T-Pack: Network Security in Real-Time Systems

Network Security & Attacks

- **Security:** Policies to detect and prevent intrusion and malware execution
- Two broad categories: **System Security** & **Network Security**
- **Network Attacks**
  - **Man-In-the-Middle (MITM)**
    - Eg. Relay and Replay attacks. (Common attacks in real time systems)
  - **Denial of service (DS)**
    - Eg. ping-of-death, syn-flooding etc.
- ***Delay Attacks*** (Common in real-time systems)
  - Delay in intended execution time of applications
- Network attacks induces delay in packet transmission
  - MITM increases end-to-end travel time by interfering.
  - DS stalls the flow of useful packet through network congestion. (Inc. travel time)

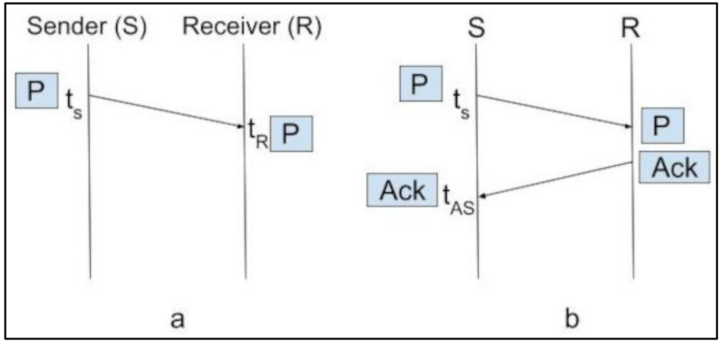
Takeaway, Motivation & Hypothesis

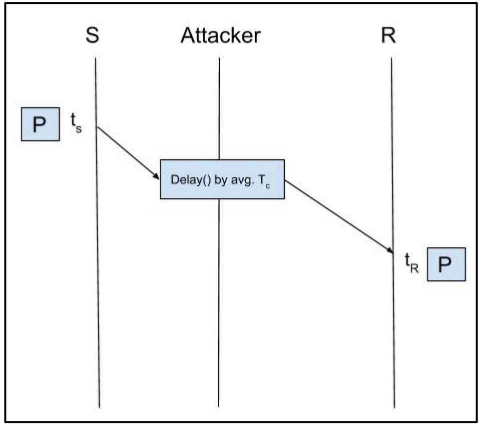
- Takeaway
  - Need to detect network attacks
    - Can cause delays (deadline misses)
- Motivation
  - **Early Detection of Network Delay** with **no cost** and **negligible performance** overhead for a Distributed Real-Time System Using **Simple** Monitoring Techniques
- Hypothesis
  - Monitoring round trip time at packet level in a distributed real-time system (periodic)
    - provides a means to detect network intrusions
    - complements conventional security methods.

Assumption & Attack Model

- Assumption
  - Distributed real-time system with end-to-end real-time guarantees of message transmission
    - Aware of the novel background traffic (Even the worst case)
    - Packet prioritization using TT-Ethernet (time-based traffic shaping) or SDN defined equipments
  - Public key encryption between subsystems
    - An attacker cannot modify novel packets
- Attack: Alter network behaviour
  - Additional, duplicate or dropped packets induces a time delay in the given real-time system

Design

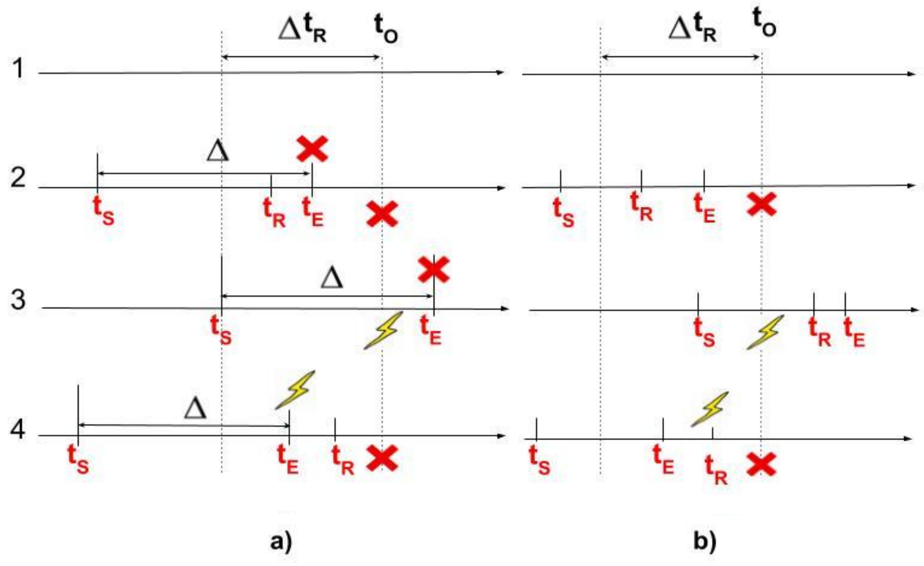




1. Expected End-to-End time(ETT) or Round-Trip Time (RTT) (Also T_{exp})
   a. ETT = $t_R - t_S + \Delta t_{rs}$ (clock offset) (UDP)
   b. RTT = $t_{AS} - t_S$ (TCP)
2. T_{exp} includes internal error $T_d + \Delta T_d$
   a. $T_{wcet} = T_{exp}$ if internal delay is $T_d + |\Delta T_d|$
3. Attacker induces delay $T_c + \Delta T_c$
   a. $T_{obs} = T_{exp} + T_c + \Delta T_c$
4. $T_{obs} > T_{wcet}$ indicates intrusion
5. $T_c + \Delta T_c + T_d + \Delta T_d \leq T_d + |\Delta T_d|$
   a. $T_{WCET} \geq T_{obs}$: vulnerability of T-Pack

- **TCP_QUICKACK & TCP_NODELAY (needed for packet level analysis)**
  - TCP Optimizations
    - Cumulative send and ack's (wait time at sender and receiver)
    - Non-deterministic execution in real-time systems
  - Controlled flow of packets in distributed real-time systems
    - TCP optimization provides only a marginal performance difference
  - Socket options: TCP_QUICKACK and TCP_NODELAY
    - to immediately transmit packet and acknowledgement.
  - Percentage of network utilization of transmission and receiver buffer for UAV paparazzi
    - (Observed for 60 sec period on Raspberry Pi 3 with preempt-RT Linux)
      - **With 'QUICKACK' & 'NODELAY' : 0.780% (tx) , 0.354% (rx)**
      - **Without 'QUICKACK' & 'NODELAY' : 0.754% (tx), 0.348% (rx)**
  - Easier for implementation of T-Pack

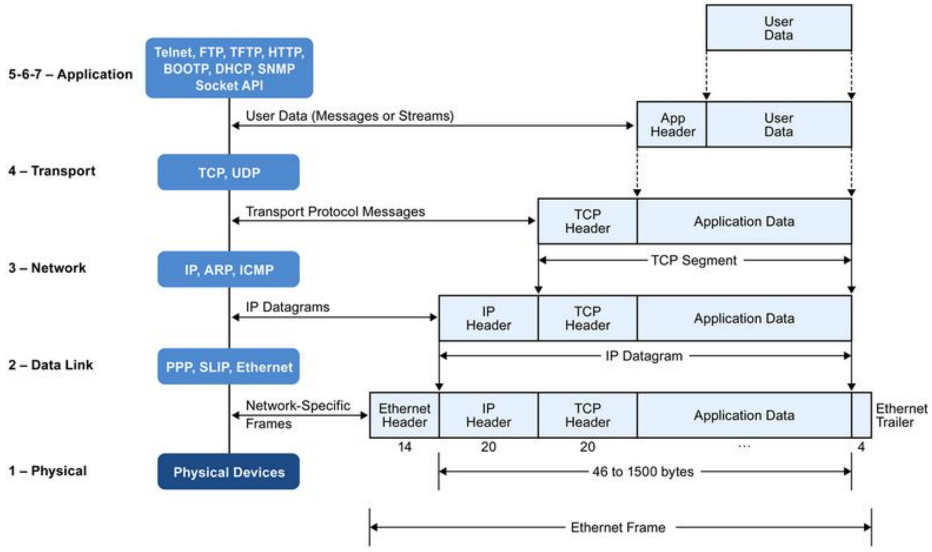
Comparison to Timeout Techniques



a) TCP b) UDP

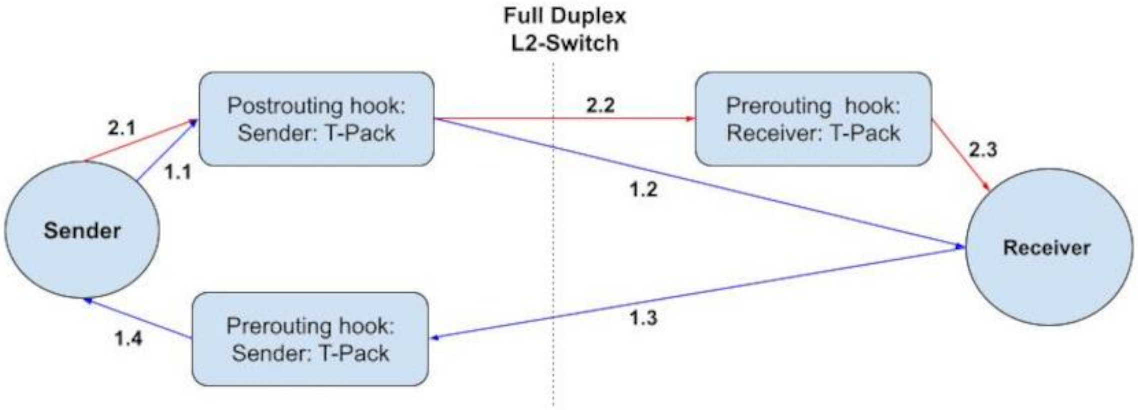
1. T-Pack works with global timeouts but providing early intrusion detection.
2. In time arrival of packet at t_R
    - Cancelling t_E and t_O
3. Long delay before packet sent or a lost packet detected by global timeout
    - Cancel t_E
    - T-Pack needs global timeouts
4. Packet sent early or simply late arrival of packet
    - Early timeout at t_E leaving t_O-t_E to transition into safe mode.
5. For UDP, T-Pack raises an exception at t_R for late arrival of a packet
    - Receiver unaware of the sender
    - t_R-t_O time to transition to safe mode
6. Early intrusion detection for T-Pack

Utilizing Linux Network Stack



- Linux packet data structure (Reading or updating packet data)
  - socket buffers (SKB)
- **Measuring time at lower layers of the network stack (Kernel level)**
  - Avoid transition time from user to kernel space
    - Packet time depends on read and write buffer capacity of sender and receiver
    - Earlier intrusion detection

Implementation

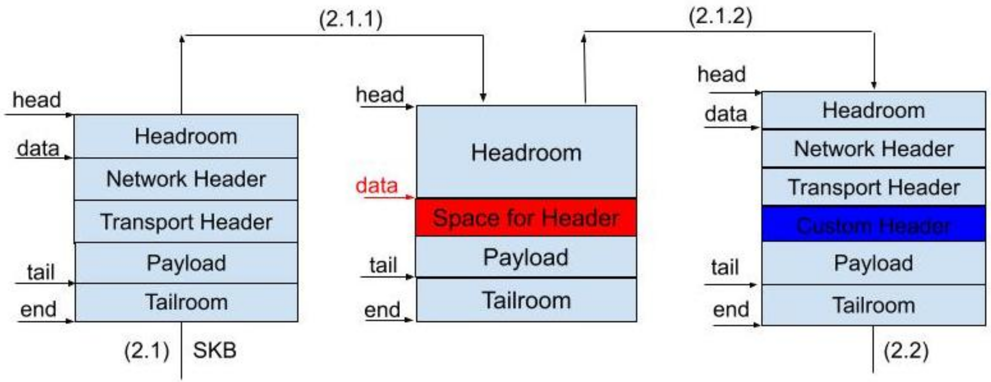


- Linux netfilter hooks execute T-Pack module
- UDP packets between sender and receiver in red.
- TCP between sender and receiver in blue.

Inside T-Pack: TCP

- Store current time in lookup table with IP address and port as key (monitoring each connection separately)
- Lookup table maintains Queues (For packet sent before ack received)
  - Send and Receive pointers
    - Send points to most recent packet sent (end of queue)
    - Receive points to packet waiting for Ack
  - Queue Nodes
    - Time
    - Expected sequence number
    - Next pointer

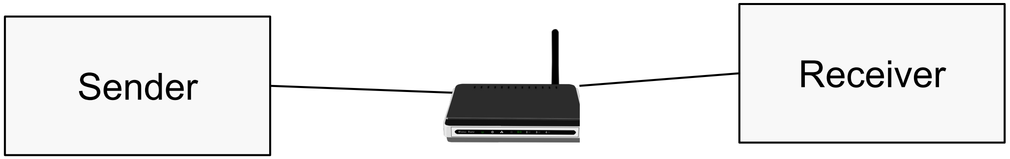
Inside T-Pack: UDP



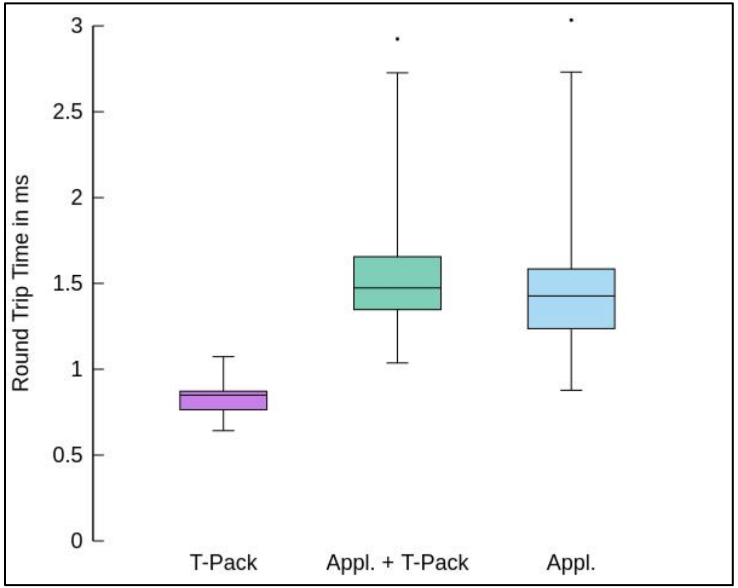
- SKB as parameter to T-Pack callback function (2.1)
- (2.1.1) `skb_pull+skb_push` functions to remove existing headers and create space for custom header
- (2.1.2) `memcpy` custom header and then removed headers
- Recalculate UDP and IP header checksum
- Custom Header
  - offset (clock sync), sender time (t_S), checksum (integrity of custom header)

Experiment 1 Framework

- Client Server Model
  - Periodic messages from client to server
    - Resembling time triggered real time system
  - TCP protocol
  - **demonstrates**
    - **performance overhead of T-Pack**
    - **benefit of recording timestamp information at the lower layer.**



Observed Round-Trip Time(RTT) - Exp 1

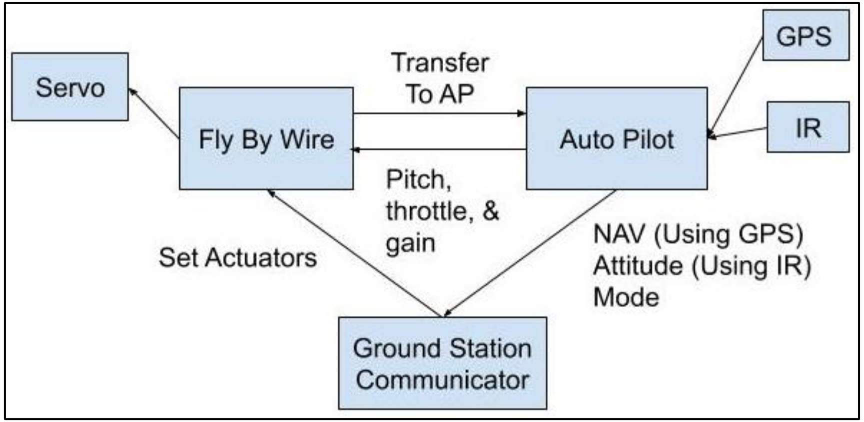


- Box plot for RTT
  - 1. Time recorded at network layer (T-Pack),
  - 2. Time recorded at Application layer with T-Pack running,
  - 3. Time recorded at Application layer without T-Pack running
- (2) and (3)
  - demonstrate overhead of introducing T-Pack.
    - Mean RTT increases by a marginal amount of approximately 0.09 msecs.
- (1) and (2)
  - Demonstrate time packet spends between application layer and network layer
    - At sender and receiver
  - Early intrusion detection at network layer (~0.3 msecs)

T-Pack vs Baseline

- **Why not monitor packet at the application layer (Not just delayed intrusion detection)?**
  - Explicit reply packets needed to replicate acknowledgement
    - Twice the number of packets saturates write buffer of receiver (delay)
      - Increase in RTT (T_{exp}) (as increase in internal delay)
    - Higher acceptable RTT because of transitions from lower to upper layers of network stack and vice versa
    - Higher RTT results in more false negatives
- DDOS attack on the setup in experiment 1
  - 1 Attacker sending 100 bytes of ICMP packets with 0.001 sec interval on 10 parallel threads
  - Observed over 300 packet; Baseline (~289 false negatives) & T-Pack (~130 false negatives)

Experiment 2 Framework

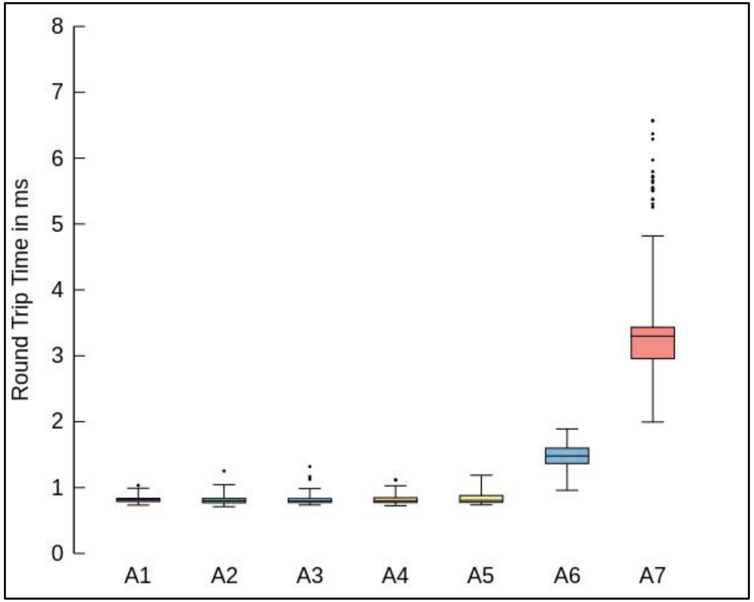


- UAV Paparazzi model
  - Unmanned aerial vehicle control software
  - Models a real-time system based on a traditional shared memory of a real-time control systems.
  - A peer-to-peer network of 3 subsystems,
    - auto pilot (AP), fly by wire (FBW) control and ground station communicator (GSC)
  - Model maintains the similar communication flow as the actual UAV paparazzi
  - Replaces processing and calculation by each subsystems with sleep operations
  - Socket operations work as intended
- Demonstrates ability of T-Pack to detect delay due to different intensities of distributed-denial-of-service attack on the network

Attack Introduced in Experiment 2 and 3

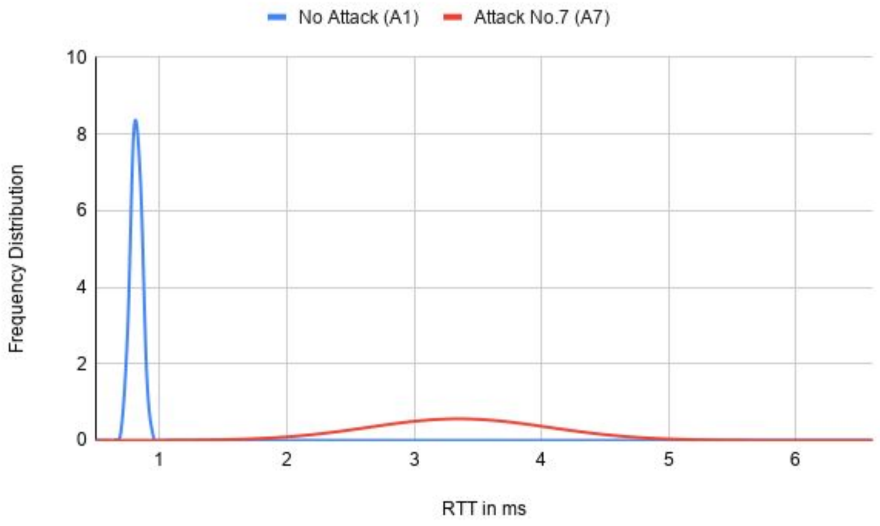
- Distributed denial of service: ping-of-death
- Attack vector defined as P(n,t,b,i)
  - P: ping
  - n: number of attackers on the network
  - t : No. of parallel threads on each attacker executing the attack
  - b: byte size of each icmp packet
  - i: time interval between each packet in seconds
- Attack Vectors
  - Attack 1: P(0,0,0,0) (No Attack)
  - Attack 2: P(1,10,500,0.5)
  - Attack 3: P(1,10,500,0.1)
  - Attack 4: P(2,10,500,0.1)
  - Attack 5: P(2,30,500,0.05)
  - Attack 6: P(2,10,500,0) ~ ping flood
  - Attack 7: P(2,30,1000,0.001) ~ ping flood

Observed Round-Trip Time(RTT) Against Different Attack Parameters - Exp2



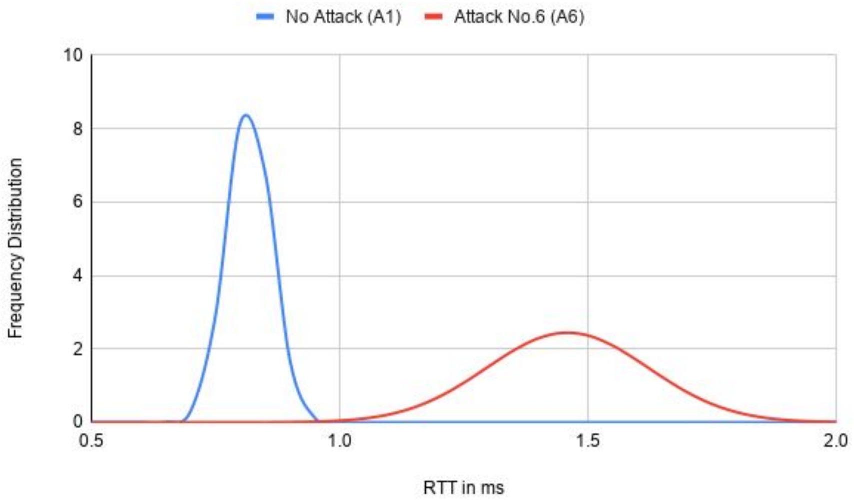
- Monitoring packets using T-Pack
  - between autopilot and ground station communicator for the UAV Paparazzi model.
- Introducing ddos attack with various attack vectors as show in the Figure.
- Increase in the attack intensity increases RTT
- Worst case RTT without attack is less than all values of RTT with UAV under Attack 7, P(2,30,1000,0.001)
  - demonstrates 100% detection of a compromised network by T-Pack

Frequency Distribution of RTT within UAV under No Attack and Attack 7 - Exp2



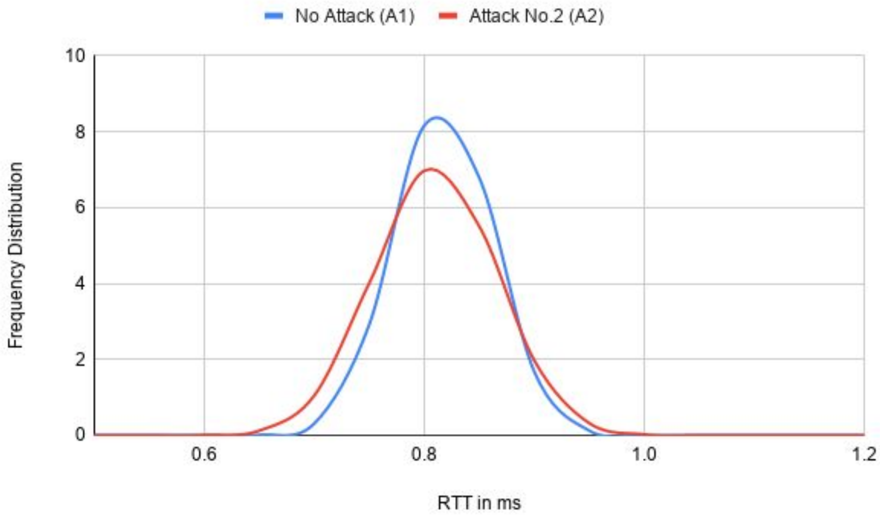
- Number of times (y-axis) and RTT is observed (x-axis)
- Results indicate no intersection between the distributions
  - both cases can be discreetly distinguished
- Attack with similar time delay effects as Attack 7, P(2,30,1000,0.001) will always be detected by T-Pack.

Frequency Distribution of RTT within UAV under No Attack and Attack 6 - Exp2



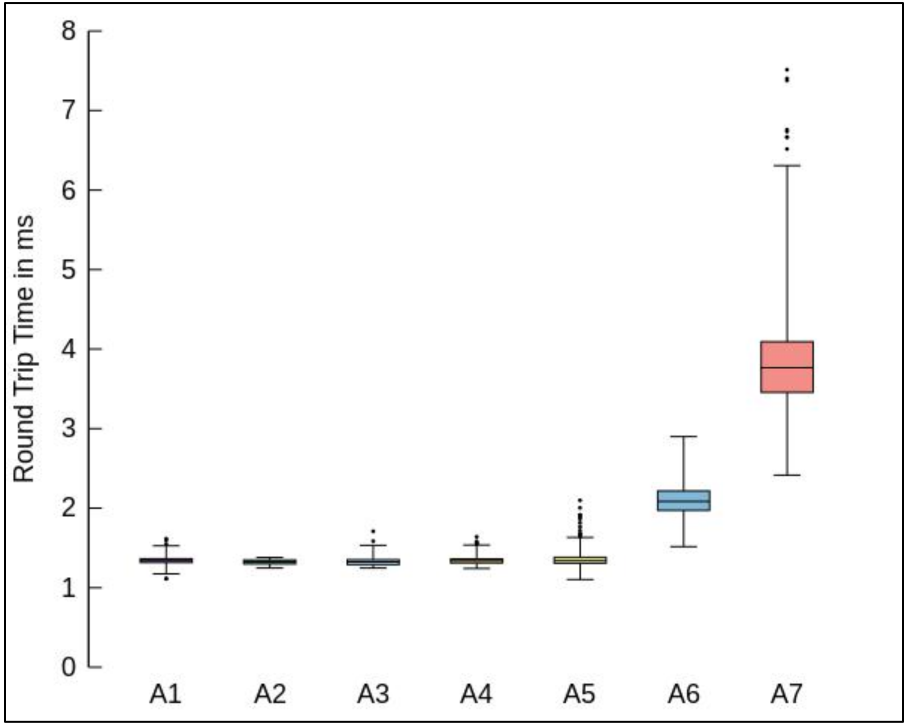
- Number of times (y-axis) and RTT is observed (x-axis)
- Results indicate ~1% intersection between the distributions.
- <1% of the samples fall into the 0.9-0.95 msecs range
  - >99% of the attacks would be detected by T-Pack
    - for a compromised network with time delay similar to that caused by Attack 6, P(2,10,500,0)

Frequency Distribution of RTT within UAV under No Attack and Attack 2 - Exp2



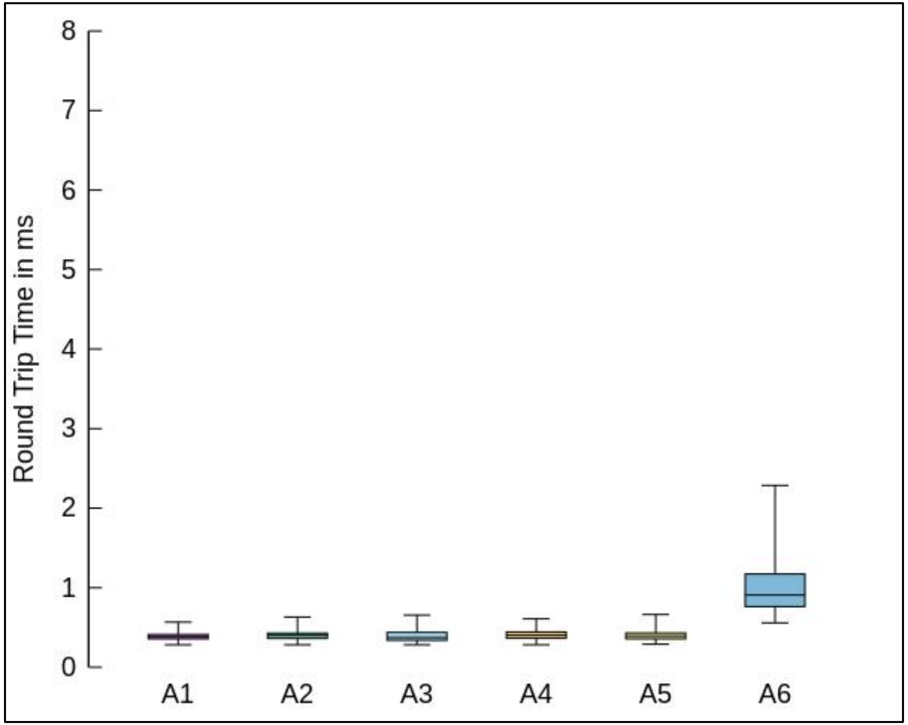
- Number of times (y-axis) and RTT is observed (x-axis)
- Results indicate significant overlap between distributions (~99%) b/w 0.65-0.85 msecs range.
- Illustrates limitations of T-Pack.
  - Any attack with similar delays of
    - Attack 2, P(1,10,500,0.5)
  - Will not be detected by T-Pack
- T-Pack can only complement other system security methods,
  - Does not replace them as not a panacea for intrusions.

Compatibility with Encryption Methods - IPSec



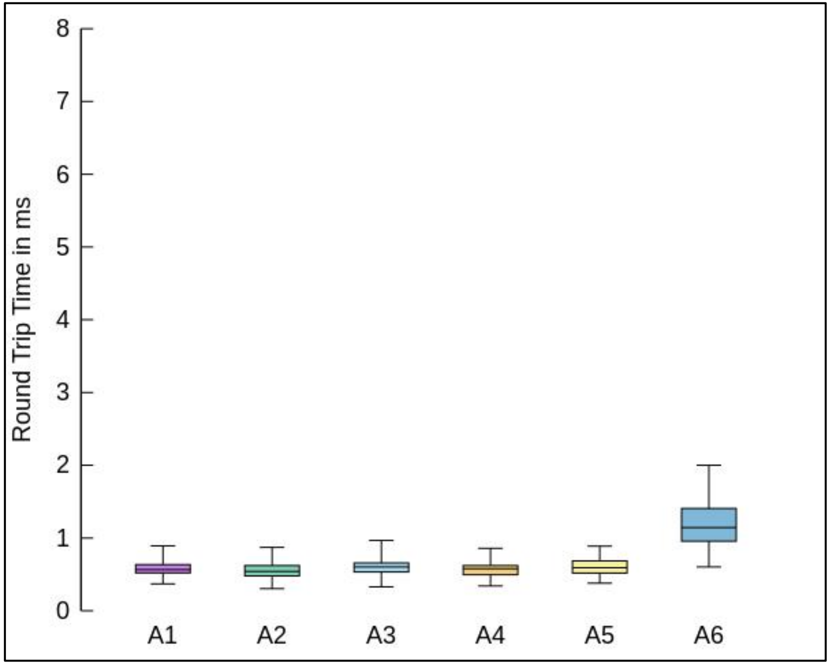
- Network layer security protocol
- Encrypts payload (everything after network header) after T-Pack implementation
- Decrypts the packet before extracting information using T-Pack.
- Protects against packet modification
  - Cannot prevent induced delays
  - T-Packs complements IPSec
- Results show T-Packs compatibility with other security protocols
- Similar trend in RTT values
  - Slightly higher due to encryption overhead

T-Pack with UDP



- Demonstrates ability of T-Pack to comply with different communication protocol
- Requires clock offset to calculate ETT (end-to-end trip time)
- Implemented NTP analogy within T-Pack
  - Clock offset calculated is used to estimate ETT
- Periodic re-synchronizations
  - Linear increase in clock-offset due to clock drift
- Results demonstrate similar trend in T-Pack for paparazzi UAV over UDP.

T-Pack for Paparazzi UAV over UDP & IPSec

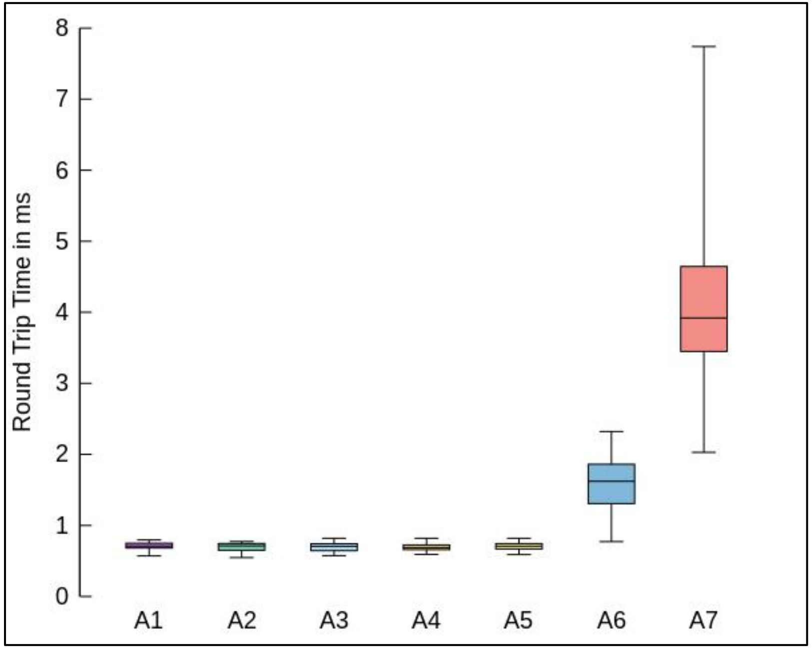


- Consistency of T-Pack over UDP with IPSec enabled
- Avg. ETT values increase slightly due to encryption compared to T-Pack over UDP without IPSec (prev. slide)

Experiment 3 Framework

- Waters Workshop Challenge 2018
  - **Drone like multisystem** demonstrating **time triggered distributed real time system**
    - A peer-to-peer network of seven subsystems (TCP):
      - Mission management system (MMS), Electrical Propulsion System (EPS), Hydraulic Braking System (HBS), Sensors (communicating also with other sensors in the Waters model), Ground Station and Maintenance System
    - Model functions and communication patterns in each subsystem
    - **The RTT monitored between EPS and MMS**
    - Attack between the two subsystems is induced to analyze the accuracy of T-Pack during the attack
    - **Demonstrates consistency in detecting delay due to distributed denial-of-service attack by T-Pack.**

Observed Round-Trip Time(RTT) Against Different Attack Parameters - Exp3



- Similar to the UAV Paparazzi model
  - T-Pack fully detects Attack 7, P(2,3,1000,0.001)
- Consistency in results compared to the UAV Paparazzi model
- Demonstrates Consistency of T-Pack with other distributed real-time systems

Conclusion

- Experimental results indicated that **T-Pack successfully detected malware intrusion with almost 100% accuracy**
  - For attacks with similar intensity as Attack 7: P(2,30,1000,0.001)
- **Better compared to baseline**
- **T-Pack can be implemented at no cost with only a small overhead**
- **Implementation and results demonstrated in this work support the hypothesis**
  - Monitoring round trip time at packet level in a real-time distributed system (periodic)
    - provides a means to detect network intrusions complementing conventional security methods.

Thank You!