Certifiable Distributed Runtime Assurance of Distributed Real-Time Systems

Sagar Chaki and Dionisio de Niz

AIAA SciTech

January 10, 2017

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM-0004363

Motivation

Distributed Real-Time Systems of great relevance to aerospace community

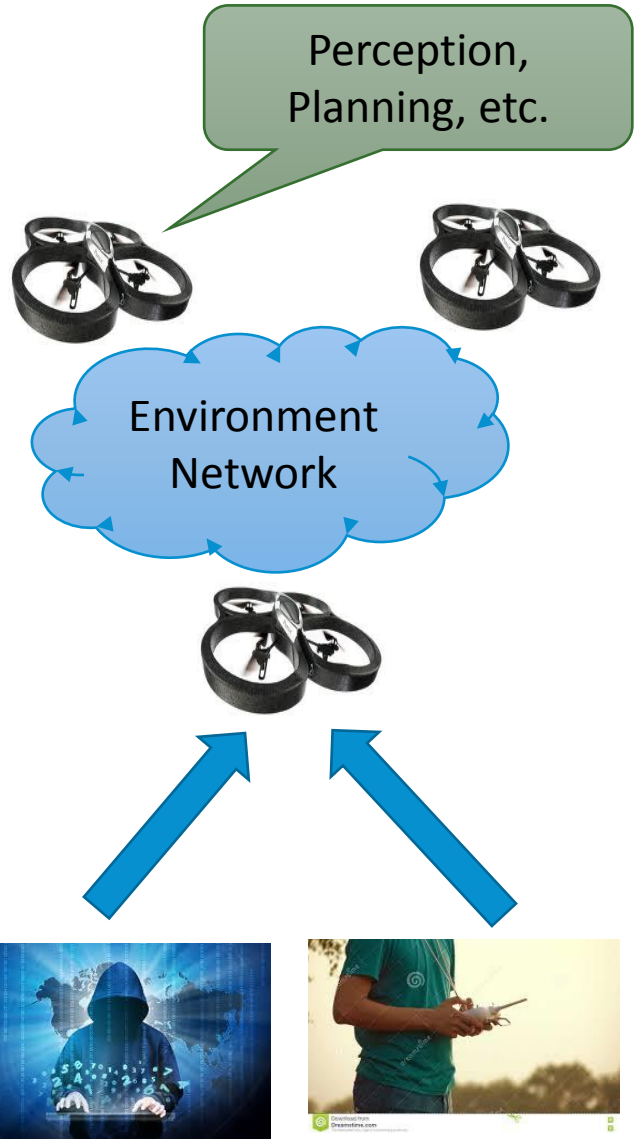
- Single aircraft with multiple sub-systems
- Multiple UASs coordinating to achieve mission

Operate in uncertain and unknown environments

- External uncertainty – normal and denied environments
- Internal uncertainty – sophisticated components with unpredictable behavior, e.g., machine learning

Safety-critical & hard real-time requirements

- Failures can be catastrophic
- How do we verify & certify?



Runtime Assurance

Suppose we want to assure that system S satisfies property Φ

Key Idea:

- Add a runtime "enforcer" to observe the behavior of S
- Step in and take enforcement action to prevent violation of Φ

Developed by AFRL & Barron Associates:

- http://www.mys5.org/Proceedings/2016/Day_1/2016-S5-Day1_1435_Schierman.pdf
- https://www.cs.indiana.edu/~lepike/pubs/RTA-CPS.pdf
- Enforcer referred to as the "reversionary system"

Prior Related Work (1)

Control theory – Simplex – CMU

- Seto, D., Krogh, B., Sha, L., and Chutinan, A., The simplex architecture for safe online control system upgrades, Proceedings of the American Control Conference, 1998.

Security Automata (Schneider) and Edit Automata (Ligatti et al.)

- Schneider, F. B., Enforceable security policies, ACM Transactions on Information and System Security (TISSEC), Vol. 3, No. 1, February 2000
- Ligatti, J., Bauer, L., and Walker, D., Edit automata: enforcement mechanisms for run-time security policies, International Journal of Information Security (IJIS), Vol. 4, No. 1-2, February 2005

Prior Related Work (2)

Runtime Verification – specific safety properties

- Kim, M., Viswanathan, M., Ben-Abdallah, H., Kannan, S., Lee, I., and Sokolsky, O., Formally specified monitoring of temporal properties, Proceedings of the 11th Euromicro Conference on Real-Time Systems (ECRTS '99), June 1999
- Havelund, K. and Rosu, G., Monitoring Programs Using Rewriting, Proceedings of the 16th International Conference on Automated Software Engineering (ASE '01), November 2001

Limitations:

- single properties over single components
- enforcers implementations not formally verified
- restricted enforcer scheduling model
- enforcer runs at the same level as S

Our Project: Certifiable Distributed Runtime Assurance (CDRA)

Problem: Runtime assurance (RA) is critical for complex non-deterministic systems.

Key idea: **monitor** the system and take preemptive action to avoid unsafe states; monitors are **simpler** more **verifiable**.

Challenges:

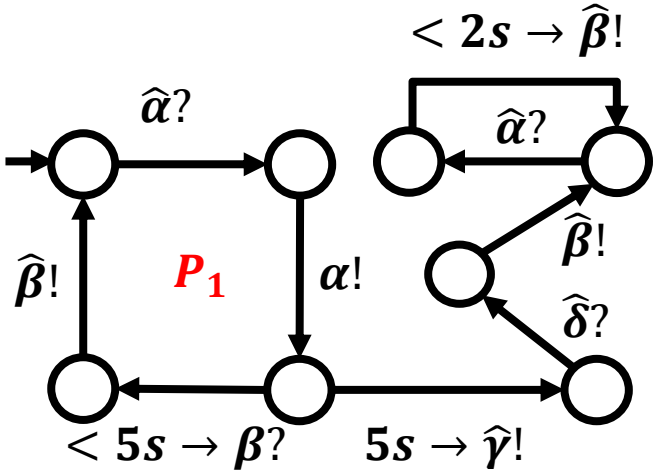
- **specifying** safety policies rigorously;
- **verifying** monitor (aka enforcer) implementations;
- preventing unsafe **inter-monitor** interactions (single-node and distributed systems);
- protecting monitors from being **circumvented**.

Solution: A combination formal policy specifications, software verification, compositional reasoning, and verified hardware-supported isolation.

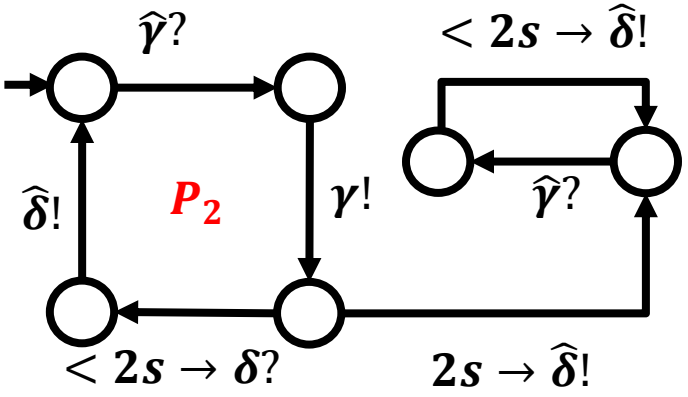
CDRA: Approach (1)

Prove $E_1 \preccurlyeq P_1$ and $E_2 \preccurlyeq P_2$ using software verification

Controller Policy



Logger Policy



Controller Enforcer Implementation (E_1)

Logger Enforcer Implementation (E_2)

Controller Component (C_1) $\{\alpha?, \beta!\}$ $\{\gamma?, \delta!\}$ **Logger Component (C_2)**

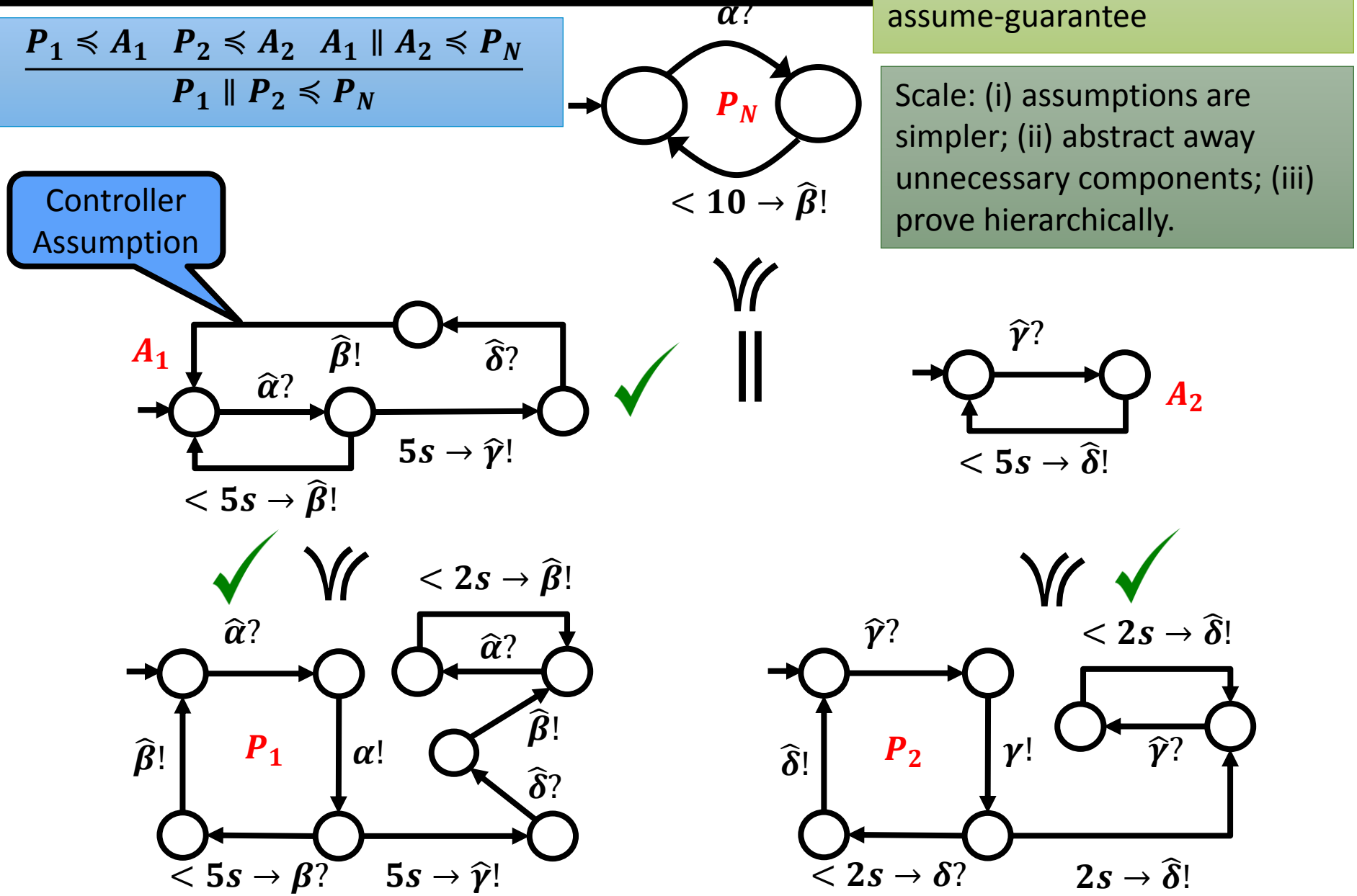
Node

CDRA: Approach (2)

Verify $P_1 \parallel P_2 \preccurlyeq P_N$ using assume-guarantee

$$\frac{P_1 \preccurlyeq A_1 \quad P_2 \preccurlyeq A_2 \quad A_1 \parallel A_2 \preccurlyeq P_N}{P_1 \parallel P_2 \preccurlyeq P_N}$$

Scale: (i) assumptions are simpler; (ii) abstract away unnecessary components; (iii) prove hierarchically.

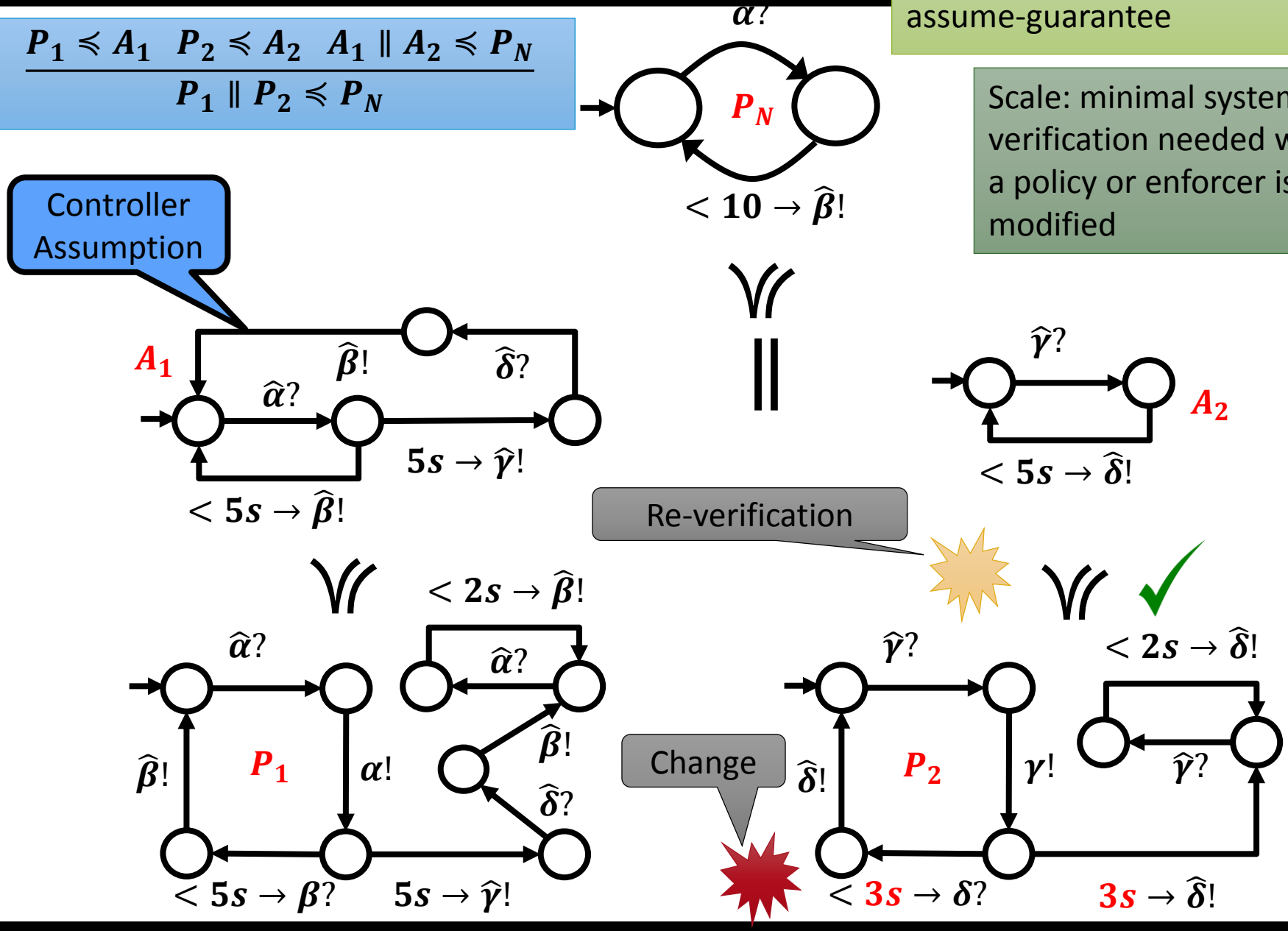


CDRA: Approach (3)

Verify $P_1 \parallel P_2 \preccurlyeq P_N$ using assume-guarantee

$$\frac{P_1 \preccurlyeq A_1 \quad P_2 \preccurlyeq A_2 \quad A_1 \parallel A_2 \preccurlyeq P_N}{P_1 \parallel P_2 \preccurlyeq P_N}$$

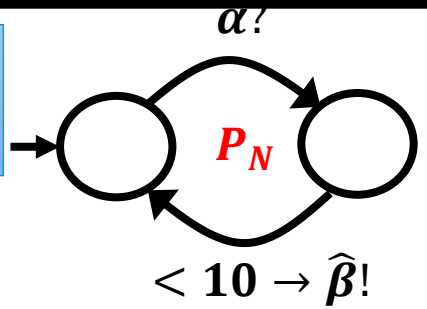
Scale: minimal system re-verification needed when a policy or enforcer is modified



CDRA: Approach (4)

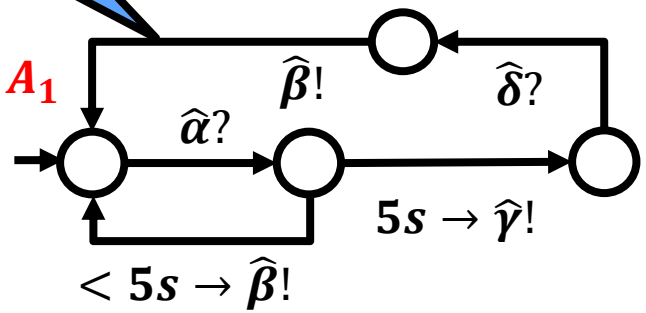
Verify $P_1 \parallel P_2 \preccurlyeq P_N$ using assume-guarantee

$$\frac{P_1 \preccurlyeq A_1 \quad P_2 \preccurlyeq A_2 \quad A_1 \parallel A_2 \preccurlyeq P_N}{P_1 \parallel P_2 \preccurlyeq P_N}$$

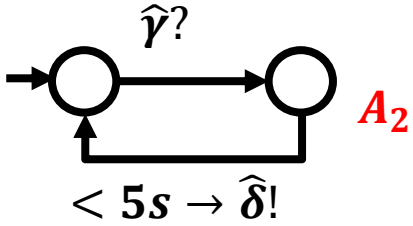


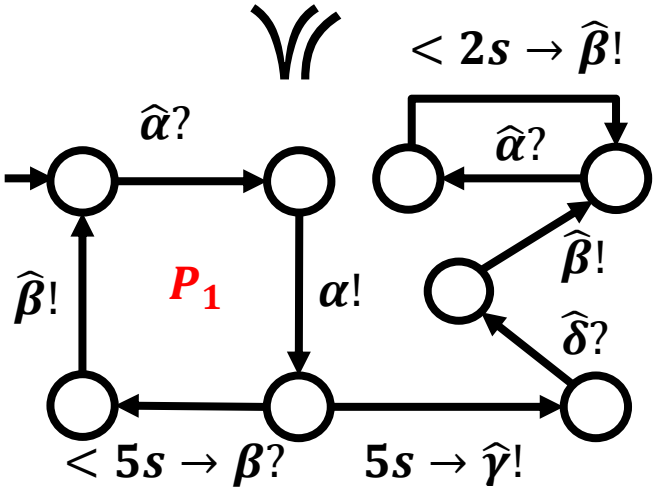
(i) Other (circular) rules exist; (ii) Challenges – (a) proving rule soundness; (b) finding right assumption.

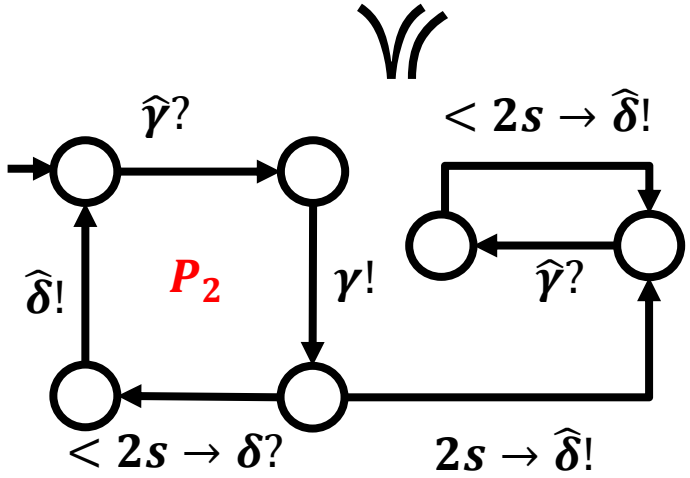
Controller Assumption



$\parallel$

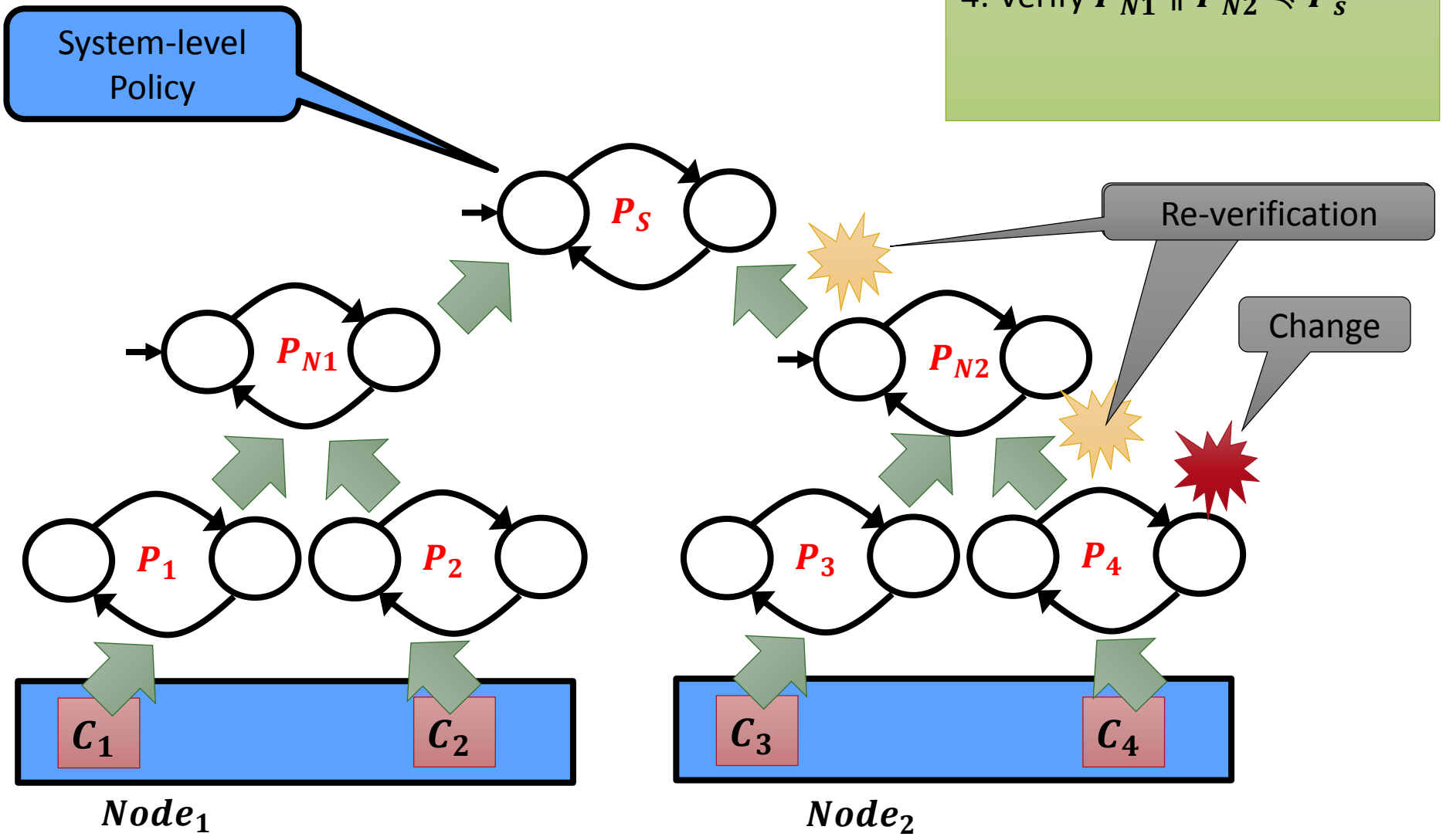






CDRA: Approach (5)

1: Verify $E_i \preccurlyeq P_i$
2: Verify $P_1 \parallel P_2 \preccurlyeq P_{N1}$
3: Verify $P_3 \parallel P_4 \preccurlyeq P_{N2}$
4: Verify $P_{N1} \parallel P_{N2} \preccurlyeq P_s$



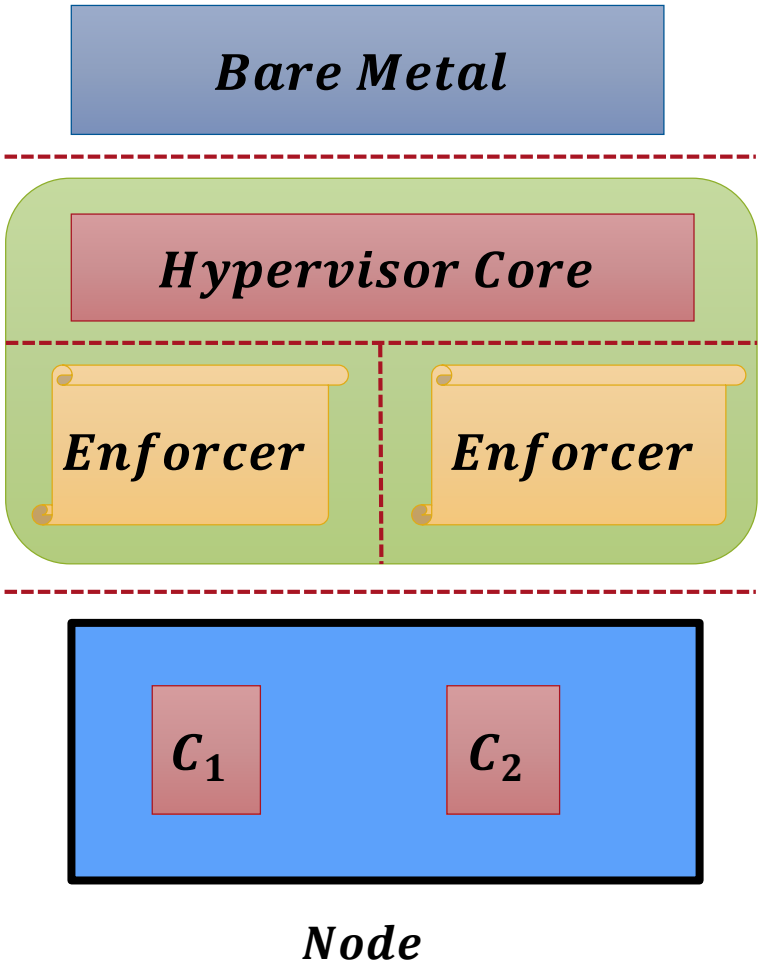
CDRA Approach (6)

Problem: Ensure that E_i is not circumvented. Inlining E_i in C_i will not work.

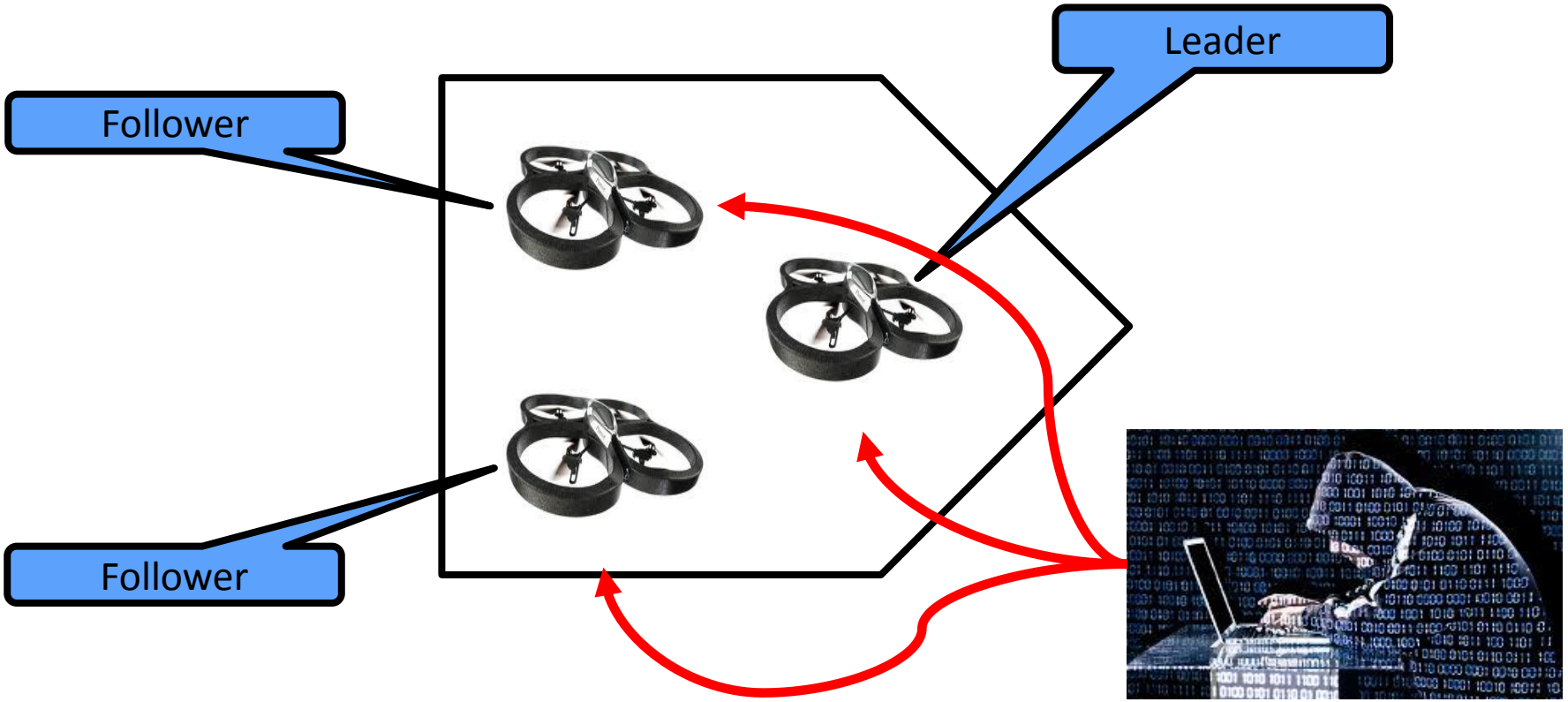
Solution: Use a hypervisor to execute E_i in an isolated environment. Prove correctness of isolation by verifying the hypervisor.

Build on XMHF: Amit Vasudevan, Sagar Chaki, Limin Jia, Jonathan M. McCune, James Newsome, Anupam Datta: *Design, Implementation and Verification of an eXtensible and Modular Hypervisor Framework.* IEEE Symposium on Security and Privacy 2013: 430-444. Many security- relevant applications already developed on top of XMHF. See references.

Challenges: Performance, Correctness

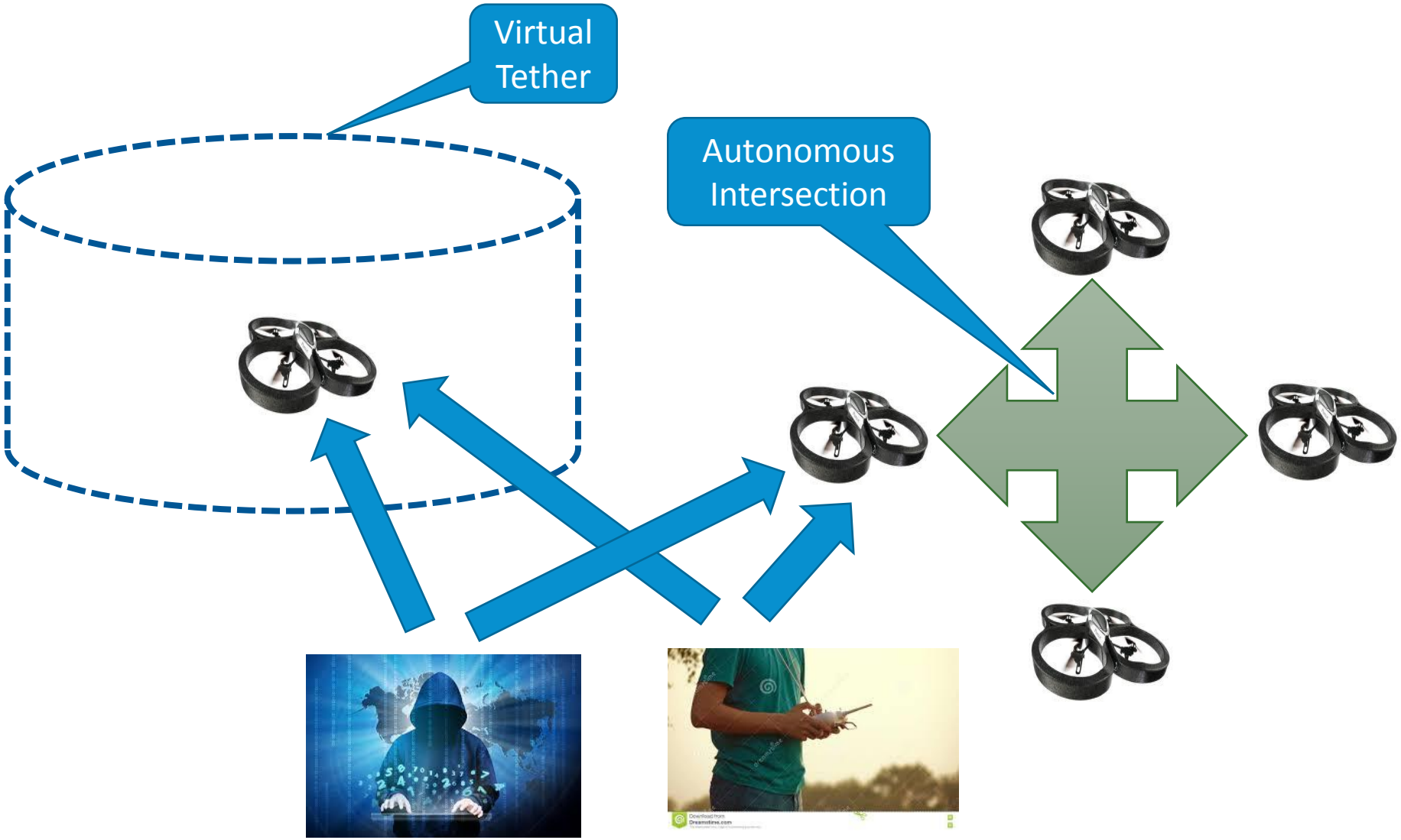


CDRA Validity



Demonstrate verified runtime assurance on a realistic scenario.
Give red team full control over applications (e.g., root access to OS)
Initially in simulation, eventually on a hardware platform.

CDRA Challenge Problems

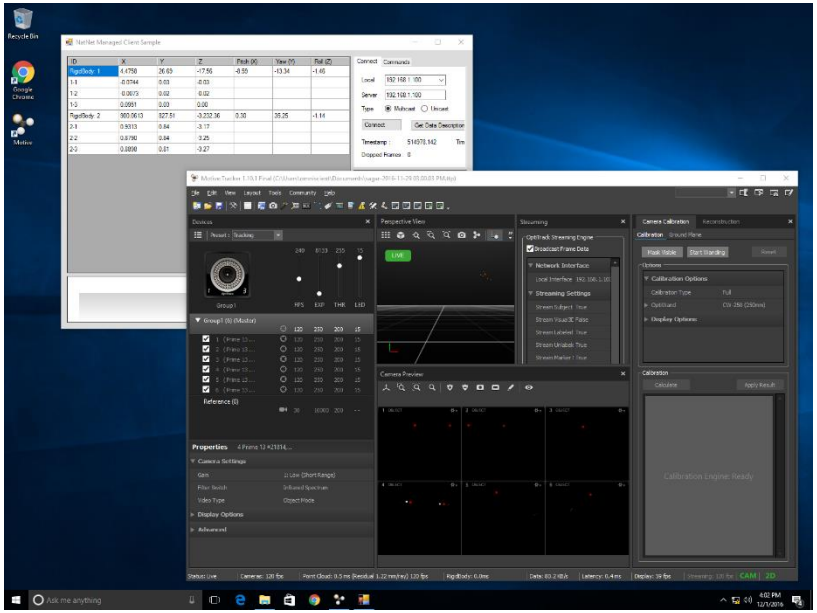


CDRA Testbed

Parrot Minidrones

Optitrack Localization

Optitrack Localization



ETH QuadCopter

Questions?