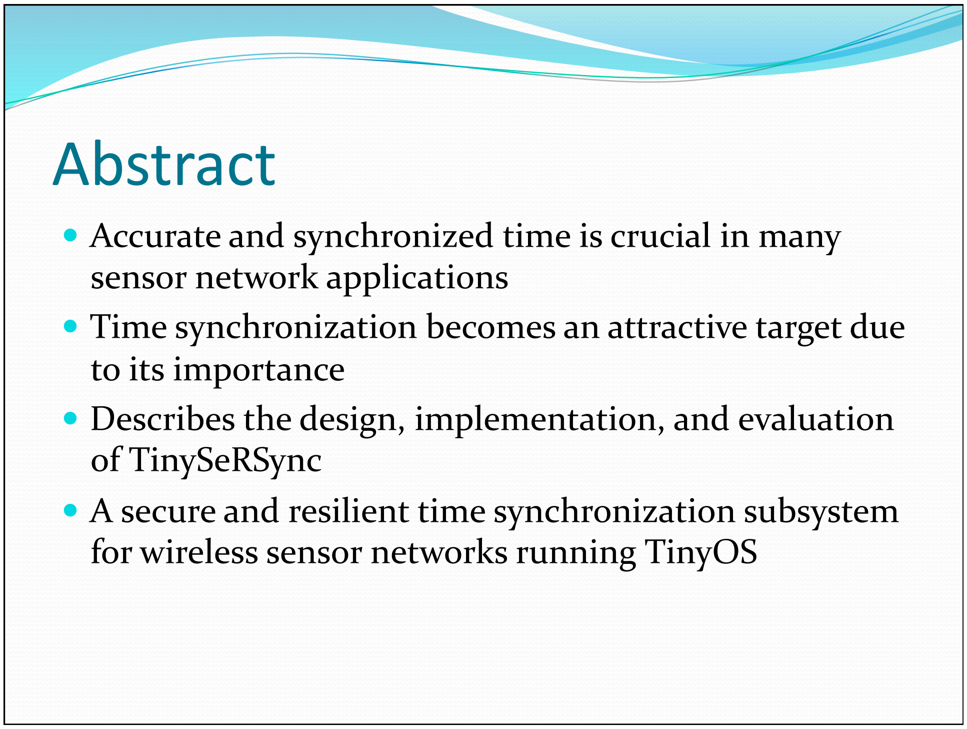
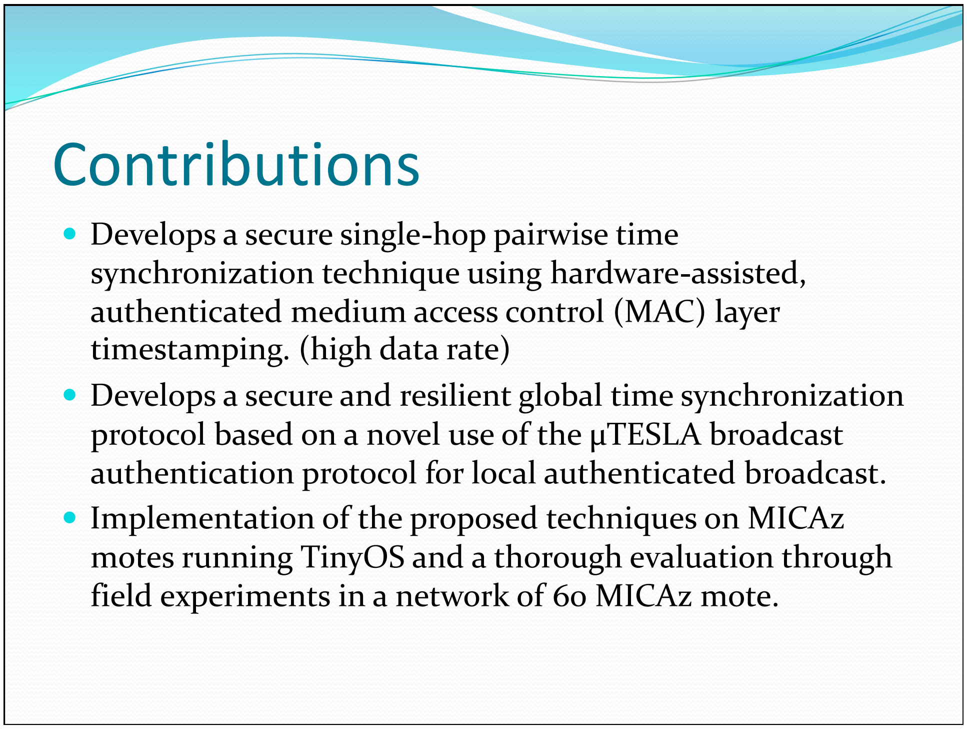
TinySeRSync: Secure and Resilient Time Synchronization in Wireless Sensor Networks

Kun Sun, Peng Ning
Cliff Wang
An Liu, Yuzheng Zhou



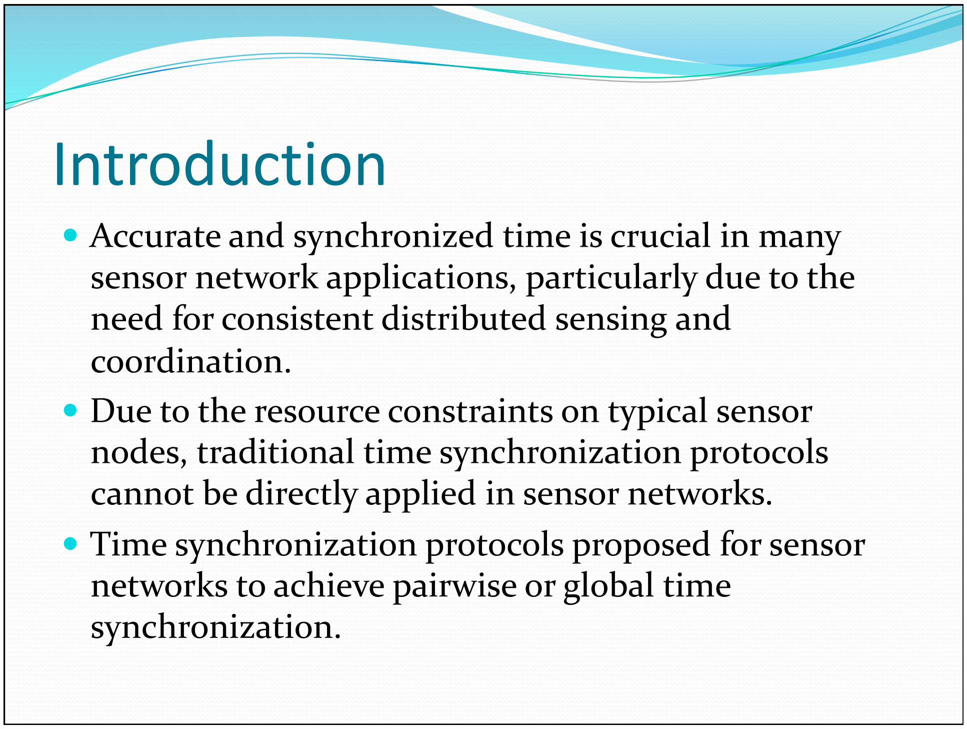
Abstract

- Accurate and synchronized time is crucial in many sensor network applications
- Time synchronization becomes an attractive target due to its importance
- Describes the design, implementation, and evaluation of TinySeRSync
- A secure and resilient time synchronization subsystem for wireless sensor networks running TinyOS



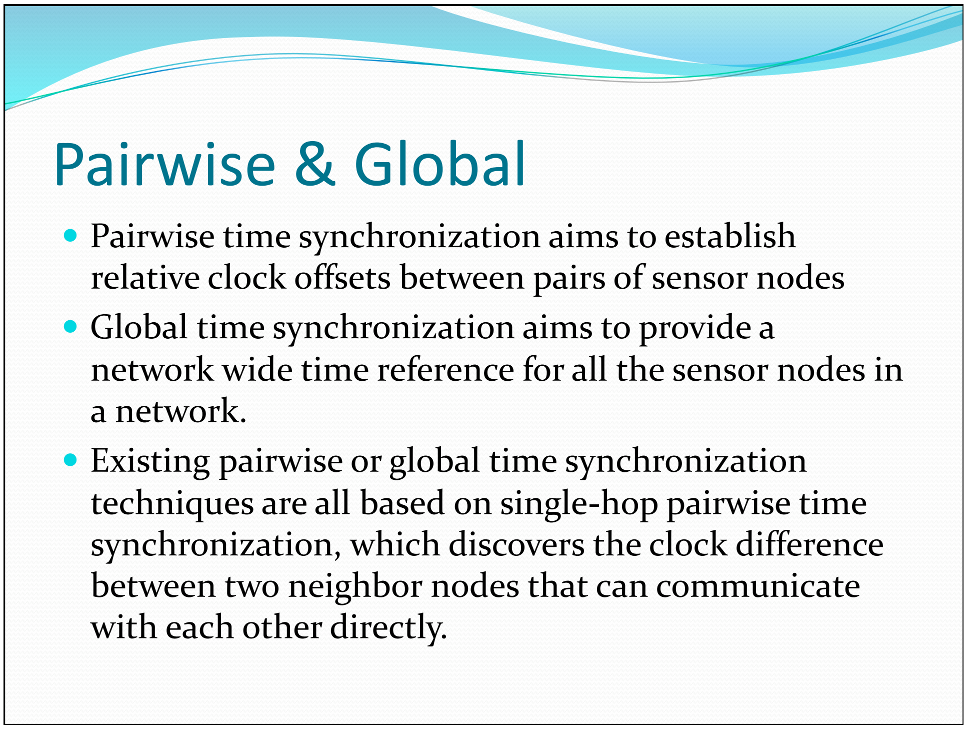
Contributions

- Develops a secure single-hop pairwise time synchronization technique using hardware-assisted, authenticated medium access control (MAC) layer timestamping. (high data rate)
- Develops a secure and resilient global time synchronization protocol based on a novel use of the μTESLA broadcast authentication protocol for local authenticated broadcast.
- Implementation of the proposed techniques on MICAz motes running TinyOS and a thorough evaluation through field experiments in a network of 60 MICAz mote.



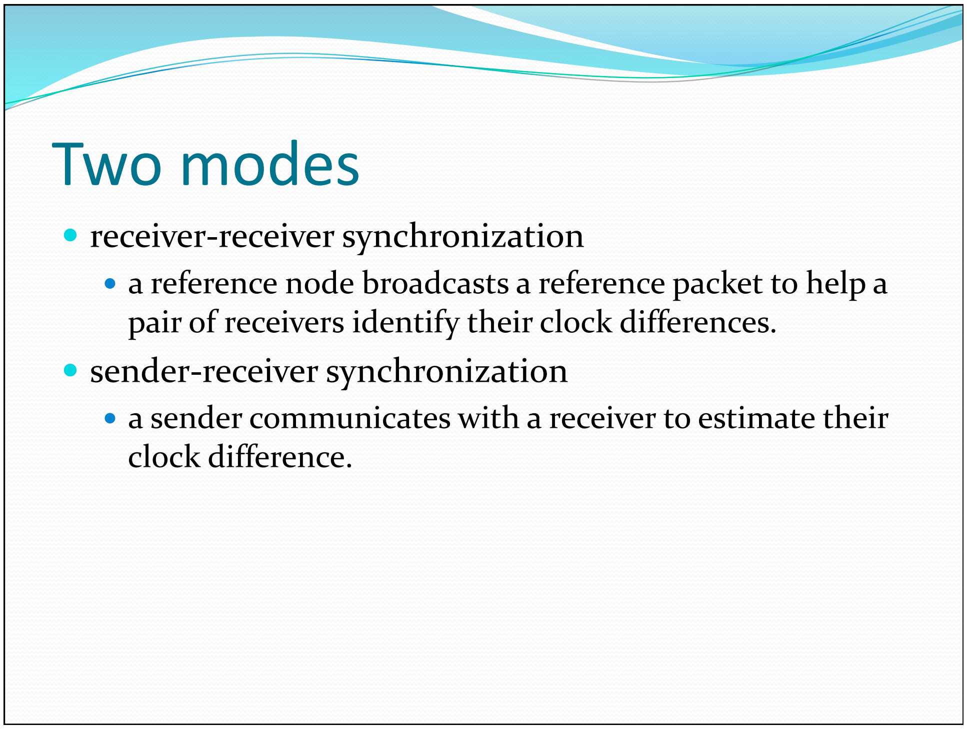
Introduction

- Accurate and synchronized time is crucial in many sensor network applications, particularly due to the need for consistent distributed sensing and coordination.
- Due to the resource constraints on typical sensor nodes, traditional time synchronization protocols cannot be directly applied in sensor networks.
- Time synchronization protocols proposed for sensor networks to achieve pairwise or global time synchronization.



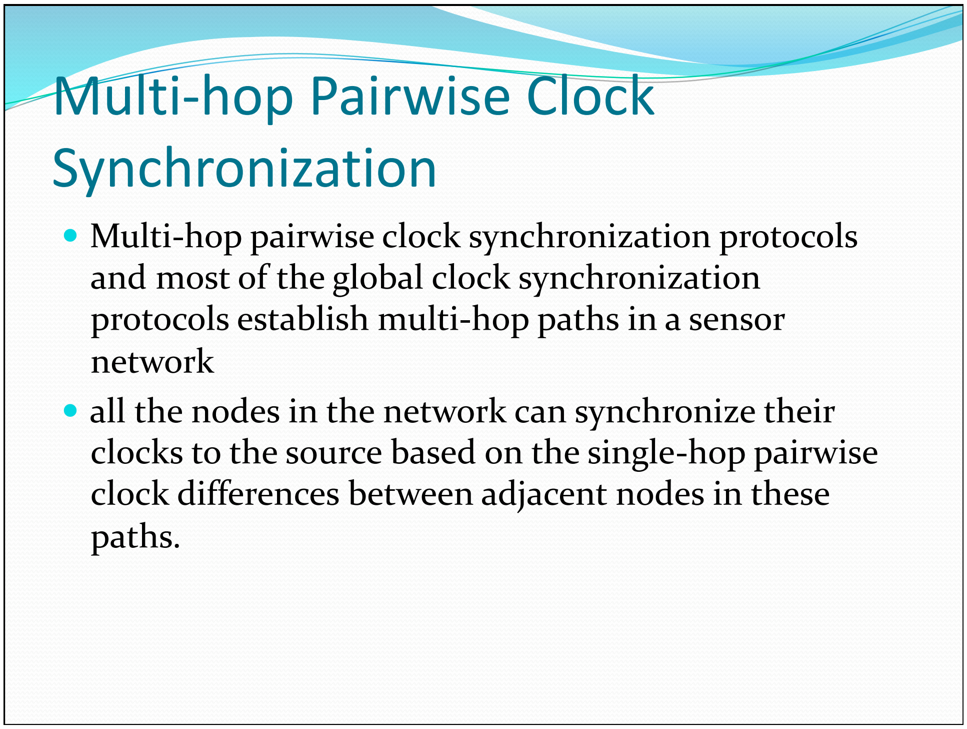
Pairwise & Global

- Pairwise time synchronization aims to establish relative clock offsets between pairs of sensor nodes
- Global time synchronization aims to provide a network wide time reference for all the sensor nodes in a network.
- Existing pairwise or global time synchronization techniques are all based on single-hop pairwise time synchronization, which discovers the clock difference between two neighbor nodes that can communicate with each other directly.



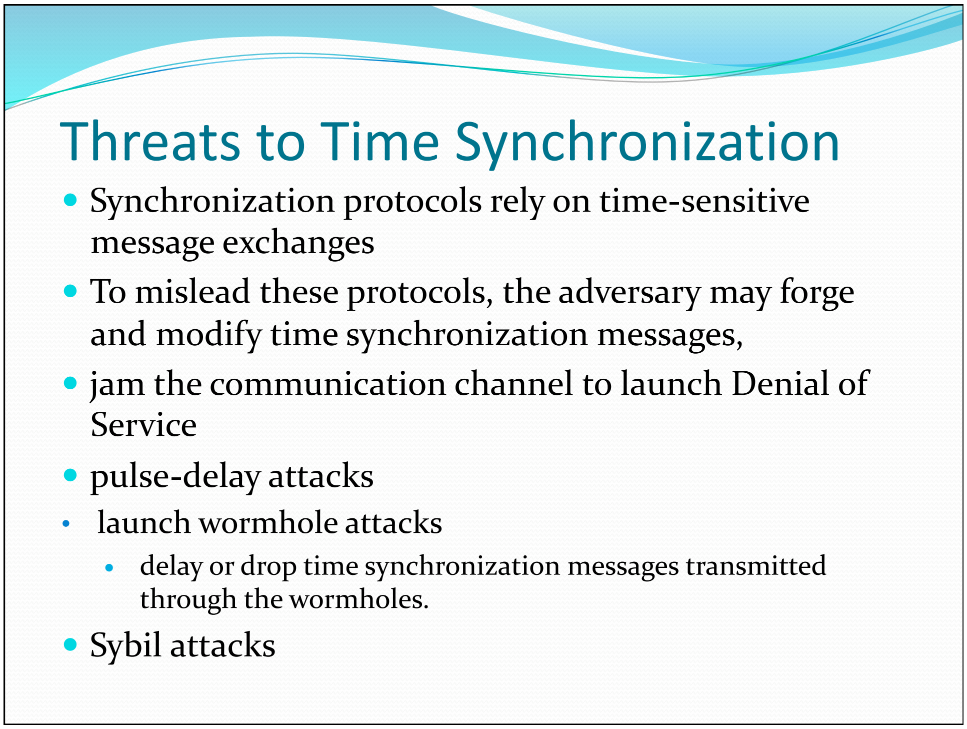
Two modes

- receiver-receiver synchronization
  - a reference node broadcasts a reference packet to help a pair of receivers identify their clock differences.
- sender-receiver synchronization
  - a sender communicates with a receiver to estimate their clock difference.



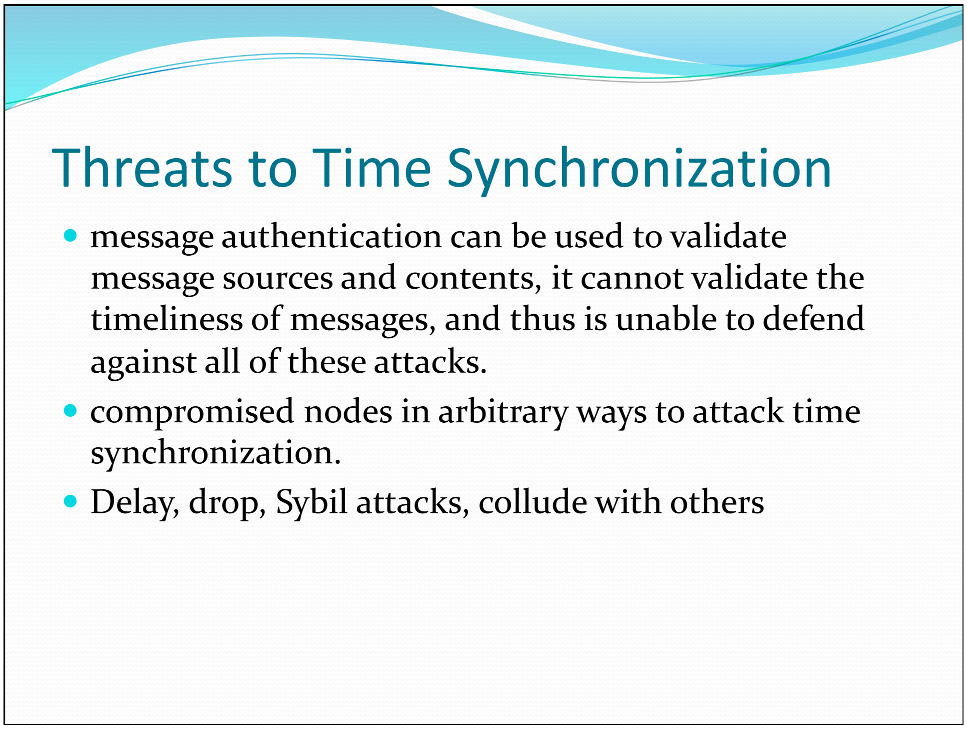
Multi-hop Pairwise Clock Synchronization

- Multi-hop pairwise clock synchronization protocols and most of the global clock synchronization protocols establish multi-hop paths in a sensor network
- all the nodes in the network can synchronize their clocks to the source based on the single-hop pairwise clock differences between adjacent nodes in these paths.



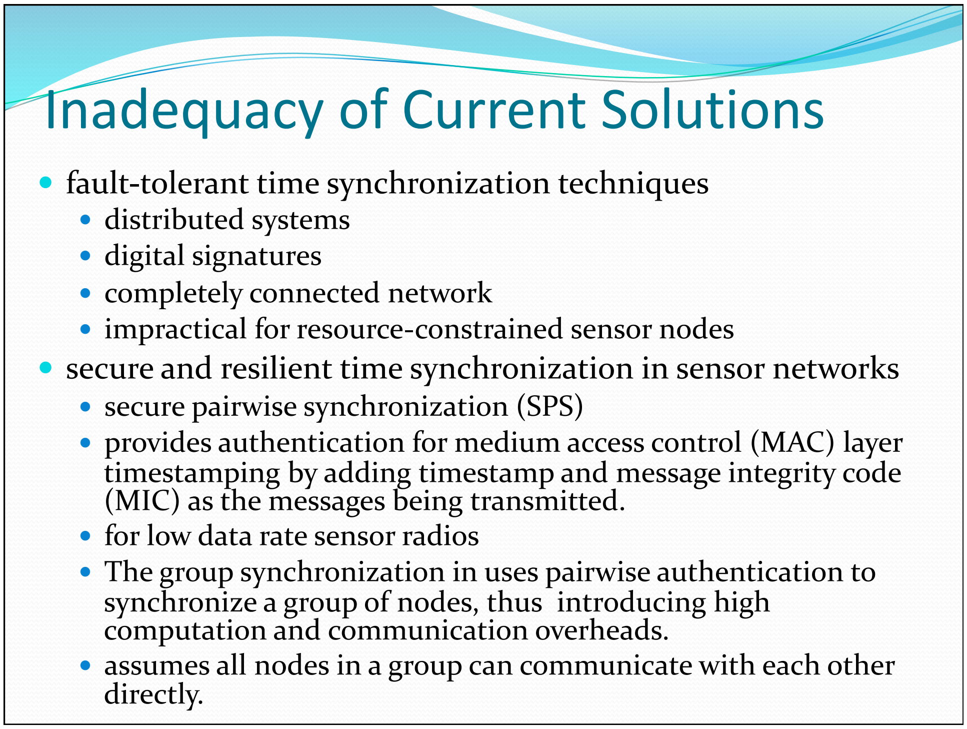
Threats to Time Synchronization

- Synchronization protocols rely on time-sensitive message exchanges
- To mislead these protocols, the adversary may forge and modify time synchronization messages,
- jam the communication channel to launch Denial of Service
- pulse-delay attacks
- launch wormhole attacks
  - delay or drop time synchronization messages transmitted through the wormholes.
- Sybil attacks



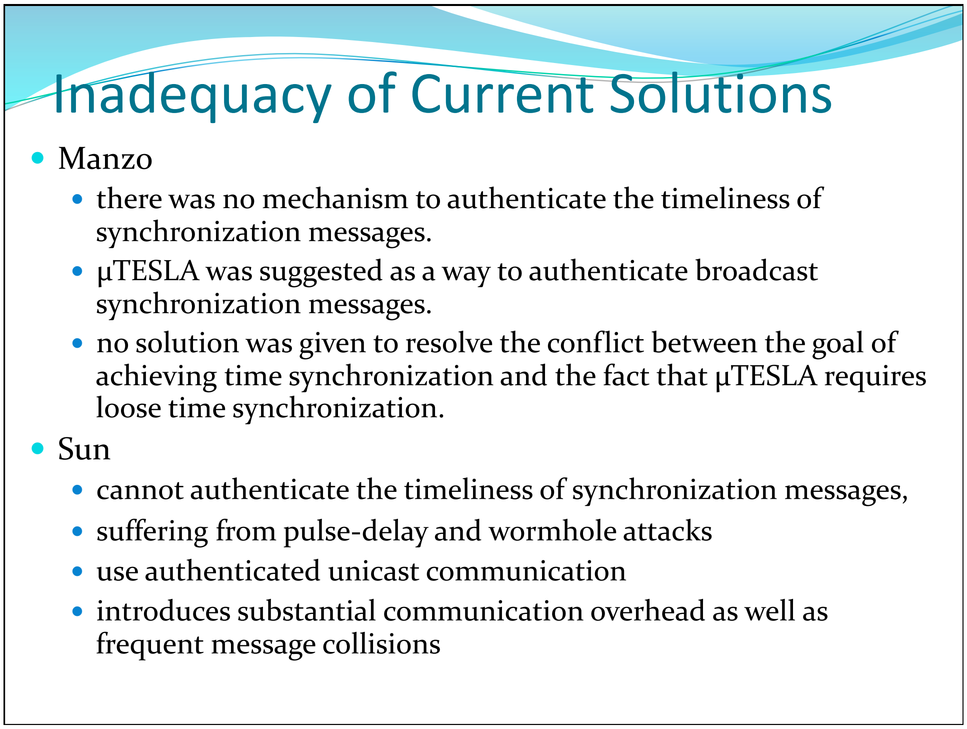
Threats to Time Synchronization

- message authentication can be used to validate message sources and contents, it cannot validate the timeliness of messages, and thus is unable to defend against all of these attacks.
- compromised nodes in arbitrary ways to attack time synchronization.
- Delay, drop, Sybil attacks, collude with others



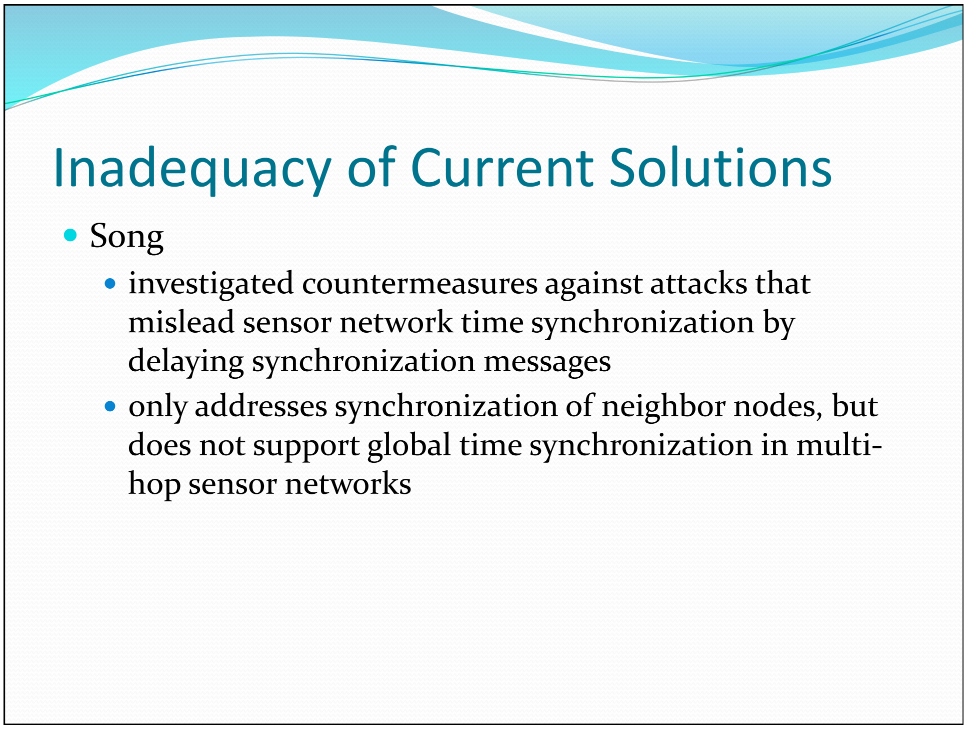
Inadequacy of Current Solutions

- fault-tolerant time synchronization techniques
  - distributed systems
  - digital signatures
  - completely connected network
  - impractical for resource-constrained sensor nodes
- secure and resilient time synchronization in sensor networks
  - secure pairwise synchronization (SPS)
  - provides authentication for medium access control (MAC) layer timestamping by adding timestamp and message integrity code (MIC) as the messages being transmitted.
  - for low data rate sensor radios
  - The group synchronization in uses pairwise authentication to synchronize a group of nodes, thus introducing high computation and communication overheads.
  - assumes all nodes in a group can communicate with each other directly.



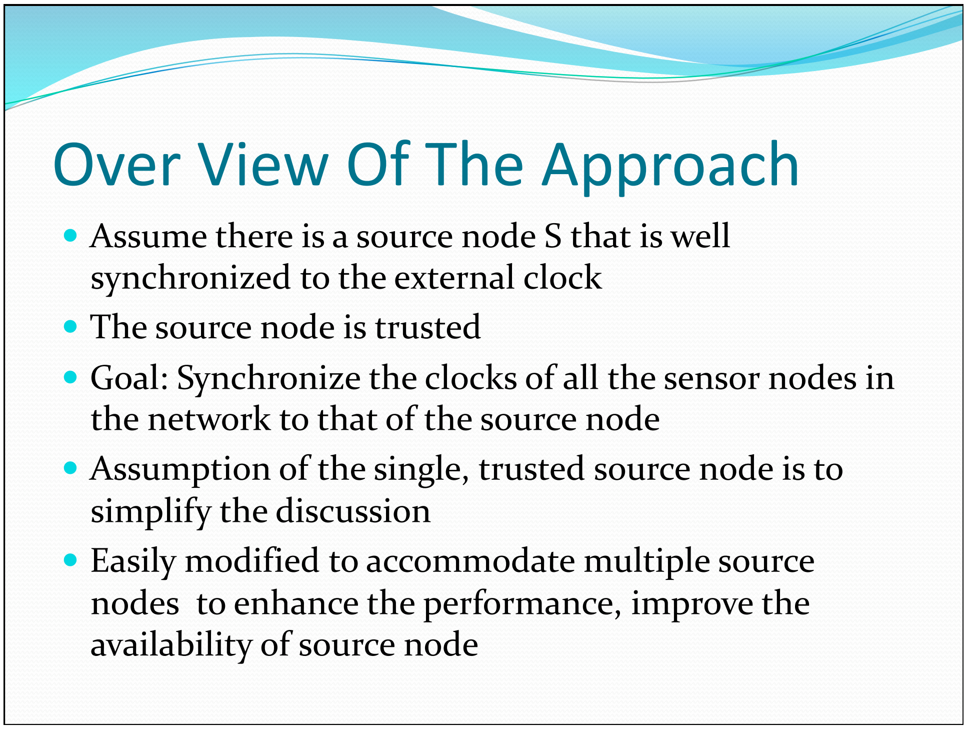
Inadequacy of Current Solutions

- Manzo
  - there was no mechanism to authenticate the timeliness of synchronization messages.
  - μTESLA was suggested as a way to authenticate broadcast synchronization messages.
  - no solution was given to resolve the conflict between the goal of achieving time synchronization and the fact that μTESLA requires loose time synchronization.
- Sun
  - cannot authenticate the timeliness of synchronization messages,
  - suffering from pulse-delay and wormhole attacks
  - use authenticated unicast communication
  - introduces substantial communication overhead as well as frequent message collisions



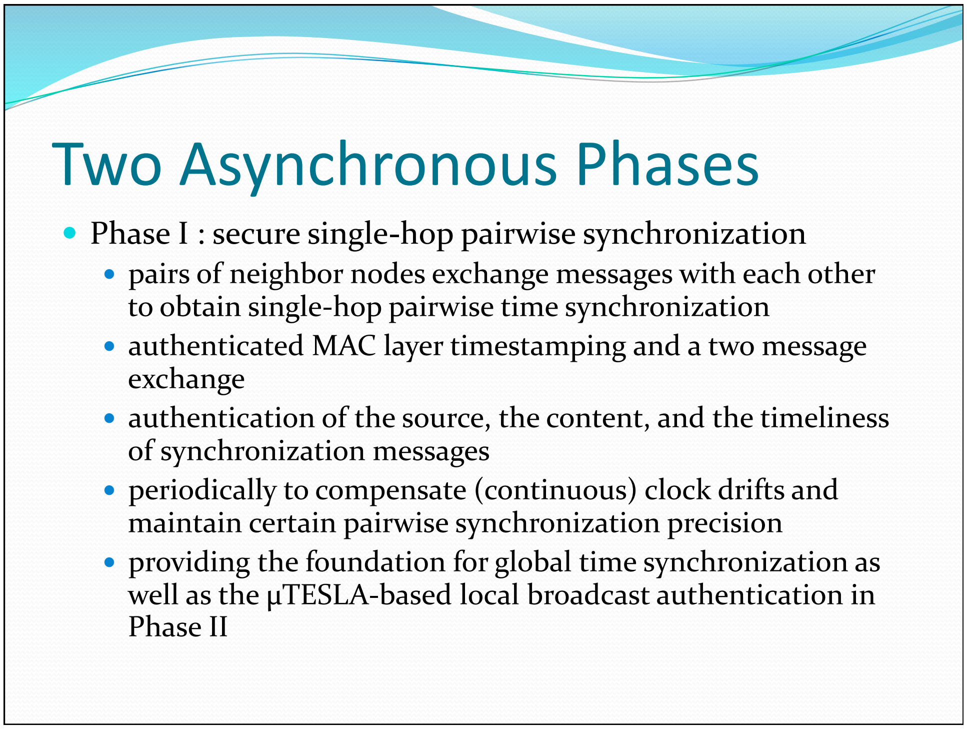
Inadequacy of Current Solutions

- Song
  - investigated countermeasures against attacks that mislead sensor network time synchronization by delaying synchronization messages
  - only addresses synchronization of neighbor nodes, but does not support global time synchronization in multi-hop sensor networks



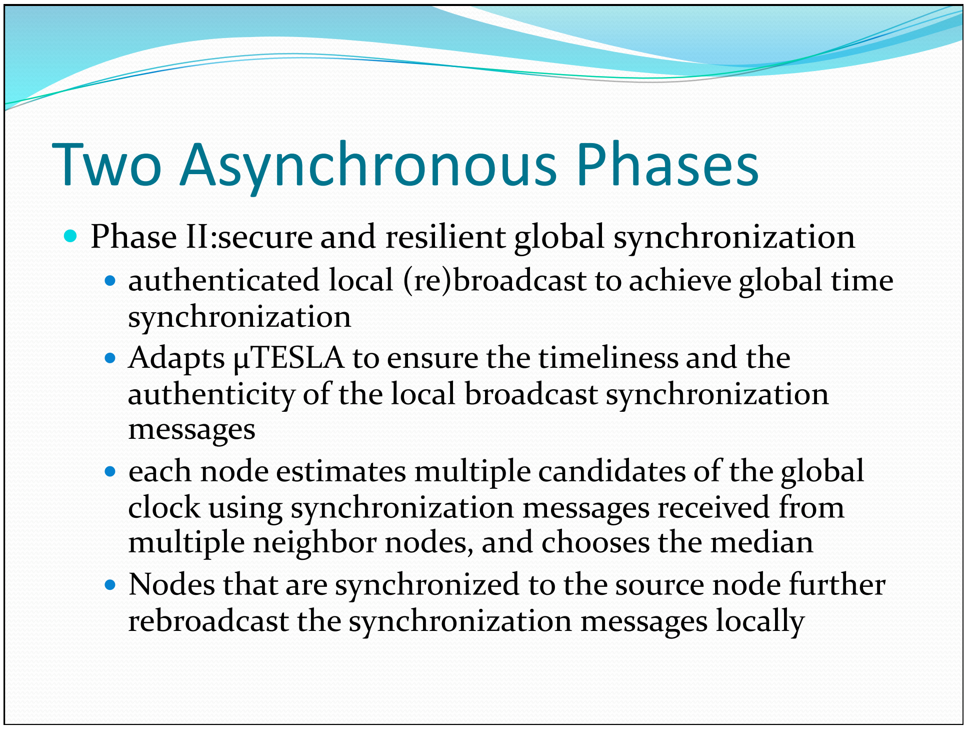
Over View Of The Approach

- Assume there is a source node S that is well synchronized to the external clock
- The source node is trusted
- Goal: Synchronize the clocks of all the sensor nodes in the network to that of the source node
- Assumption of the single, trusted source node is to simplify the discussion
- Easily modified to accommodate multiple source nodes to enhance the performance, improve the availability of source node



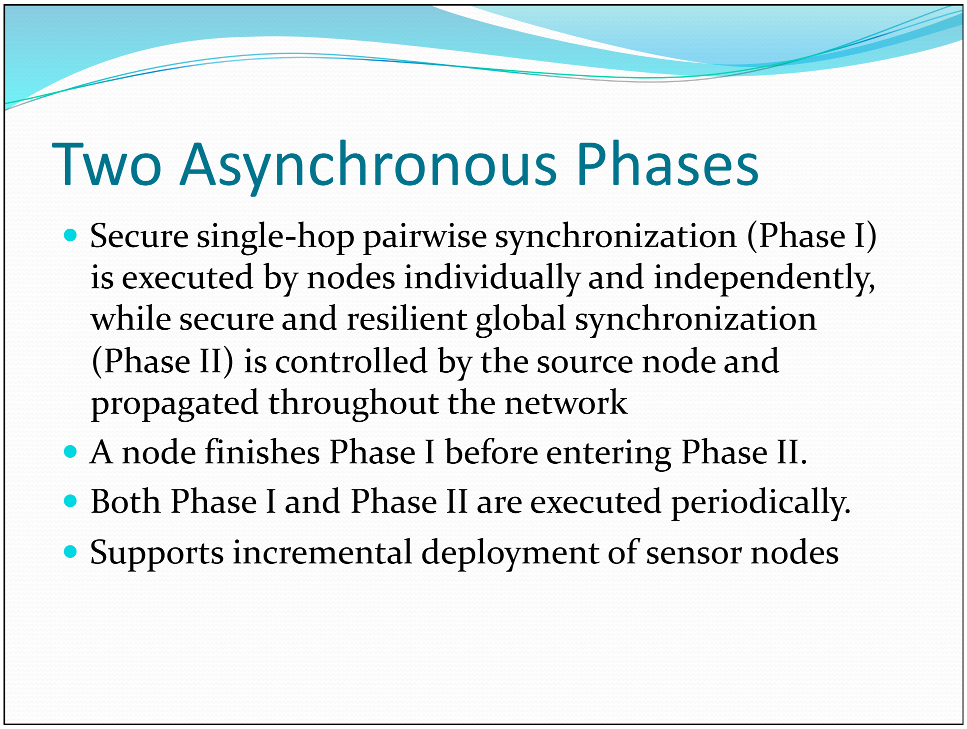
Two Asynchronous Phases

- Phase I : secure single-hop pairwise synchronization
  - pairs of neighbor nodes exchange messages with each other to obtain single-hop pairwise time synchronization
  - authenticated MAC layer timestamping and a two message exchange
  - authentication of the source, the content, and the timeliness of synchronization messages
  - periodically to compensate (continuous) clock drifts and maintain certain pairwise synchronization precision
  - providing the foundation for global time synchronization as well as the μTESLA-based local broadcast authentication in Phase II



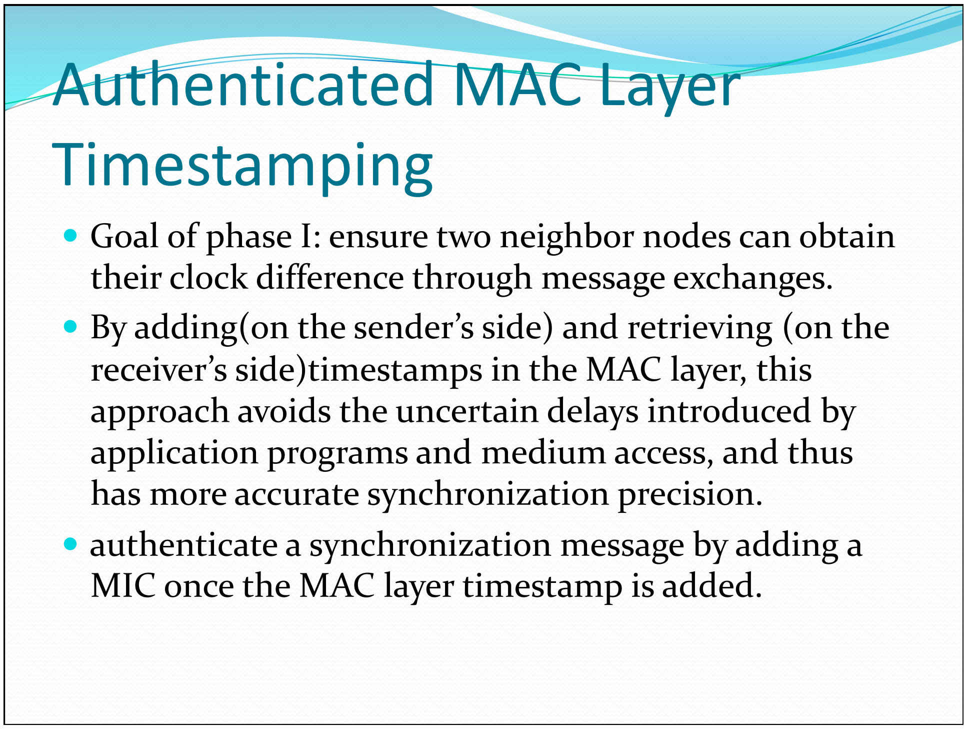
Two Asynchronous Phases

- Phase II:secure and resilient global synchronization
  - authenticated local (re)broadcast to achieve global time synchronization
  - Adapts μTESLA to ensure the timeliness and the authenticity of the local broadcast synchronization messages
  - each node estimates multiple candidates of the global clock using synchronization messages received from multiple neighbor nodes, and chooses the median
  - Nodes that are synchronized to the source node further rebroadcast the synchronization messages locally



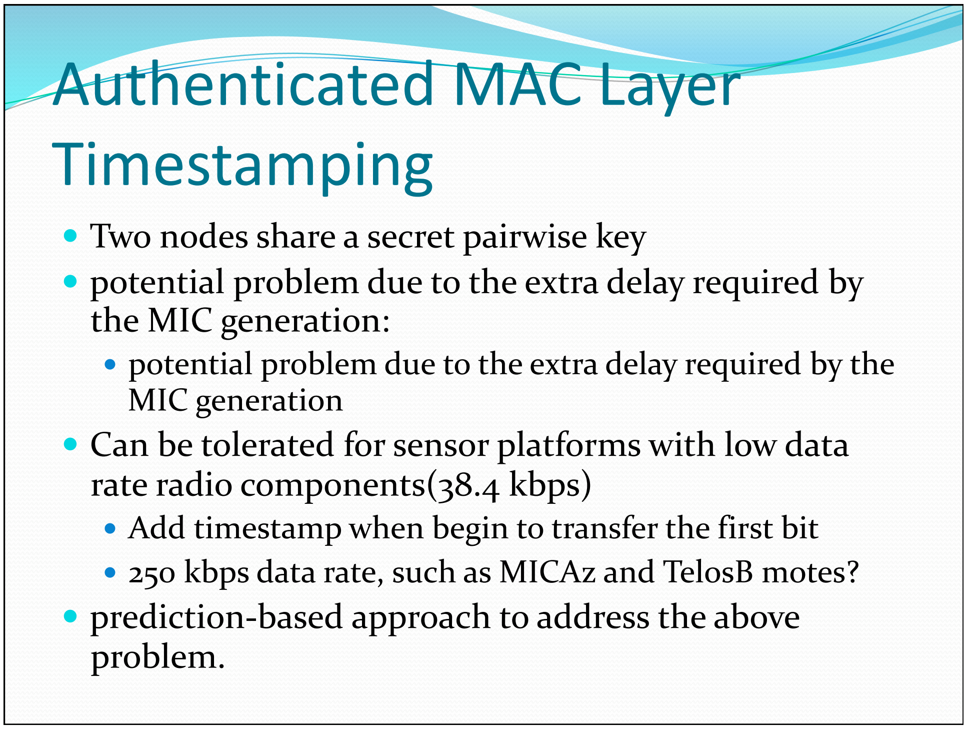
Two Asynchronous Phases

- Secure single-hop pairwise synchronization (Phase I) is executed by nodes individually and independently, while secure and resilient global synchronization (Phase II) is controlled by the source node and propagated throughout the network
- A node finishes Phase I before entering Phase II.
- Both Phase I and Phase II are executed periodically.
- Supports incremental deployment of sensor nodes



Authenticated MAC Layer Timestamping

- Goal of phase I: ensure two neighbor nodes can obtain their clock difference through message exchanges.
- By adding(on the sender's side) and retrieving (on the receiver's side)timestamps in the MAC layer, this approach avoids the uncertain delays introduced by application programs and medium access, and thus has more accurate synchronization precision.
- authenticate a synchronization message by adding a MIC once the MAC layer timestamp is added.



Authenticated MAC Layer Timestamping

- Two nodes share a secret pairwise key
- potential problem due to the extra delay required by the MIC generation:
  - potential problem due to the extra delay required by the MIC generation
- Can be tolerated for sensor platforms with low data rate radio components(38.4 kbps)
  - Add timestamp when begin to transfer the first bit
  - 250 kbps data rate, such as MICAz and TelosB motes?
- prediction-based approach to address the above problem.

Prediction Based MACLayer Timestamping

- Facts:
  - time required for a MIC generation for messages with a given length is fixed
  - the process to transmit a packet (starting from observing the channel vacancy to the actual transmission of data payload) in CC2420 is also deterministic.
- May predict the time required by MIC generation and at the same time predict the delay between the start of transmission and the transmission a given bit in the packet

CC2420 radio packet transmission

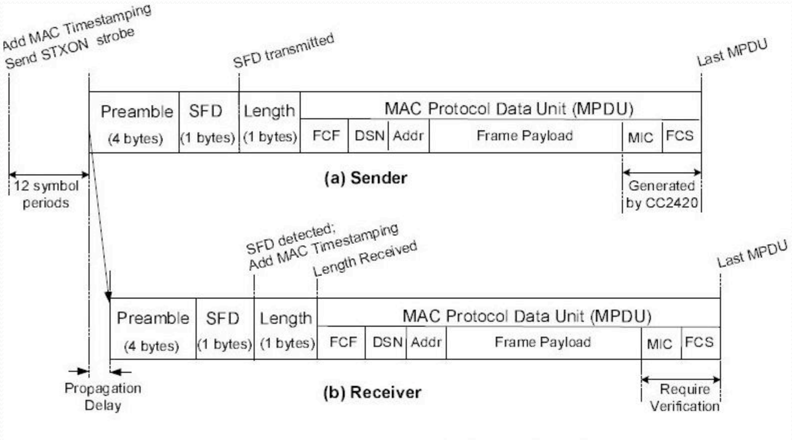
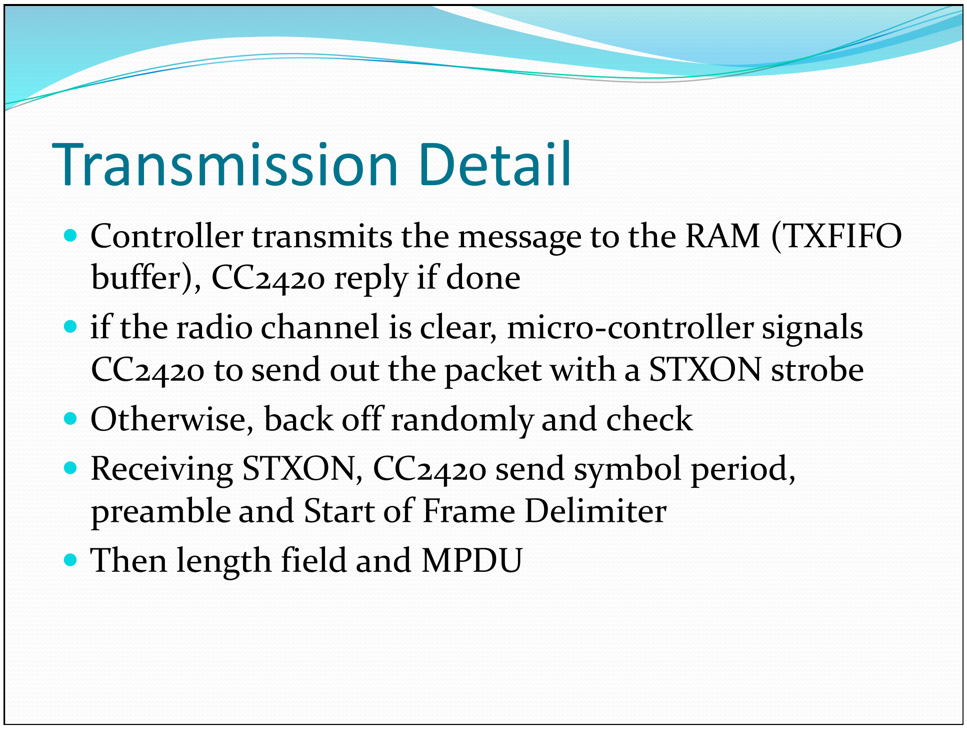
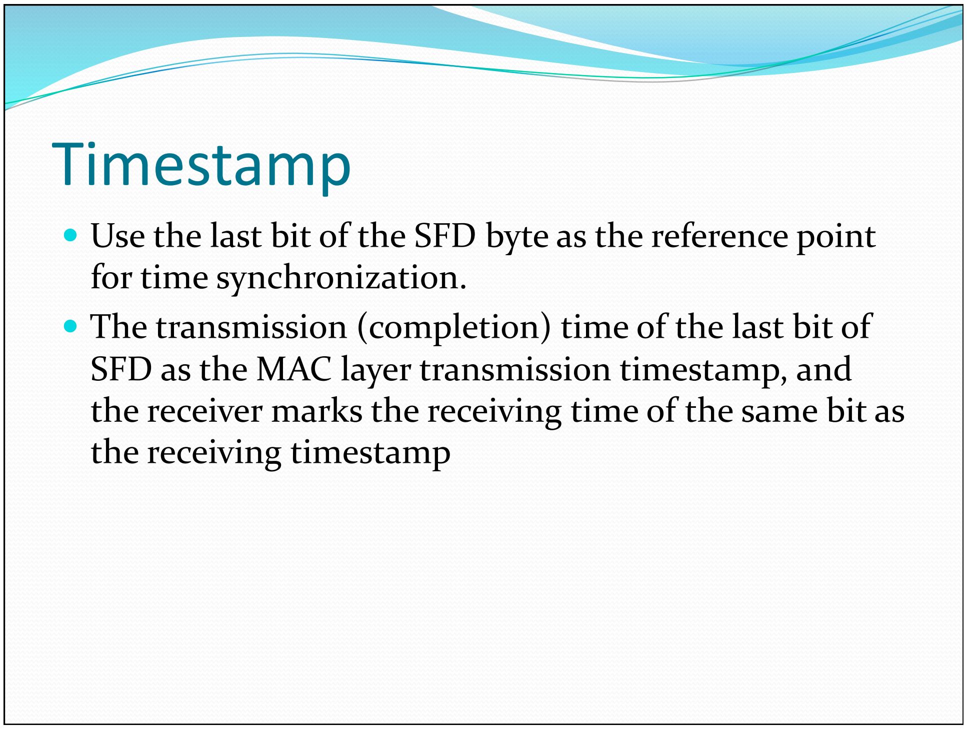


Figure 1: Packet Sending and Receiving Process



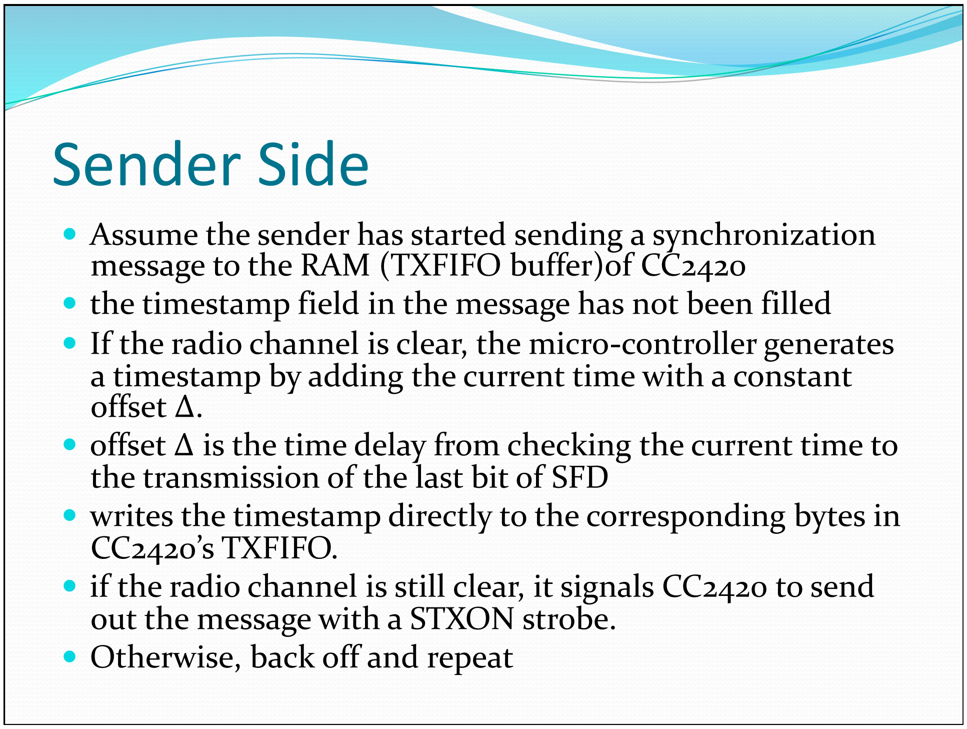
Transmission Detail

- Controller transmits the message to the RAM (TXFIFO buffer), CC2420 reply if done
- if the radio channel is clear, micro-controller signals CC2420 to send out the packet with a STXON strobe
- Otherwise, back off randomly and check
- Receiving STXON, CC2420 send symbol period, preamble and Start of Frame Delimiter
- Then length field and MPDU



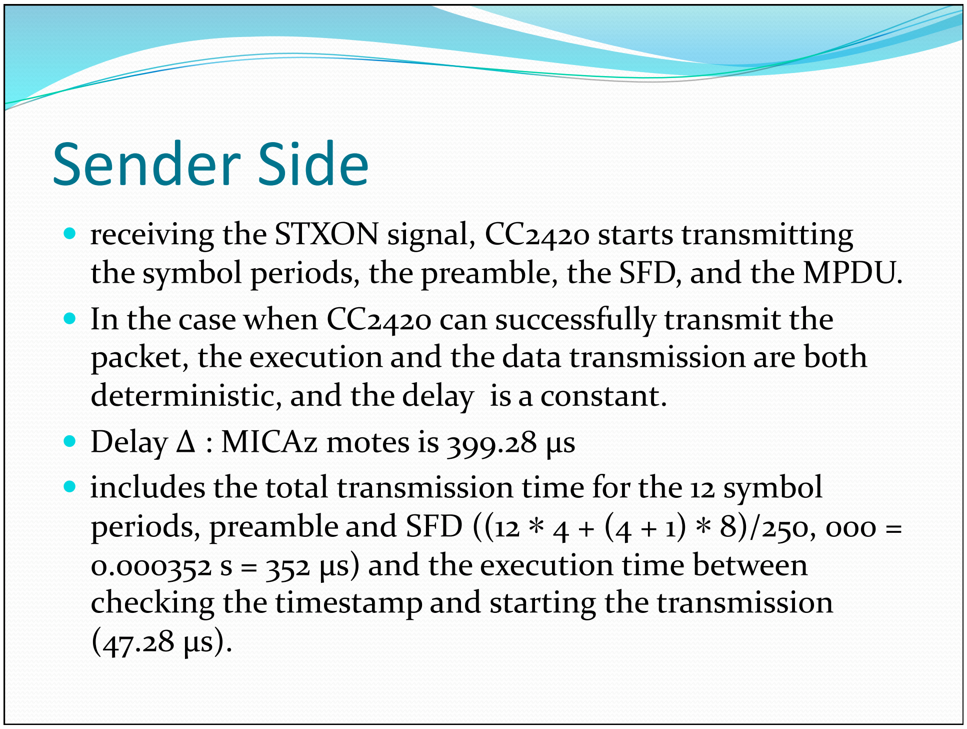
Timestamp

- Use the last bit of the SFD byte as the reference point for time synchronization.
- The transmission (completion) time of the last bit of SFD as the MAC layer transmission timestamp, and the receiver marks the receiving time of the same bit as the receiving timestamp



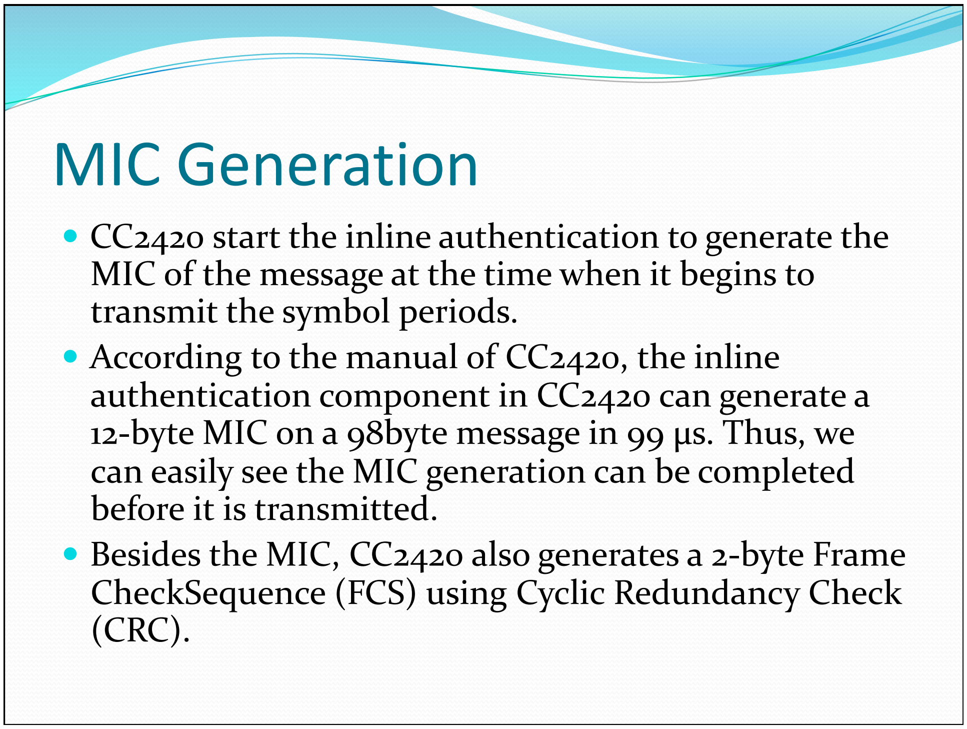
Sender Side

- Assume the sender has started sending a synchronization message to the RAM (TXFIFO buffer)of CC2420
- the timestamp field in the message has not been filled
- If the radio channel is clear, the micro-controller generates a timestamp by adding the current time with a constant offset Δ.
- offset Δ is the time delay from checking the current time to the transmission of the last bit of SFD
- writes the timestamp directly to the corresponding bytes in CC2420's TXFIFO.
- if the radio channel is still clear, it signals CC2420 to send out the message with a STXON strobe.
- Otherwise, back off and repeat



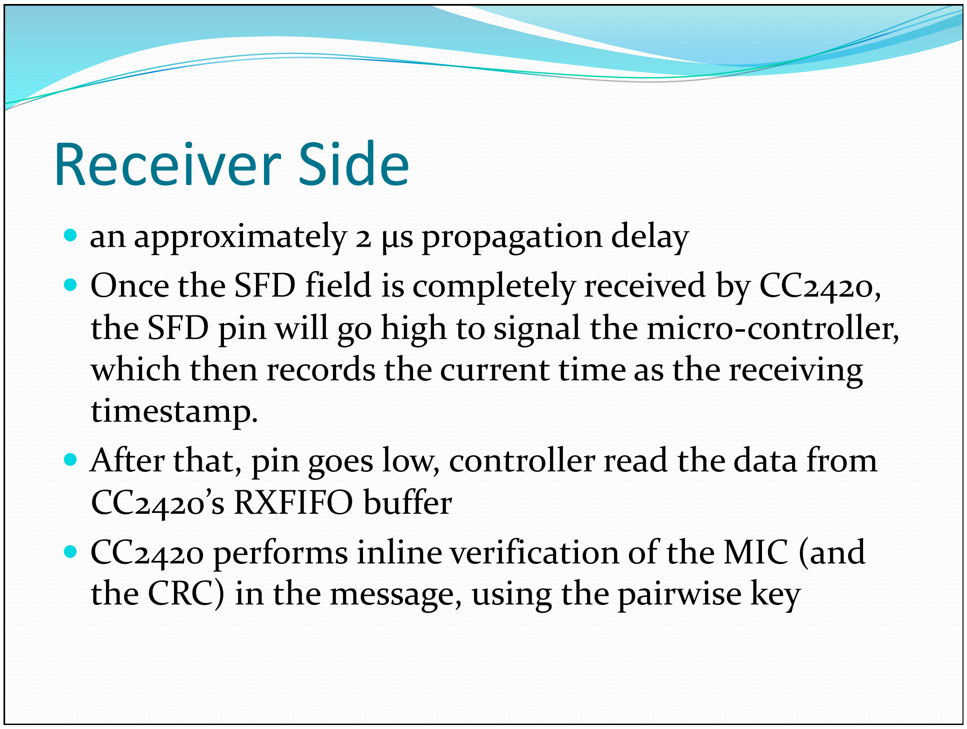
Sender Side

- receiving the STXON signal, CC2420 starts transmitting the symbol periods, the preamble, the SFD, and the MPDU.
- In the case when CC2420 can successfully transmit the packet, the execution and the data transmission are both deterministic, and the delay is a constant.
- Delay Δ : MICAz motes is 399.28 μs
- includes the total transmission time for the 12 symbol periods, preamble and SFD ((12 ∗ 4 + (4 + 1) ∗ 8)/250, 000 = 0.000352 s = 352 μs) and the execution time between checking the timestamp and starting the transmission (47.28 μs).



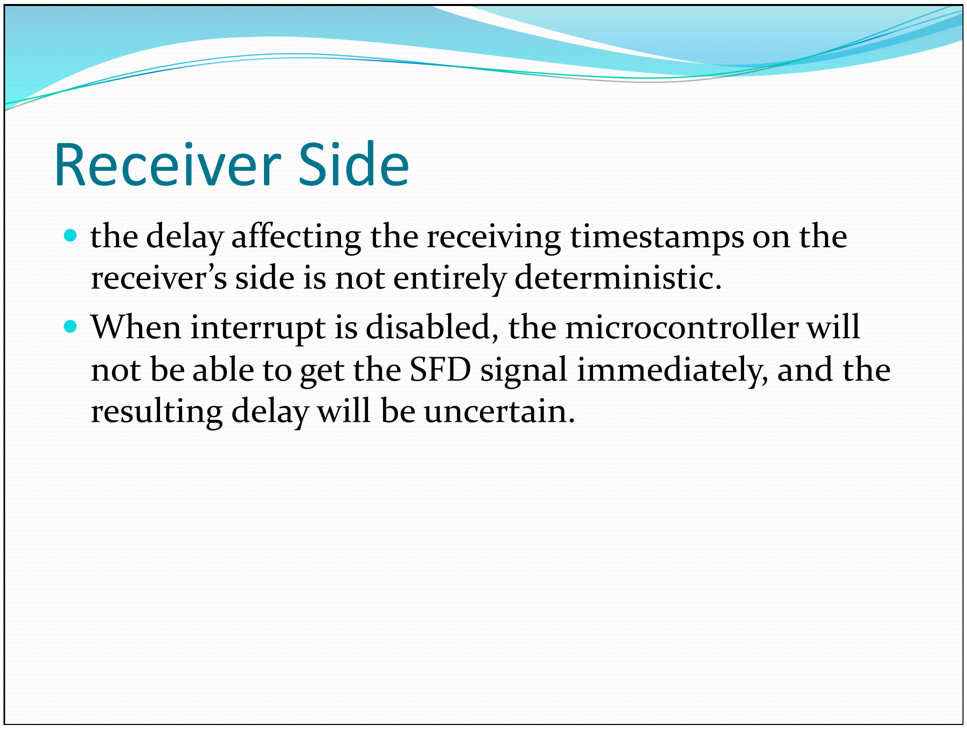
MIC Generation

- CC2420 start the inline authentication to generate the MIC of the message at the time when it begins to transmit the symbol periods.
- According to the manual of CC2420, the inline authentication component in CC2420 can generate a 12-byte MIC on a 98byte message in 99 μs. Thus, we can easily see the MIC generation can be completed before it is transmitted.
- Besides the MIC, CC2420 also generates a 2-byte Frame CheckSequence (FCS) using Cyclic Redundancy Check (CRC).



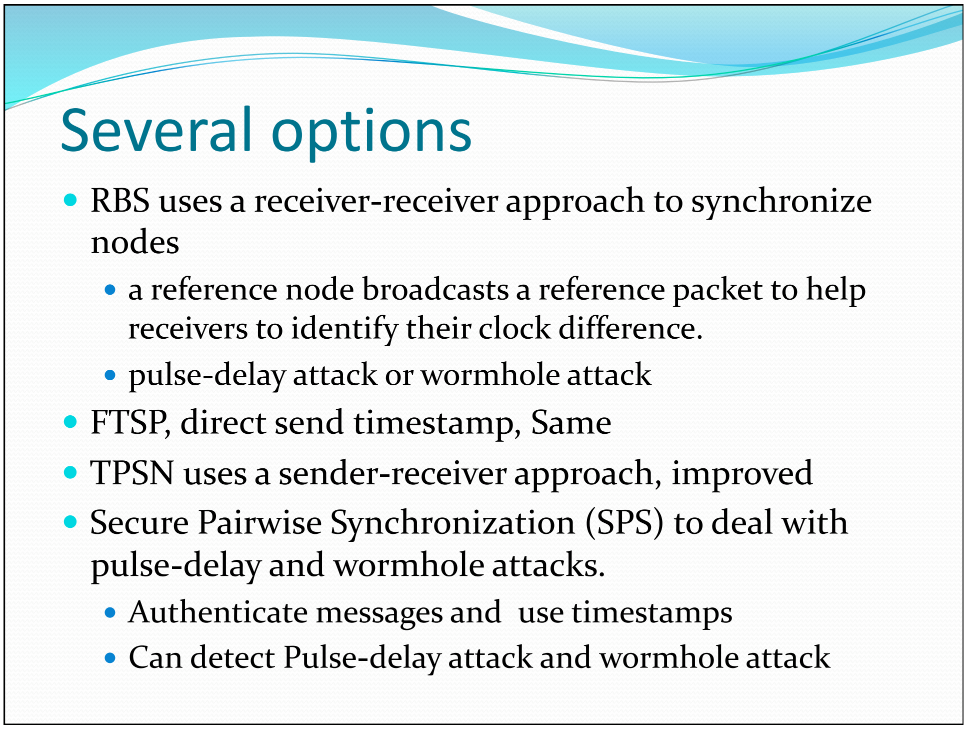
Receiver Side

- an approximately 2 μs propagation delay
- Once the SFD field is completely received by CC2420, the SFD pin will go high to signal the micro-controller, which then records the current time as the receiving timestamp.
- After that, pin goes low, controller read the data from CC2420's RXFIFO buffer
- CC2420 performs inline verification of the MIC (and the CRC) in the message, using the pairwise key



Receiver Side

- the delay affecting the receiving timestamps on the receiver’s side is not entirely deterministic.
- When interrupt is disabled, the microcontroller will not be able to get the SFD signal immediately, and the resulting delay will be uncertain.



Several options

- RBS uses a receiver-receiver approach to synchronize nodes
  - a reference node broadcasts a reference packet to help receivers to identify their clock difference.
  - pulse-delay attack or wormhole attack
- FTSP, direct send timestamp, Same
- TPSN uses a sender-receiver approach, improved
- Secure Pairwise Synchronization (SPS) to deal with pulse-delay and wormhole attacks.
  - Authenticate messages and use timestamps
  - Can detect Pulse-delay attack and wormhole attack

Revised SPS protocol

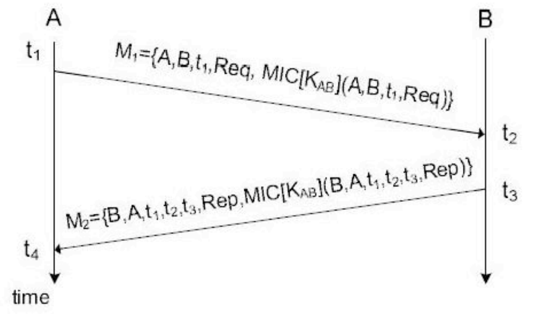

A
B
t_1
M_1={A,B,t_1,Req, MIC[K_{AB}](A,B,t_1,Req)}
t_2
M_2={B,A,t_1,t_2,t_3,Rep,MIC[K_{AB}](B,A,t_1,t_2,t_3,Rep)}
t_3
t_4
time

Figure 2: (Revised) Secure Pairwise Synchronization (The authors would like to thank Glenn Wurster for pointing out a typo in an earlier version of this figure.)

Revised SPS protocol

- Modify SPS with sender's timestamp in reply message
- all messages are timestamped and authenticated with the key KAB
- t1 is also included in M2 to detect replay attacks
- A receives the message at t4, it can calculate the clock difference ΔA,B = ((t2–t1)–(t4–t3))/2
- the estimated one-way transmission delay dA = (t2–t1 +t4–t3)/2
- Node A verifies that the one-way transmission delay is less than the maximum expected delay

Revised SPS protocol

- Enable the sender obtain the clock difference with the receiver
- An alternative is to perform a three-way message exchange so that both nodes will get the clock difference at the end of the protocol execution
- Have to maintain their states at the intermediate protocol steps

Sources of delay

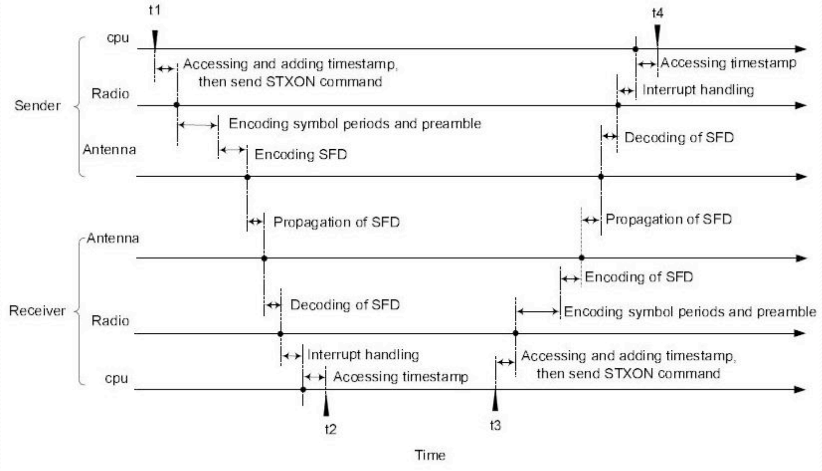


Figure 3: Delay Uncertainty (Certain Details are Omitted for Clarity)

Sources of delay

- CPU to access current time is deterministic and less than 1 tick.
- The time for adding timestamp in the buffered packet is deterministic, less than 2 ticks.
- The encoding time is for the radio to encode and transform a part of the packet (4 bits a time in case of CC2420) to electromagnetic waves, and the decoding time is for the radio to transform and decode electromagnetic waves into binary data. These times are controlled by a chip sequence at 2 MChips/s, and thus are deterministic.
- The propagation time depends on the distance between the sender and the receiver, which is also deterministic.
- The uncertainty in the packet delay is MAINLY BECAUSE of the jitter of the interrupt handling, which is caused by temporarily disabled interrupt handling on the receiving side.
- For example, the receiver may be executing a piece of code that disables interrupt handling when the SFD is received by the RF module, and thus cannot access the receiving timestamp promptly.

Sync Error Distribution in SPS

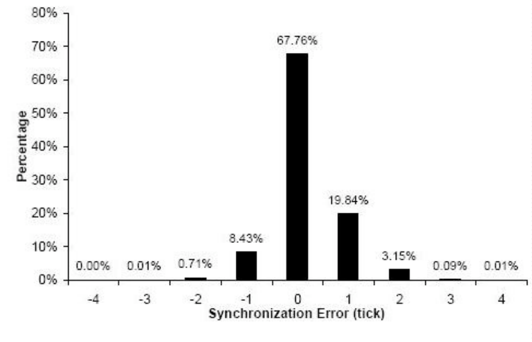
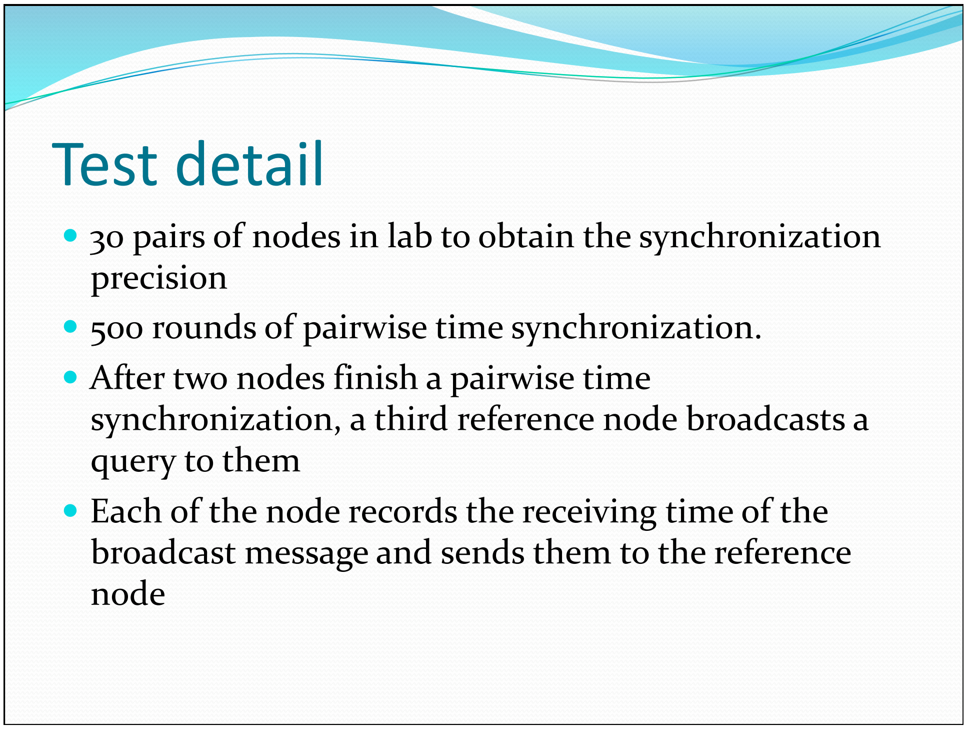
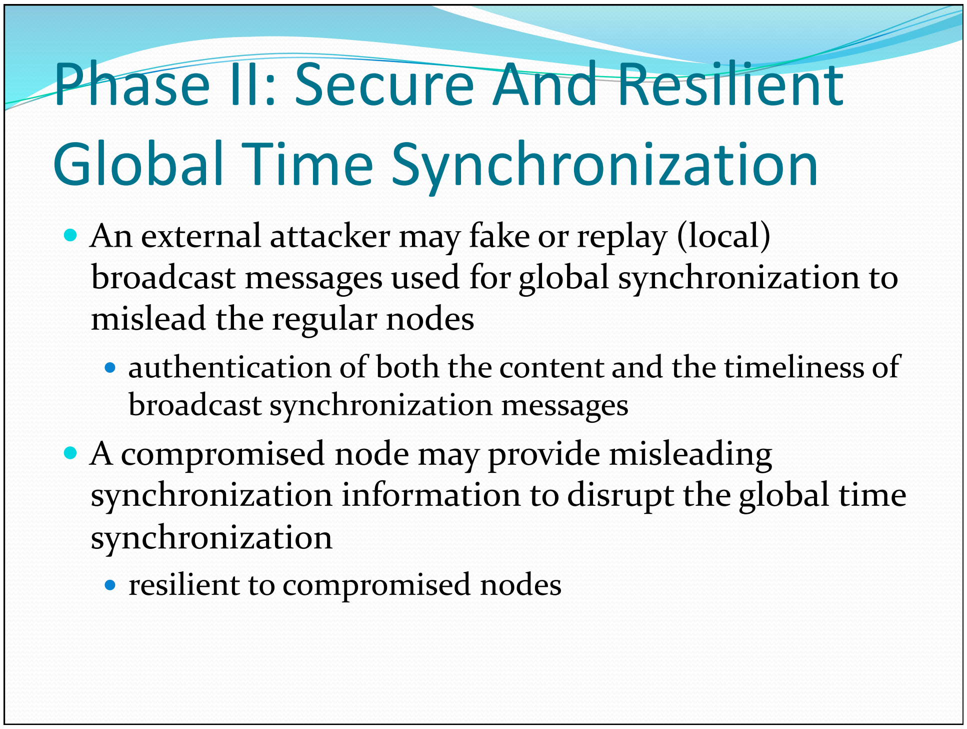


Figure 4: Synchronization Error Distribution in Secure Pairwise Synchronization (1 Tick = 8.68 μs)



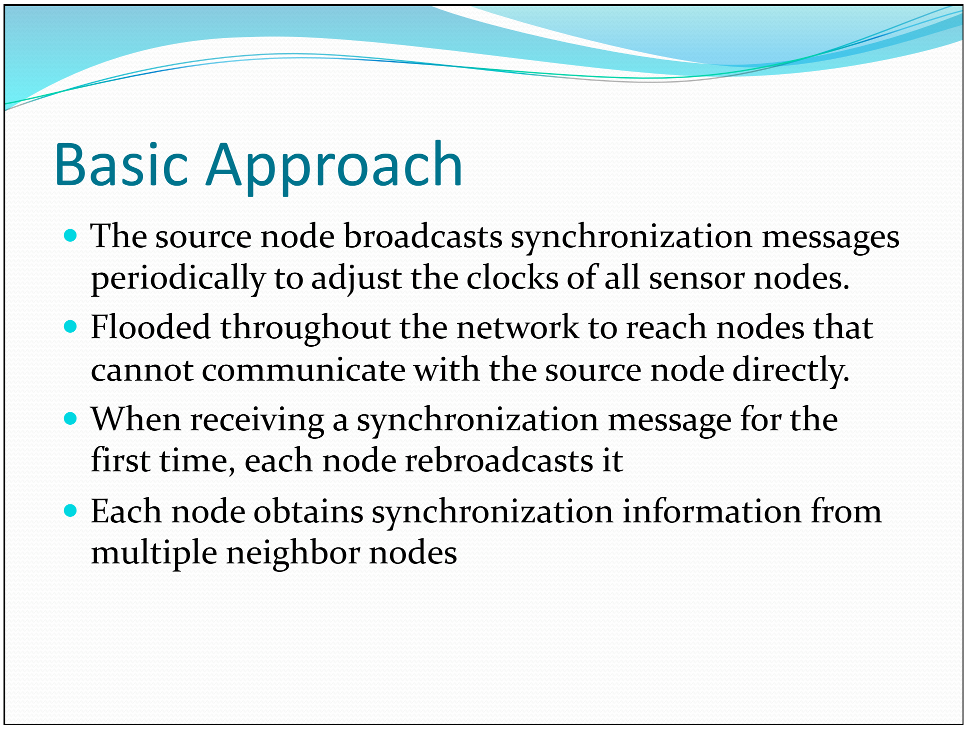
Test detail

- 30 pairs of nodes in lab to obtain the synchronization precision
- 500 rounds of pairwise time synchronization.
- After two nodes finish a pairwise time synchronization, a third reference node broadcasts a query to them
- Each of the node records the receiving time of the broadcast message and sends them to the reference node



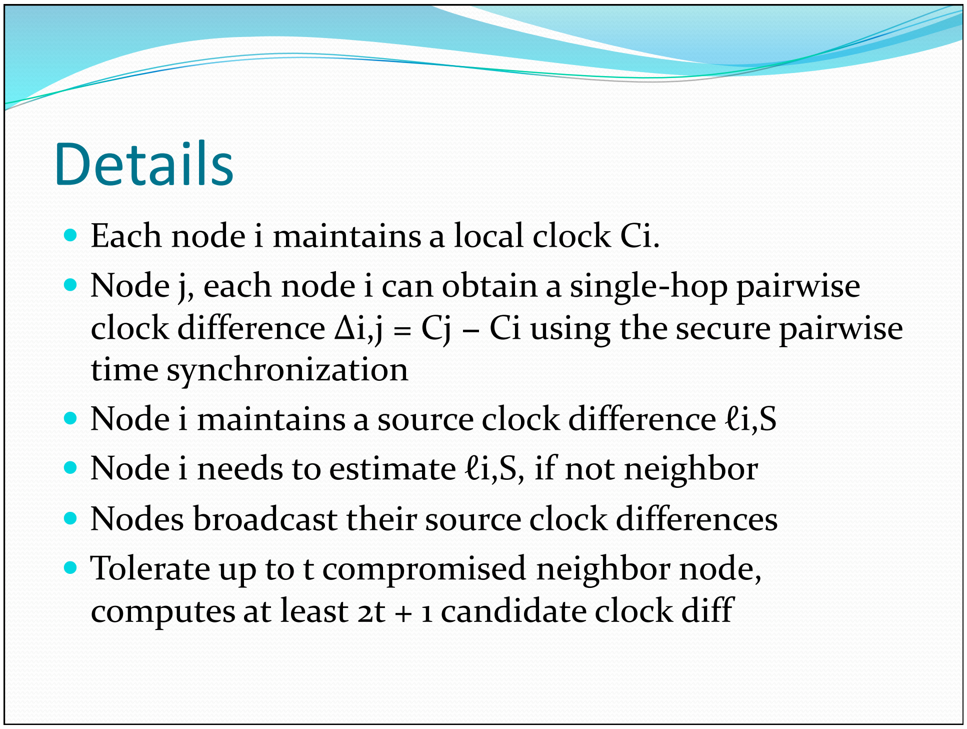
Phase II: Secure And Resilient Global Time Synchronization

- An external attacker may fake or replay (local) broadcast messages used for global synchronization to mislead the regular nodes
  - authentication of both the content and the timeliness of broadcast synchronization messages
- A compromised node may provide misleading synchronization information to disrupt the global time synchronization
  - resilient to compromised nodes



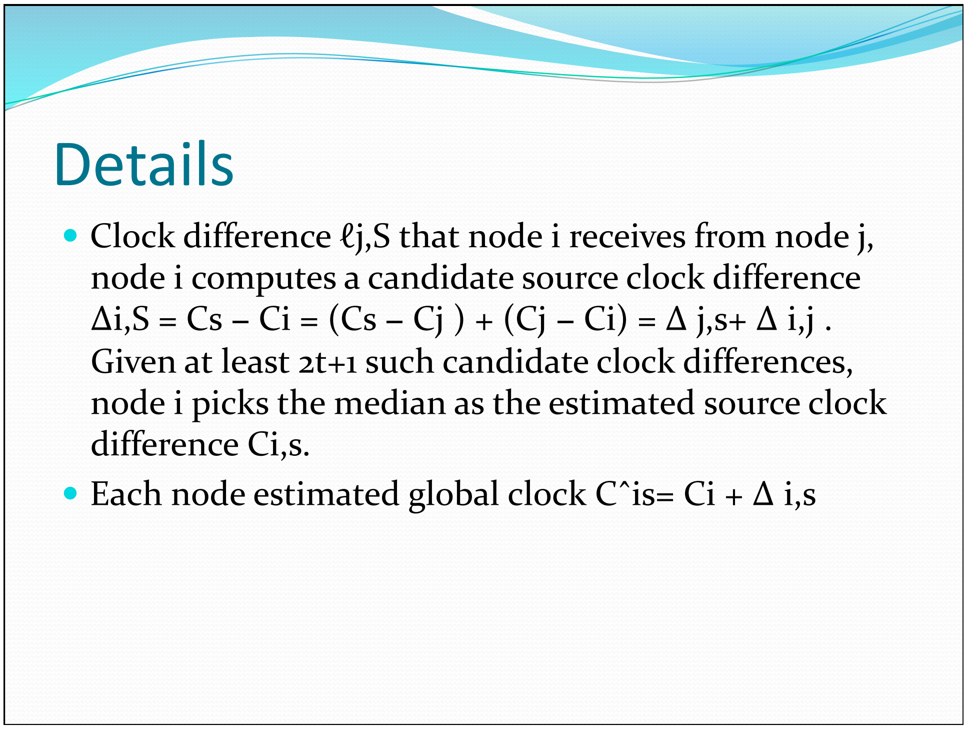
Basic Approach

- The source node broadcasts synchronization messages periodically to adjust the clocks of all sensor nodes.
- Flooded throughout the network to reach nodes that cannot communicate with the source node directly.
- When receiving a synchronization message for the first time, each node rebroadcasts it
- Each node obtains synchronization information from multiple neighbor nodes



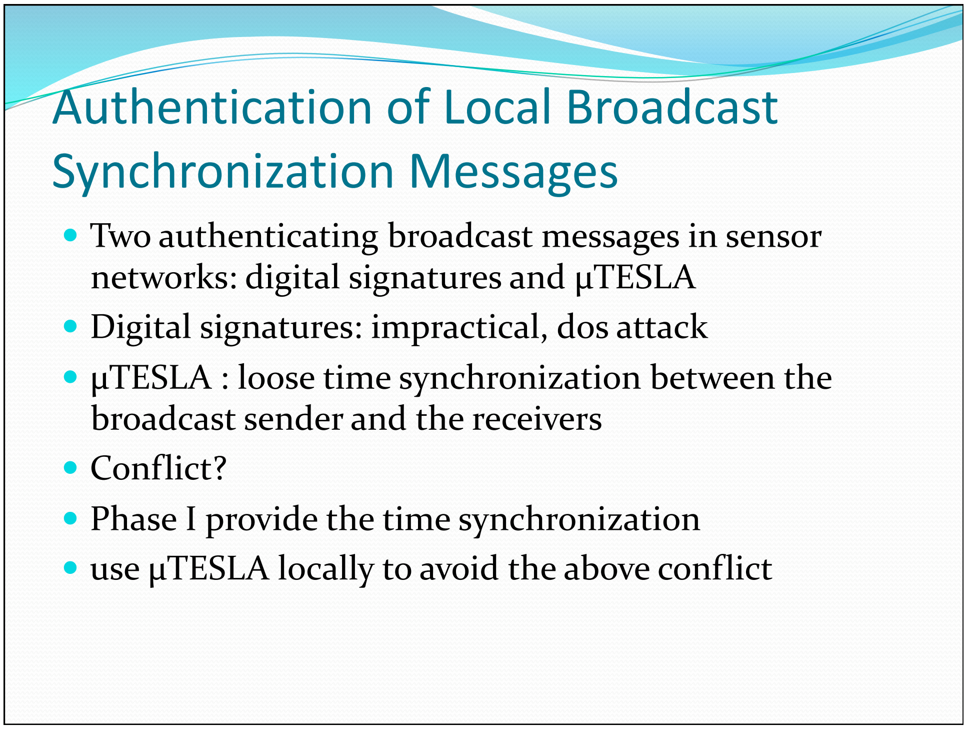
Details

- Each node i maintains a local clock Ci.
- Node j, each node i can obtain a single-hop pairwise clock difference $\Delta i,j = Cj - Ci$ using the secure pairwise time synchronization
- Node i maintains a source clock difference $\ell i,S$
- Node i needs to estimate $\ell i,S$, if not neighbor
- Nodes broadcast their source clock differences
- Tolerate up to t compromised neighbor node, computes at least $2t + 1$ candidate clock diff



Details

- Clock difference $\ell_{j,S}$ that node i receives from node j, node i computes a candidate source clock difference $\Delta_{i,S} = C_s - C_i = (C_s - C_j) + (C_j - C_i) = \Delta_{j,s} + \Delta_{i,j}$. Given at least $2t+1$ such candidate clock differences, node i picks the median as the estimated source clock difference $C_{i,s}$.
- Each node estimated global clock $\hat{C}_{is} = C_i + \Delta_{i,s}$



Authentication of Local Broadcast Synchronization Messages

- Two authenticating broadcast messages in sensor networks: digital signatures and μTESLA
- Digital signatures: impractical, dos attack
- μTESLA : loose time synchronization between the broadcast sender and the receivers
- Conflict?
- Phase I provide the time synchronization
- use μTESLA locally to avoid the above conflict

uTESLA

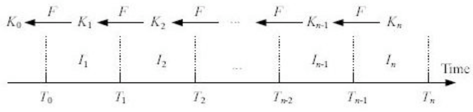
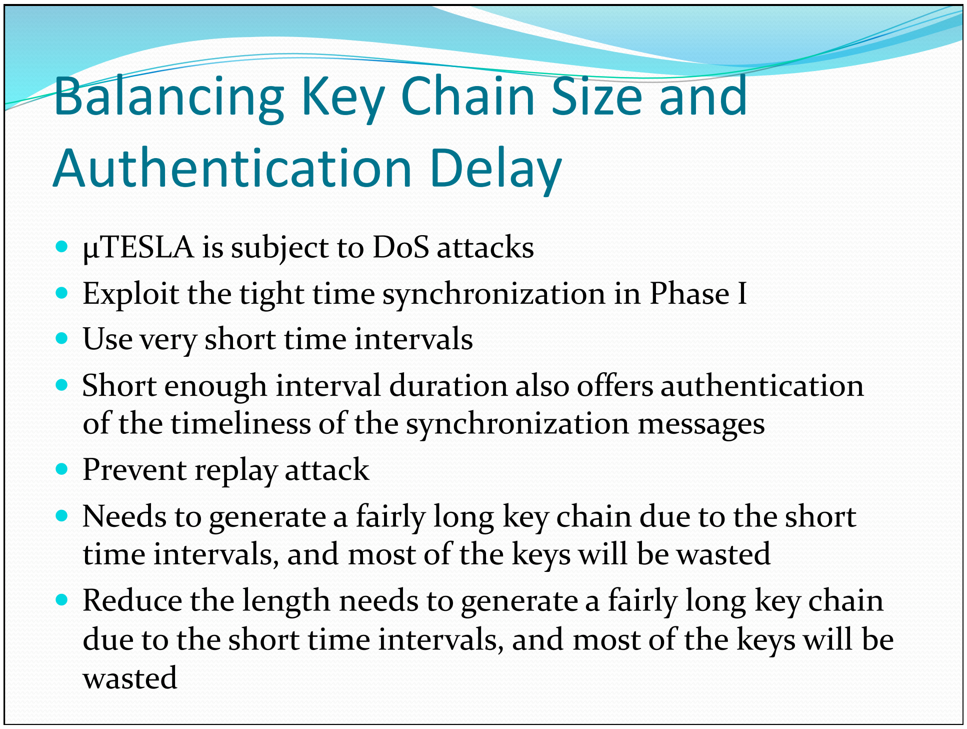


Figure 5: μTESLA

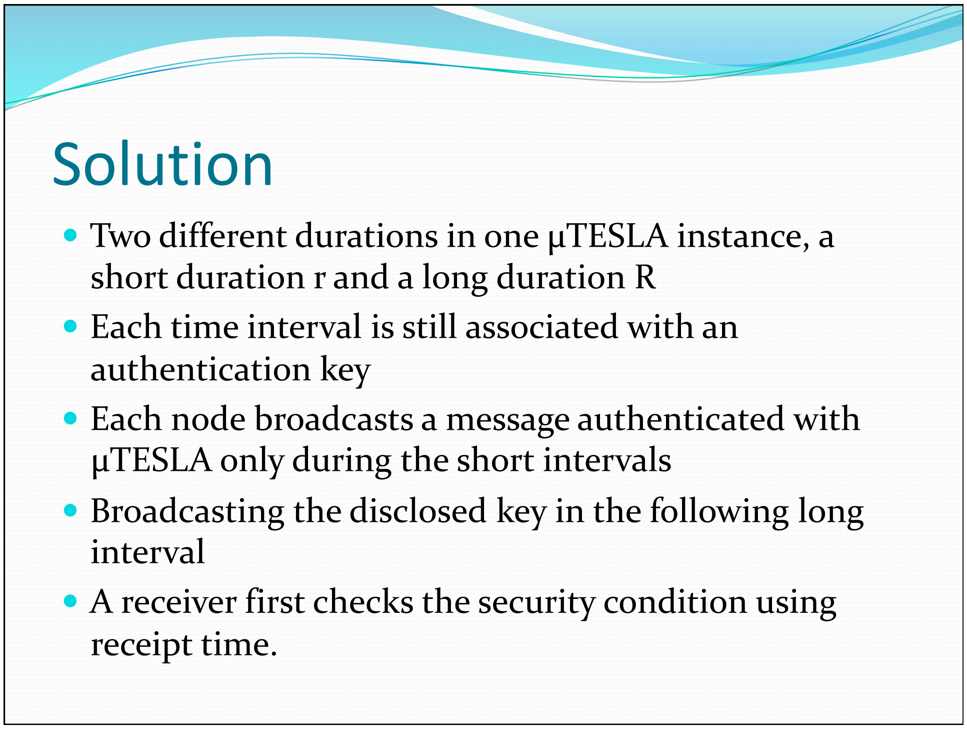
Short Delayed µTESLA

- µTESLA parameters: the key chain ID, the key chain commitment, the duration of each time interval, and the starting time of the first time interval
- Fix the duration of time intervals and the length of each key chain
- Transmission other µTESLA parameters with the single-hop pairwise synchronization
- When one key chain is about to expire, each node needs to communicate with each neighbor node again to transmit the parameters for the next



Balancing Key Chain Size and Authentication Delay

- μTESLA is subject to DoS attacks
- Exploit the tight time synchronization in Phase I
- Use very short time intervals
- Short enough interval duration also offers authentication of the timeliness of the synchronization messages
- Prevent replay attack
- Needs to generate a fairly long key chain due to the short time intervals, and most of the keys will be wasted
- Reduce the length needs to generate a fairly long key chain due to the short time intervals, and most of the keys will be wasted



Solution

- Two different durations in one μTESLA instance, a short duration r and a long duration R
- Each time interval is still associated with an authentication key
- Each node broadcasts a message authenticated with μTESLA only during the short intervals
- Broadcasting the disclosed key in the following long interval
- A receiver first checks the security condition using receipt time.

Security Condition

- B may calculate i = └ (ti–To)/(r+R) ┘ and checks the following security condition:
  - ti – To + ΔB,A + ℓmax < i ∗ (R + r) + r,
  - where Δ B,A is the pairwise clock difference between A and B,
  - ℓmax is maximum synchronization error between two neighbor nodes
- After B obtains Ki, verifies with the commitment
- If the key is valid, B then uses K_i to verify the MIC in Mi

Short Delayed µTESLA

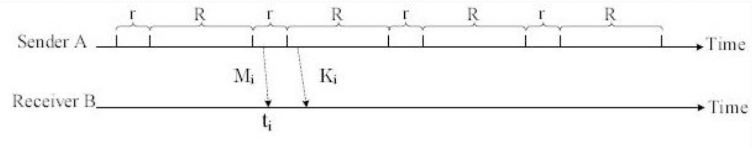
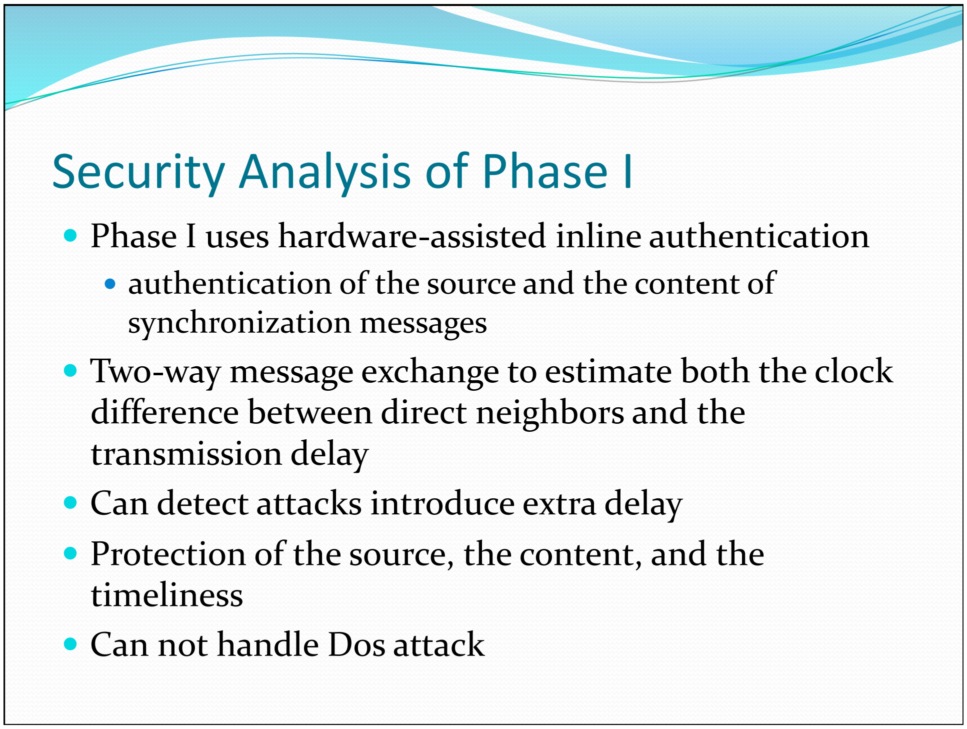
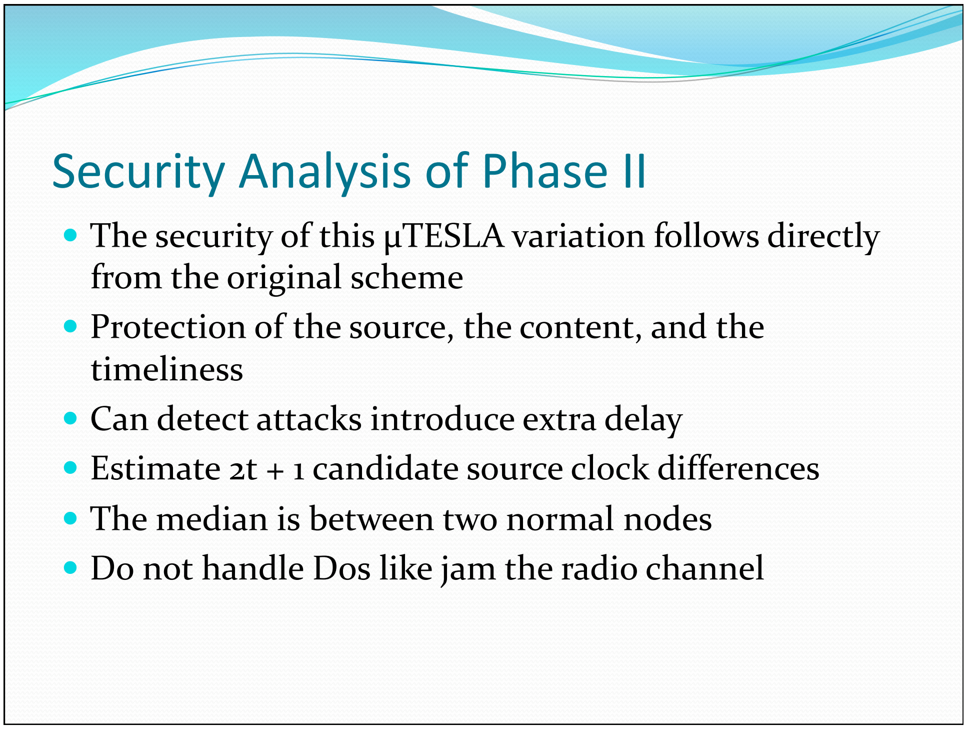


Figure 6: Short Delayed μTESLA



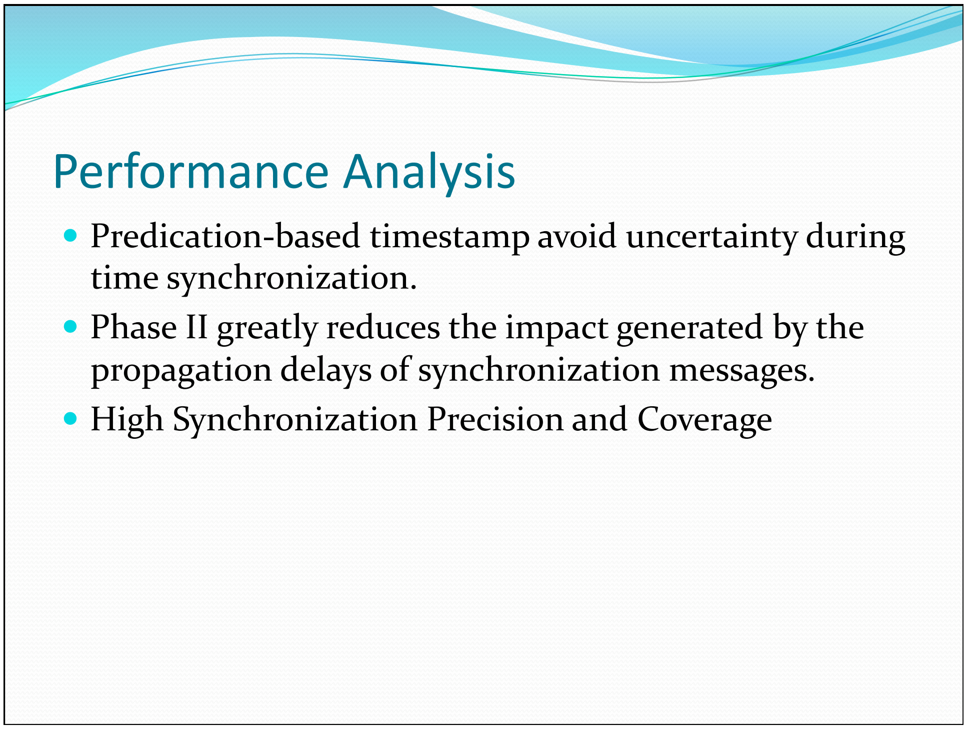
Security Analysis of Phase I

- Phase I uses hardware-assisted inline authentication
  - authentication of the source and the content of synchronization messages
- Two-way message exchange to estimate both the clock difference between direct neighbors and the transmission delay
- Can detect attacks introduce extra delay
- Protection of the source, the content, and the timeliness
- Can not handle Dos attack



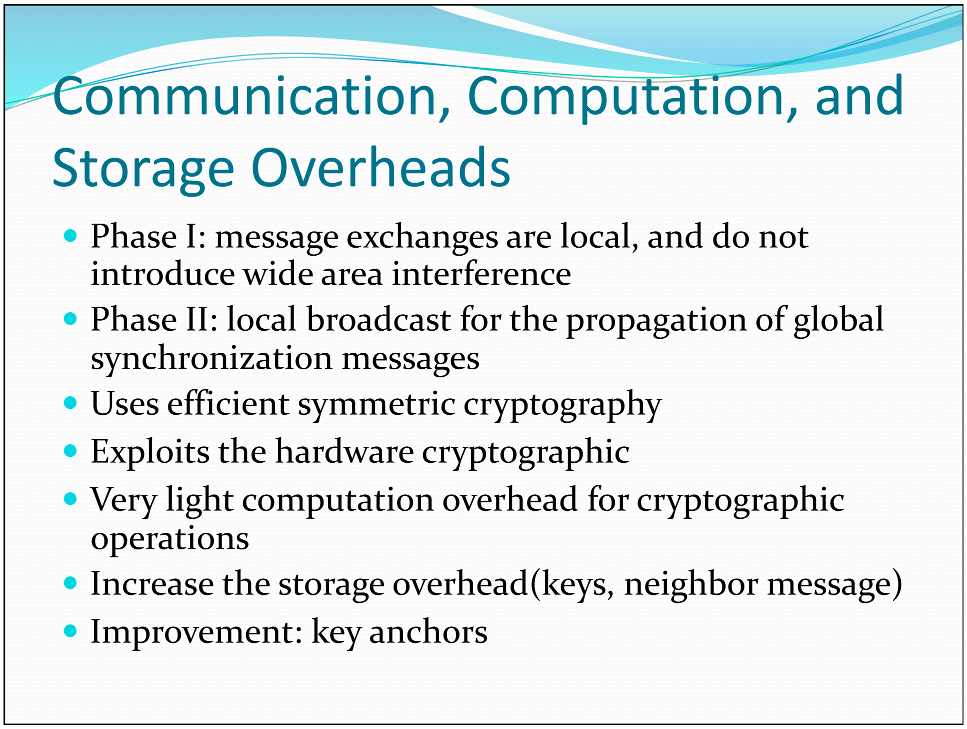
Security Analysis of Phase II

- The security of this μTESLA variation follows directly from the original scheme
- Protection of the source, the content, and the timeliness
- Can detect attacks introduce extra delay
- Estimate 2t + 1 candidate source clock differences
- The median is between two normal nodes
- Do not handle Dos like jam the radio channel



Performance Analysis

- Predication-based timestamp avoid uncertainty during time synchronization.
- Phase II greatly reduces the impact generated by the propagation delays of synchronization messages.
- High Synchronization Precision and Coverage



Communication, Computation, and Storage Overheads

- Phase I: message exchanges are local, and do not introduce wide area interference
- Phase II: local broadcast for the propagation of global synchronization messages
- Uses efficient symmetric cryptography
- Exploits the hardware cryptographic
- Very light computation overhead for cryptographic operations
- Increase the storage overhead(keys, neighbor message)
- Improvement: key anchors

Incremental Deployment

- Both of which are executed periodically
- The newly deployed nodes first obtain the pairwise time differences and the commitments of the key chains from its neighbor nodes in Phase I, and then join the Phase II global time synchronization.

Implementation Details

- MICAz mote
  - 8-bit micro-controller ATMega128L
  - 128 kB program memory and 4kB SRAM.
  - the ChipCon CC2420 radio component
  - 2.4GHz radio frequency ,up to 250 kbps data rate.
- CC2420 provides two types of security operations:
  - stand-alone:plain AES encryption with 128 bit
  - in-line security operations: on a per frame basis, RXFIFO,TXFIFO. CBC-MIC generate MIC

Experiment Results

- A network of 60 MICAz motes
- Nodes performs a spairwise synchronization every d1 = 4 seconds
- The source node starts a global synchronization every d2 seconds(5,10)
- t =0, 1, 2, 3, 4
- Use sink node to collect data
- Each node marks the receiving time and converts it to the global time using its source clock difference

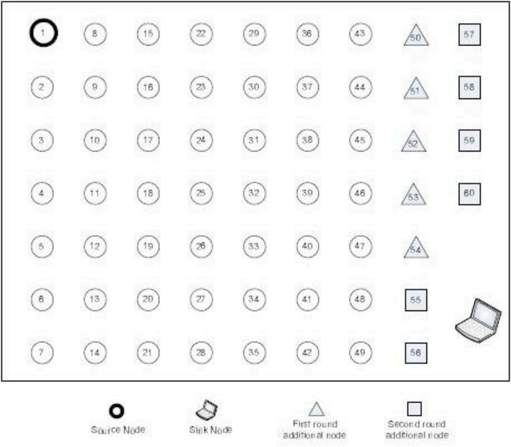


Figure 7: Network Topology

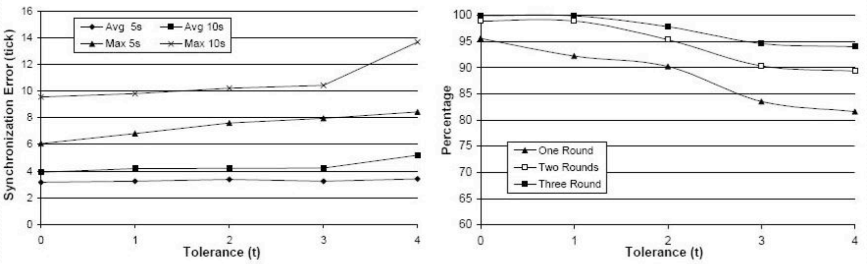
Table 1: Code Size

Memory	Size (bytes)
RAM	1,977
ROM	24,814

Performance in Static Deployment

- Average and Maximum Synchronization Error:
  - maximum synchronization error is below 14 ticks (121.52 μs),
  - average synchronization error is below 6 ticks (52.08 μs).
- Synchronization Rate:
  - Tolerance increases, the synchronization rate decreases.
  - after three rounds, about 95%

Results



(a) Maximum and average synchronization error

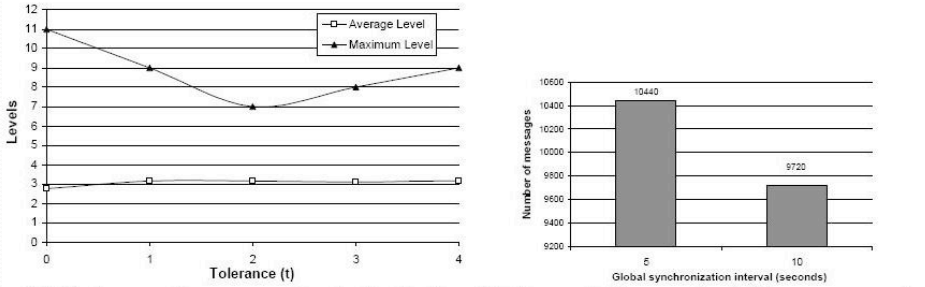
(b) Synchronization rate

Figure 8: Synchronization Error and Synchronization Rate

Performance in Static Deployment

- Synchronization Level:
  - average synchronization level is around 3.
  - maximum synchronization level initially decreases as the tolerance t increases, but then goes up as t is greater than 2.
- Communication Overhead:
  - n neighbor nodes, In a given long time interval T, each node sends at most messages.
  - 10 neighbor nodes, pairwise interval is 4 seconds
  - One hour

Results



(a) Maximum and average synchronization level

(b) Communication overhead (# messages each node sends per hour)

Figure 9: Synchronization Level and Communication Overhead

Incremental Deployment

- 49 nodes marked as circles. Then added 5 new nodes into the network about 10 minutes later, and added another 6 new nodes about 1 minute later.
- Set t = 2,and the global synchronization interval is set to 10s.
- Could not be synchronized immediately, and the average synchronization error was large and the synchronization rate dropped to around 90%.
- After a few rounds of global synchronization, all these new nodes were correctly synchronized

Results

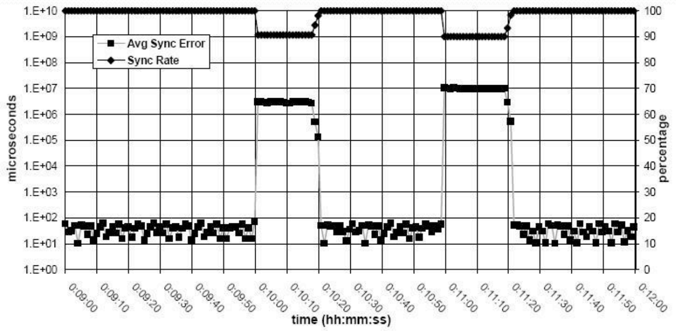
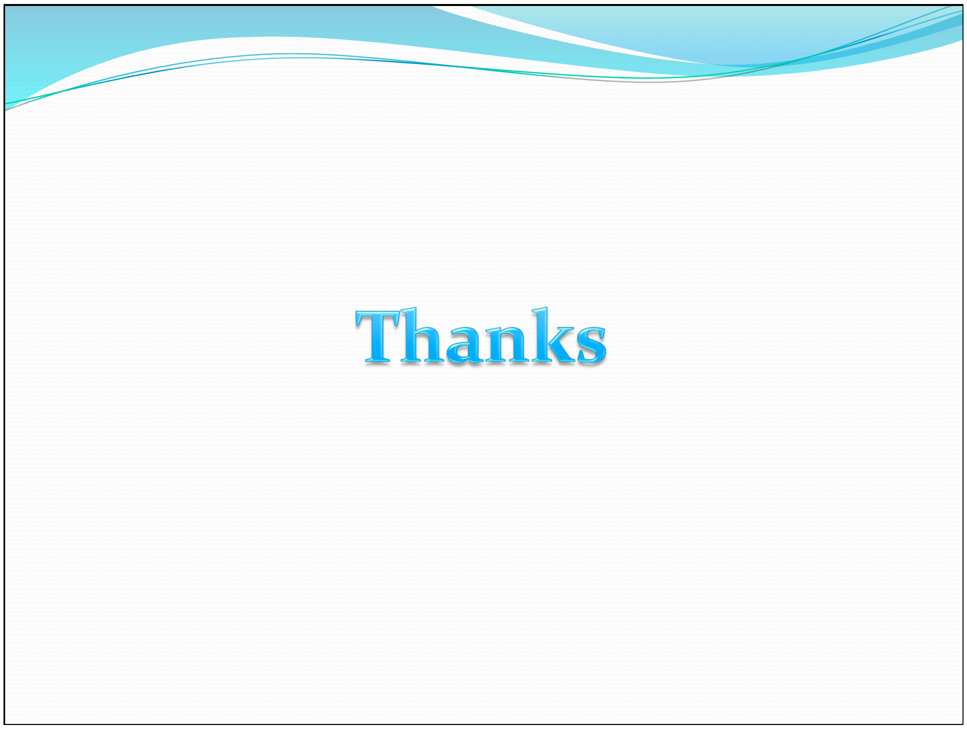
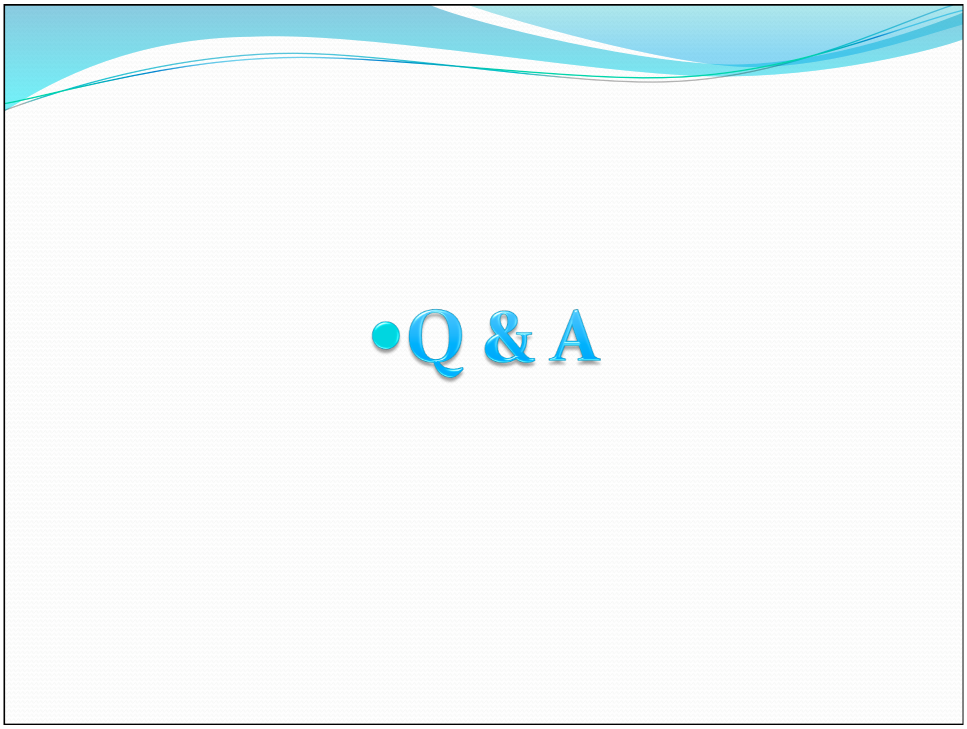


Figure 10: Average Synchronization Error (Left Y-axis) and Coverage (Right Y-axis) During Incremental Deployment ($t = 2$)



Thanks



- Q & A