From High-Level Types to Low-Level Monitors: Synthesizing Verified Runtime Checkers for MAVLink

Arthur Amorim, Paul Gazzillo, Max Taylor, and Lance Joneckis.

Five inputs in, certified C monitor out — 4x faster and 40% leaner than the state of the art

Cyber-Physical Systems Are Now Targets for Physical Damage via Cyber Attacks

Stuxnet

5 OCTOBER 1986 No 8,461 Price 50p

THE SUNDAY TIMES

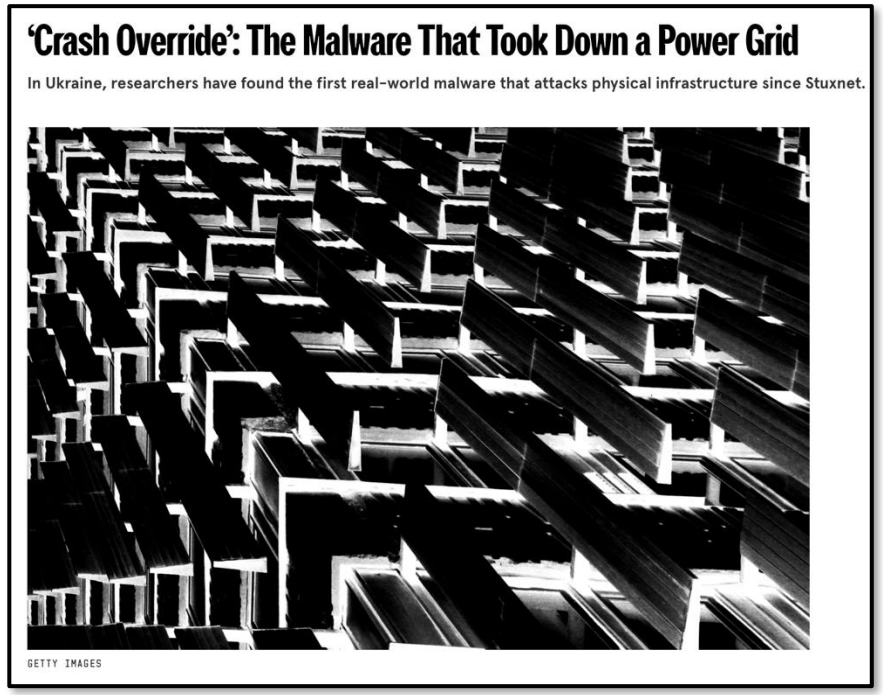
Revealed: the secrets of Israel's nuclear arsenal

INSIGHT

Pictured for the first time in detail: Israel's nuclear bomb plant is built deep beneath the Negev desert and has been secretly operating for more than 20 years

Inside Dimona: full story pages 2 & 3

Crash Override



'Crash Override': The Malware That Took Down a Power Grid

In Ukraine, researchers have found the first real-world malware that attacks physical infrastructure since Stuxnet.

GETTY IMAGES

German Steel Mill

TECH + ENGINEERING

Cyber Attack on German Steel Mill Leads to 'Massive' Real World Damage

A steel mill in Germany lost control of its blast furnace. Hackers had infiltrated the mill's control system, according to the German government's office for information security.

BY R.A. BECKER THURSDAY, JANUARY 8, 2015 NOVA NEXT

Idaho National Laboratory

Stealthy Attacks Use Only Valid Commands in the Wrong Order

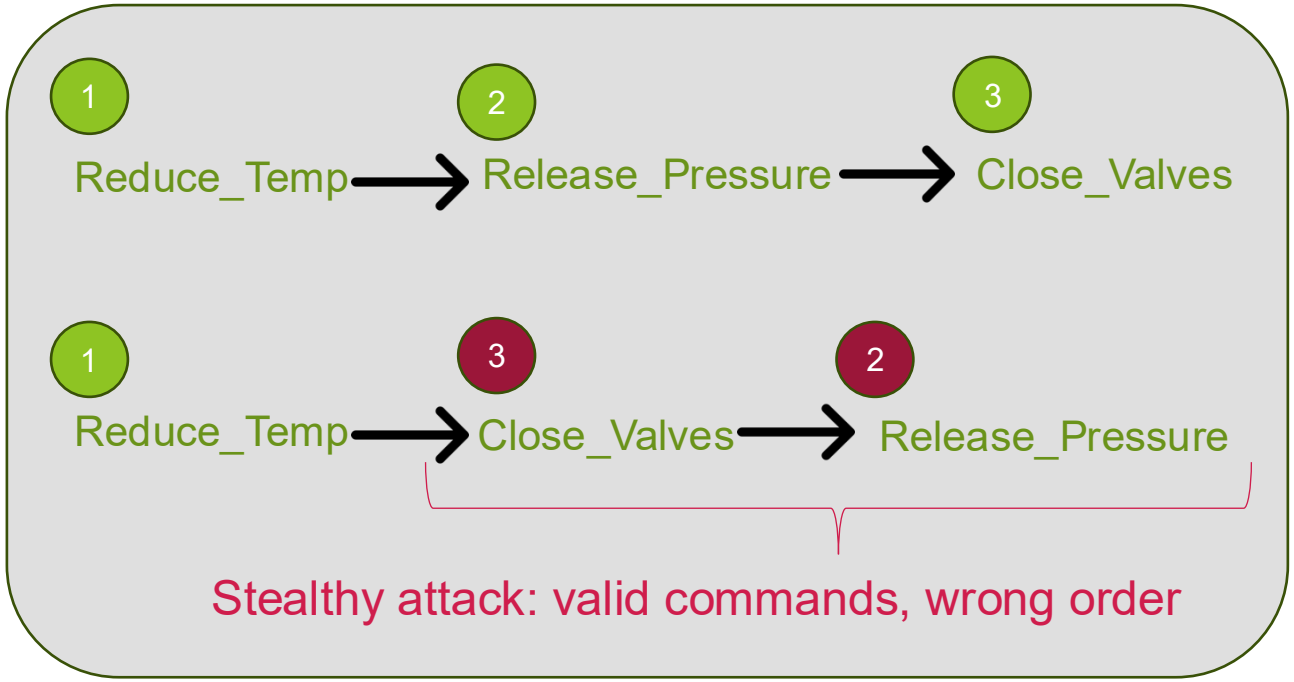
TECH + ENGINEERING

Cyber Attack on German Steel Mill Leads to 'Massive' Real World Damage

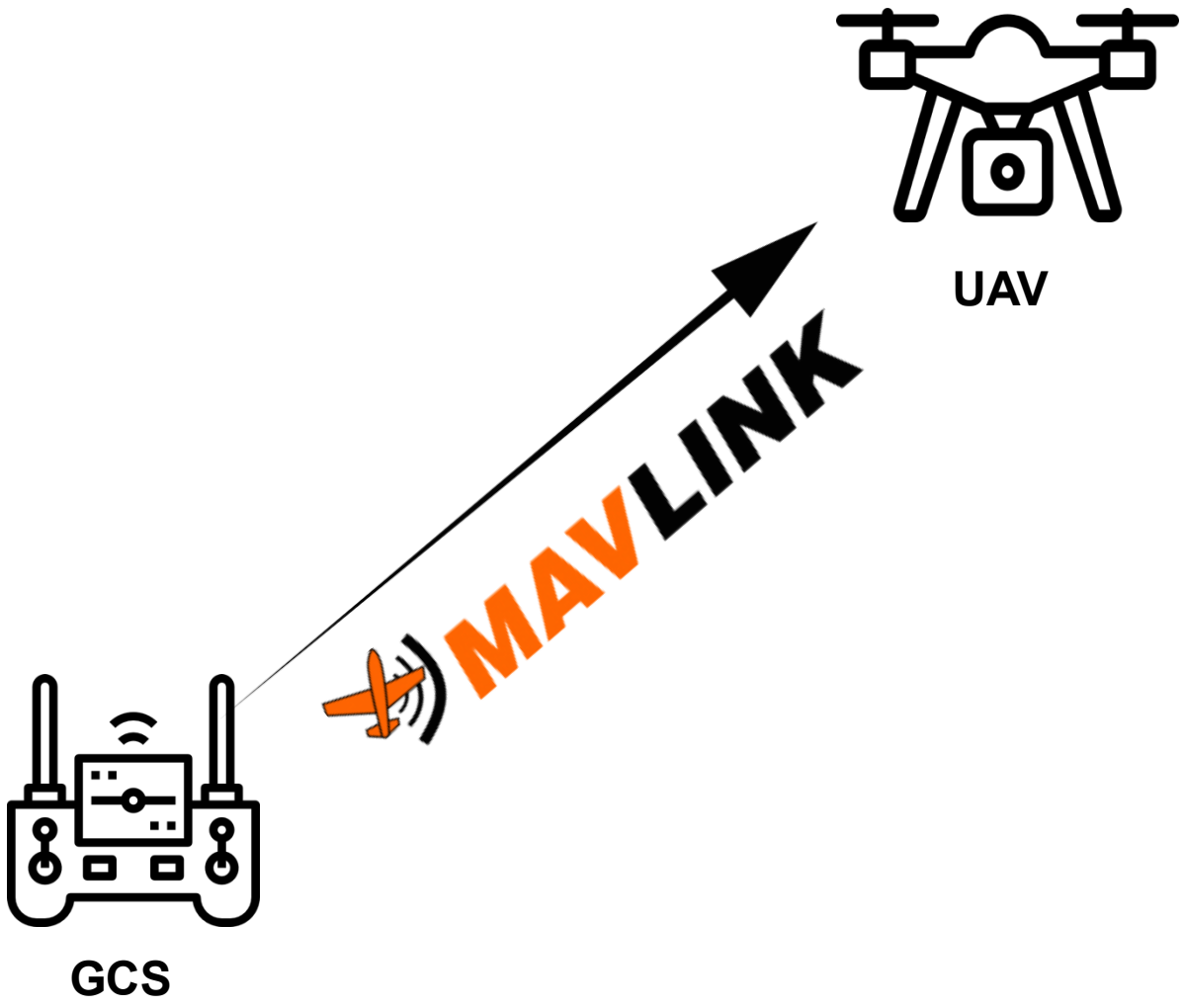
A steel mill in Germany lost control of its blast furnace. Hackers had infiltrated the mill's control system, according to the German government's office for information security.

BY R.A. BECKER THURSDAY, JANUARY 8, 2015 NOVA NEXT

Blasting Furnace Shutdown Sequence:



Widely Used Protocols Are Susceptible to Stealthy Attacks



RVFUZZER: Finding Input Validation Bugs in Robotic Vehicles Through Control-Guided Testing

Taegyu Kim†, Chung Hwan Kim*, Junghwan Rhee*, Fan Fei†, Zhan Tu†, Gregory Walkup†
Xiangyu Zhang†, Xinyan Deng†, Dongyan Xu†
†*Purdue University, {tgkim, feif, tu17, gwalkup, xyzhang, xdeng, dxu}@purdue.edu*
**NEC Laboratories America, {chungkim, rhee}@nec-labs.com*

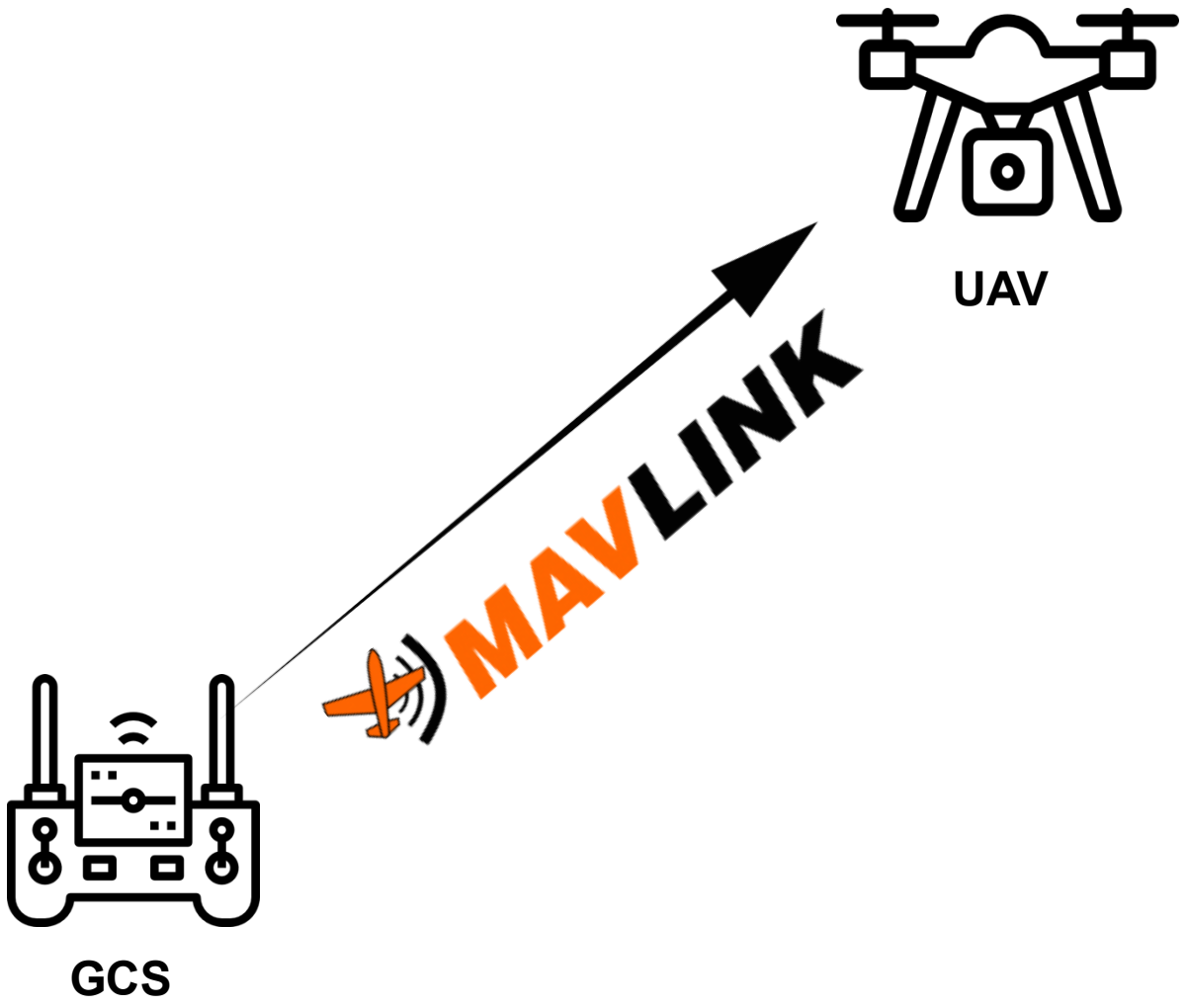
Found 89 input validation bugs

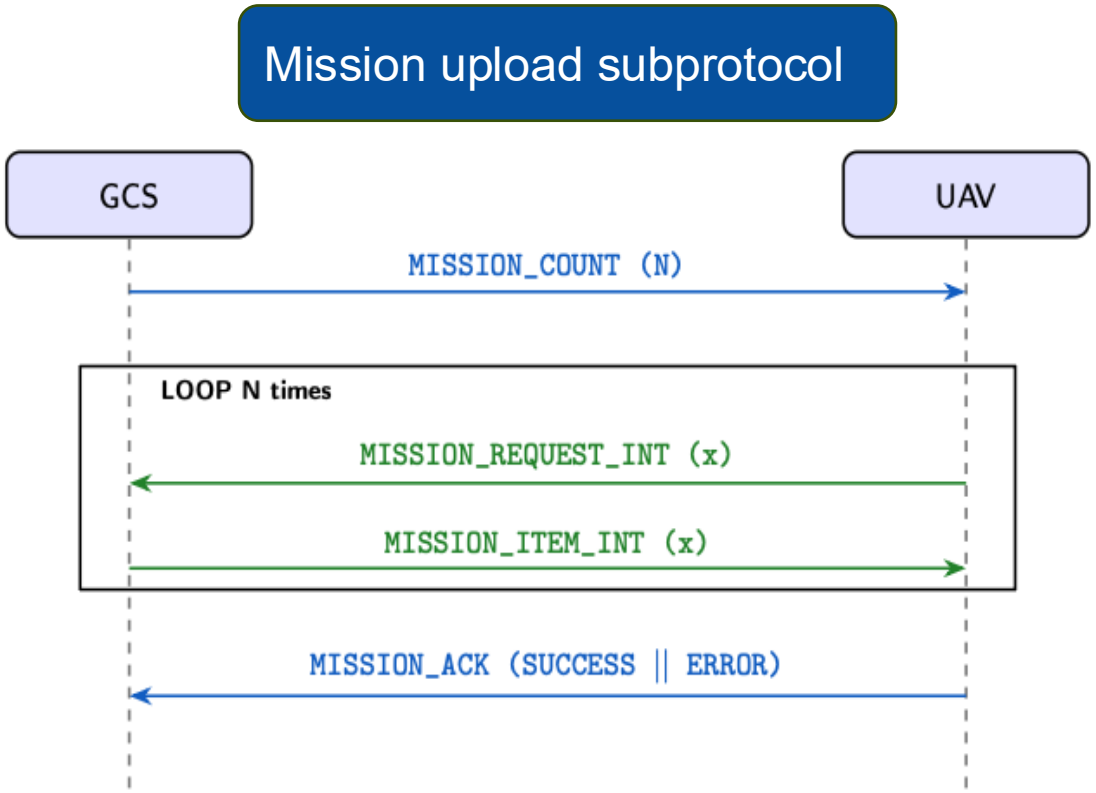
PGFUZZ: Policy-Guided Fuzzing for Robotic Vehicles

Hyungsub Kim, Muslum Ozgur Ozmen, Antonio Bianchi, Z. Berkay Celik, and Dongyan Xu
Purdue University
{kim2956, mozmen, antoniob, zcelik, dxu}@purdue.edu

Found 156 policy violation bugs

Widely Used Protocols Are Susceptible to Stealthy Attacks

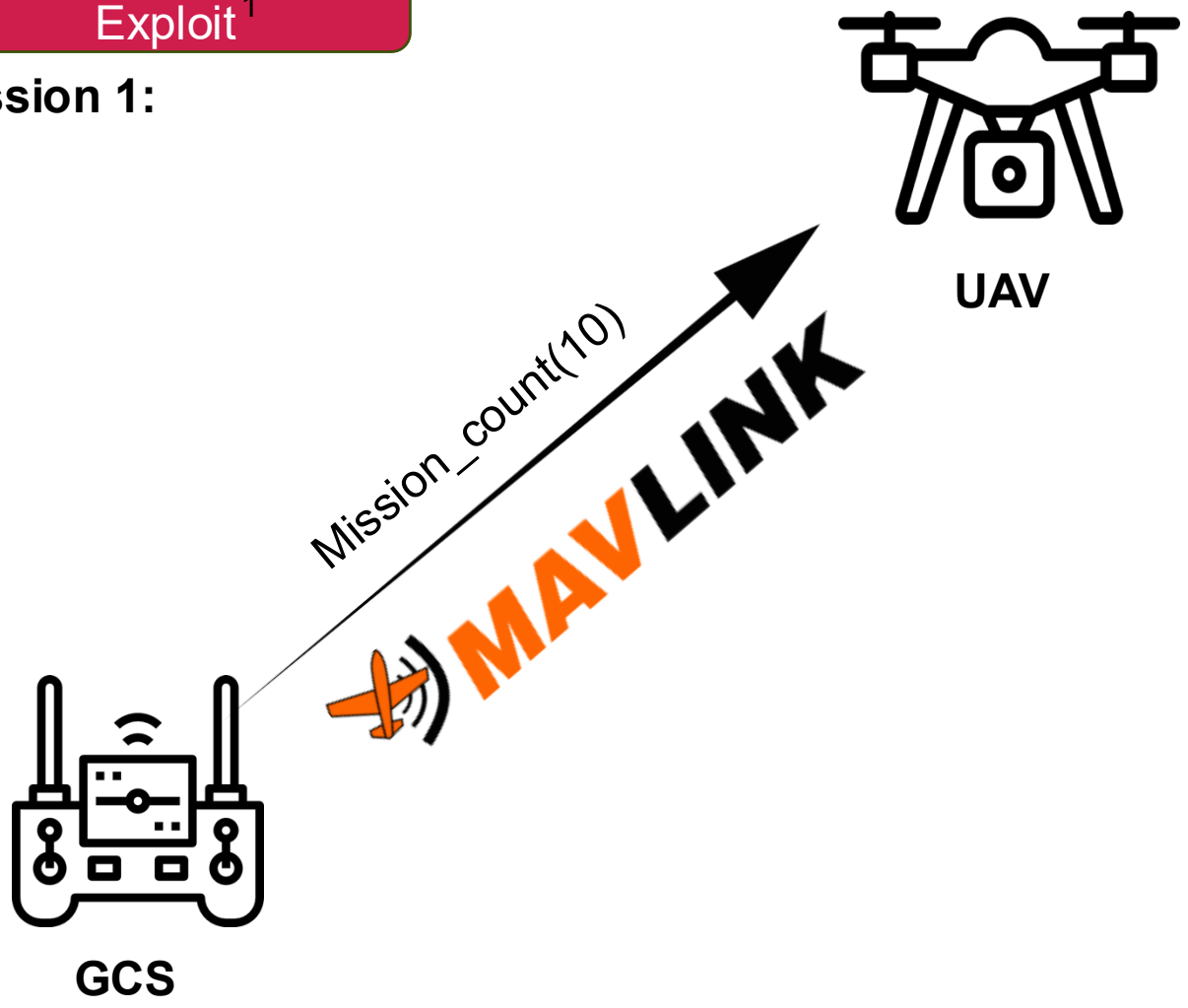




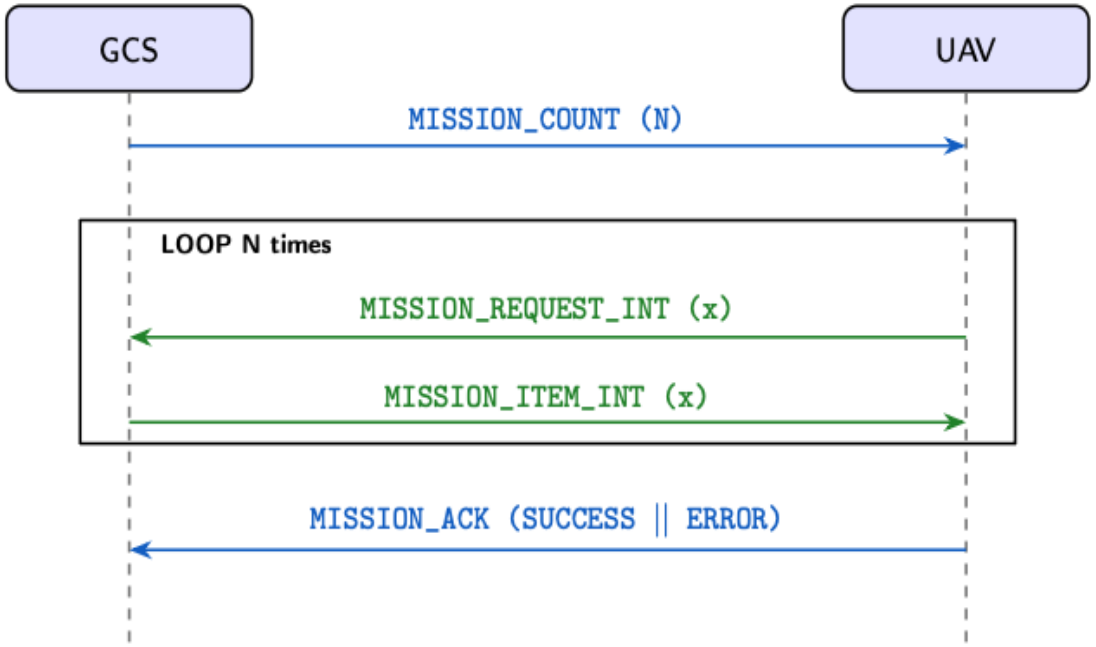
Widely Used Protocols Are Susceptible to Stealthy Attacks

Resource Usage Exploit[1]

Mission 1:



Mission upload subprotocol

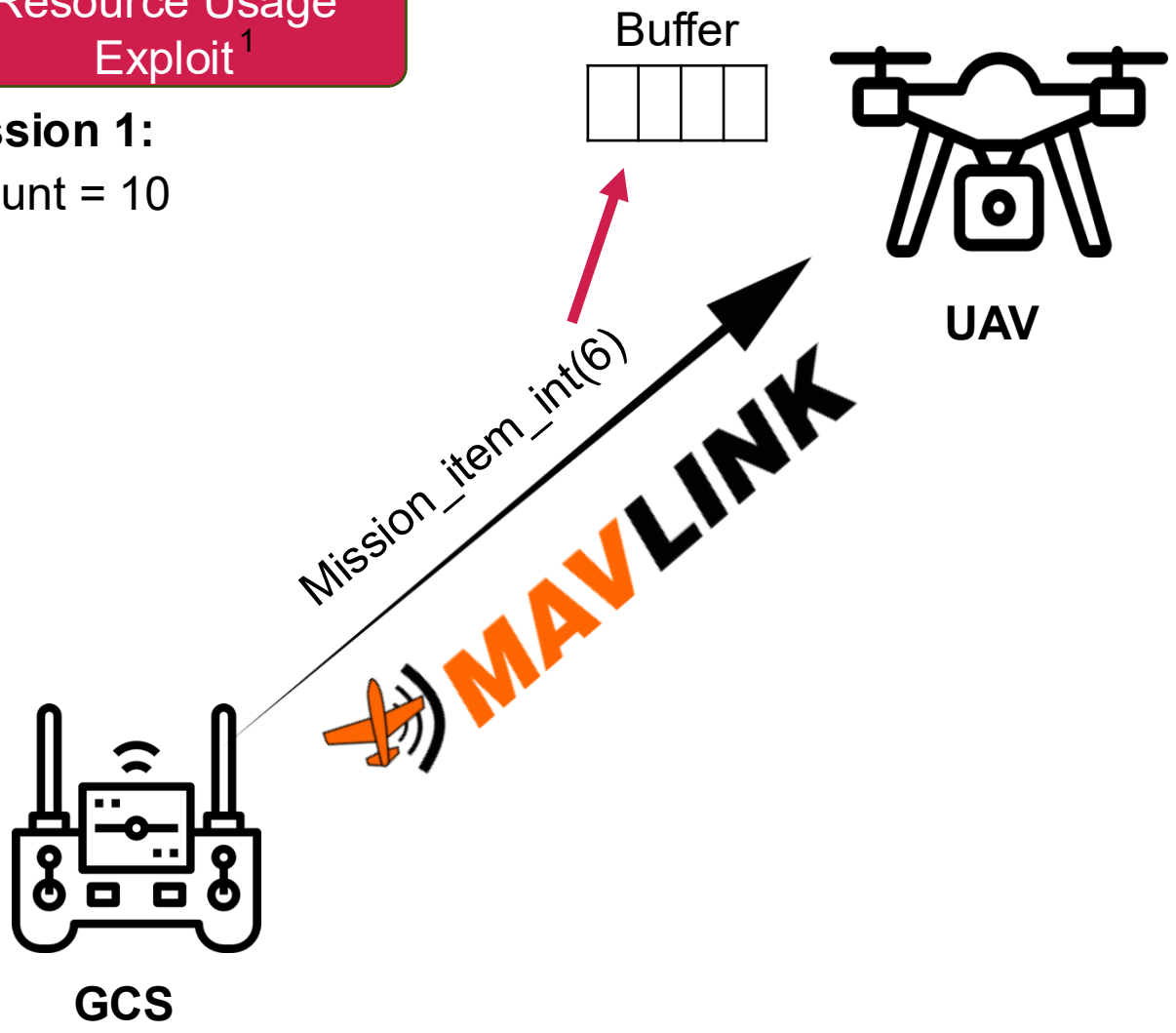


1- https://github.com/ArduPilot/MissionPlanner/issues/1248

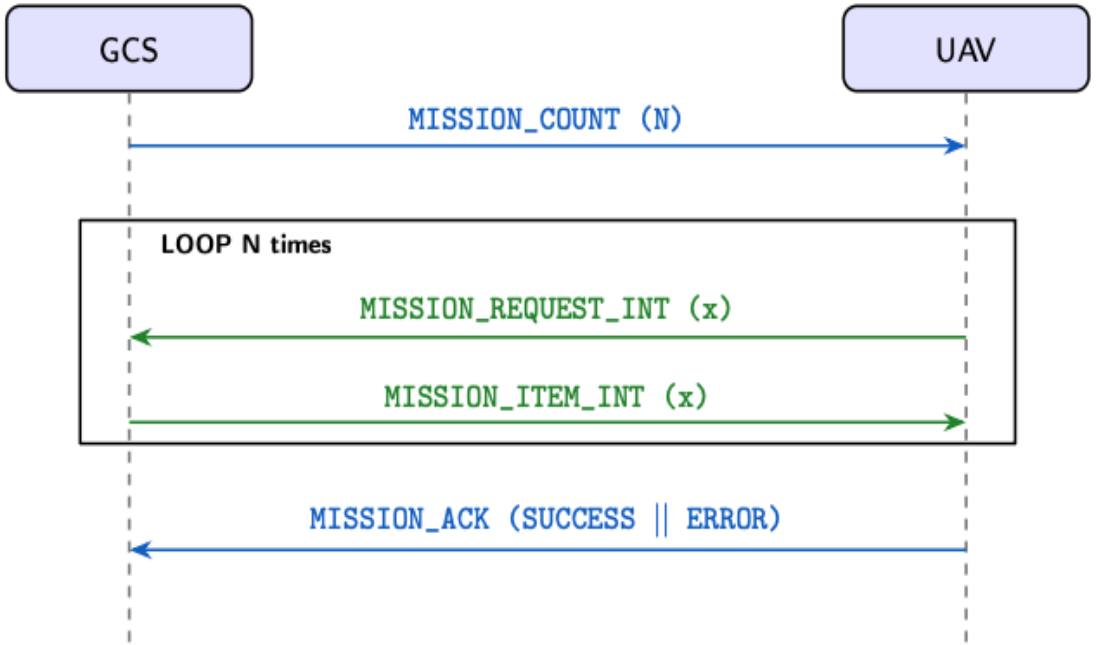
Widely Used Protocols Are Susceptible to Stealthy Attacks

Resource Usage Exploit[1]

Mission 1:
Count = 10



Mission upload subprotocol



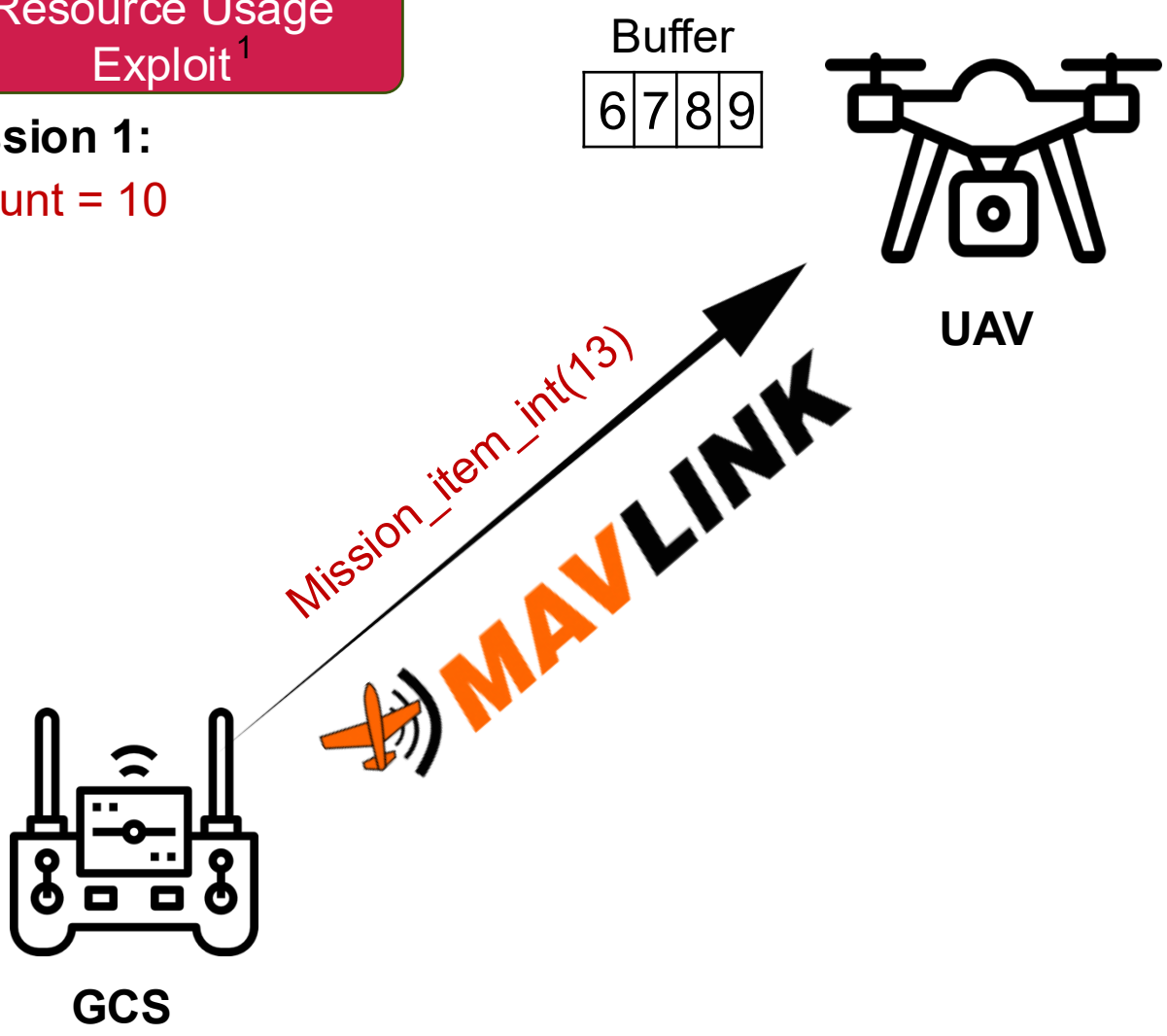
1- https://github.com/ArduPilot/MissionPlanner/issues/1248

Widely Used Protocols Are Susceptible to Stealthy Attacks

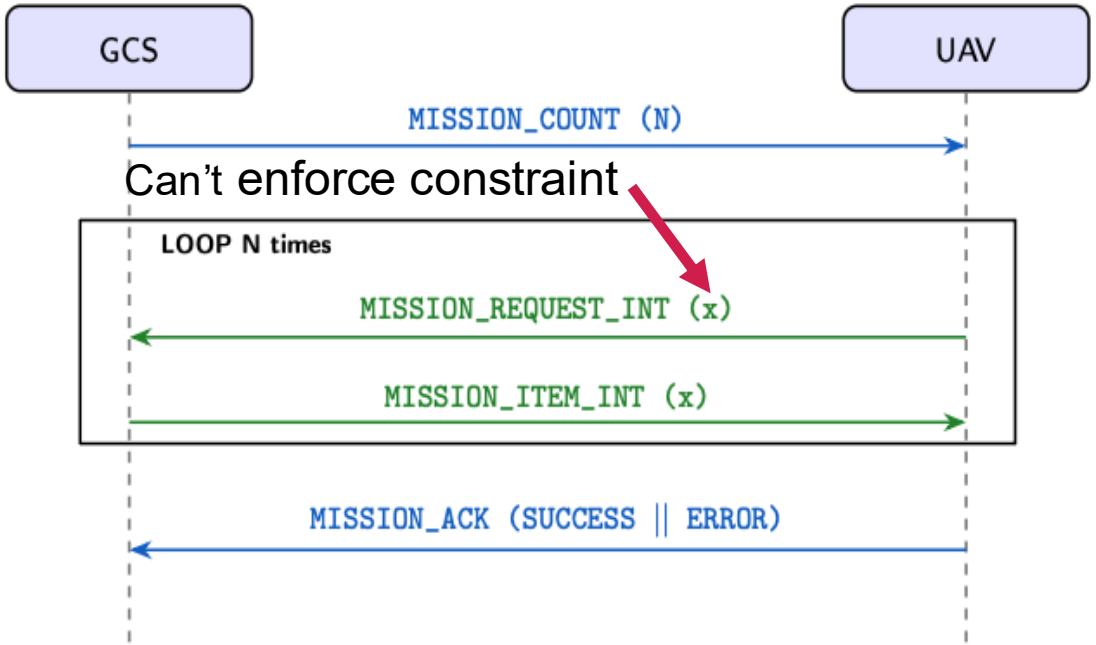
Resource Usage Exploit[1]

Mission 1:

Count = 10



Mission upload subprotocol



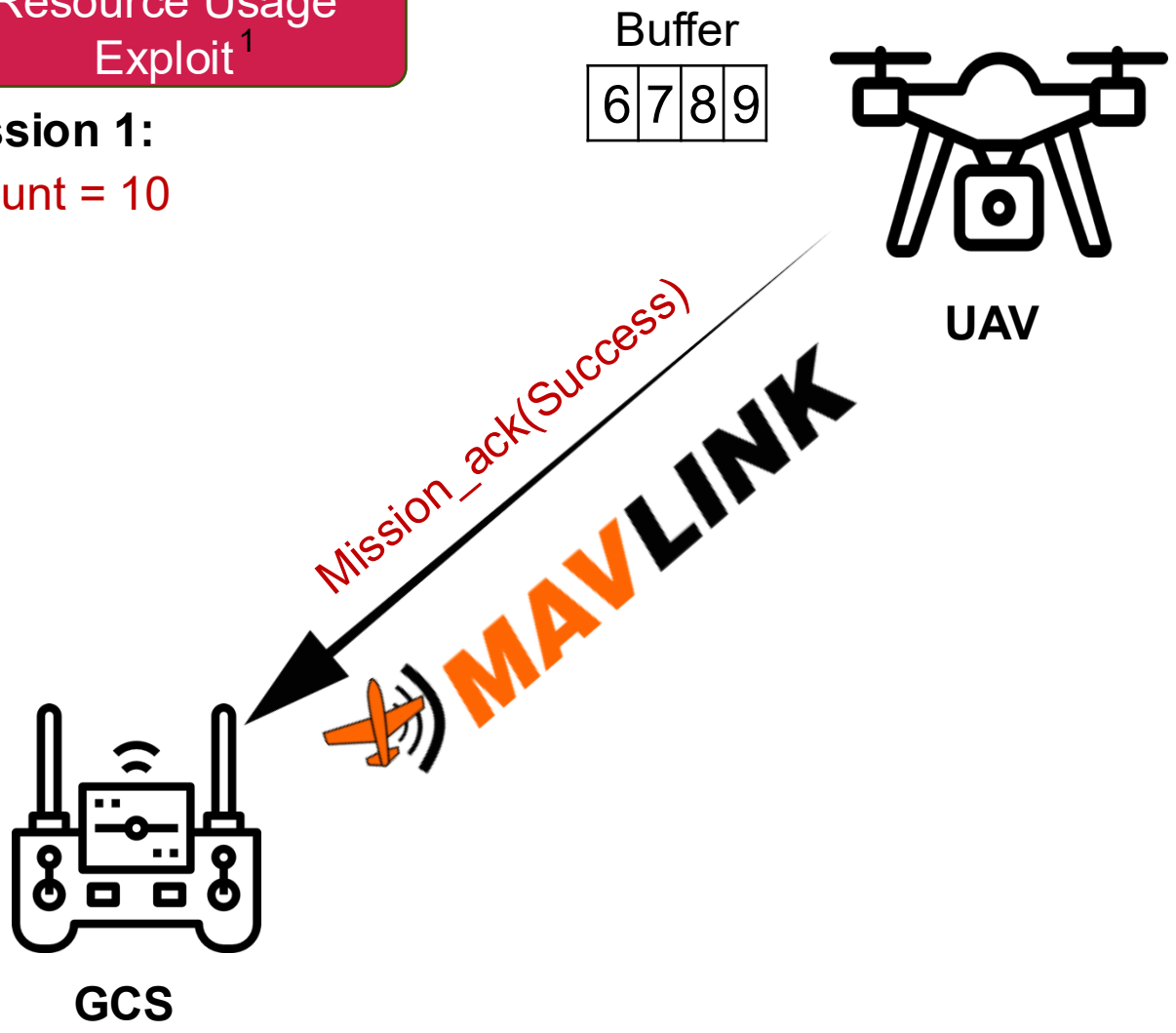
1- https://github.com/ArduPilot/MissionPlanner/issues/1248

Widely Used Protocols Are Susceptible to Stealthy Attacks

Resource Usage Exploit[1]

Mission 1:

Count = 10



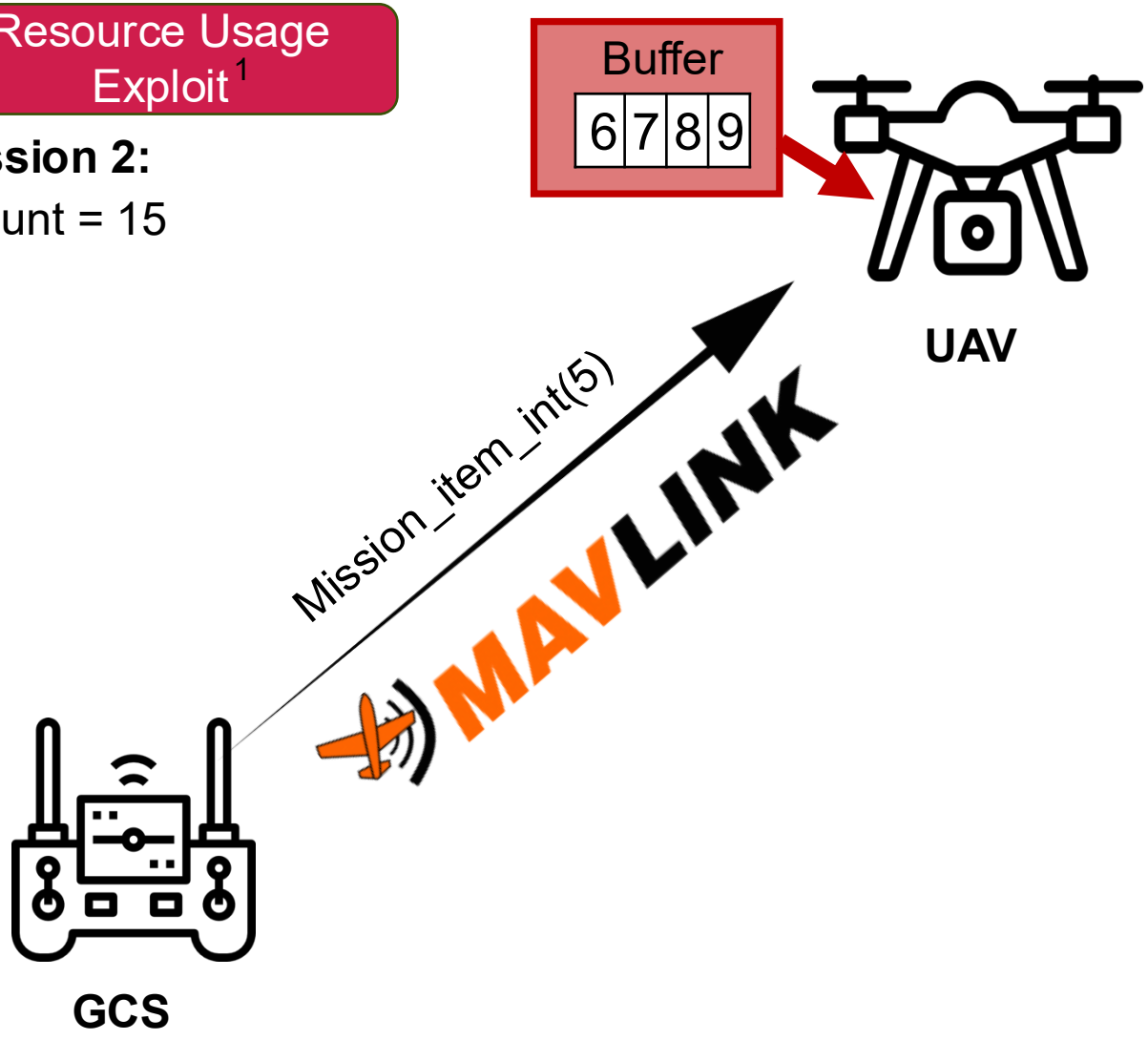
1- https://github.com/ArduPilot/MissionPlanner/issues/1248

Widely Used Protocols Are Susceptible to Stealthy Attacks

Resource Usage Exploit[1]

Mission 2:

Count = 15



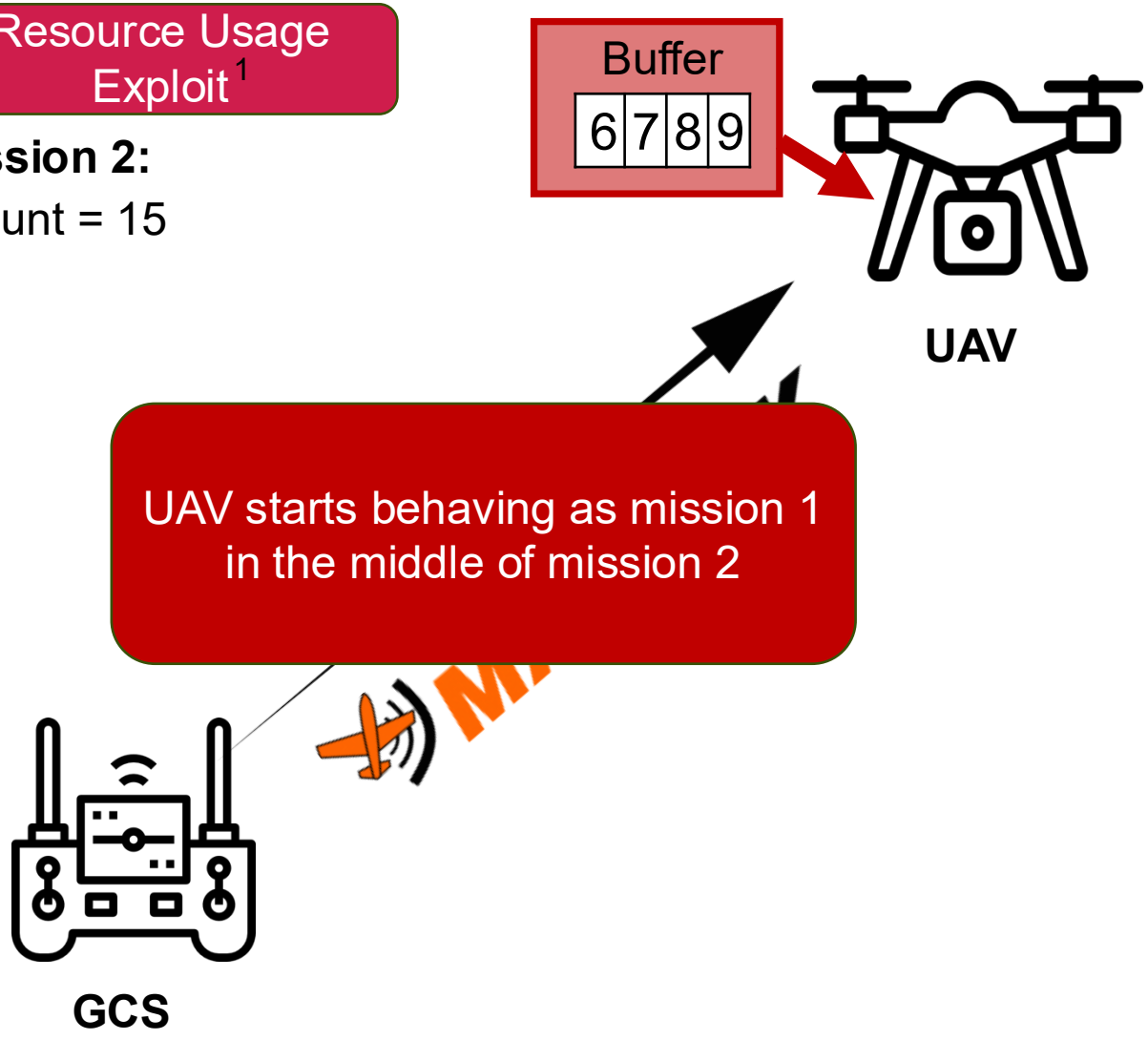
1- https://github.com/ArduPilot/MissionPlanner/issues/1248

Widely Used Protocols Are Susceptible to Stealthy Attacks

Resource Usage Exploit[1]

Mission 2:

Count = 15

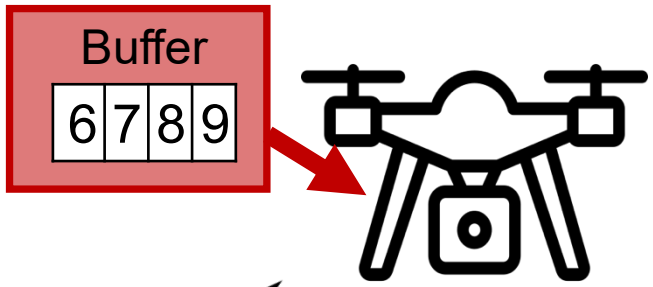


1- https://github.com/ArduPilot/MissionPlanner/issues/1248

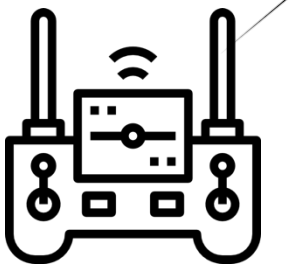
Existing Defenses Miss the Logical Layer

Resource Usage Exploit[1]

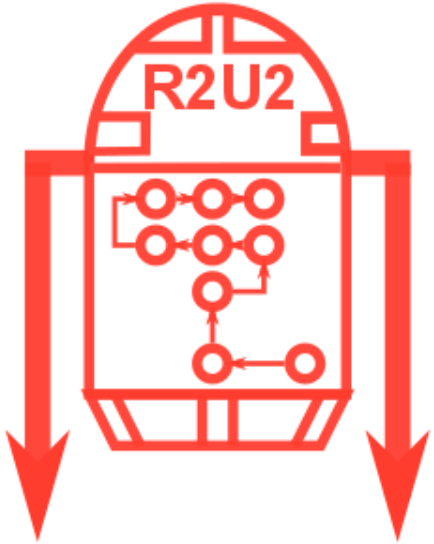
Mission 2:
Count = 15



UAV starts behaving as mission 1 in the middle of mission 2



Problem: A stealthy attack uses the protocol exactly as specified to induce undesired behavior, which makes it invisible to the defenses we usually rely on



Intrusion detection systems: No anomalous behavior or virus signature

Physics based anomaly detection: no anomalous physical deviation.

1- https://github.com/ArduPilot/MissionPlanner/issues/1248

DATUM: Proof of Concept, Two Blockers to Deployment

Global Refined Multiparty Session Types (GRMPST)

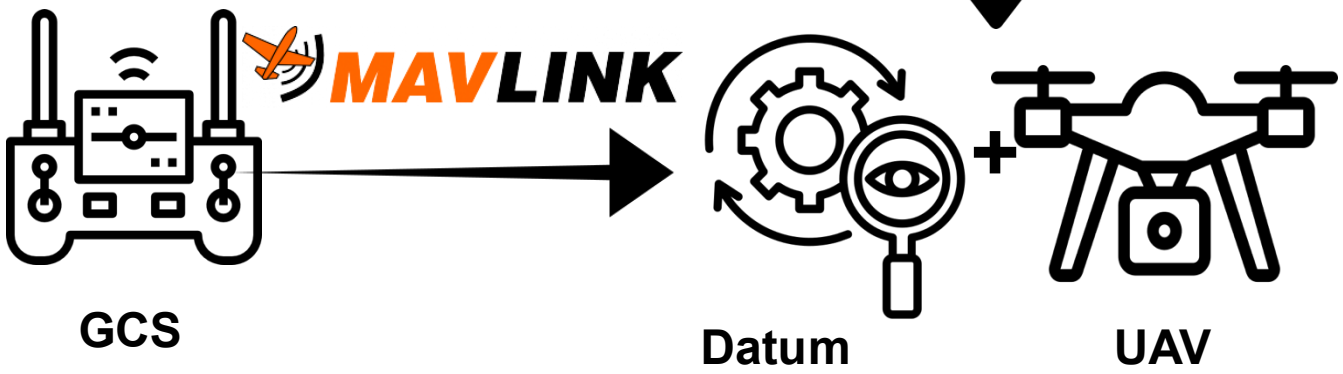
$$
\begin{aligned}
&\mathtt{GCS} \rightarrow \mathtt{UAV}:\mathtt{MISSION\ COUNT}(n\{1 \leq n < \mathrm{LIMIT}\}) \\
&\mu.T(curr\{0 \leq curr < n\})(curr = 0) \\
&\quad \mathbf{UAV} \rightarrow \mathbf{GCS}: \\
&\qquad \mathtt{MISSION\ REQUEST\ INT}(req\{req = curr < n\}) \\
&\qquad \mathtt{MISSION\ ITEM\ INT}(item\{item = curr < n\}) \\
&\qquad T(curr + 1) \\
&\quad \oplus \mathtt{MISSION\ ACK}(ack\{\mathrm{type} = \mathrm{ERROR} \vee curr = n\}).\mathbf{End}
\end{aligned}
$$

F* Specification

compile

Compile time

Runtime



Idaho National Laboratory

DATUM: Proof of Concept, Two Blockers to Deployment

Global Refined Multiparty Session Types (GRMPST)

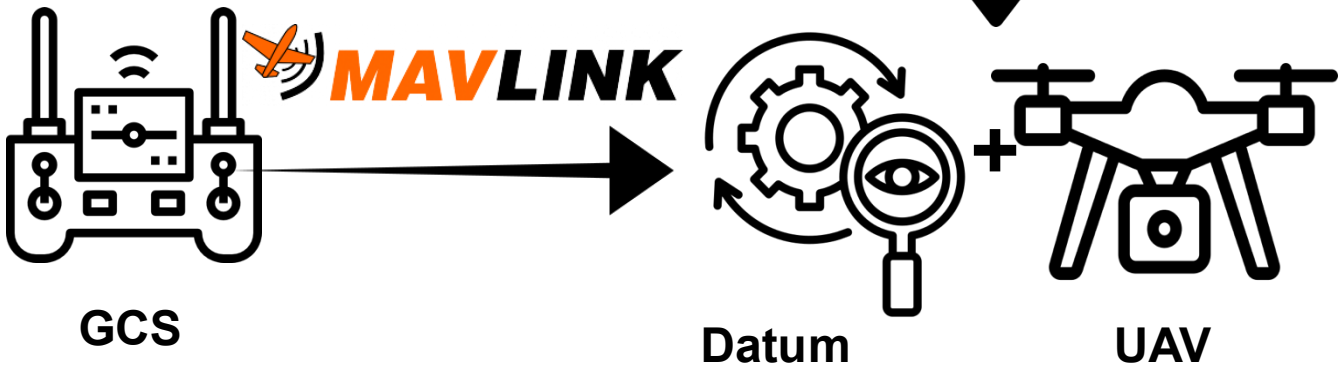
$$
\begin{aligned}
&\mathbf{GCS} \rightarrow \mathbf{UAV}\colon \mathtt{MISSION\ COUNT}(n\{1 \leq n < \mathrm{LIMIT}\}) \\
&\mu.T(curr\{0 \leq curr < n\})(curr = 0) \\
&\quad \mathbf{UAV} \rightarrow \mathbf{GCS}\colon \\
&\qquad \mathtt{MISSION\ REQUEST\ INT}(req\{req = curr < n\}) \\
&\qquad \mathtt{MISSION\ ITEM\ INT}(item\{item = curr < n\}) \\
&\qquad T(curr + 1) \\
&\quad \oplus\, \mathtt{MISSION\ ACK}(ack\{\mathrm{type} = \mathrm{ERROR} \vee curr = n\}).\mathbf{End}
\end{aligned}
$$

F* Specification

compile

Compile time

Runtime



Limitation #1: Specification Burden

Reason: Extrinsic Verification

```
(#nparticipants : nat) ->
(#rec_contexts : list (session_type u))
(from : nat) -> (to : nat) ->
(l : (list (label * (dtuple4 ...
```

Extra proof arguments

DATUM: Proof of Concept, Two Blockers to Deployment

Global Refined Multiparty Session Types (GRMPST)

$$
\begin{aligned}
&\mathbf{GCS} \rightarrow \mathbf{UAV}: \texttt{MISSION\_COUNT}(n\{1 \leq n < \text{LIMIT}\}) \\
&\mu.T(curr\{0 \leq curr < n\})(curr = 0) \\
&\quad \mathbf{UAV} \rightarrow \mathbf{GCS}: \\
&\qquad \texttt{MISSION\_REQUEST\_INT}(req\{req = curr < n\}) \\
&\qquad \texttt{MISSION\_ITEM\_INT}(item\{item = curr < n\}) \\
&\qquad T(curr + 1) \\
&\quad \oplus \texttt{MISSION\_ACK}(ack\{\text{type} = \text{ERROR} \vee curr = n\}).\mathbf{End}
\end{aligned}
$$

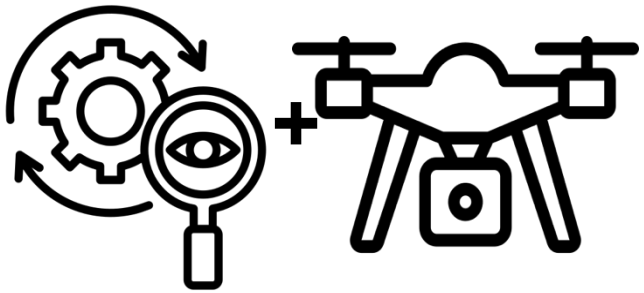
F* Specification

compile

Compile time

Runtime

GCS



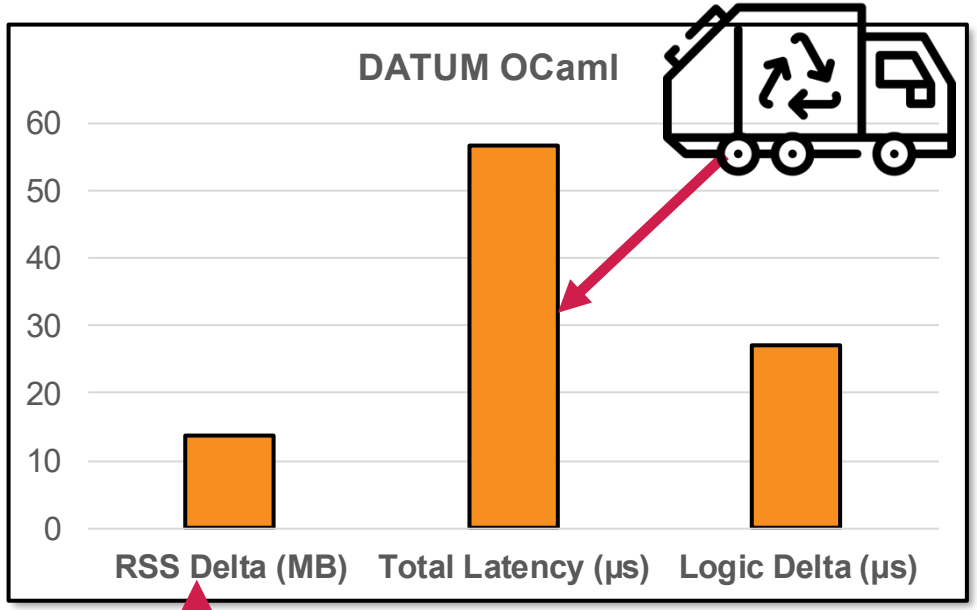
Datum

UAV

Limitation #2 : The Deployment Gap

Reason: OCaml runtime

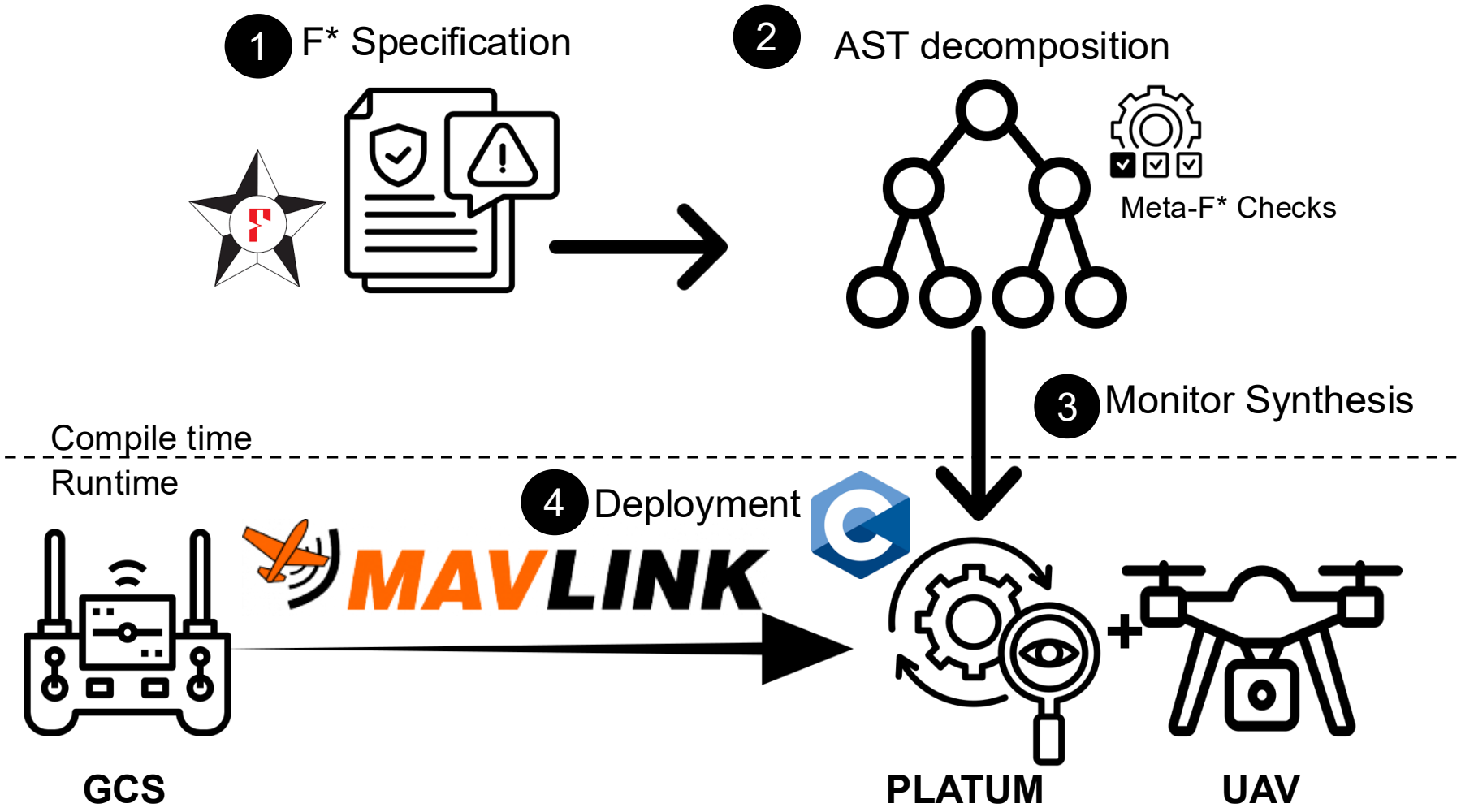
Garbage Collector



OCaml Compiler

Idaho National Laboratory

Platum: *Five Inputs In. Certified C Monitor Out*



Contributions

1- A minimal DSL: session inputs, no proof scaffolding, and automated checks.

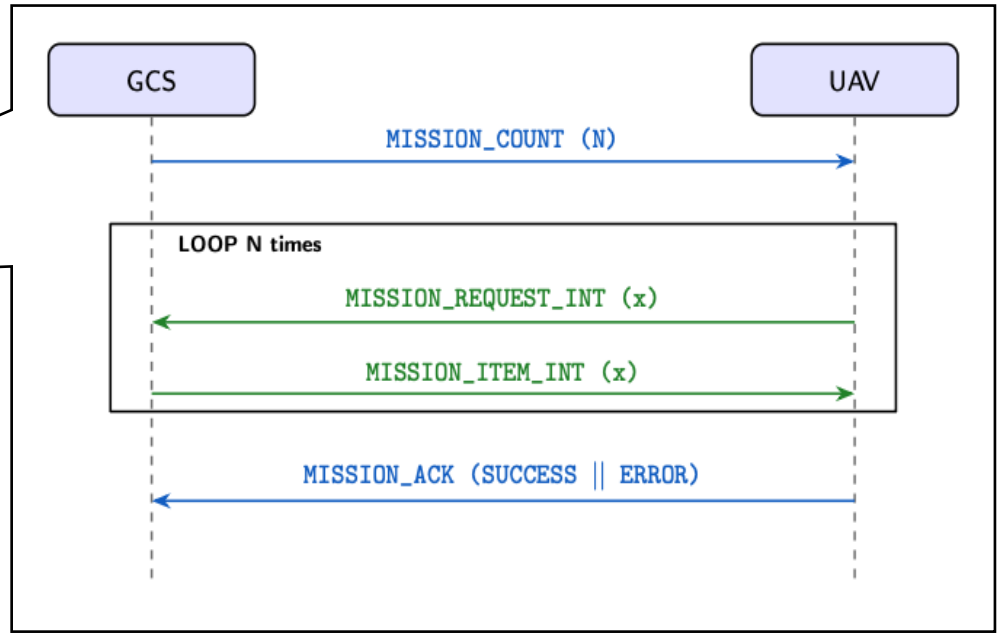
2- Synthesis of performant monitors from wellformed session types.

3- 4x latency and 40% memory reduction vs. DATUM under ArduPilot SITL.

GRMPSTs Enforce Ordering, Preconditions and Value Constraints at the Network Boundary

Global Refined Multiparty Session Types (GRMPST)

$$
\begin{aligned}
&\mathbf{GCS} \to \mathbf{UAV}: \texttt{MISSION\_COUNT}(n\{1 \leq n < \text{LIMIT}\}) \\
&\mu.T(curr\{0 \leq curr < n\})(curr = 0) \\
&\quad \mathbf{UAV} \to \mathbf{GCS}: \\
&\qquad \texttt{MISSION\_REQUEST\_INT}(req\{req = curr < n\}) \\
&\qquad \texttt{MISSION\_ITEM\_INT}(item\{item = curr < n\}) \\
&\qquad T(curr + 1) \\
&\quad \oplus \texttt{MISSION\_ACK}(ack\{\text{type} = \text{ERROR} \vee curr = n\}).\mathbf{End}
\end{aligned}
$$



GRMPSTs Enforce Ordering, Preconditions and Value Constraints at the Network Boundary

Global Refined Multiparty Session Types (GRMPST)

$$\begin{aligned}
&\mathbf{GCS} \rightarrow \mathbf{UAV}: \texttt{MISSION COUNT}(n\{1 \leq n < \text{LIMIT}\}) \\
&\mu.T(curr\{0 \leq curr < n\})(curr = 0) \\
&\quad \mathbf{UAV} \rightarrow \mathbf{GCS}: \\
&\qquad \texttt{MISSION REQUEST INT}(req\{req = curr < n\}) \\
&\qquad \texttt{MISSION ITEM INT}(item\{item = curr < n\}) \\
&\qquad T(curr + 1) \\
&\quad \oplus\, \texttt{MISSION ACK}(ack\{\text{type} = \text{ERROR} \vee curr = n\}).\mathbf{End}
\end{aligned}$$

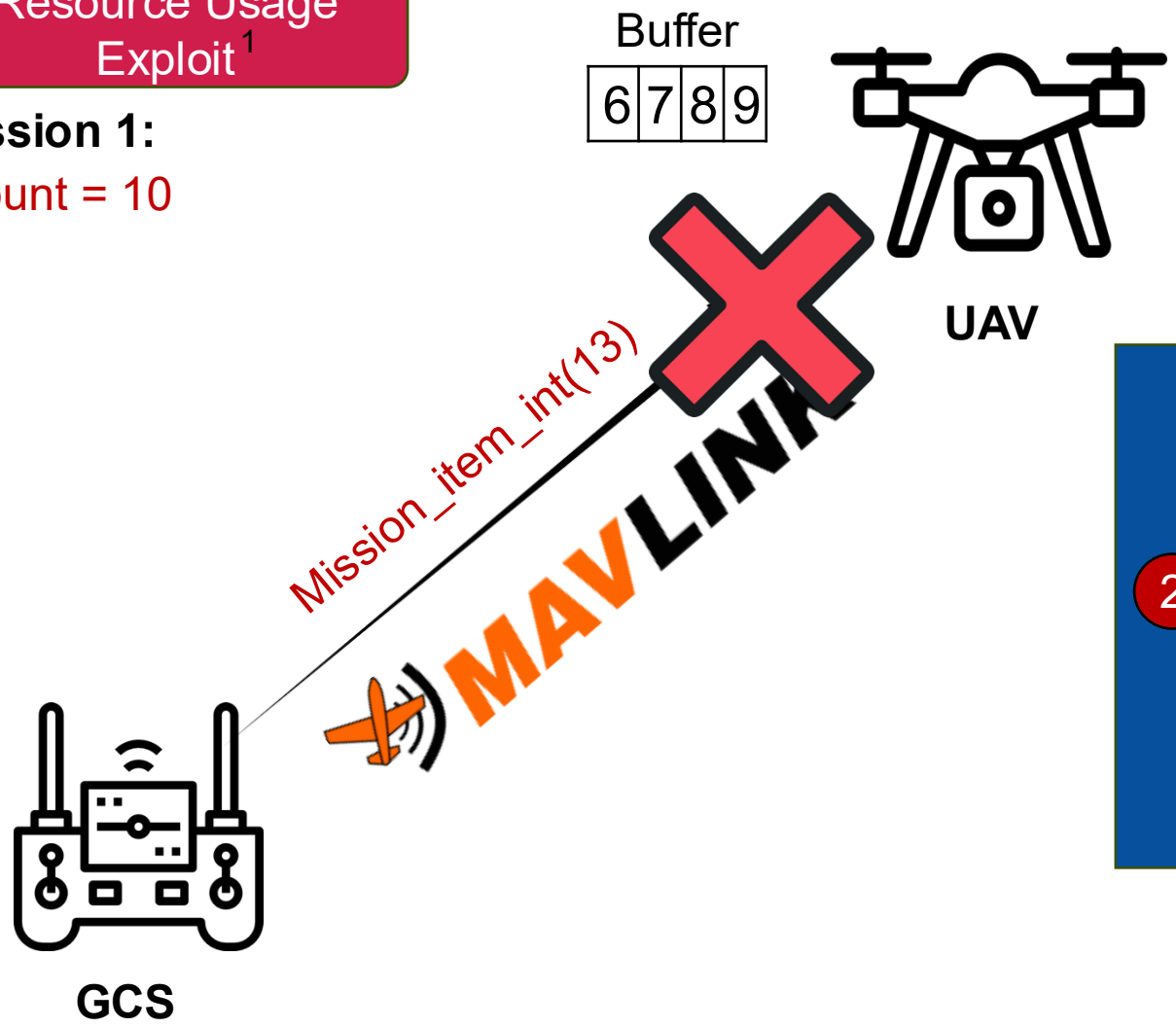
1. Create a recursive variable *curr* that must be less than or equal to n
2. Ensure messages are uploaded in the correct order and not skipped
3. The protocol must only complete with *curr* = *n* or with an error message.

GRMPSTs Enforce Ordering, Preconditions and Value Constraints at the Network Boundary

Resource Usage Exploit[1]

Mission 1:

Count = 10



2 Ensure messages are uploaded in the correct order and not skipped

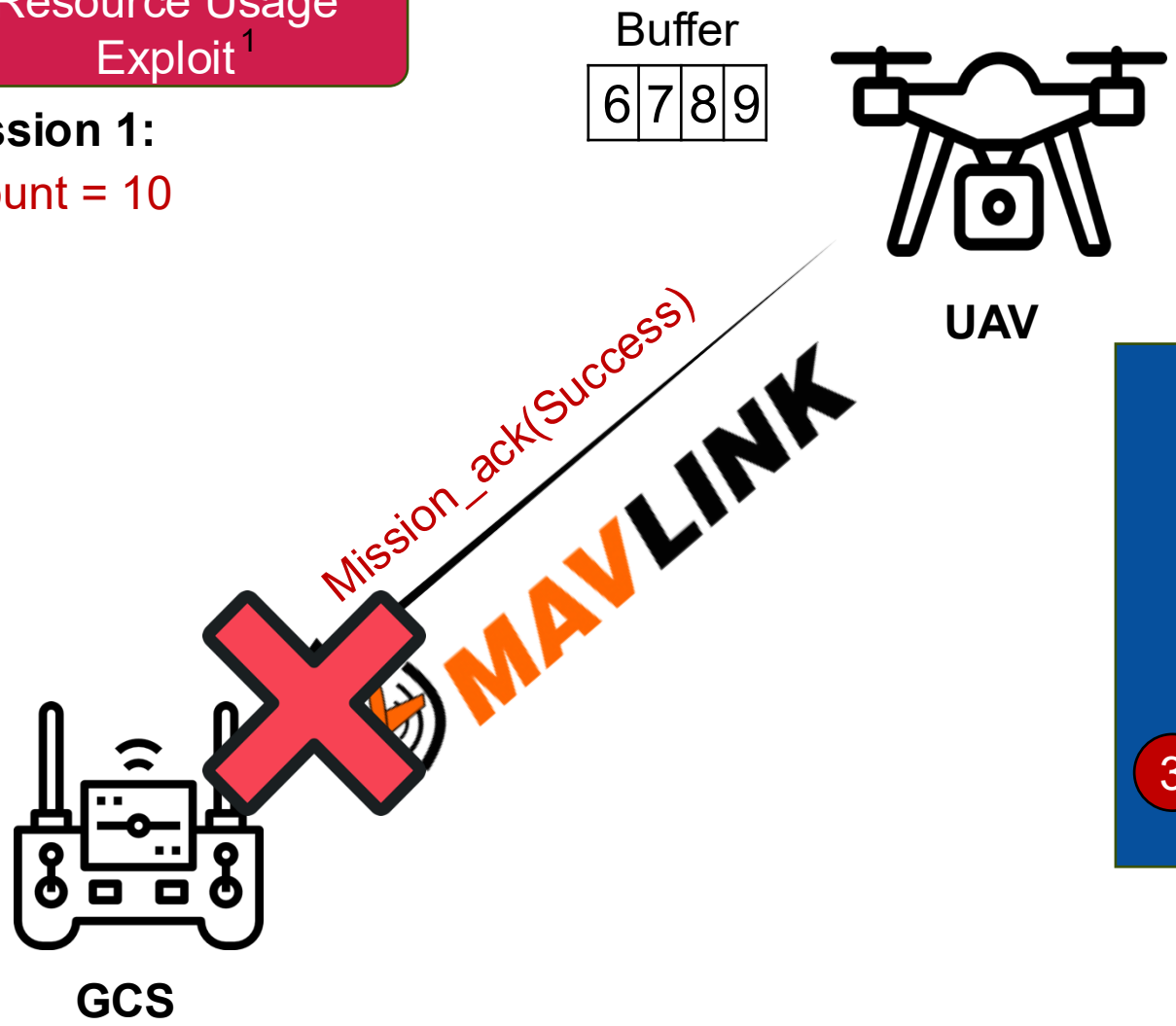
Idaho National Laboratory

GRMPSTs Enforce Ordering, Preconditions and Value Constraints at the Network Boundary

Resource Usage Exploit [1]

Mission 1:

Count = 10



3 The protocol must only complete with *curr* = *n* or with an error message.

Idaho National Laboratory

Extrinsic vs. Intrinsic — The DSL Design Choice

1 F* Specification

5 key arguments for a GRMPST

$$GCS \rightarrow UAV: MISSION\ COUNT(n: mission\ count\ \{1 \leq count(n) < LIMIT\})$$

1-sender

2- receiver

3- label

4- variable

5-refinement

Extrinsic vs. Intrinsic — The DSL Design Choice

1 F* Specification

Must check these properties

1- Label Uniqueness
2- Guarded Recursion
3- Session Fidelity
4- Global Progress

5 key arguments for a GRMPST

2- receiver

4- variable

$$GCS \rightarrow UAV: MISSION\ COUNT(n: mission\ count\ \{1 \leq count(n) < LIMIT\})$$

1-sender

3- label

5-refinement

Extrinsic vs. Intrinsic — The DSL Design Choice

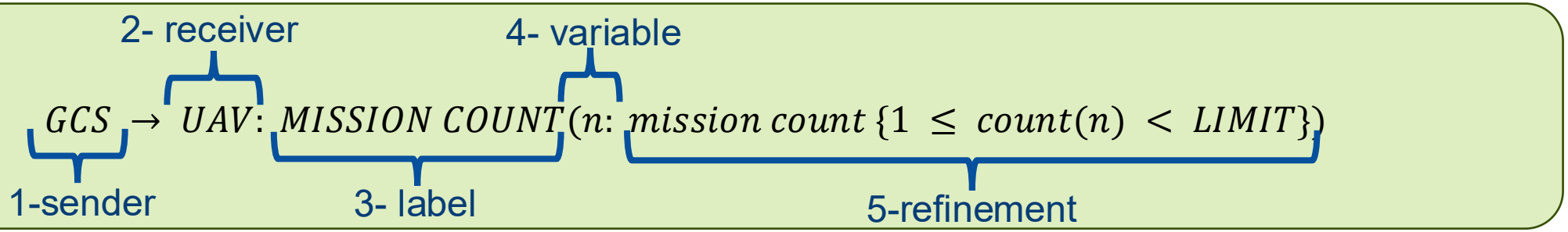
1 F* Specification

Intrinsic Verification

Type checks = is well-formed

Pros: Cannot constructs/compile a malformed term
Cons: Must provide the type checker in enough information to check well-formedness (extra inputs)

5 key arguments for a GRMPST



Extrinsic Verification

Type checks ≠ is well-formed (checked separately)

Pros: Minimal syntactic surface (only required inputs)
Cons: Must ensure checks happen at some point.

Extrinsic vs. Intrinsic — The DSL Design Choice

OLD - Intrinsic verification (Datum)

```
(#nparticipants : nat) ->
(#rec_contexts : list (session_type u)) ->
(from : nat) -> (to : nat) ->
(l : (list (label * (dtuple4 ...
```

5 key arguments for a GRMPST

Extrinsic Verification

Type checks ≠ is well-formed (checked separately)

Pros: Minimal syntactic surface (only required inputs)
Cons: Must ensure checks happen at some point.

Extrinsic vs. Intrinsic — The DSL Design Choice

1 F* Specification

OLD - Intrinsic verification (Datum)

```
(#nparticipants : nat) ->
(#rec_contexts : list (session_type u)) ->
(from : nat) -> (to : nat) ->
(l : (list (label * (dtuple4 ...
```

Extra proof inputs

5 key arguments for a GRMPST

Extrinsic Verification

Type checks ≠ is well-formed (checked separately)

Pros: Minimal syntactic surface (only required inputs)
Cons: Must ensure checks happen at some point.

Extrinsic vs. Intrinsic — The DSL Design Choice

1 F* Specification

OLD - Intrinsic verification (Datum)

```
(#nparticipants : nat) ->
(#rec_contexts : list (session_type u)) ->
(from : nat) -> (to : nat) ->
(l : (list (label * (dtuple4 ...
```

Extra proof inputs

5 key arguments for a GRMPST

2- receiver

4- variable

$$GCS \rightarrow UAV: MISSION\ COUNT(n: mission\ count\ \{1 \leq count(n) < LIMIT\})$$

1-sender

3- label

5-refinement

NEW - Intrinsic Verification (Platum)

2- receiver

4- variable

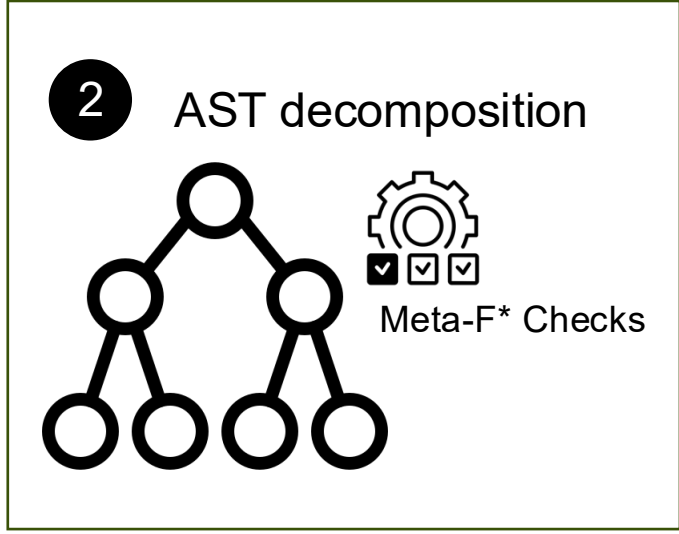
$$(0 \text{ --> } 1)\ [Option\ \text{"MISSION\_COUNT"}\ (\lambda\ (cnt : mission\_count\ \{\ cnt\ >=\ 1\ \&\&\ cnt\ <\ mission\_limit\})$$

1-sender

3- label

5-refinement

Meta-F* Decomposes the AST Into an Inspectable Protocol Graph



Must check these properties

1- Label Uniqueness
2- Guarded Recursion
3- Session Fidelity
4- Global Progress

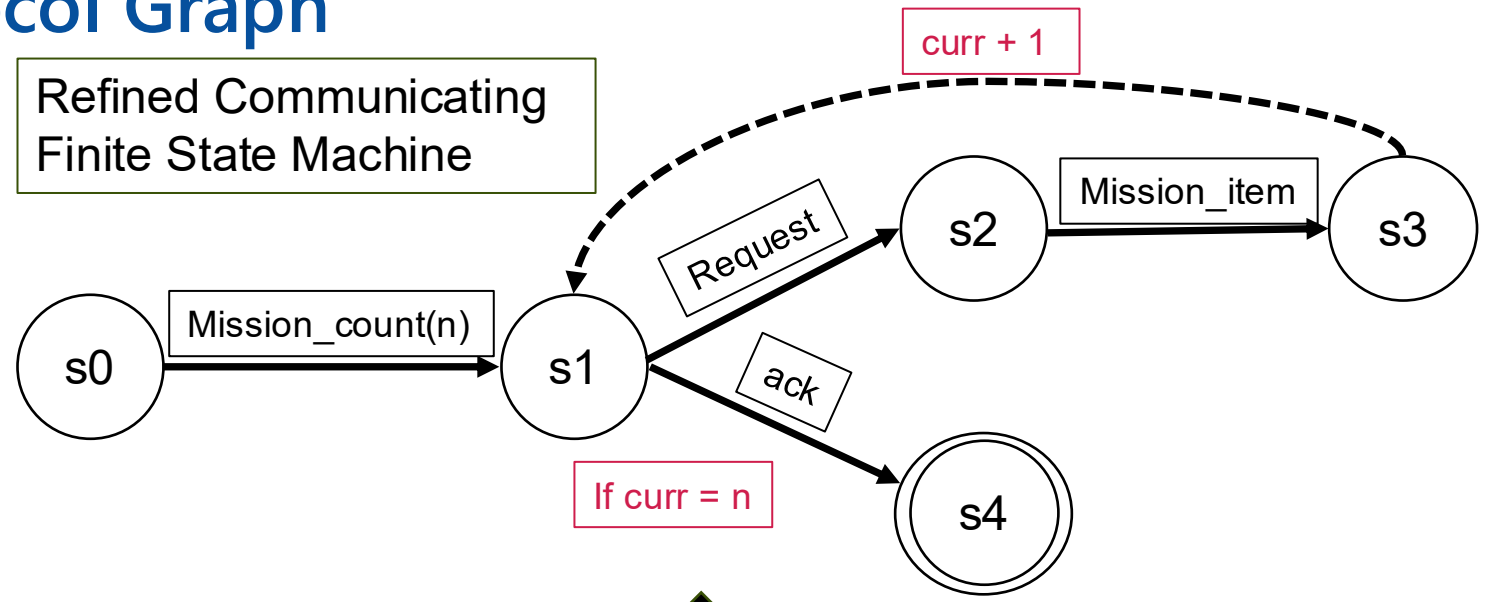
NEW - Intrinsic Verification (Platum)

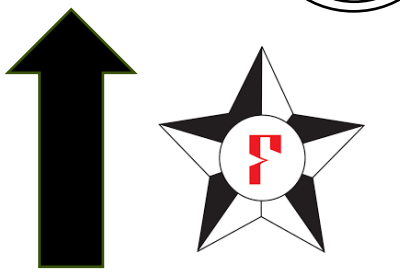
$$(0 \texttt{-->} 1)\ [Option\ \text{"MISSION\_COUNT"}\ (\lambda\ (cnt : mission\_count\ \{\ cnt\ >=\ 1\ \&\&\ cnt\ <\ mission\_limit\})$$

1-sender
2- receiver
3- label
4- variable
5-refinement

Meta-F* Decomposes the AST Into an Inspectable Protocol Graph

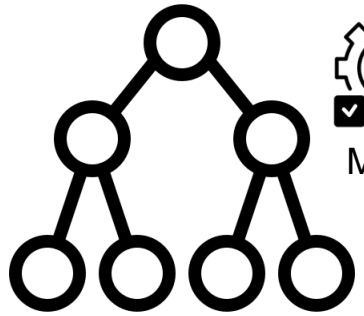
Refined Communicating Finite State Machine





Meta-F*

2 AST decomposition



Meta-F* Checks

Must check these properties

1- Label Uniqueness
2- Guarded Recursion
3- Session Fidelity
4- Global Progress

NEW - Intrinsic Verification (Platum)

2- receiver

4- variable

$$(0 --> 1)\ [Option\ "\text{MISSION\_COUNT}"\ (\lambda\ (cnt : mission\_count\ \{\ cnt\ >=\ 1\ \&\&\ cnt\ <\ mission\_limit\})$$

1-sender

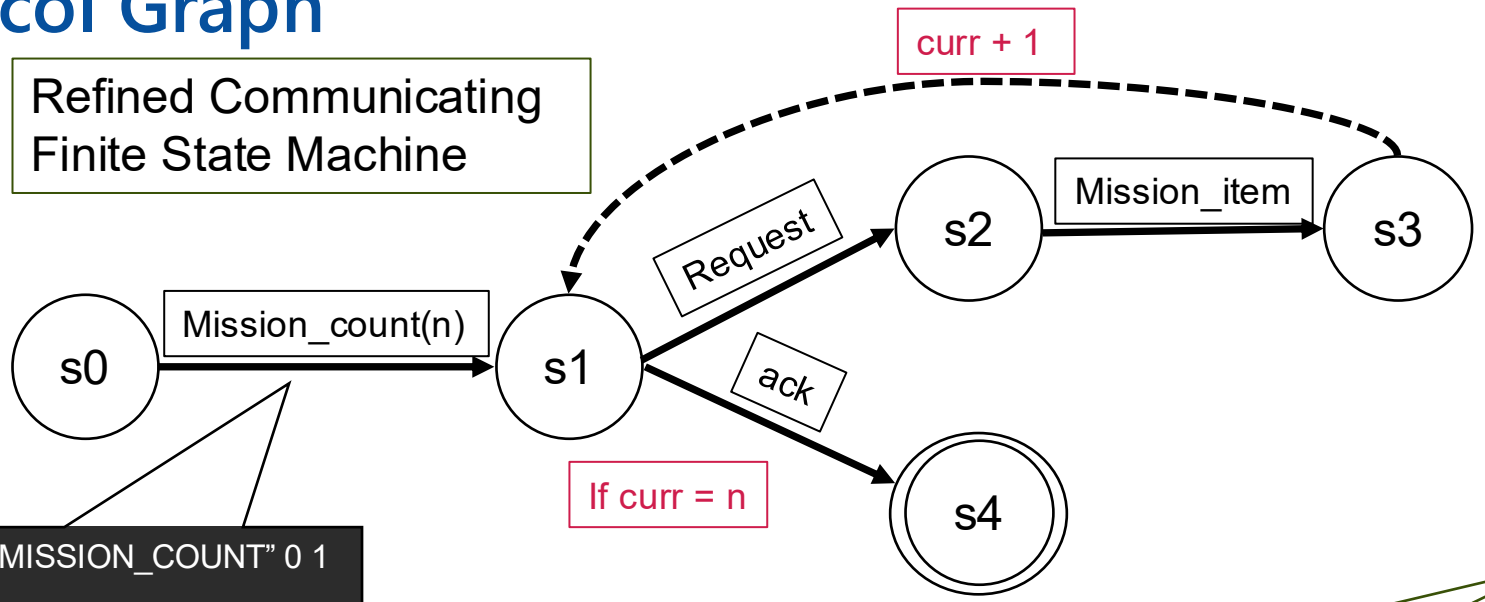
3- label

5-refinement

Idaho National Laboratory

Meta-F* Decomposes the AST Into an Inspectable Protocol Graph

Refined Communicating Finite State Machine

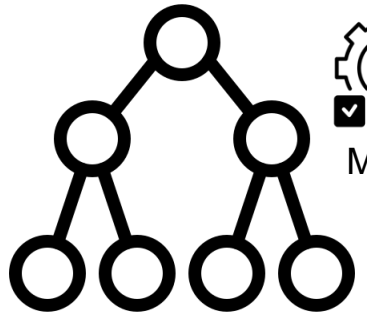



```
Mkprotocol_transition "MISSION_COUNT" 0 1
[
        Mkvar "mc_target_system" KindInt;
        Mkvar "mc_target_component" KindInt;
        Mkvar "mc_count" KindInt;
        Mkvar "mc_mission_type" KindInt;
        Mkvar "mc_opaque_id" KindInt
      ]
      (GAnd (GGe (TVar "mc_count") (TIntLit
1)) (GLt (TVar "mc_count") (TIntLit 65535)))
```

```
let rec has_duplicates (l: list label) (seen: list label)
    Tot bool =
  match l with
  | [] → false
  | hd :: tl →
      if L.mem hd seen && hd <> "RECUR" then true
      else has_duplicates tl (hd :: seen)

let check_unique_labels (labels: list label) : bool =
  not (has_duplicates labels [])
```

2 AST decomposition

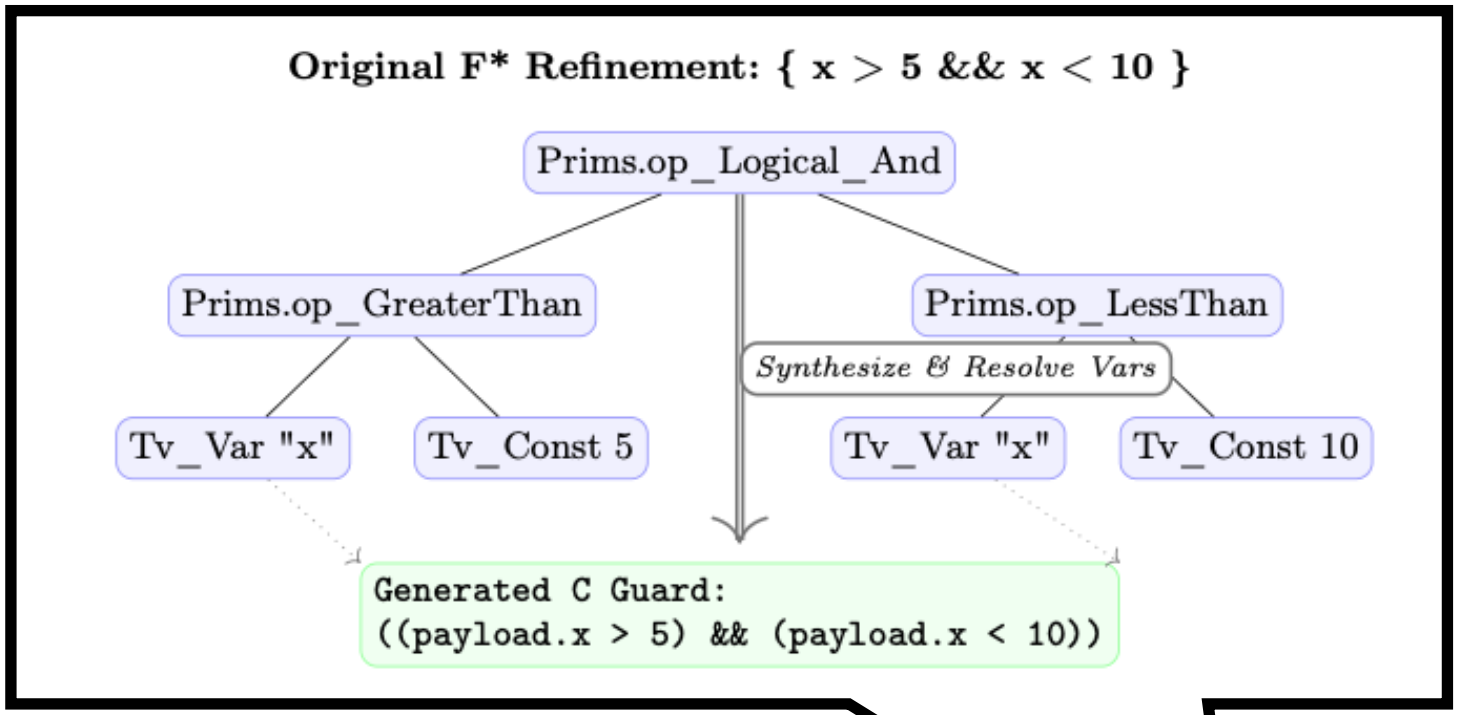


Meta-F* Checks

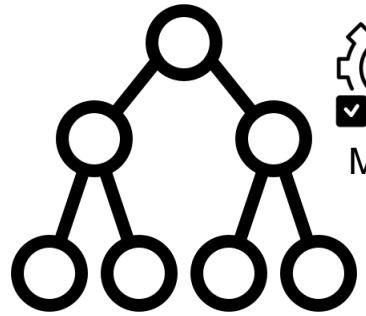
Must check these properties

1- Label Uniqueness
2- Guarded Recursion
3- Session Fidelity
4- Global Progress

Type Erasure Without Losing the Constraints



2 AST decomposition



Meta-F* Checks

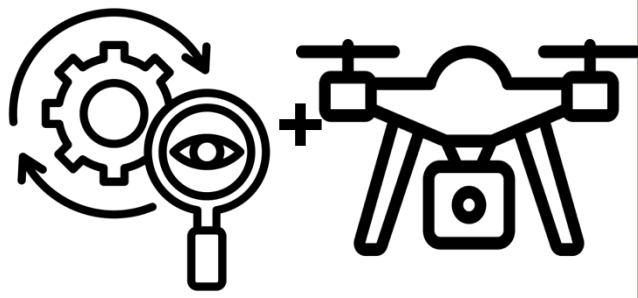
NEW - Intrinsic Verification (Platum)

$(0 \text{ --> } 1)$ [$Option$ "MISSION_COUNT" (λ (cnt : $mission\_count$ { cnt >= 1 && cnt < $mission\_limit$})

5-refinement

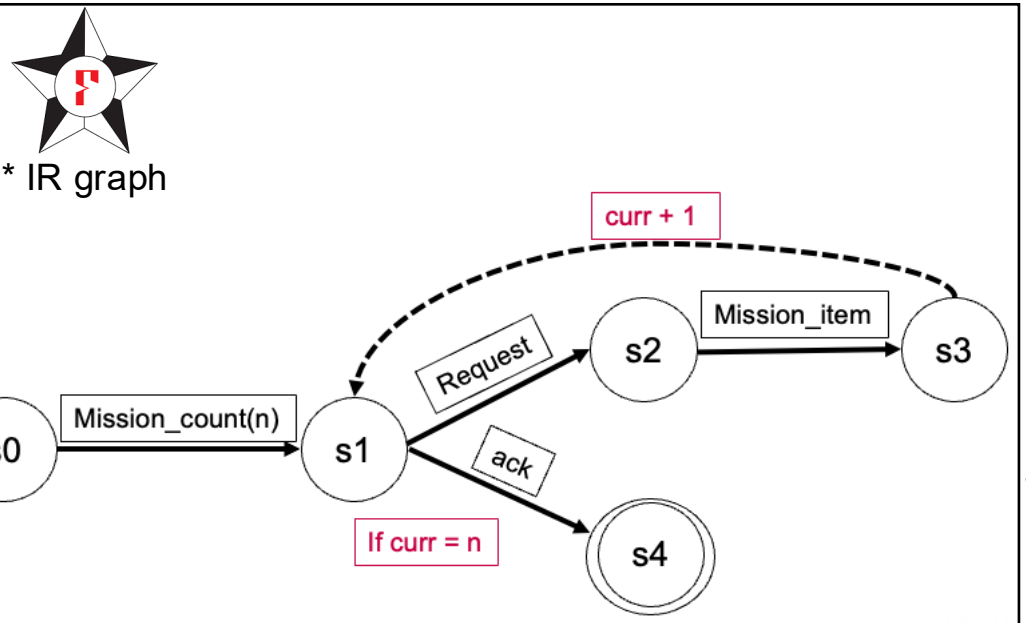
Type Erasure Without Losing the Constraints

3 Monitor Synthesis



PLATUM UAV

Advantages:
1- Allocation-free
2- O(1) state lookup (jump table)
3- Bounded WCET (no GC pauses)



Synthesis

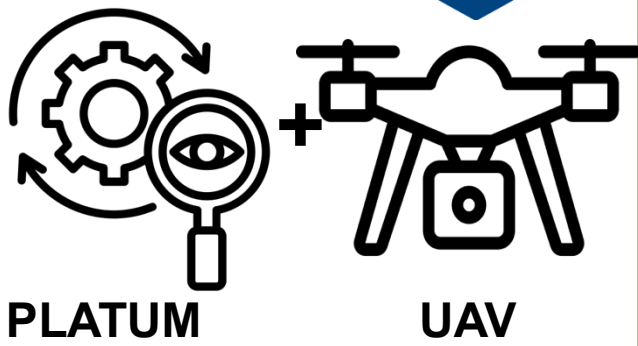
C FSM

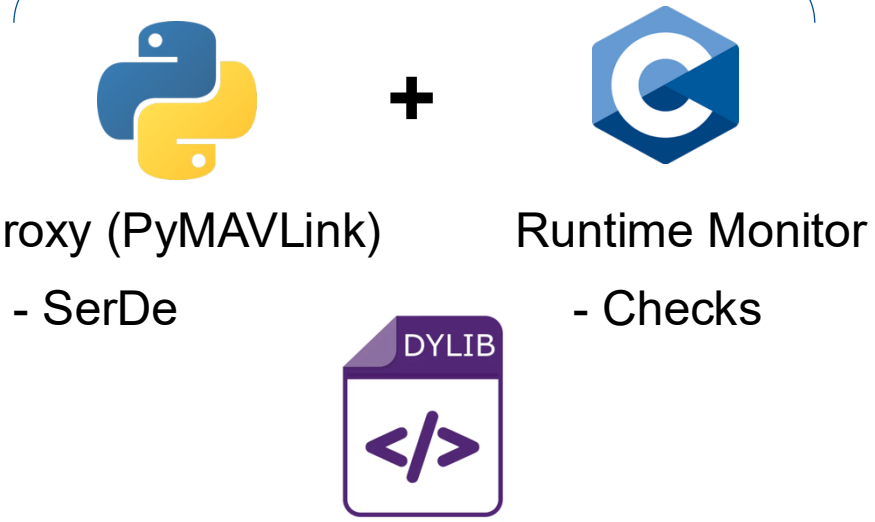
```
bool monitor_step(monitor_t* mon, const mavlink_message_t*
    msg) {
  switch (mon->state) {
// State 0: Expecting MISSION_COUNT
// GCS -> UAV: MISSION_COUNT(N: Z {N >= 1 /\ N < LIMIT})
    case STATE_IDLE_0: {
      if (msg->msgid == MAVLINK_MSG_ID_MISSION_COUNT) {
        mavlink_mission_count_t payload;
        mavlink_msg_mission_count_decode(msg, &payload);
// Check Refinement: {cnt >= 1 && cnt < mission_item_limit}
        if(payload.count >= 1 && payload.count < 65535) {
          // Update Context (F* bound variables)
          mon->ctx.cnt = payload.count;
          mon->ctx.curr = 0; // Initialize Mu variable
// Transition
          mon->state = STATE_LOOP_REQ_1;
          return true;
        }
```

Must pass all 4 wellformedness checks

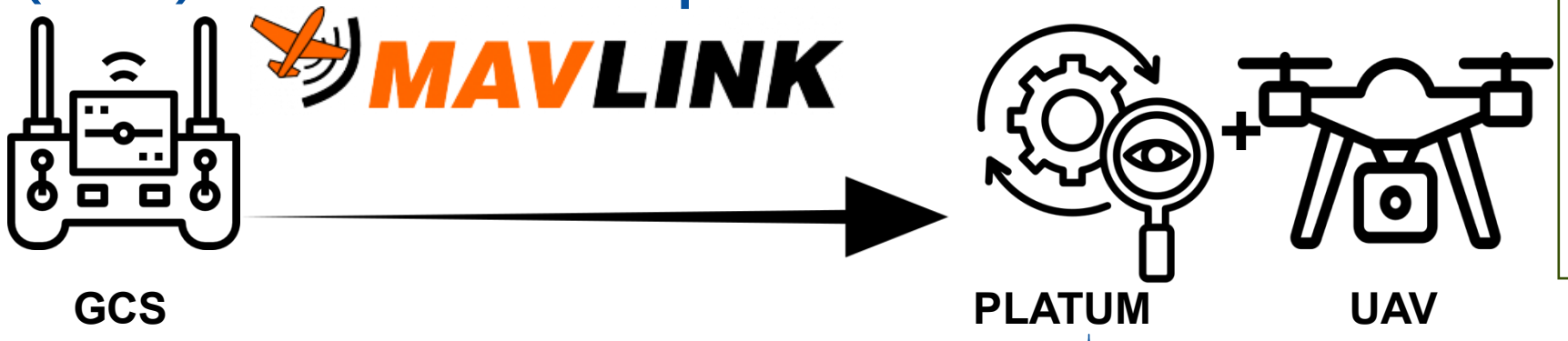
Platum decreases Latency (4x) and Memory (40%) Overhead Compared to Datum

4 Deployment

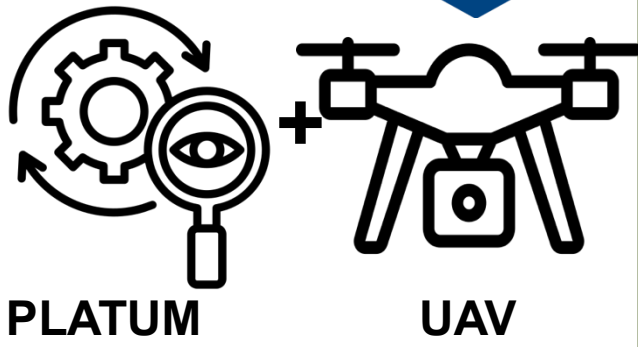


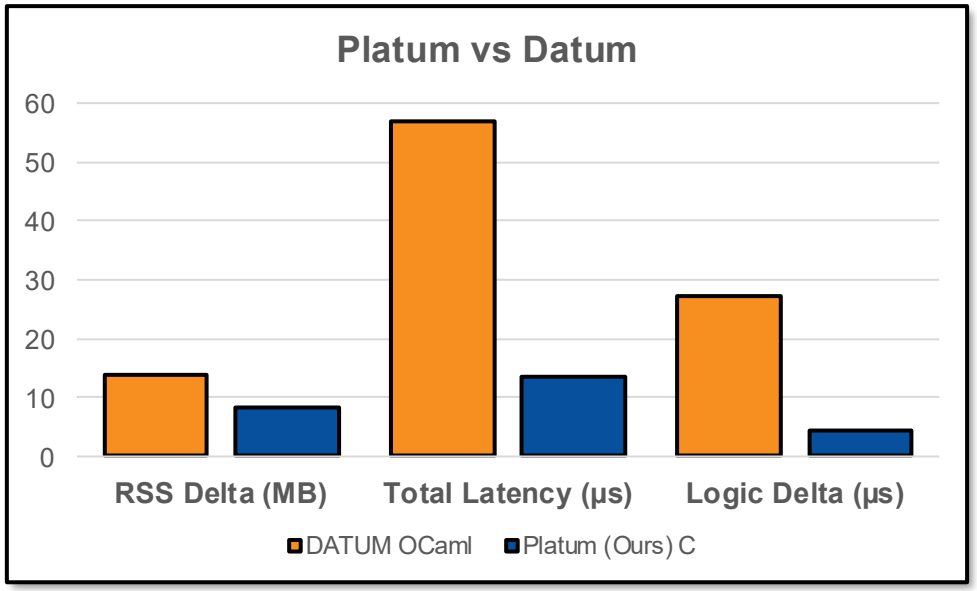


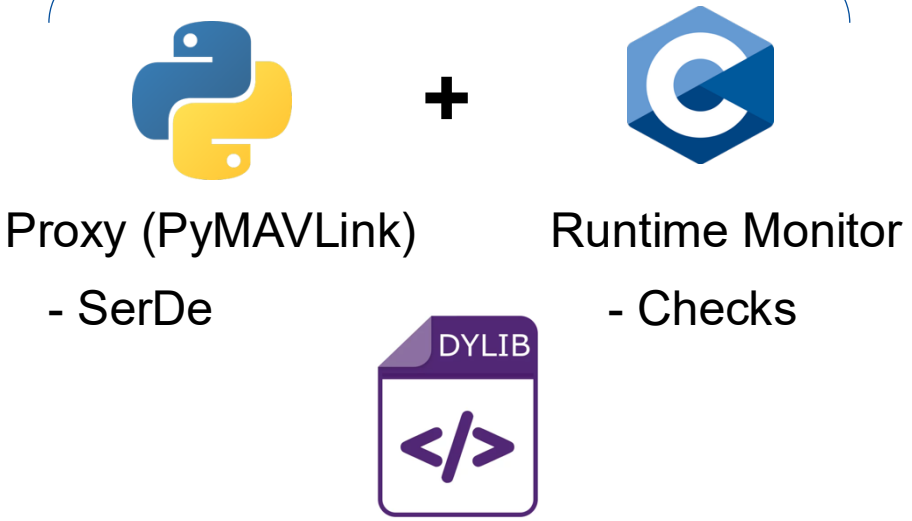
Platum decreases Latency (4x) and Memory (40%) Overhead Compared to Datum



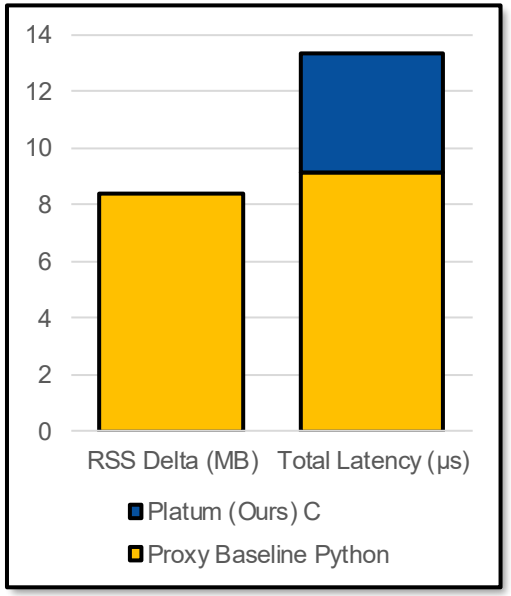
4 Deployment







Shared Overhead

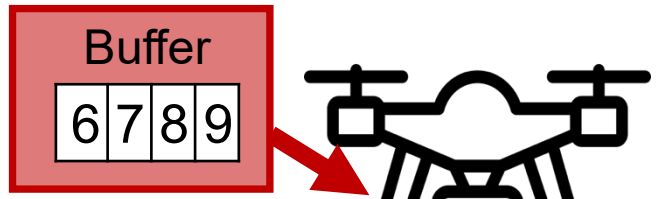


Idaho National Laboratory

Platum: *Five Inputs In. Certified C Monitor Out*

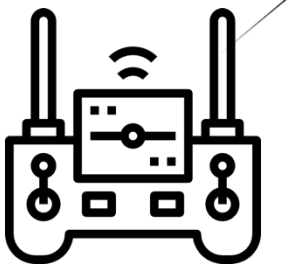
Resource Usage Exploit[1]

Mission 2:
Count = 15



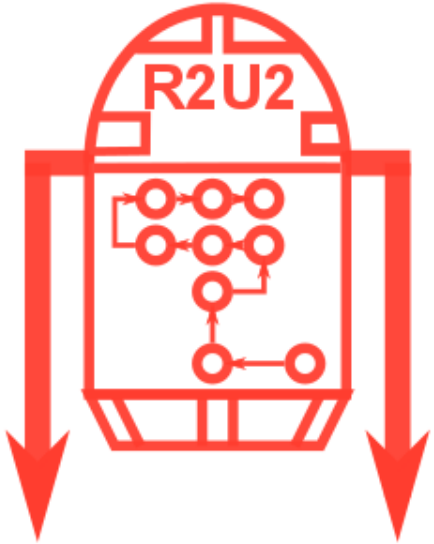
UAV

UAV starts behaving as mission 1 in the middle of mission 2



GCS

Problem: A stealthy attack uses the protocol exactly as specified to induce undesired behavior, which makes it invisible to the defenses we usually rely on



Intrusion detection systems: No anomalous behavior or virus signature

Physics based anomaly detection: no anomalous physical deviation.

Platum: *Five Inputs In. Certified C Monitor Out*

1 F* Specification

2- receiver 4- variable

$$GCS \rightarrow UAV: MISSION\ COUNT(n: mission\ count\ \{1 \leq count(n) < LIMIT\})$$

1-sender 3- label 5-refinement

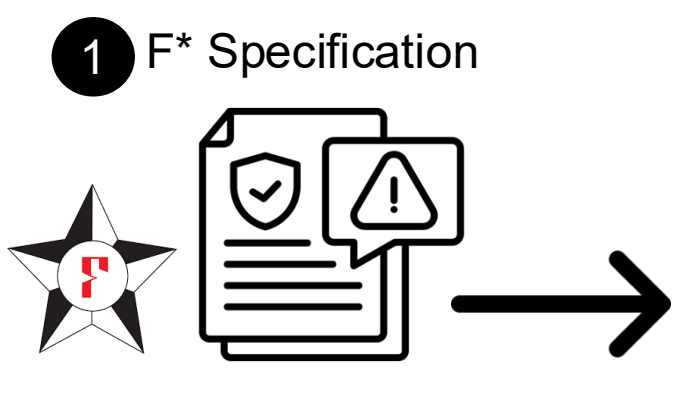
NEW - Intrinsic Verification (Platum)

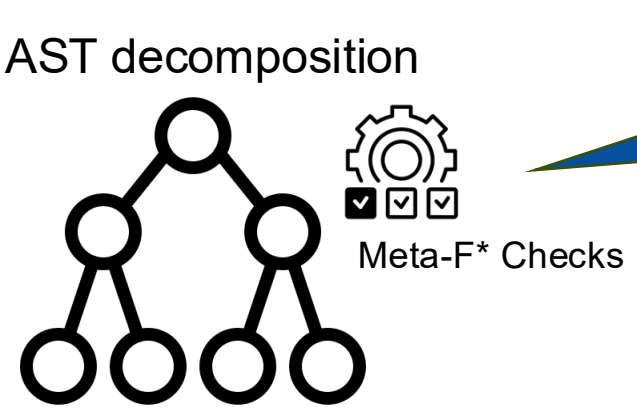
2- receiver 4- variable

$$(0 --> 1)\ [Option\ "\text{MISSION\_COUNT}"\ (\lambda\ (cnt : mission\_count\ \{\ cnt\ >=\ 1\ \&\&\ cnt\ <\ mission\_limit\})$$

1-sender 3- label

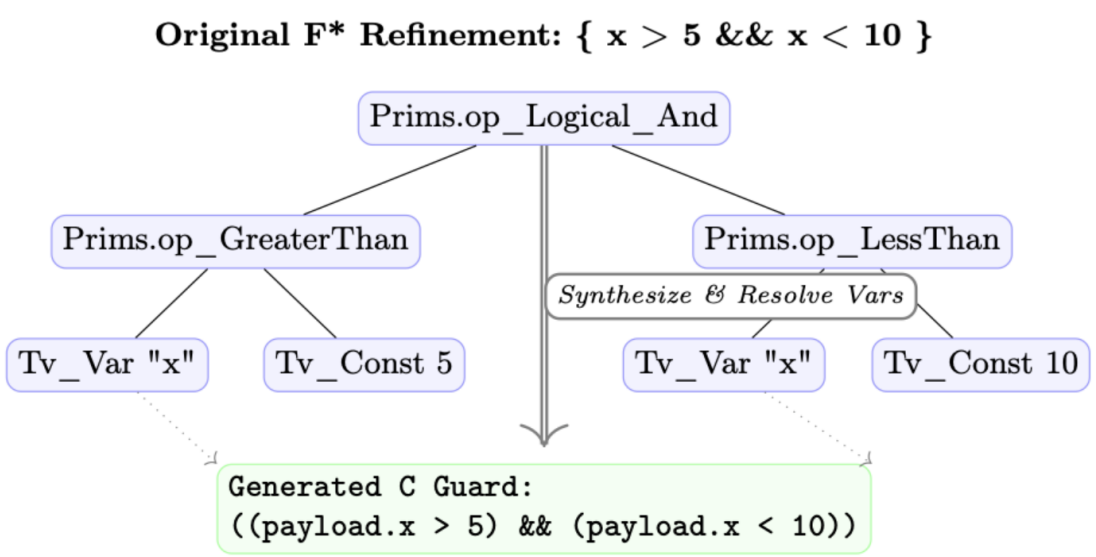
Platum: *Five Inputs In. Certified C Monitor Out*



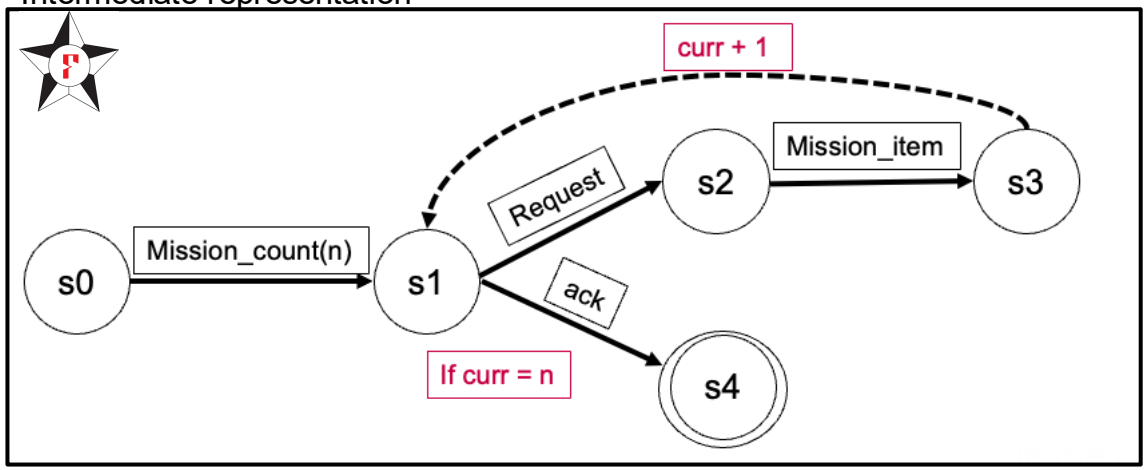


Wellformedness Checks

1- Label Uniqueness
2- Guarded Recursion
3- Session Fidelity
4- Global Progress

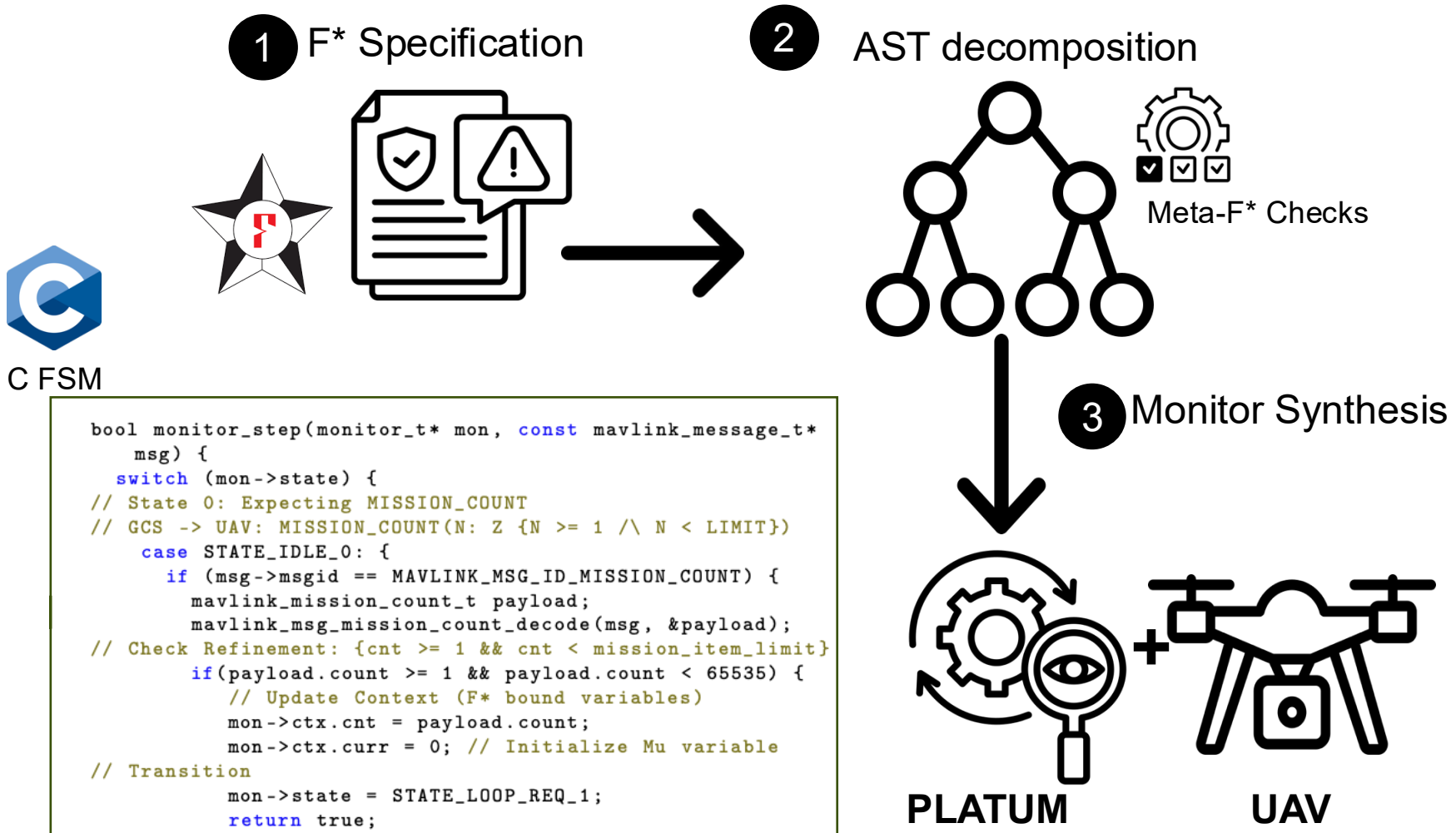


Intermediate representation



Idaho National Laboratory

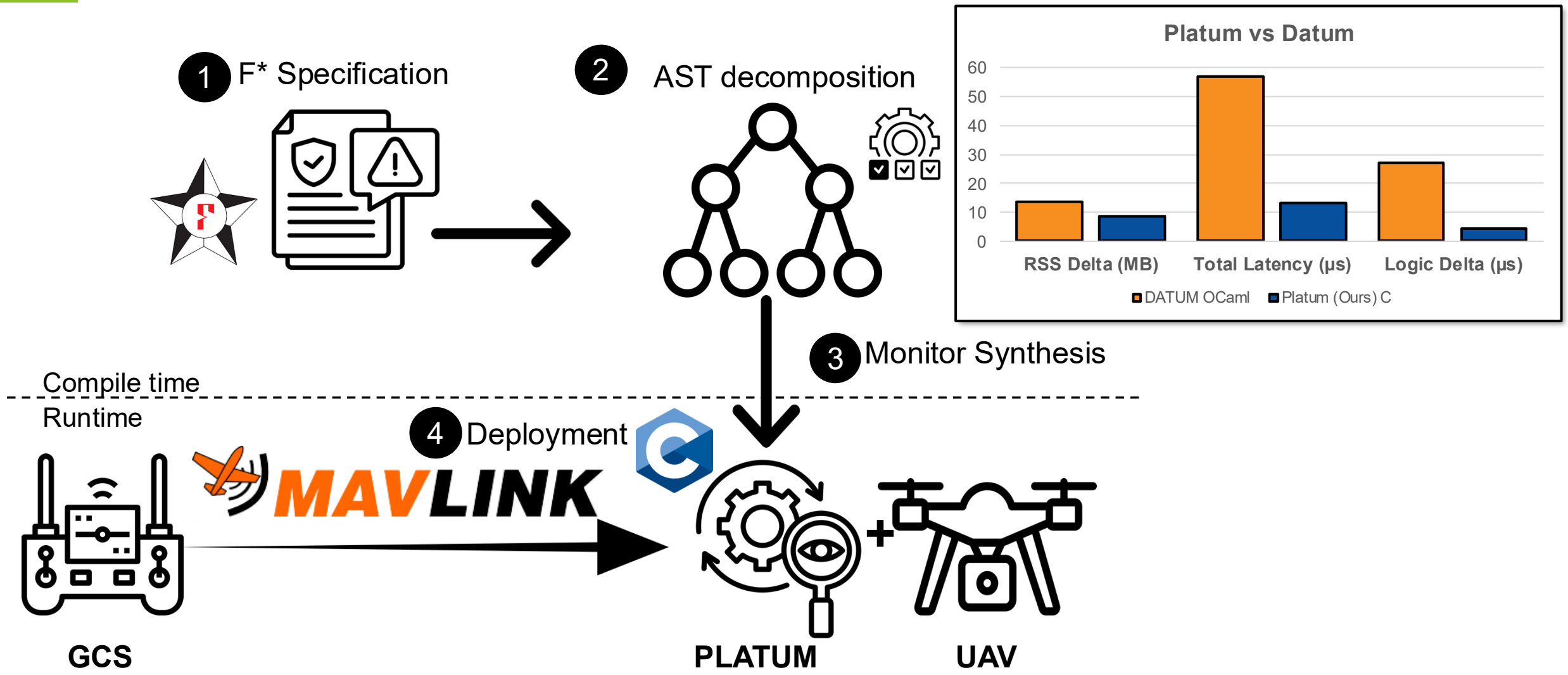
Platum: *Five Inputs In. Certified C Monitor Out*




```
bool monitor_step(monitor_t* mon, const mavlink_message_t*
    msg) {
  switch (mon->state) {
// State 0: Expecting MISSION_COUNT
// GCS -> UAV: MISSION_COUNT(N: Z {N >= 1 /\ N < LIMIT})
    case STATE_IDLE_0: {
      if (msg->msgid == MAVLINK_MSG_ID_MISSION_COUNT) {
        mavlink_mission_count_t payload;
        mavlink_msg_mission_count_decode(msg, &payload);
// Check Refinement: {cnt >= 1 && cnt < mission_item_limit}
        if(payload.count >= 1 && payload.count < 65535) {
           // Update Context (F* bound variables)
           mon->ctx.cnt = payload.count;
           mon->ctx.curr = 0; // Initialize Mu variable
// Transition
           mon->state = STATE_LOOP_REQ_1;
           return true;
        }
```

Advantages:
1- Allocation-free
2- O(1) state lookup (jump table)
3- Bounded WCET (no GC pauses)

Platum: *Five Inputs In. Certified C Monitor Out*



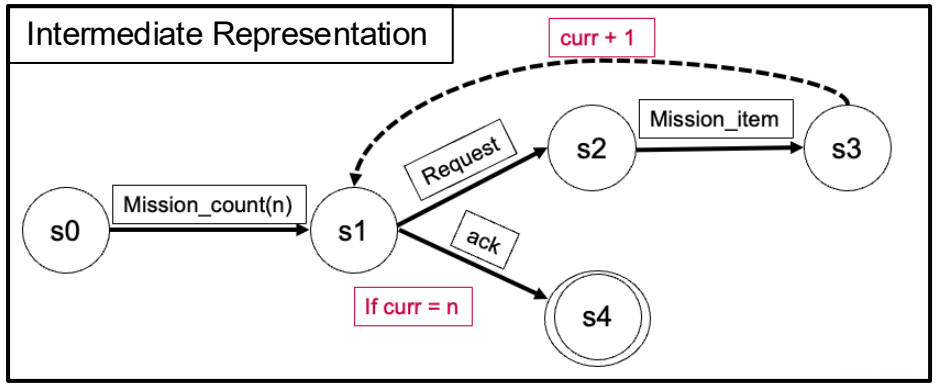
Idaho National Laboratory

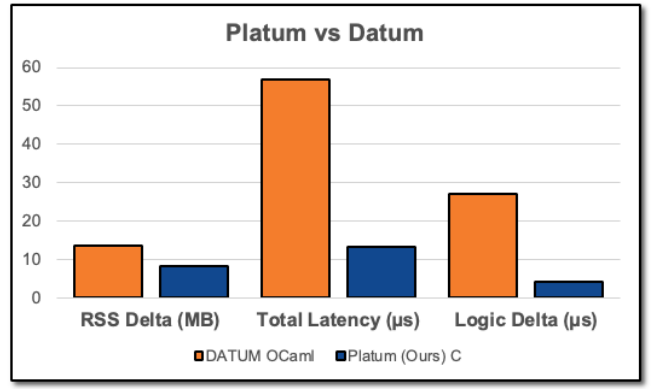
Five inputs in, verified C monitor out — 4x faster and 40% leaner than the state of the art.

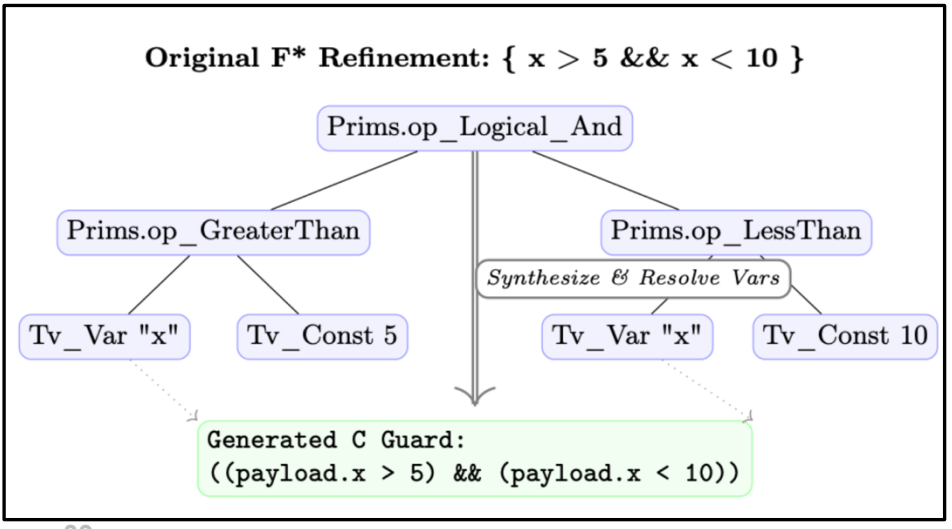
OPEN TO WORK

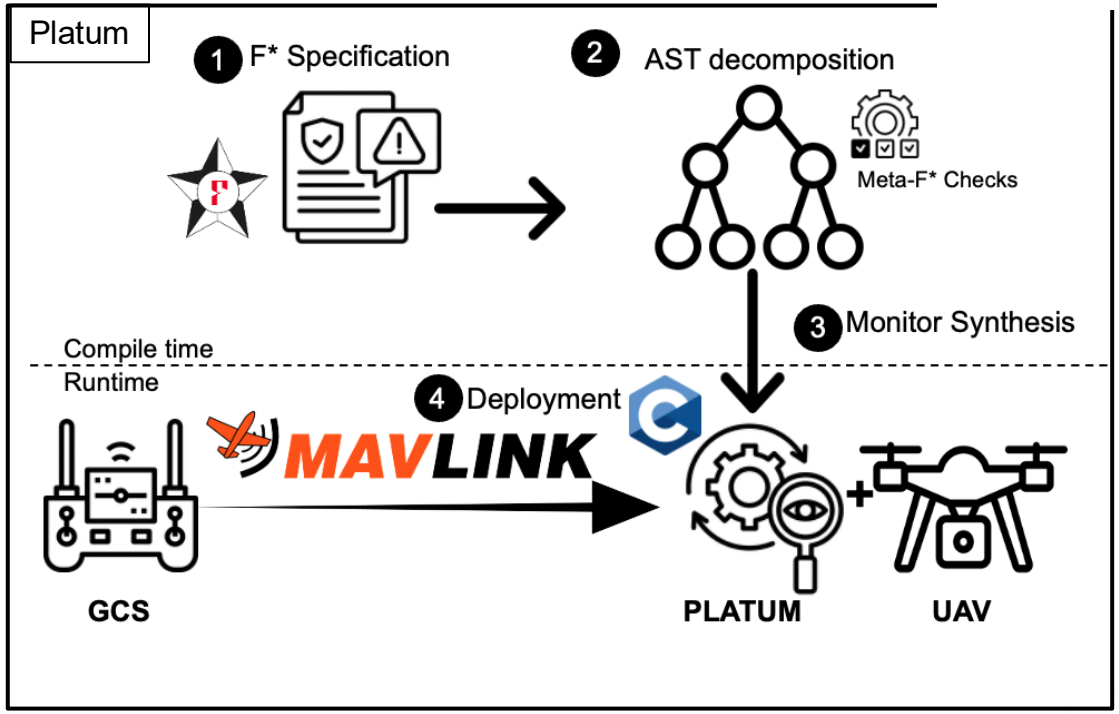
Arthur Amorim
arthur.amorim@ucf.edu
arthur.amorim@inl.gov
Website

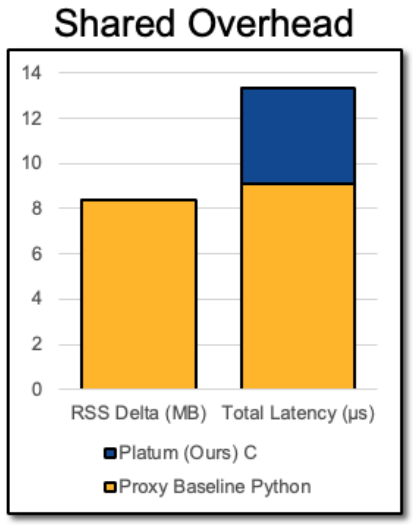
Idaho National Laboratory





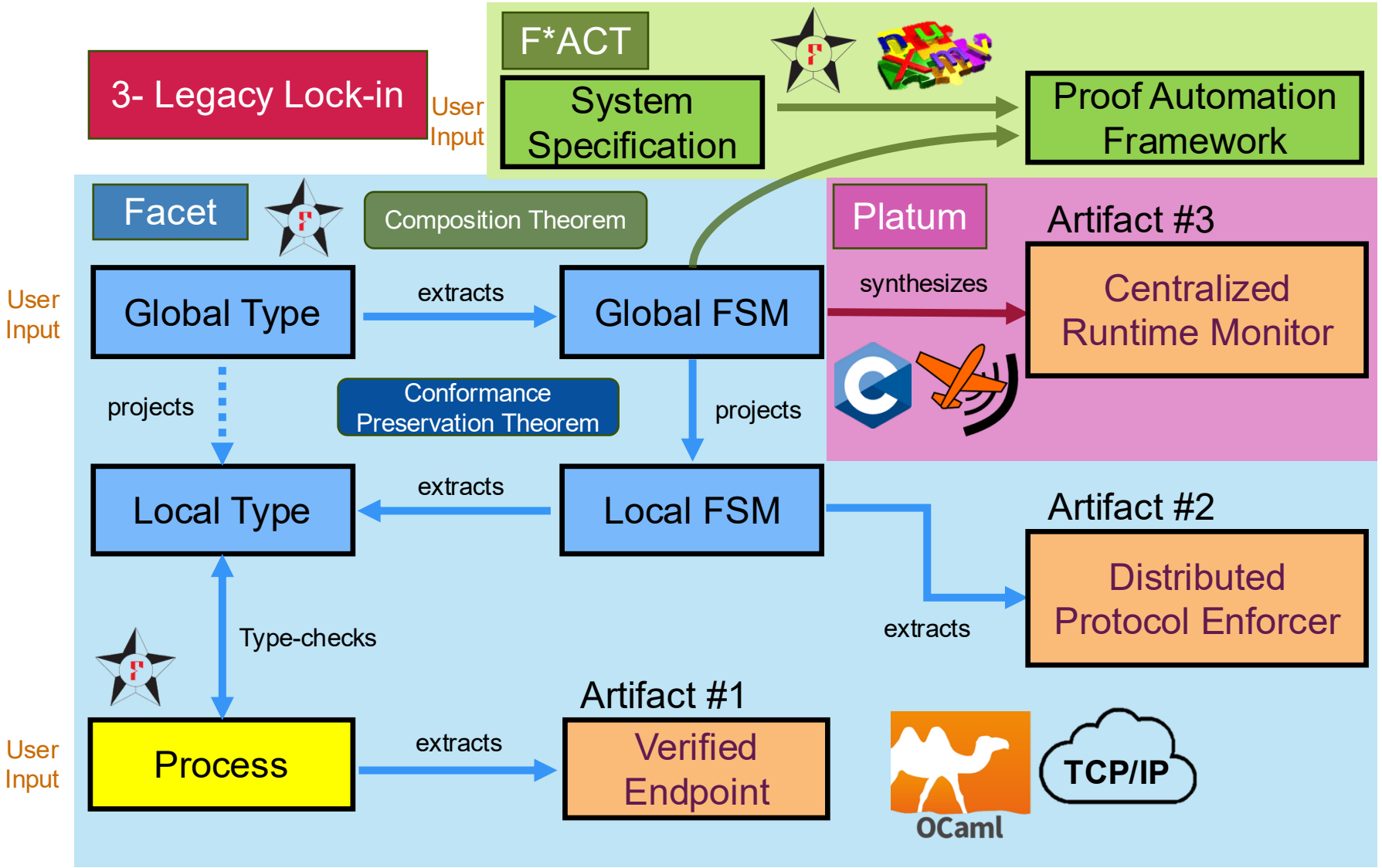






Idaho National Laboratory

Platum: Retrofitting Legacy Protocols.



Evaluation numbers

System	Language	RSS Delta (MB)	Total Latency (µs)	Logic Delta (µs)
DATUM	OCaml	13.72	56.74	27.14
Proxy Baseline	Python	8.39	9.11	0
Platum (Ours)	C	8.39	13.33	4.22