

Formal Methods and Tool-suites for CPS Security, Safety and Verification

Lecture 5: Vignettes in: Security and Trust in Networked Systems, Automotive CPS, Stable Path Routing in MANET, Composable and Assured Autonomy

John S. Baras

Institute for Systems Research and Dept. of Electr. and Comp. Engin.

University of Maryland College Park, USA,

and ACCESS Centre Royal Instit. of Technology (KTH), Stockholm, Sweden,

and Instit. for Advanced Study Technical Univ. of Munich (TUM), Germany

August 3, 2018

**MARKTOBERDORF INTERNATIONAL SUMMER SCHOOL ON ENGINEERING SECURE AND
DEPENDABLE SOFTWARE SYSTEMS**

Part I: Security and Trust in Networks and Networked Systems

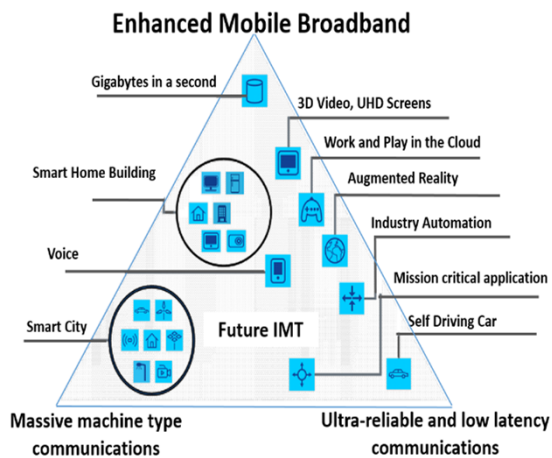
Part II: Hardware Software Co-design for Automotive CPS using Architecture Analysis and Design Language

Part III: Distributed Topology Control for Stable Path Routing in Multi-Hop Wireless Networks

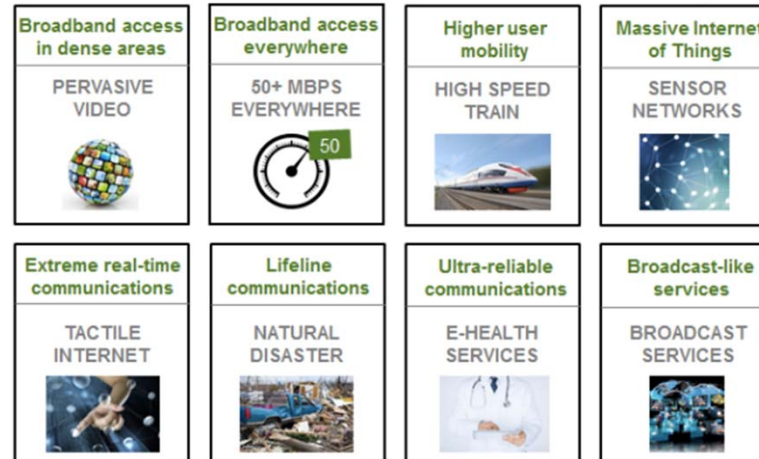
Part IV: Composable and Assured Autonomy

Part I: Security and Trust in Networks and Networked Systems

5G Use Cases



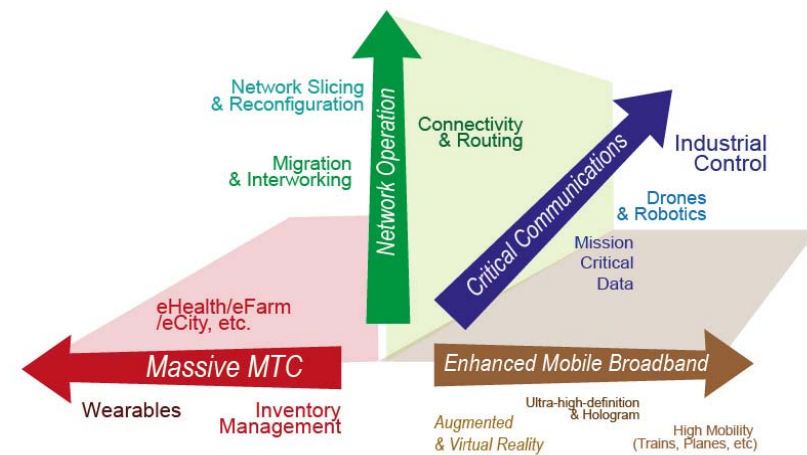
SRC: ITU-R



SRC: NGMN

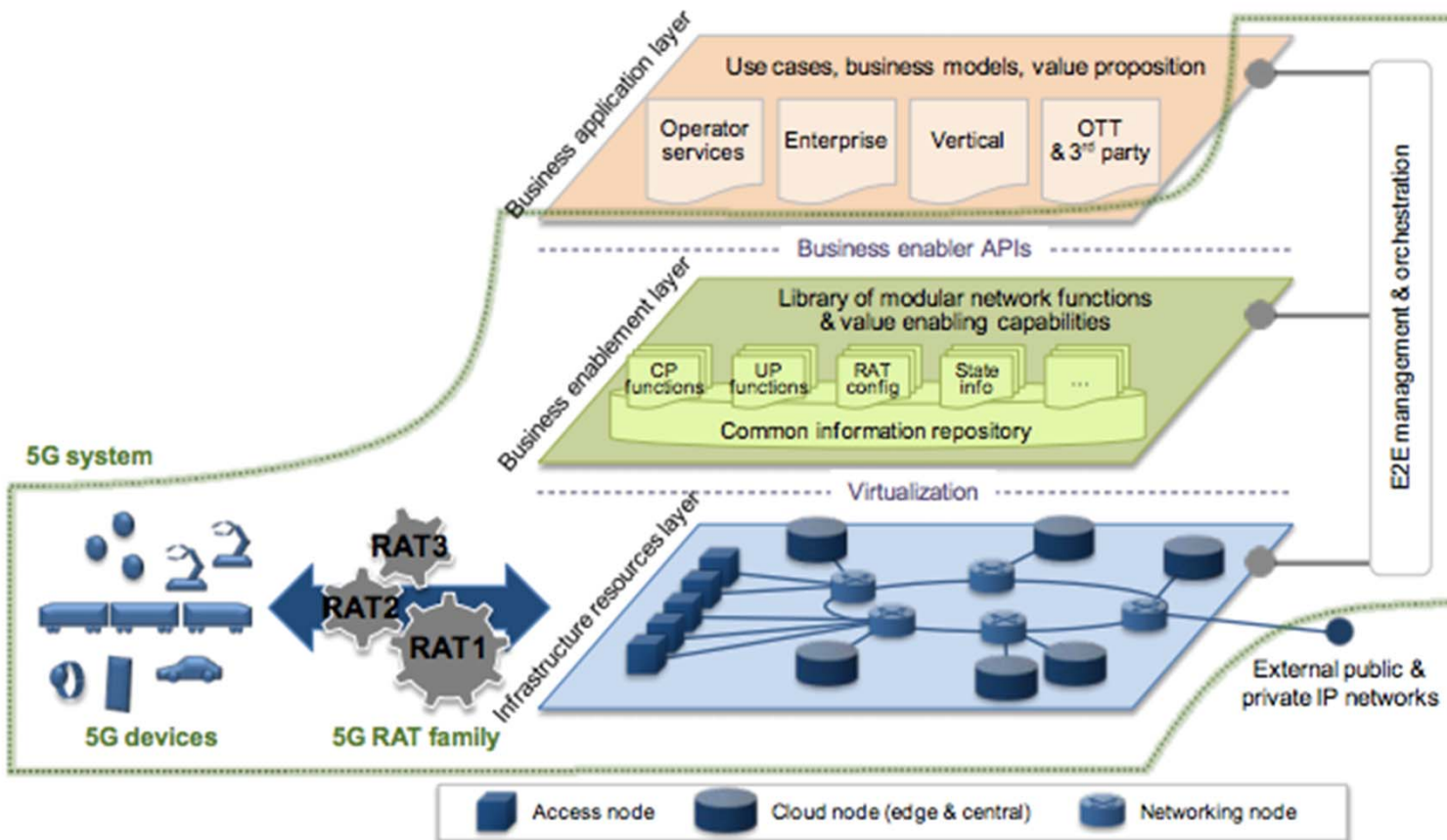


SRC: 5G PPP



SRC: 3GPP

5G Architecture



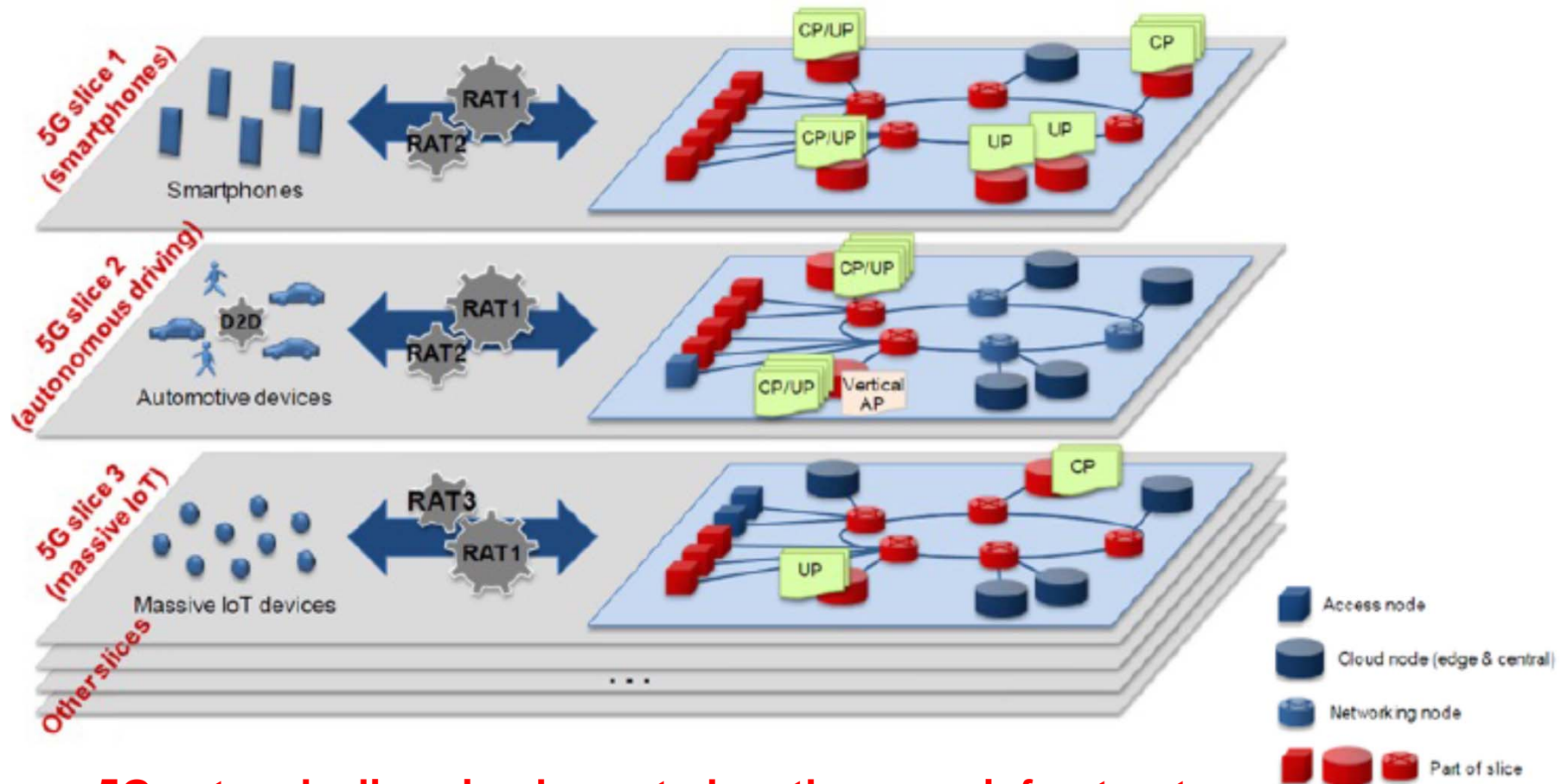
SRC: NGMN

Virtual Networking: NFV & SDN – Key Enablers for 5G and IoT



- Network Functions Virtualization (NFV) decouples network functions from dedicated hardware devices
 - Network services (routers, firewalls, load balancers , etc.) can now be hosted on virtual machines
- SDN is an architectural model that offers network virtualization and programmability
 - SDN abstracts the network control plane from the data plane
 - Some definitions are less focused on decoupling the planes, and more on APIs and integration

Network Slicing



5G network slices implemented on the same infrastructure

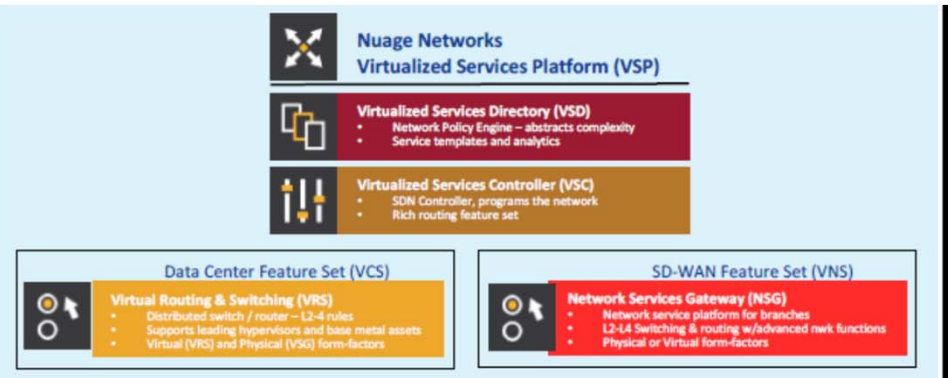
SRC: NGMN

Virtualizing the Network – Network as a Service (NaaS)?

Connecting USERS with APPLICATIONS

*Single Policy based Network Automation
Platform from the DC to the Branch*

Virtualized Services Platform (VSP)



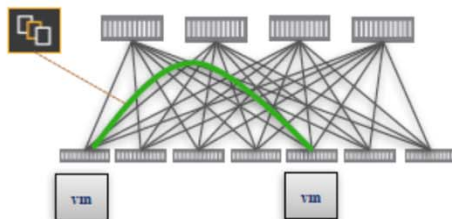
Virtualized Cloud
Services (VCS)

Virtualized Network
Services (VNS)

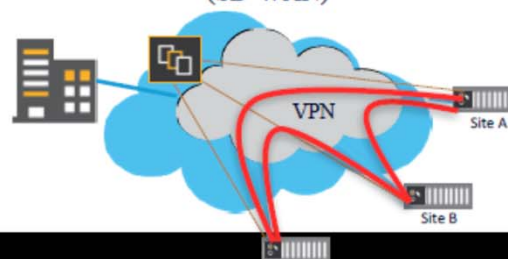
Virtualized Services
Assurance Platform
(VSAP)

Virtualized Security Services
(VSS)

Data Center
(Private Cloud)



Connecting & Serving
Disparate Locations
(SD-WAN)



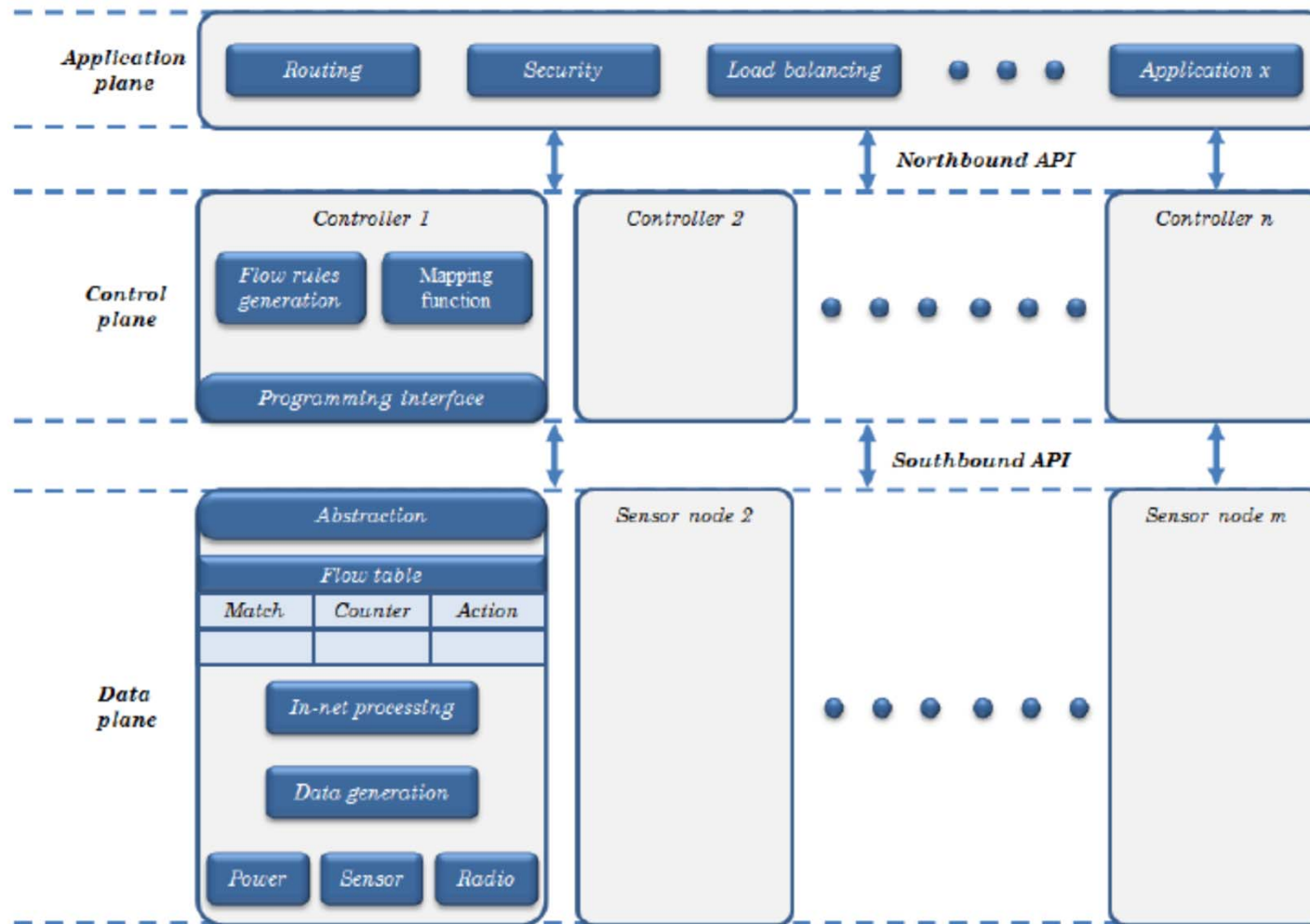
Operational Tools
(Monitoring / Correlation)



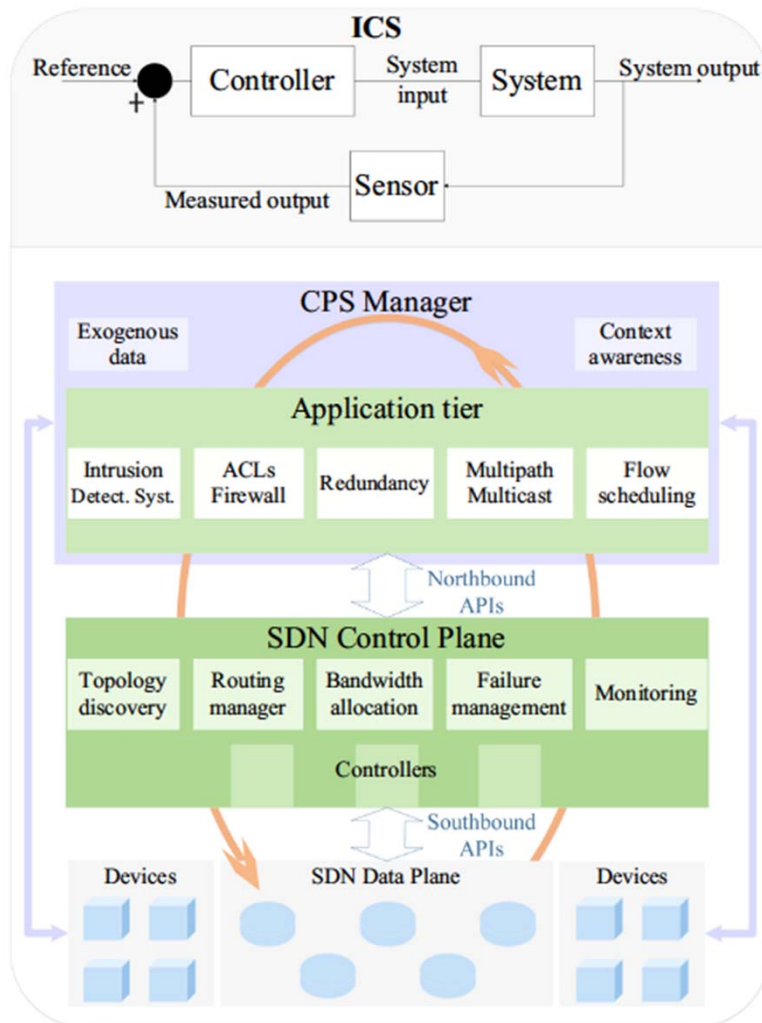
Micro-segmentations & Analytics
Prevent, Detect & Respond



Basic SDWN Architecture



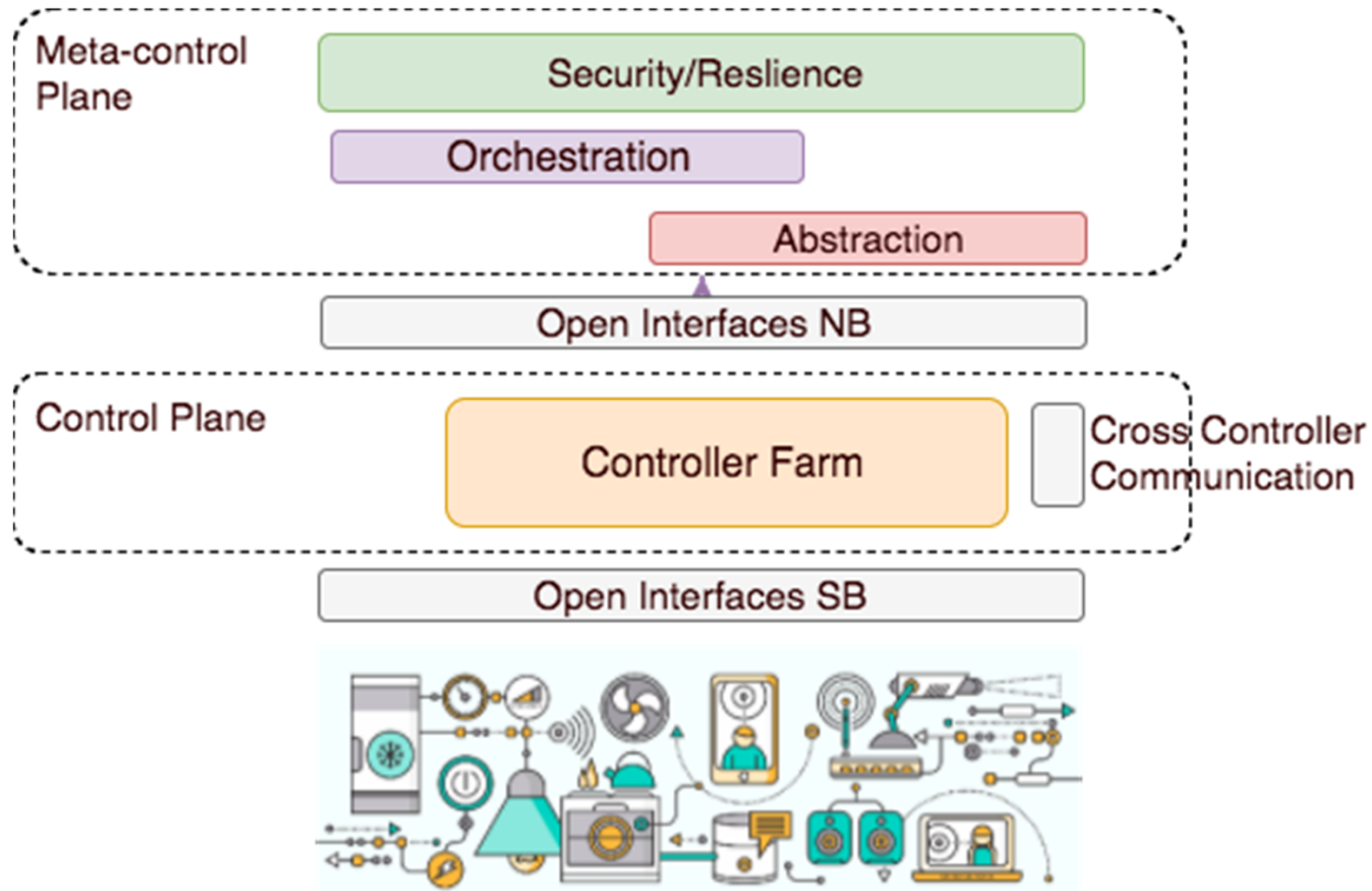
Different functionalities distributed along three planes



E. Molina and E. Jacob, “SDN in CPS: A Survey”, 2017

- CPS, Net-CPS, Net-HCPS can be viewed as distributed partially asynchronous implementations of the sense-decide-actuate cycle in control systems
- The collaboration and information network in our multi-layer model of CPS, net-CPS and Net-HCPS impose requirements on the communication network including: priority, bandwidth constraints, predictability, timeliness, robustness, survivability, network security
- The SDN paradigm and associated concepts such as Network Function Virtualization (NFV) can play a key role in meeting these requirements

SD-CPS Architecture



Challenges and Benefits

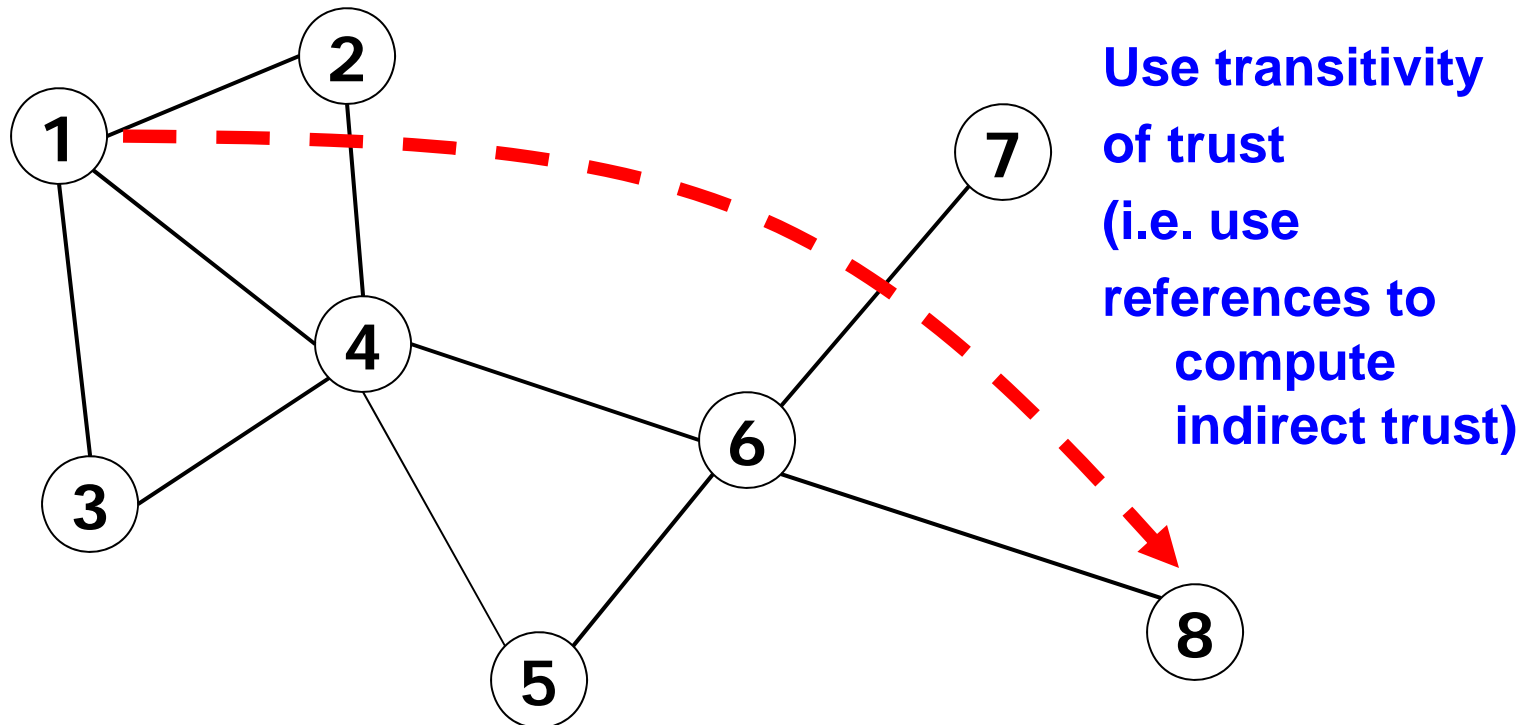
- ✓ Manageability
 - e.g., Automation, orchestration and network re-configuration
- ✓ Resource Allocation
- ✓ Real-time
- ✓ Reliability
 - e.g. failure detection and recovery
- ✓ Security
 - e.g. security policies applied on demand
- ✓ Interoperability
 - e.g., open standard interfaces

- **Trust and reputation critical for collaboration**
- Characteristics of trust relations:
 - *Integrative* (Parsons 1937) – main source of social order
 - *Reduction of complexity* – without it bureaucracy and transaction complexity increases (Luhmann 1988)
 - *Trust as a lubricant for cooperation* (Arrow 1974) – rational choice theory
- **Social Webs, Economic Webs**
 - MySpace, Facebook, Windows Live Spaces, Flickr, Classmates Online, Orkut, Yahoo! Groups, MSN Groups
 - e-commerce, e-XYZ, services and service composition
 - **Reputation** and **recommender** systems

Indirect Network Trust

User 8 asks for access to User 1's files.
User 1 and User 8 have no previous interaction

What should User 1 do?



Indirect Trust: System Model



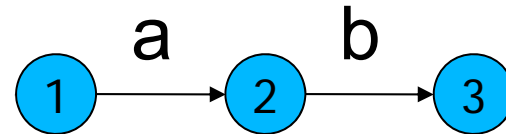
- System mapped to a **weighted, directed graph**
 - **Vertices** : entities/users
 - **Edges** : direct trust relations
 - **Weights** : $w(i,j)$ = How much i trusts j
- Establish an indirect trust relation, between users that have not had direct interactions
 - We assume that trust is **transitive (at least partially)**
- **Trust computation: path problem on a graph**
 - Information about j that is useful to $i \Leftrightarrow$
Directed paths from i to j
 - **Combine information along each path, and then aggregate across paths**

- **Shortest Path Problem**
 - Semiring: $(\mathcal{R}_+, \min, +)$
 - \otimes is $+$ and computes **total path delay**
 - \oplus is \min and **picks shortest path**
- **Bottleneck Problem**
 - Semiring: $(\mathcal{R}_+, \max, \min)$
 - \otimes is \min and computes **path bandwidth**
 - \oplus is \max and **picks highest bandwidth**

Trust Semiring Properties: Partial Order

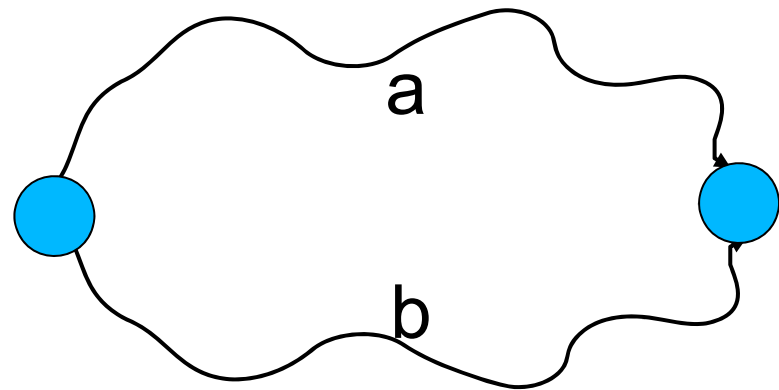
- Combined **along-a-path weight should not increase** :

$$a \otimes b \leq a, b$$



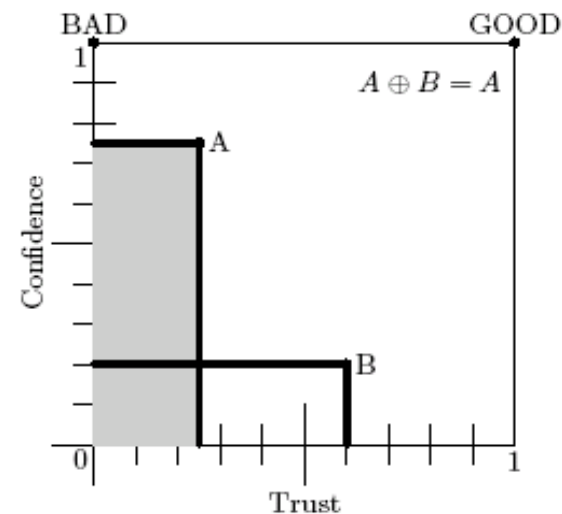
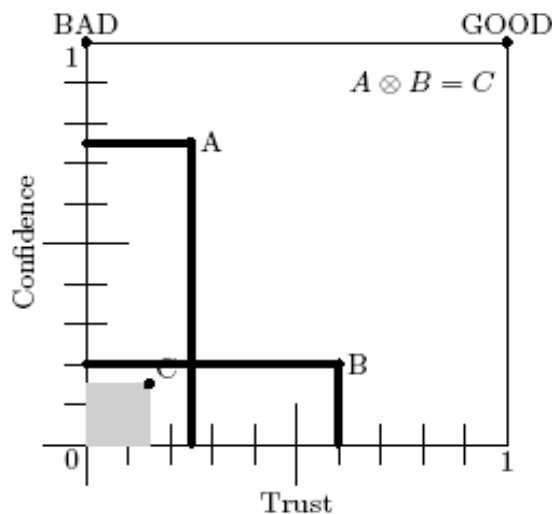
- Combined **across-paths weight should not decrease** :

$$a \oplus b \geq a, b$$



Trust Path Semiring

- $0 \leq \text{trust}, \text{confidence} \leq 1$
- \otimes is $(t_{ik}, c_{ik}) \otimes (t_{kj}, c_{kj}) = (t_{ik}t_{kj}, c_{ik}c_{kj})$
- \oplus is $(t_{ij}^{p1}, c_{ij}^{p1}) \oplus (t_{ij}^{p2}, c_{ij}^{p2}) = \begin{cases} (t_{ij}^{p1}, c_{ij}^{p1}) & \text{if } c_{ij}^{p1} > c_{ij}^{p2} \\ (t_{ij}^{p2}, c_{ij}^{p2}) & \text{if } c_{ij}^{p1} < c_{ij}^{p2} \\ (\max\{t_{ij}^{p1}, t_{ij}^{p2}\}, c_{ij}^{p1}) & \text{if } c_{ij}^{p1} = c_{ij}^{p2} \end{cases}$



Trust Distance Semiring

- Motivated from Eisner's Expectation Semiring (2002) (speech/language processing)

$$(a_1, b_1) \otimes (a_2, b_2) = (a_1 b_2 + a_2 b_1, b_1 b_2)$$

$$(a_1, b_1) \oplus (a_2, b_2) = (a_1 + a_2, b_1 + b_2)$$

- $0 \leq \text{trust}, \text{confidence} \leq 1, (t, c) \rightarrow (c/t, c)$

$$S = [0, \infty] \times [0, 1]$$

- \otimes is $(t_{ik}, c_{ik}) \otimes (t_{kj}, c_{kj}) \rightarrow \left(\frac{1}{\frac{1}{t_{ik}} + \frac{1}{t_{kj}}}, c_{ik} c_{kj} \right)$

- \oplus is $(t_{ij}^{p_1}, c_{ij}^{p_1}) \oplus (t_{ij}^{p_2}, c_{ij}^{p_2}) \rightarrow \left(\frac{\frac{c_{ij}^{p_1} + c_{ij}^{p_2}}{c_{ij}^{p_1} c_{ij}^{p_2}}}{\frac{1}{t_{ij}^{p_1}} + \frac{1}{t_{ij}^{p_2}}}, c_{ij}^{p_1} + c_{ij}^{p_2} \right)$

- Path interpretation

$$t_{i \rightarrow j} = \bigoplus_{\text{path } p: i \rightarrow j} t_{i \rightarrow j}^p$$

- Linear system interpretation

$$t_{i \rightarrow j} = \bigoplus_{\text{User } k} t_{i \rightarrow k} \bigoplus w_{k \rightarrow j}$$

$$\vec{t}_n = W \otimes \vec{t}_{n-1} \bigoplus \vec{b}$$

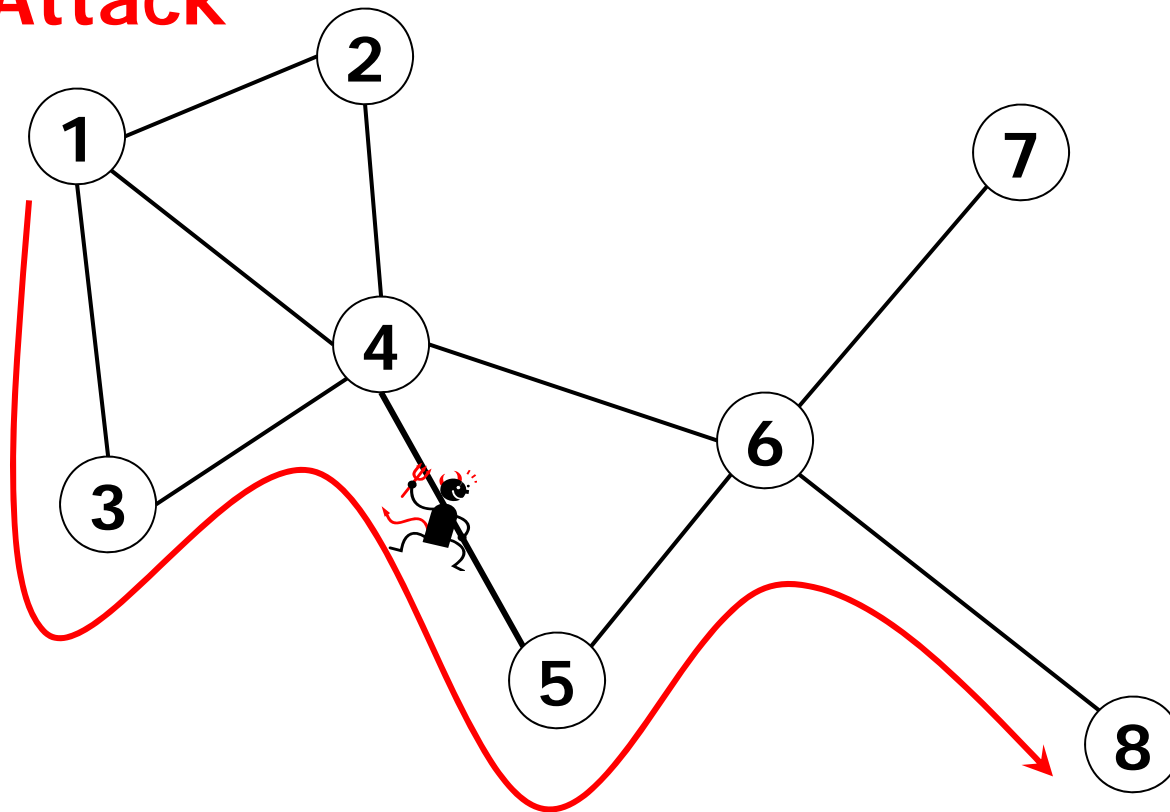
Indicator vector of pre-trusted nodes

- Treat as a **linear system**
 - We are looking for its **steady state**.

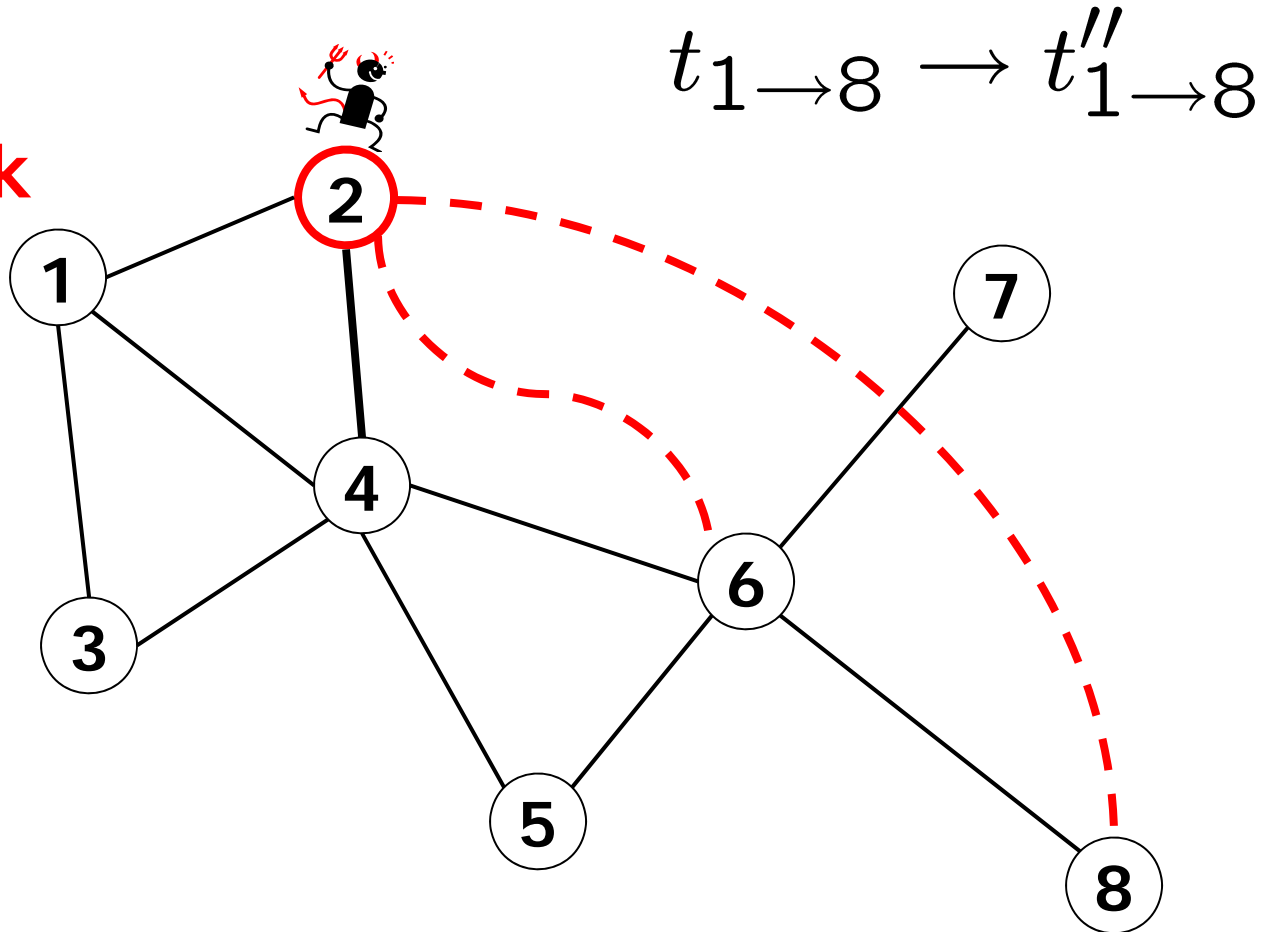
- **ATTACK** the trust computation!
 - **Aim:** Increase $t_{1 \rightarrow 8}$ to a level that would grant access.
- How?
 - **Edge attack:** change opinion on an edge (trick a node into forming false opinion)
 - **Node attack:** change any opinion emanating from a node (gain complete control of a node)


**Edge
Attack**

$$t_{1 \rightarrow 8} \rightarrow t'_{1 \rightarrow 8}$$



**Node
Attack**



- **Model**: Combined x-node, y-edge attack
- **Given**: topology, weights and semiring
 - What is the **maximum damage**  can cause?
 - Which nodes/edges are **more likely to be attacked**? (these will need extra protection)
- **Given**: topology and semiring
 - Designer chooses weights secretly from attacker to **Minimize the Maximum damage** the attacker can cause.

Most Vital Edge

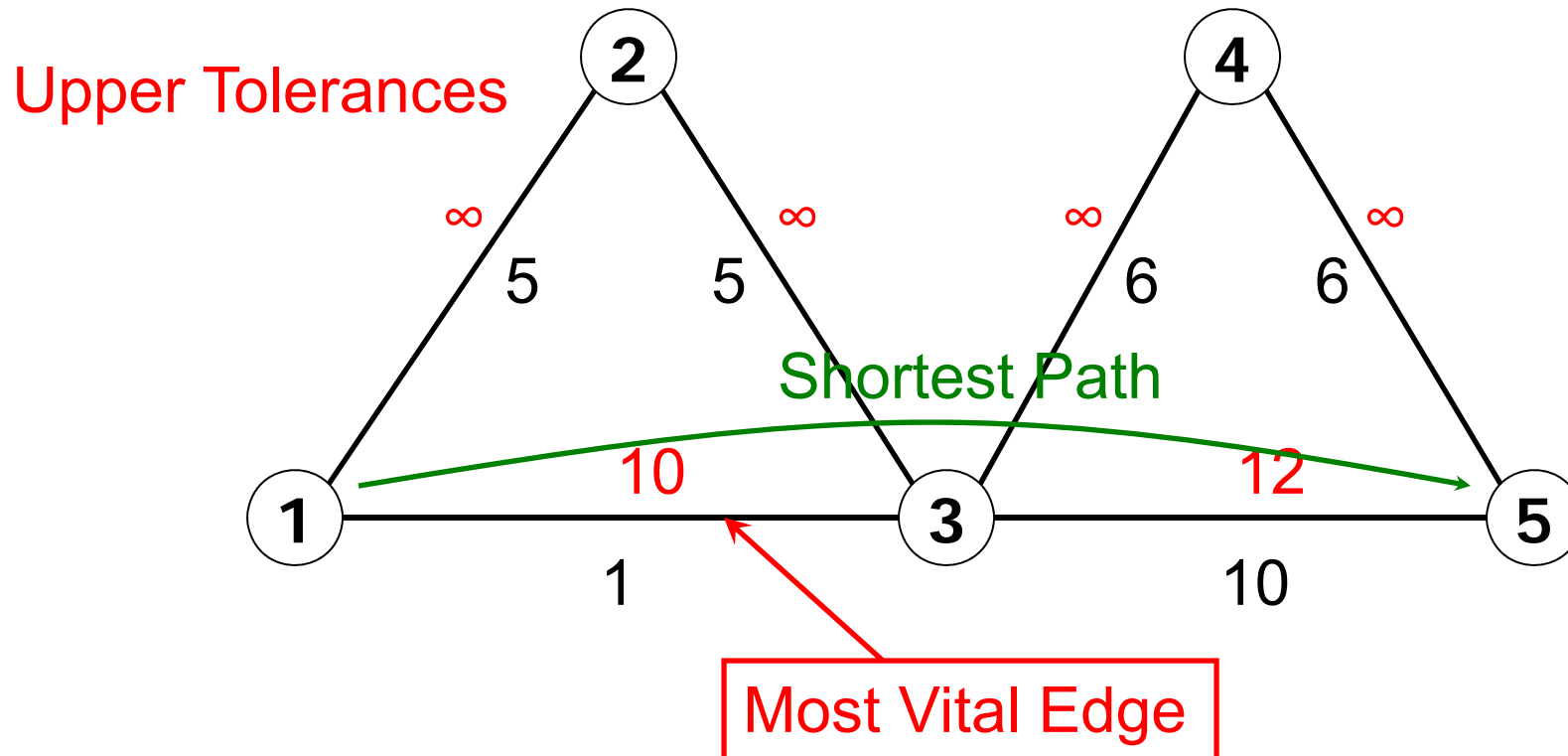
- In all cases, the Attacker attacks a single edge, called the “Most Vital Edge”.
- All that changes is the semiring and the interpretation of the weights.
- Ramaswamy, Orlin, and Chakravarti found two different characterizations of the Most Vital Edge for the $(\min, +)$ and the (\max, \min) semiring.
- Is there a unified characterization?

Edge Tolerances

- Upper (Lower) edge tolerance of an edge e , w.r.t. an optimal path p^* , is the highest (lowest) weight of e that would preserve the optimality of p^* .
- In a **shortest path** problem (min, +), the most vital edge is the path edge whose weight has the largest difference with the **upper tolerance**.
- In a **maximum capacity** problem (max, min), the most vital edge is the path edge whose weight has the largest difference with the **lower tolerance**.

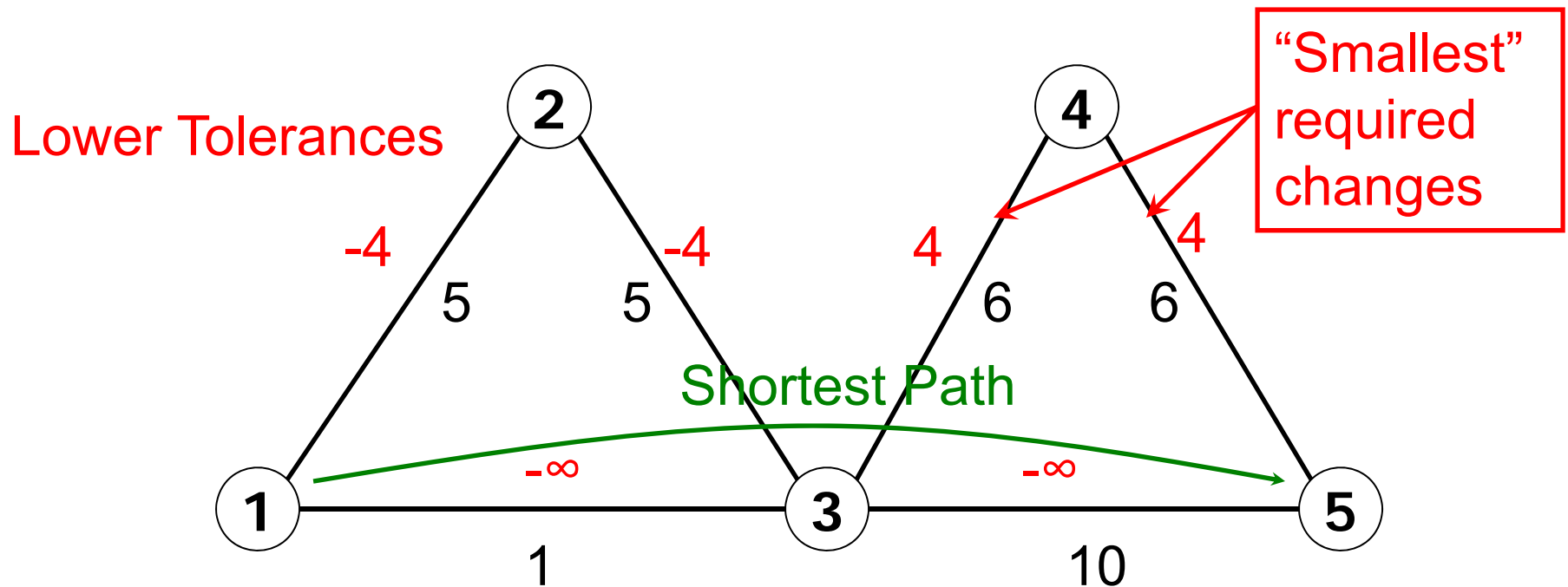
Upper Tolerance Example

- Upper Tolerances for the Shortest Path Problem



Lower Tolerance Example

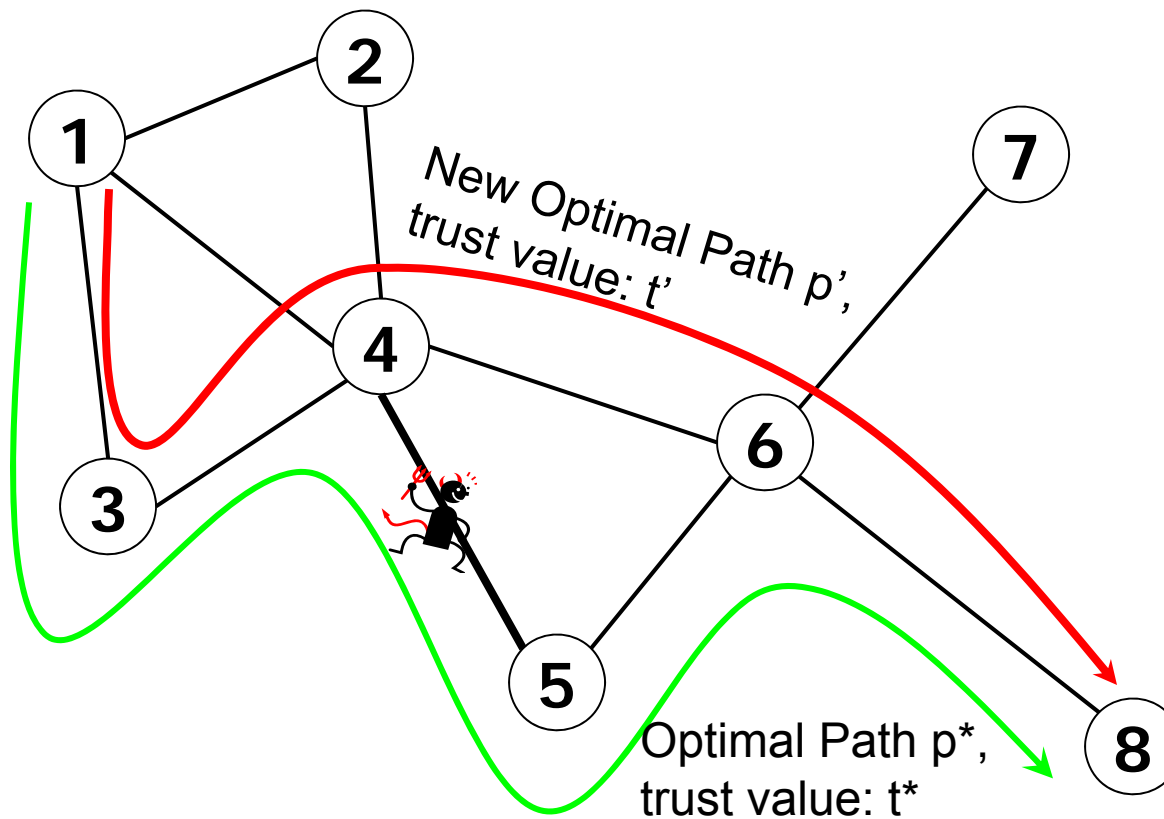
- Lower Tolerances for the Shortest Path Problem



Attacked Edge on the Path

Trust Edge Attack

$$t_{1 \rightarrow 8}^* > t'_{1 \rightarrow 8}$$

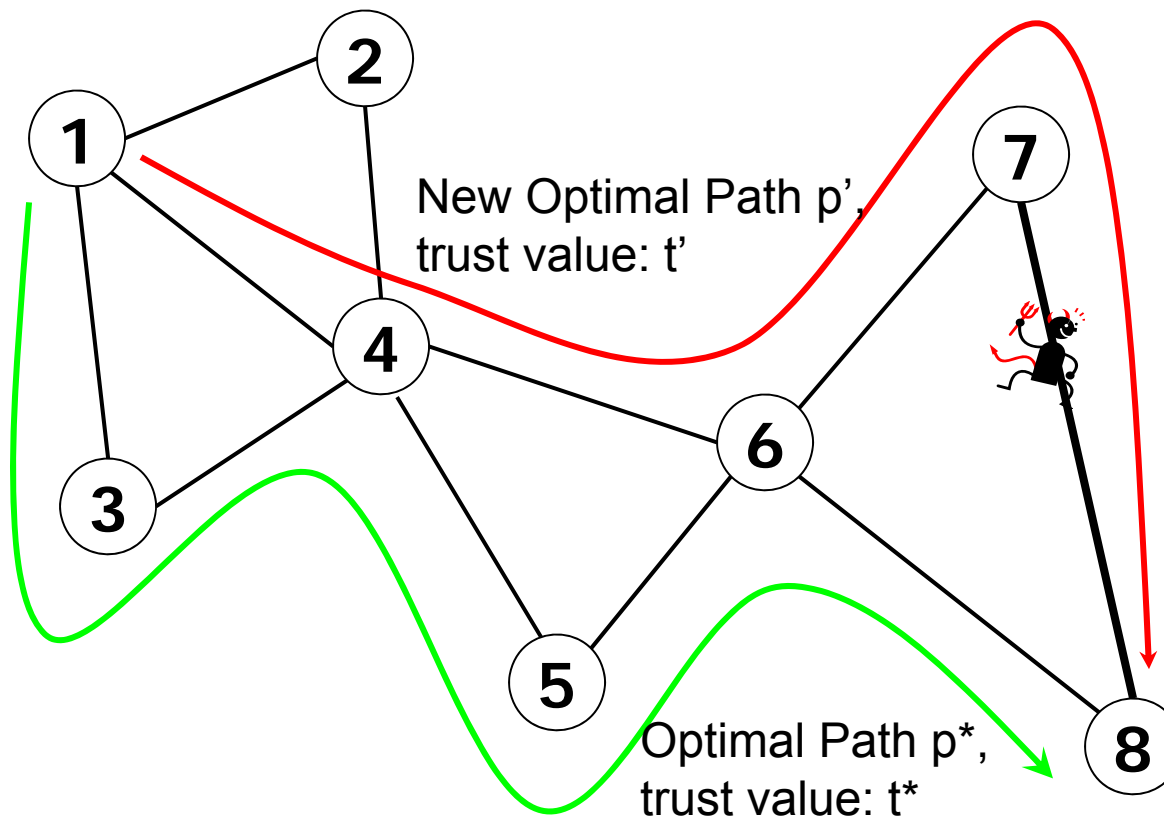


RESULT:
Decrease Trust!

Attacked Edge not on the Path

Trust Edge Attack

$$t_{1 \rightarrow 8}^* < t'_{1 \rightarrow 8}$$



RESULT:
Increase Trust!
Change Path!

Tolerances for *any* Optimization Semiring

- Optimization semirings: \oplus is min or max
- \oplus -minimal (maximal) tolerance α_e (β_e) of edge e instead of lower (upper) tolerance.
- \oslash is the inverse of \otimes defined by: $a \otimes x = b \Leftrightarrow x = b \oslash a$
- $w(e)$ is the weight of edge e . $w(p)$ is the weight of path p .

If $e \in p^*$

$$\bullet \alpha_e = \left(\bigoplus_{\substack{p:s \rightsquigarrow d \\ w(e) \leftarrow \textcircled{0}}} w(p) \right) \oslash w(p^* \setminus e).$$

$$\bullet \beta_e = \textcircled{1}.$$

If $e \notin p^*$

$$\bullet \alpha_e = \textcircled{0}.$$

$$\bullet \beta_e = w(p^*) \oslash \left(\bigoplus_{\substack{p:s \rightsquigarrow d \\ w(e) \leftarrow \textcircled{1}}} w(p) \right)$$

Tolerances for the Trust Semiring

- Assume (\max, \cdot) semiring; essentially equivalent to our trust semiring.
- Tolerances:

If $e \in p^*$

- $\alpha_e = \frac{w(e)}{w(p^*)} \cdot \left(\max_{w(e) \leftarrow 0} w(p) \right).$

- $\beta_e = 1.$

If $e \notin p^*$

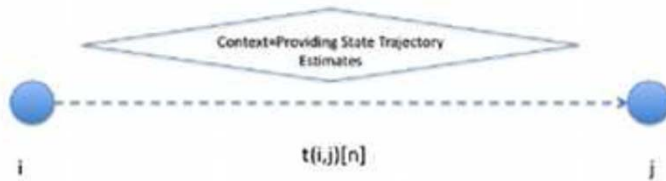
- $\alpha_e = 0.$

- $\beta_e = \frac{w(p^*)}{\max_{w(e) \leftarrow 1} w(p)}$

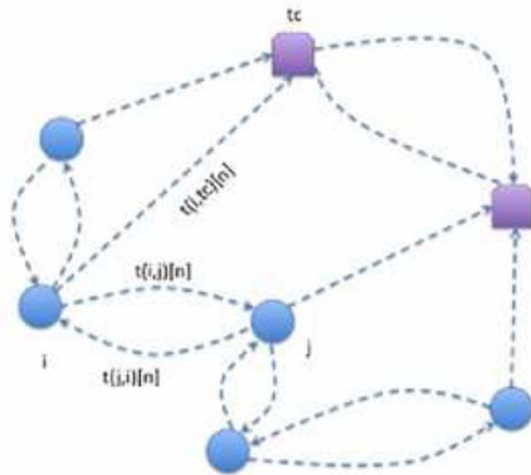
Distributed Kalman Filtering and Tracking: Performance Improvements from Trusted Core

- **Realistic sensor networks**: Normal nodes, faulty or corrupted nodes, malicious nodes
- **Hierarchical scheme** – provide global trust on a particular context without requiring direct trust on the same context between all agents
- Combine techniques from fusion centric, collaborative filtering, estimation propagation
- **Trusted Core**
 - **Trust Particles**, higher security, additional sensing capabilities, broader observation of the system, confidentiality and integrity, multipath comms
 - Every sensor can communicate with one or more trust particles **at a cost**

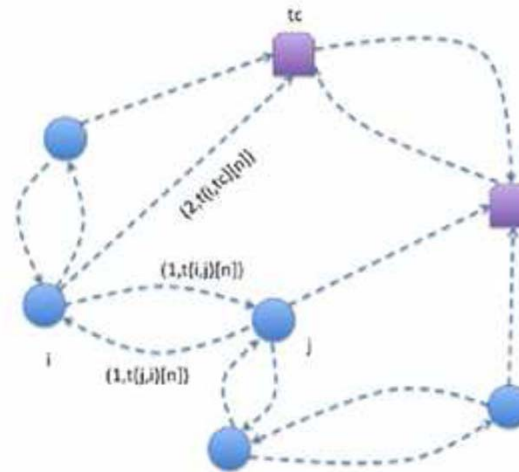
Trust and Induced Graphs



Trust relation



Weighted Directed Dynamic Trust Graph $G_t(V, A_t)$



Induced Graph $G(V, A)$

$$V_{tc} \subset V$$

$$w(i, j) = (c(i, j), t(i, j)[n])$$

Goals of Trusted System

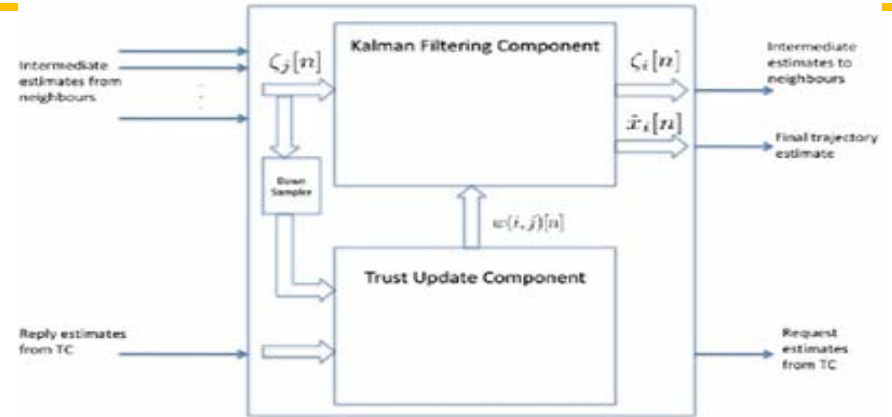
1. All the sensors which abide by the protocols of sensing and message passing, should be able to track the trajectories.
2. This implies that those nodes which have poor sensing capabilities, **nodes with corrupted sensors**, **should be aided** by their neighbors in tracking.
3. Those nodes which are **malicious and pass false estimates**, should be **quickly detected** by the trust mechanism and their estimates should be discarded.

$$x[n + 1] = A x[n] + B w[n]$$

$$z_i[n] = H_i[n] x[n] + v_i[n]$$

$$z_{tc} = H_{tc}[n] x[n] + v_{tc}[n]$$

- Can use **any valid trust system** as trust update component
- Can replace DKF with **any Distributed Sequential MMSE or other filter**
- Trust update mechanism: Linear credit and exponential penalty



Algorithm 1 Trusted Kalman Filter

```

Init  $M[0], \hat{x}_i = \underline{x}(0), n = 0$ 
repeat
   $n \leftarrow n + 1$ ;
  Prediction MSE
   $P[n] = AM[n-1]A^T + BQB^T$ 
  Kalman Gain
   $K[n] = P[n]H_i^T (R_i + H_i P[n]H_i^T)^{-1}$ 
  Local correction
   $\zeta_i[n] = A\hat{x}_i[n-1] + K[n](z_i[n] - H_i A\hat{x}_i[n-1])$ 
  The nodes exchanges the local estimates  $\hat{x}_j, \forall j \in \mathcal{N}^+(i)$ 
  Trust sensitive filtering
   $\hat{x}_i[n] = \sum_{j \in \mathcal{N}^+(i)} w_{ij} \times \zeta_j[n]$ 
  Estimation MSE
   $M[n] = (I - K[n]H_i[n])P[n]$ 
until Forever
  
```

Algorithm 2 Trust Update for the inclusive neighborhood

```

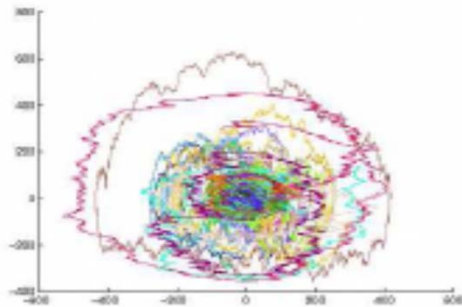
Init  $t(i,j)[0] = \frac{1}{|\mathcal{N}^+(i)|}, \forall j \in \mathcal{N}^+(i), \text{ and } k = 0$ 
repeat
  Wait for Exponential time  $\tau$ 
   $k \leftarrow k + \tau$ 
  Request Estimate update from the TC
  The TC replies with its trustworthy estimate  $\hat{x}_{tc}$ 
  for all  $j \in \mathcal{N}^+(i)$  do
     $dev(j) = \|\zeta_j - \zeta_{tc}\|_2$ 
     $t(i,j)[k] = \begin{cases} \min(max_T, t(i,j)[k-1]) + \delta & dev(j) \leq Dev_T \\ t(i,j)[k-1]/2 & dev(j) > Dev_T \end{cases}$ 
  end for
until Forever
  
```

Trusted DKF Performance

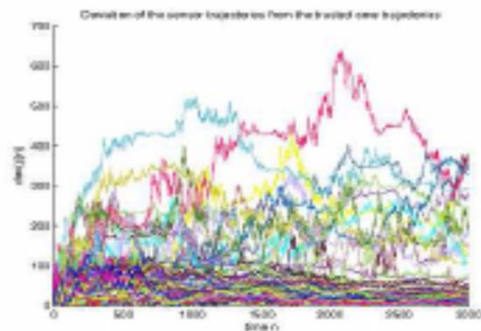
$$A = 2 \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

$$Q = 25I_2, \quad \underline{x}(0) = (15, -10)^T$$

$$H_{tc} = I_2, \quad R_{tc} = 30I_2$$

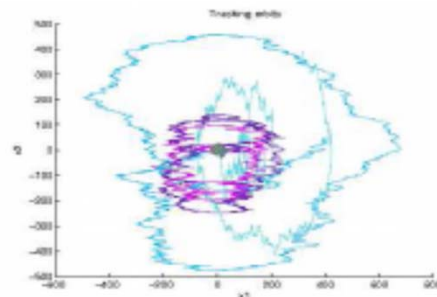


(a) Orbits

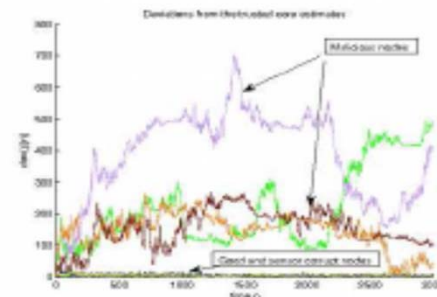


(b) Deviations

Open Loop Performance

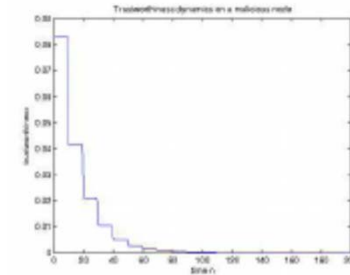


(a) Orbits

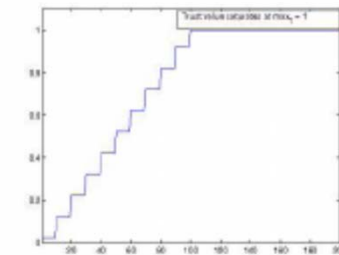


(b) Deviations

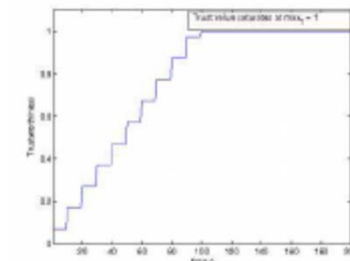
Closed Loop Performance



(a) Malicious Nodes



(b) Sensor Corrupt Nodes



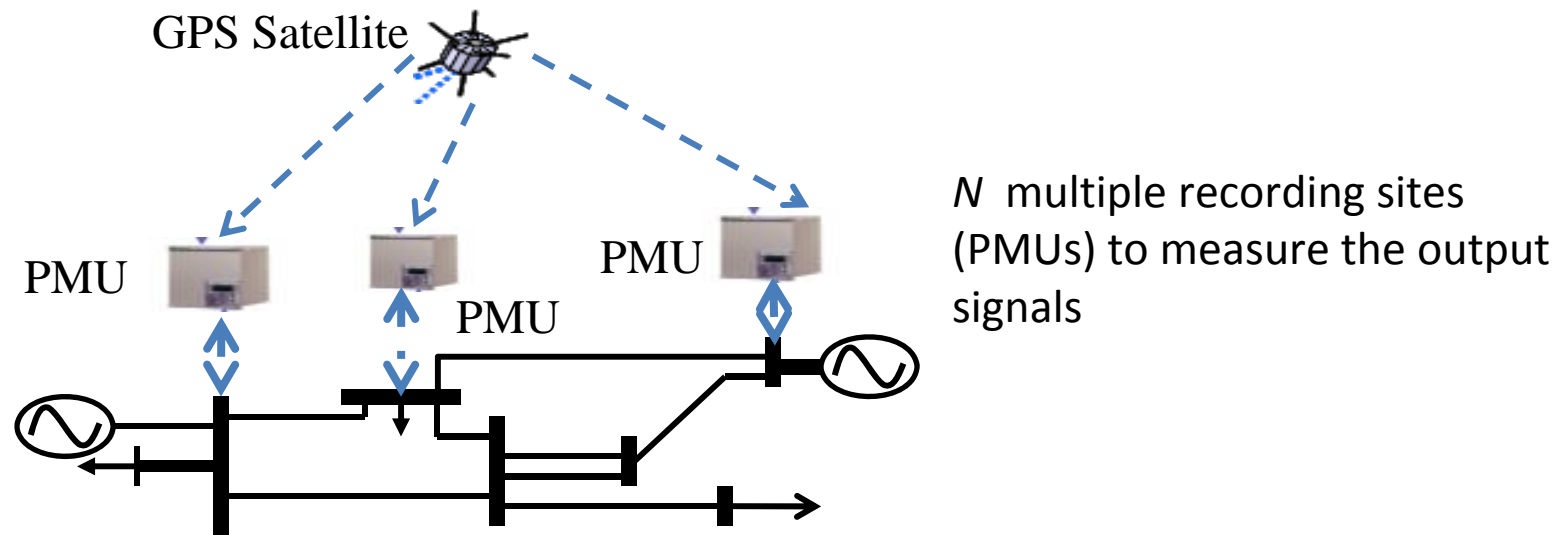
(c) Good Nodes

Trust System
Performance

Power Grid Cyber-security

- Inter-area oscillations (modes)
 - Associated with large inter-connected power networks between clusters of generators
 - Critical in system **stability**
 - Requiring **on-line** observation and control
- Automatic estimation of modes
 - Using currents, voltages and angle differences measured by PMUs (Power Management Units) that are distributed throughout the power system

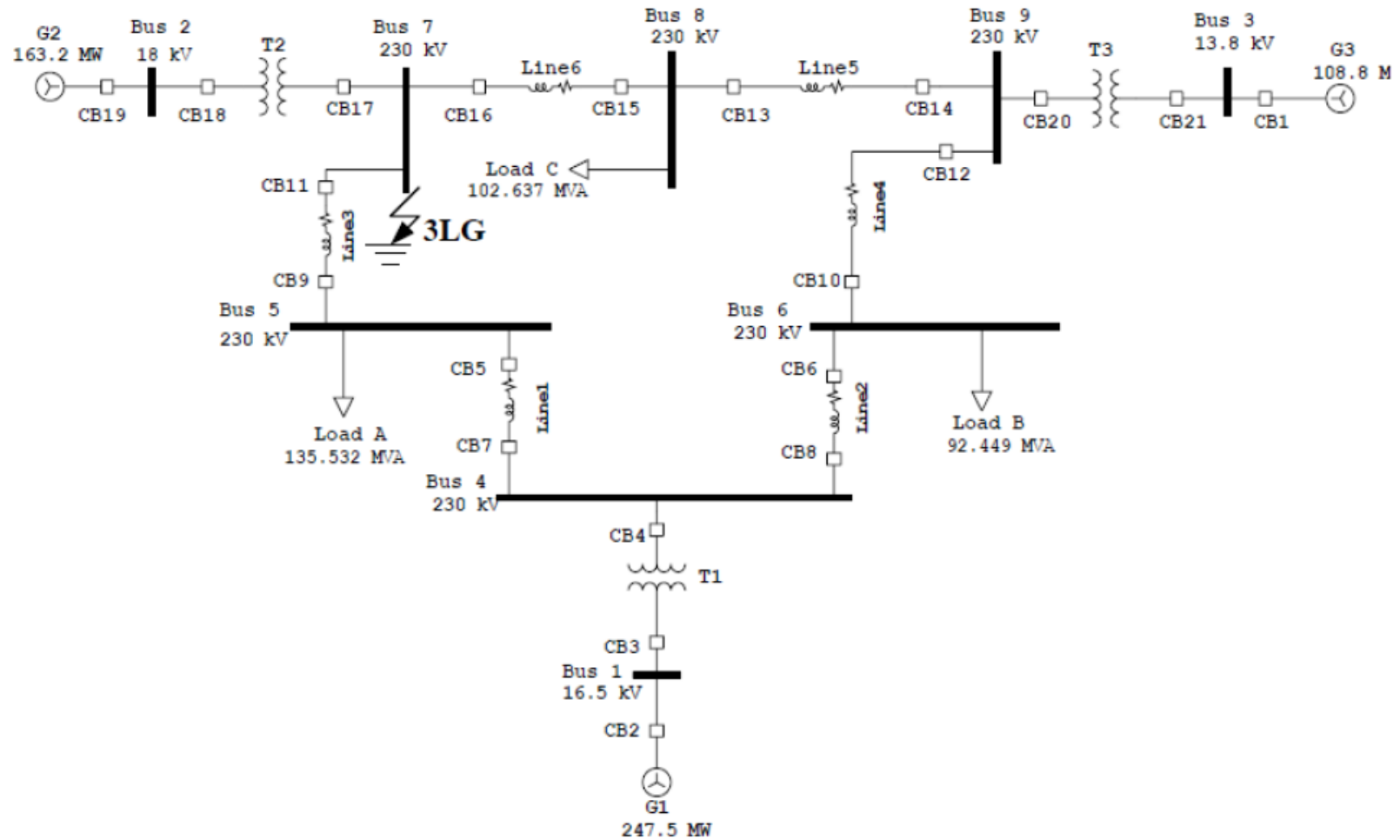
Distributed Estimation



- To compute an accurate estimate of the state $x(k)$, using:
 - **local measurements** $y_j(k)$;
 - information received from the PMUs in its **communication neighborhood**;
 - confidence in the information received from other PMUs provided by the **trust model**

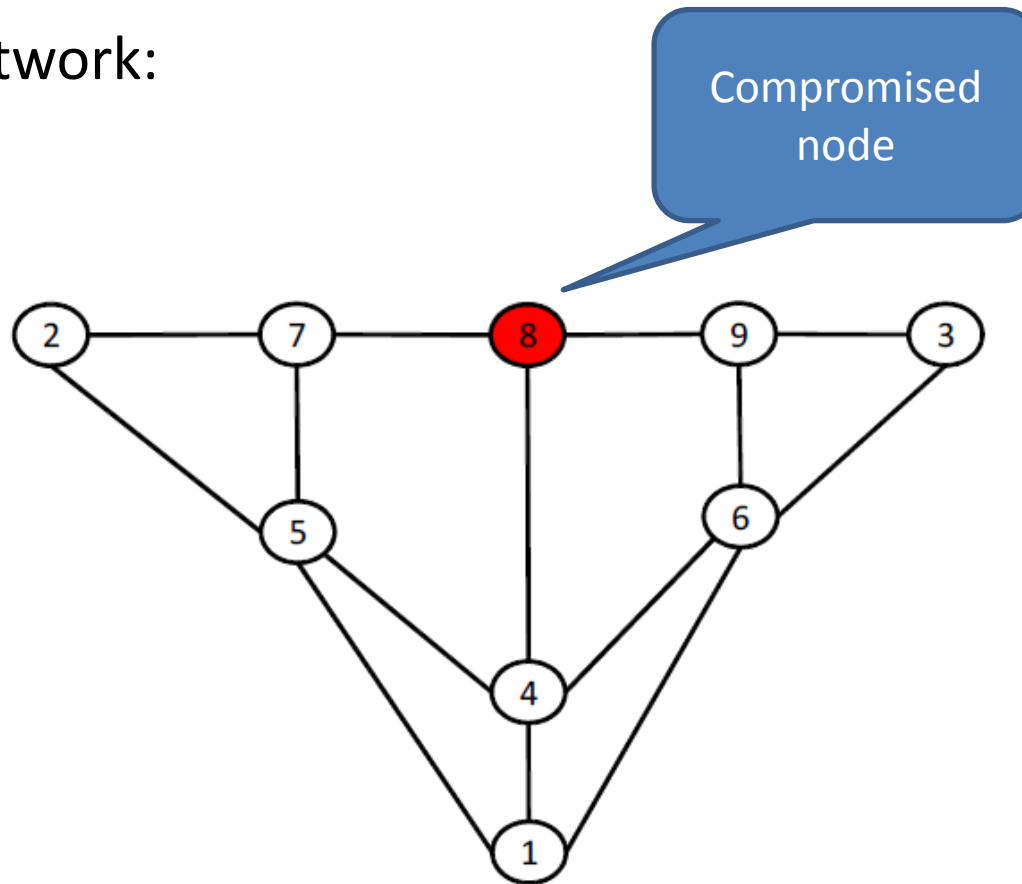
Numerical Example

- 3-generators, 9-bus system:

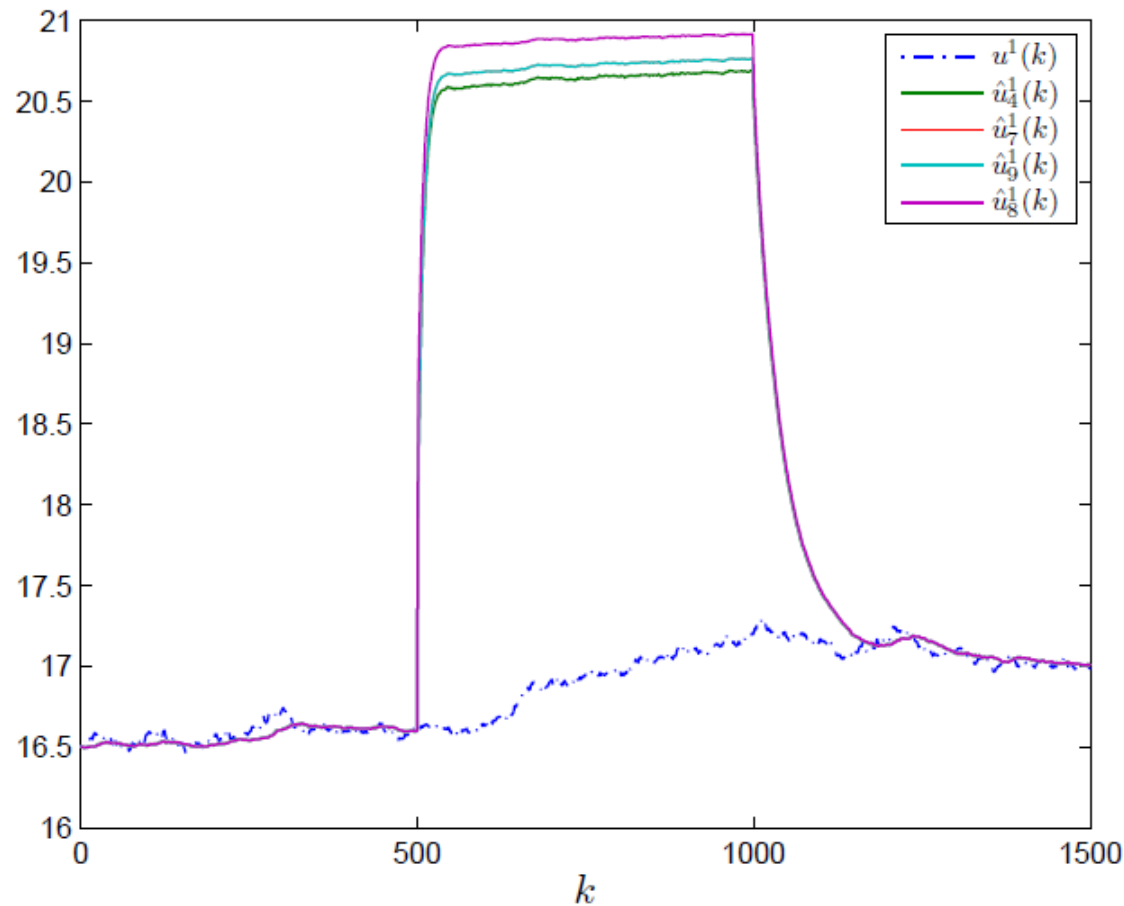


Numerical Example (cont.)

- PMU network:

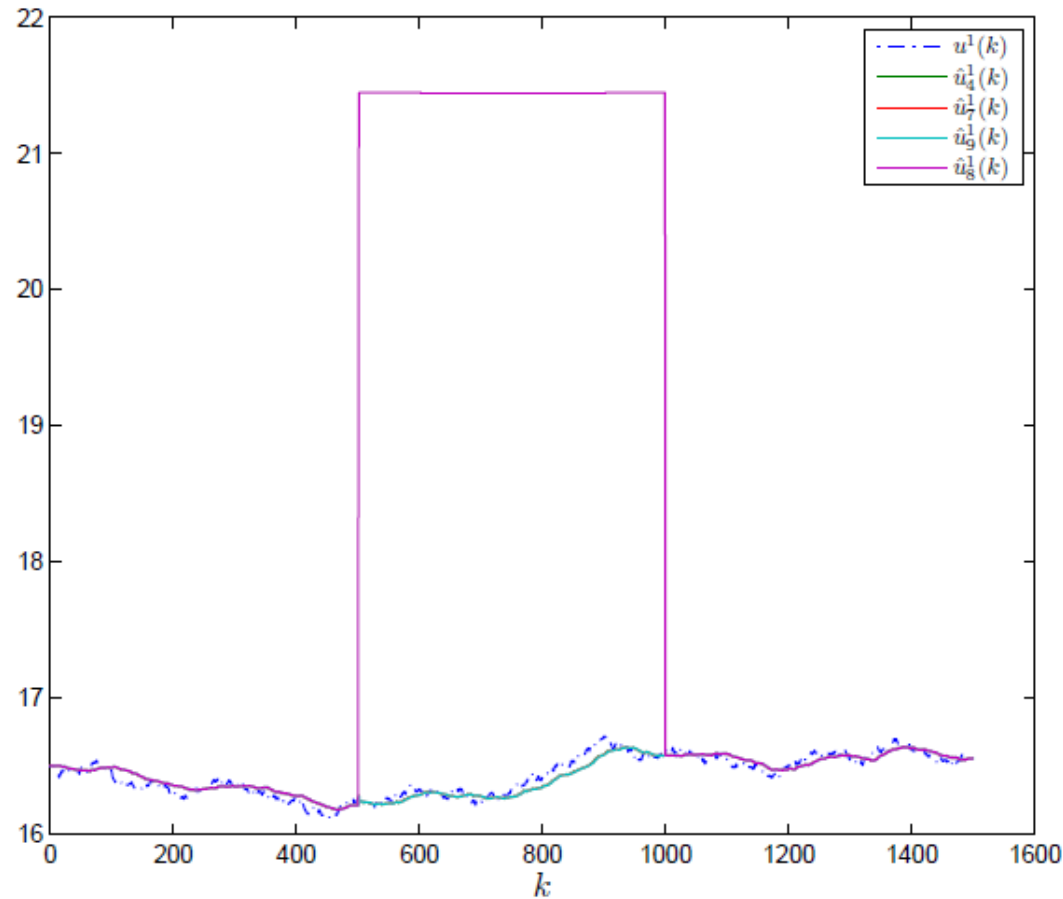


Numerical Example (cont.)



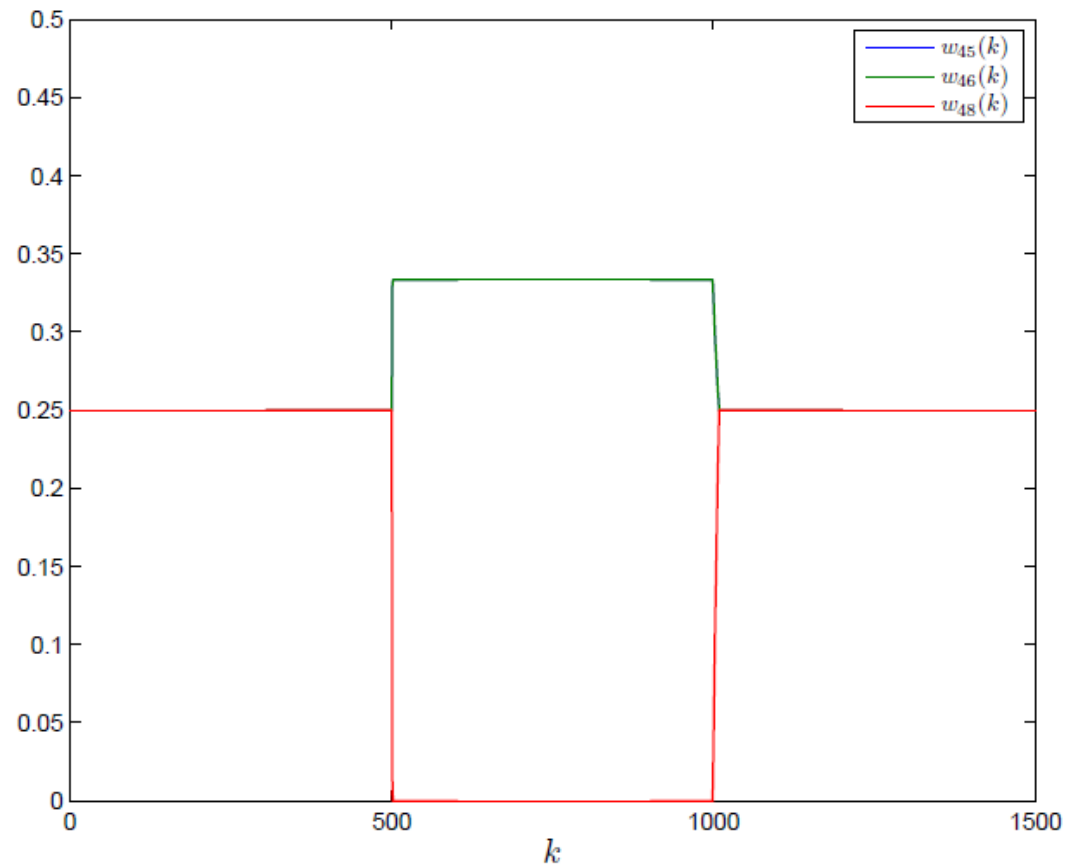
- Estimates of the voltage at bus 1 using Algorithm 1, with agent 8 injecting false data

Numerical Example (cont.)



- Estimates of the voltage at bus 1 using Algorithm 3, with agent 8 injecting false data

Numerical Example (cont.)

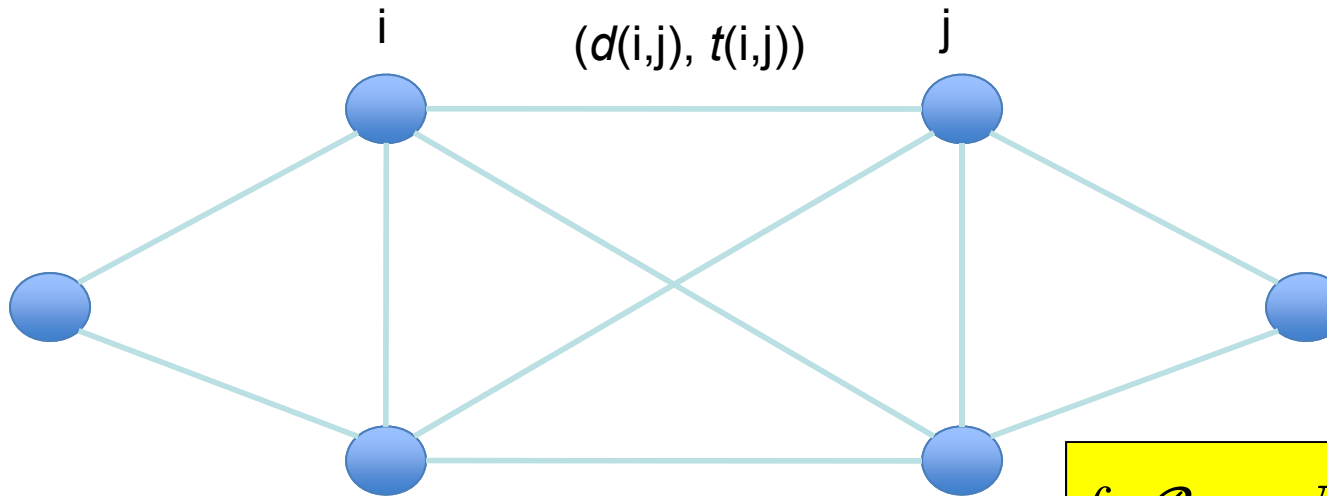


- The evolution of agent 4's weights

MANET Trust Aware Routing - -Trust/Reputation Systems

- Components
 - Monitoring and Detection Modules
 - Reputation Control System
 - Response Modules
- Our approach is different: build and use a Trusted Sentinel Sub-Network (SSN), which is responsible for monitoring and flooding reputation measures
- Logically decouple the Trust/Reputation System from other network functionalities
- Demonstrate that logical constraints on the SSN translate to constraints on the communication graph topology of the network
- Trade-off analysis between security and performance

Path Problems on Graphs – Delay and Trust Semirings



$$\min_{p \in \mathcal{P}_{SD}} d(p) = \min_{p \in \mathcal{P}_{SD}} \sum_{(i,j) \in p} d(i,j)$$

$$\max_{p \in \mathcal{P}_{SD}} t(p) = \max_{p \in \mathcal{P}_{SD}} \left(\min_{(i,j) \in p} t(i,j) \right) = -\min_{p \in \mathcal{P}_{SD}} \left(\max_{(i,j) \in p} (-t(i,j)) \right)$$

$$f : \mathcal{P}_{SD} \rightarrow \mathbb{R}^2$$

$$f(p) = (d(p), -t(p)), \quad \forall p \in \mathcal{P}_{SD}$$

Delay Semiring: $(\mathcal{R}_+ \cup \{0\}, \min, +)$

Trust Semiring: $(-\mathcal{R}_+ \cup \{0\}, \min, \max)$

Notions of Optimality: Pareto, Lexicographic, Max-Ordering, Approximation Semirings

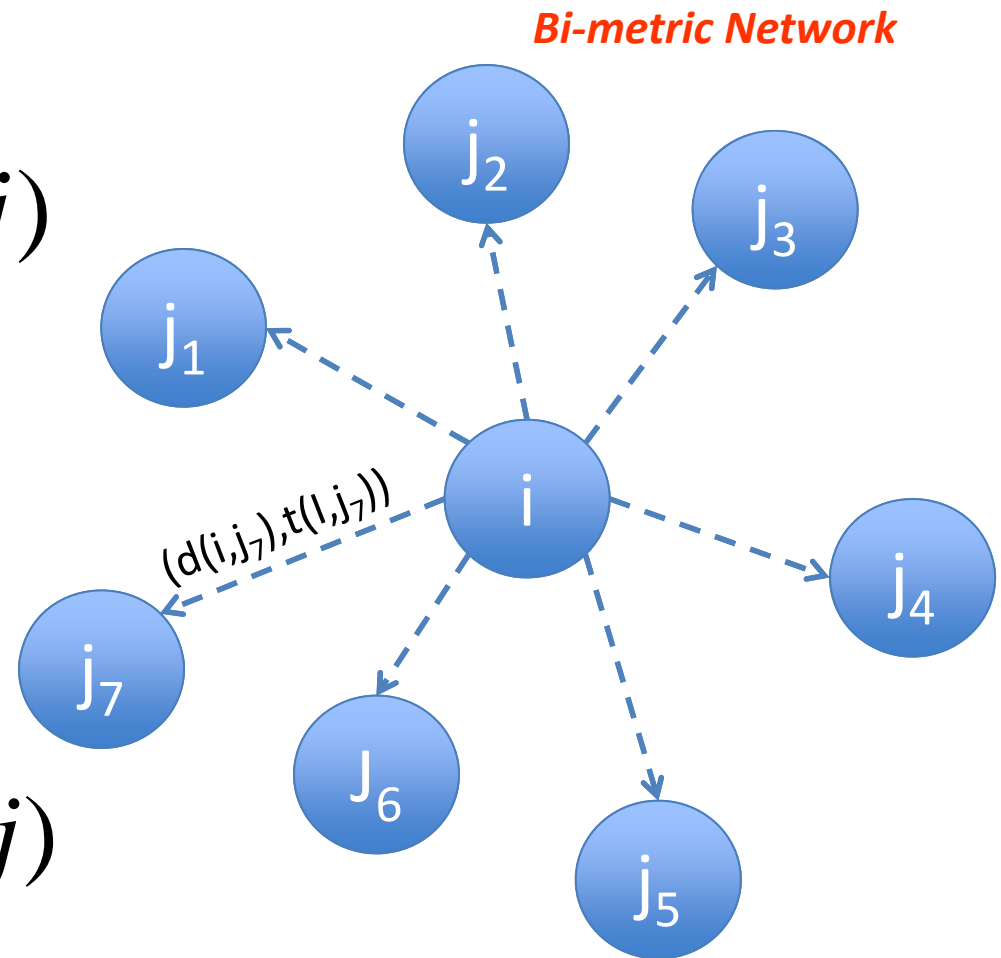
Trust Aware Routing – Multi-Criteria Optimization Problem

- **Delay** of a path “p”

$$d(p) = \sum_{(i,j) \in p} d(i,j)$$

- **Trust** of a path “p” –
bottleneck trust

$$t(p) = \min_{(i,j) \in p} t(i,j)$$

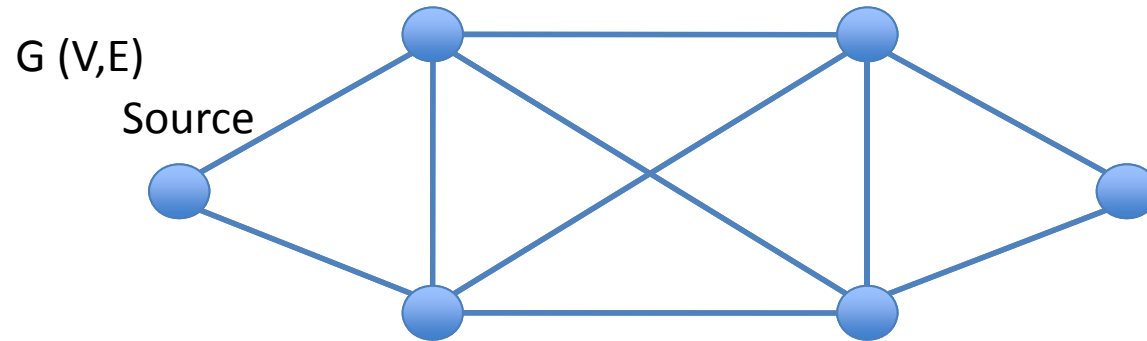


- How to build routing tables based on these metrics?
- The two metrics are not trivially comparable

$$MCOP : (\mathcal{P}_{SD}, f, X) / \theta / (\mathcal{R}^Q, \preceq)$$

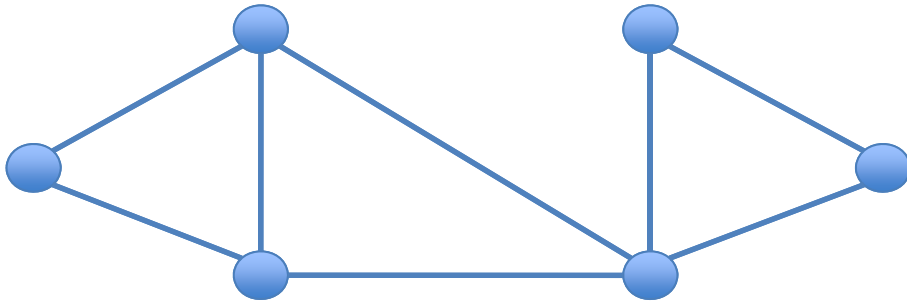
Notation	Definition	Name
$x \leq y$	$x_i \leq y_i \quad i = 1, 2, \dots, Q$	Weak component-wise order
$x < y$	$x_i \leq y_i \quad i = 1, 2, \dots, Q \text{ and } x \neq y$	Component-wise order
$x \ll y$	$x_i < y_i \quad i = 1, 2, \dots, Q$	Strict component-wise order
$x \leq_{lex} y$	$x_k < y_k \text{ or } x = y \quad k = \min\{i : x_i \neq y_i\}$	Lexicographic component-wise order
$x \leq_{MO} y$	$\max_i x_i \leq \max_i y_i$	Max order

Haimes Method – Two Stage Recipe



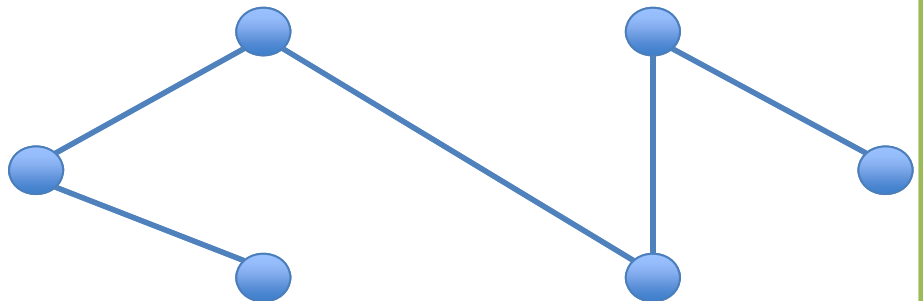
1. G' – reduced graph

$O(|E|)$



2. G'^{SP} – SP on reduced graph

$O(|V| \cdot |E|)$



Wireless Multihop Networks



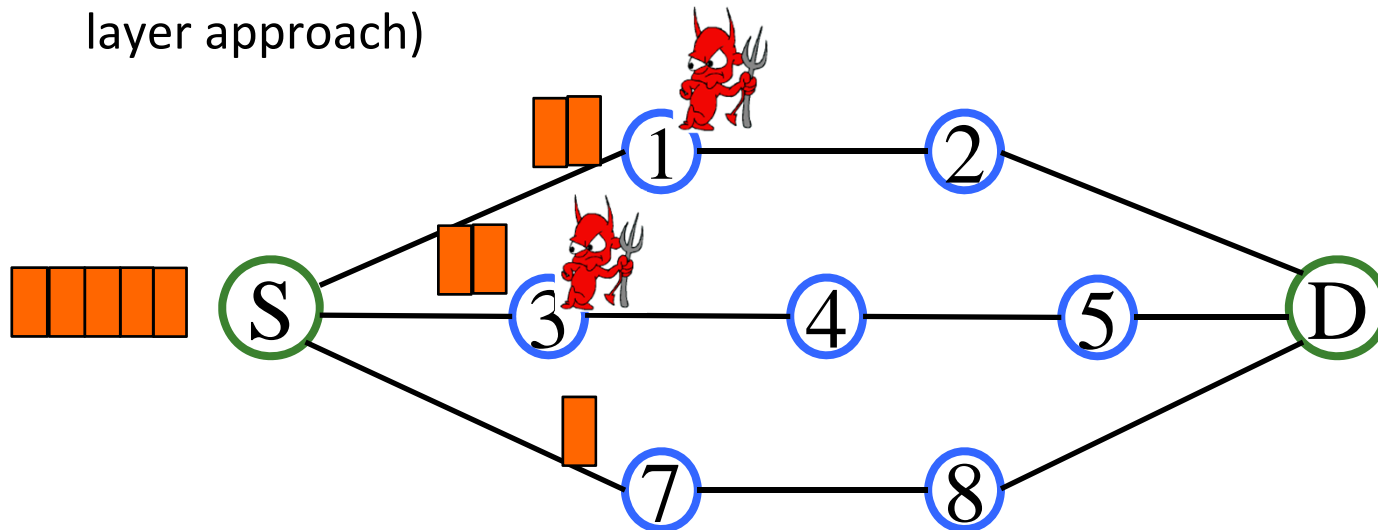
- Properties of the networks
 - Small devices: low power, low computational power
 - Distributed architecture: distributed resource allocation algorithms
- Attacker landscape
 - Adversaries more powerful and motivated
 - Devices less capable of defense: distributed, computing limits
 - Networks are more open: communication medium

Defend against attacks and maintain performance
requirements

Network Utility Maximization (NUM)



- Network Utility Maximization (NUM)
 - Resource allocation based on constrained maximization of a utility function
 - Joint source rate control and scheduling via dual decomposition (cross-layer approach)

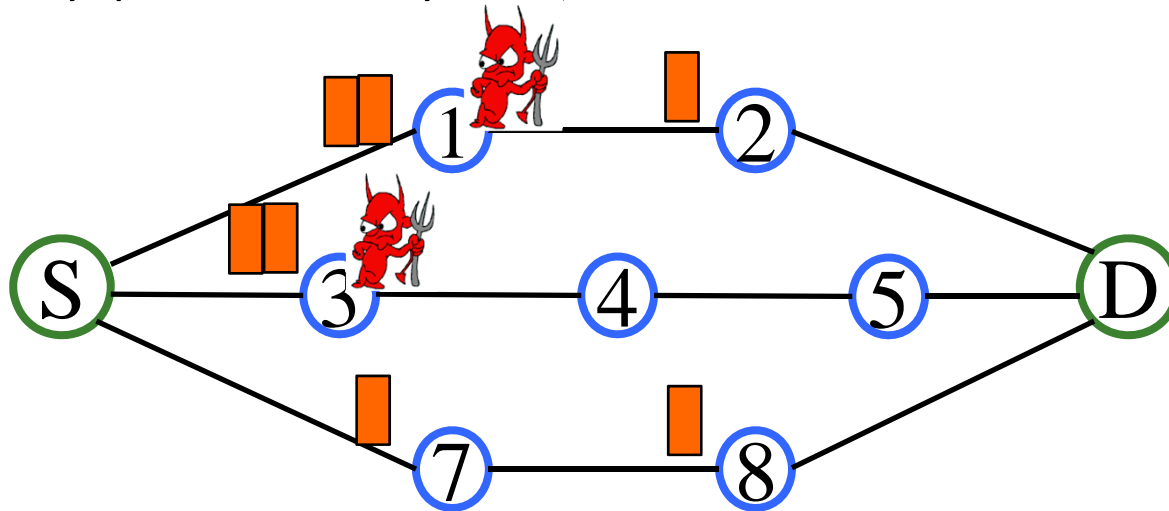


How to incorporate the notion of security in the NUM formulation, but at the same time maintain performance guarantees?

Adversary Model



- Adversary is capable of deploying Denial-of-Service (DoS) attack
 - ❑ Selective forwarding attack with some probability
 - ❑ Lack of **availability** in the network
 - ❑ Affects significantly Quality-of-Service (QoS) requirements (e.g., end-to-end delay, packet delivery ratio)



Trust Estimates



- Based on distributed trust evidence and proper monitoring mechanisms
- Notation: the estimated trust value of node i is denoted as v_i and takes values $[0,1]$
- Updated over time based on the attacker's observed strategy
- Define an update period T_{update} , where the trust evaluation mechanism provides fresh trust estimates
- To prevent significant variation on trust estimates, we use exponential weighted moving average (EWMA)

$$\nu_i(t) = (1 - \alpha)\nu_i(t - T_{\text{update}}) + \alpha\nu_i^{\text{new}}$$

System Model and Notation



- Wireless Multihop Network as graph $G(\mathcal{N}, \mathcal{L})$
- Multiple source-destination pairs (traffic flows)
- Set of alternative paths from a source node s to destination node d_s (one traffic flow) denoted by $\mathcal{P}_s = \{p_{s1}, \dots, p_{sP_s}\}$
- Traffic rate vector for each source-destination pair denoted by \mathbf{x}_s
- Maximum data rate for each traffic flow denoted by \mathcal{R}_s .
- Rate constraints

$$\begin{aligned}\mathbf{x}_s &\geq \mathbf{0} \quad , \forall s \in \mathcal{S} \\ \mathbf{1}^T \mathbf{x}_s &\leq \mathcal{R}_s \quad , \forall s \in \mathcal{S}\end{aligned}$$

- Routing matrix for each traffic flow denoted by $\mathbf{R}_s = [(R_s)_{k(i,j)}]_{(P_s \times |\mathcal{L}|)}$

$$(R_s)_{k(i,j)} = \begin{cases} 1, & \text{if } p_{sk} \text{ passes through link } (i,j). \\ 0, & \text{otherwise.} \end{cases}$$

Aggregate Trust Value



- **Aggregate trust value** of a path is the product of the corresponding trust values of the intermediate node of the path

$$t_{sk} = \prod_{j:(i,j) \in p_{sk}} \nu_j$$

- **Aggregate trust incidence matrix** \mathcal{T}_s denotes the aggregate trust values of every possible sub-path $p_{sk}^{(i,j)}$

$$t(p_{sk}, (i, j)) = \begin{cases} \prod_{j':(i',j') \in p_{sk}^{(i,j)}} \nu_{j'} & , \text{if } (i, j) \in p_{sk} \\ 0 & , \text{otherwise} \end{cases}$$

Optimization Constraints



- Link capacity constraint
 - Link capacity margin, denoted by $\sigma_{i,j}$ controls link delay
 - Capacity allocated to a wireless link is denoted by $\hat{c}_{i,j}$

$$\sum_{s \in \mathcal{S}} \sum_{k: (i,j) \in p_{sk}} t_{sk}^{(i,j)} x_{sk} \leq \hat{c}_{i,j} - \sigma_{i,j}, \quad \forall (i,j) \in \mathcal{L}$$

- Average end-to-end delay constraint
 - Indicates the imposed end-to-end delay (QoS) requirements imposed to each traffic flow
 - Link delay is denoted by $\phi(\sigma_{i,j}) = \phi_{i,j} = 1/\sigma_{i,j}$.

$$\mathbf{R}_s \phi(\sigma) \leq \mathbf{1D}_s, \quad \forall s \in \mathcal{S}$$

- Scheduling constraint: $\hat{c} \in \Lambda$.
- Reliability constraint: achieve minimum allowable traffic rate

$$\sum_{p_{sk} \in \mathcal{P}_s} t_{sk} x_{sk} \geq \mathcal{R}_s^{thres}, \quad \forall s \in \mathcal{S}$$

Trust-Aware Network Utility Maximization Formulation



- Allocate more traffic rate to trustworthy paths to avoid malicious nodes
- Maintain end-to-end delay guarantees (QoS requirements)
- **Concave utility function** $\mathcal{U}_s(\mathbf{x}_s)$ of each source node

$$\mathcal{U}_s(\mathbf{x}_s) = \sum_{p_{sk} \in \mathcal{P}_s} \left(t_{sk} \log(x_{sk}) \right)$$

- Primal trust-aware NUM optimization problem

$$\begin{aligned} \max_{\mathbf{x}, \sigma, \hat{\mathbf{c}}} \quad & \sum_{s \in \mathcal{S}} \mathcal{U}_s(\mathbf{x}_s) \\ \text{s. t.} \quad & \sum_{s \in \mathcal{S}} \sum_{k: (i,j) \in p_{sk}} t_{sk}^{(i,j)} x_{sk} \leq \hat{c}_{i,j} - \sigma_{i,j}, \forall (i,j) \\ & \mathbf{R}_s \phi(\sigma) \leq \mathbf{1} \mathcal{D}_s, \quad \forall s \in \mathcal{S} \\ & 0 \leq \mathbf{1}^T \mathbf{x}_s \leq \mathcal{R}_s, \quad \forall s \in \mathcal{S} \\ & \sum_{p_{sk} \in \mathcal{P}_s} t_{sk} x_{sk} \geq \mathcal{R}_s^{thres}, \quad \forall s \in \mathcal{S} \\ & \hat{\mathbf{c}} \in \Lambda \end{aligned}$$

Dual Objective Function



- Dual variables (Lagrange multipliers)
 - Vector of *link prices* denoted by λ
 - Vector of dual variables associated with the average end-to-end delay constraints imposed to each traffic flow denoted by μ_s
- Dual Optimization Problem

$$\begin{aligned} h(\lambda, \mu) &= \sup_{\mathbf{x} \in \mathcal{X}} \left\{ \sum_{s \in \mathcal{S}} \left(u_s(\mathbf{x}_s) - (\lambda^s)^T \mathcal{T}_s^T \mathbf{x}_s \right) \right\} \\ &\quad + \sup_{\sigma \geq 0} \left\{ - \sum_{(i,j) \in \mathcal{L}} \left(\phi_{i,j} \mu^{(i,j)} + \lambda_{i,j} \sigma_{i,j} \right) \right\} \\ &\quad + \sup_{\hat{\mathbf{c}} \in \Lambda} \left\{ \lambda^T \hat{\mathbf{c}} \right\} \\ &\quad + \sum_{s \in \mathcal{S}} \mu_s^T \mathbf{1} \mathcal{D}_s \end{aligned}$$

- **Three maximization sub-problems:** source rate control, average end-to-end delay control and scheduling

Dual Decomposition



- Source Rate Control (solve numerically)

$$\nabla \mathcal{U}_s(x_s^*) = \mathcal{T}_s \lambda^s, \quad \forall s \in \mathcal{S}$$

- Average end-to-end delay control (solve numerically)

$$\frac{d\phi}{d\sigma}(\sigma_{\mathbf{i},\mathbf{j}}^*) \mu^{(i,j)} = -\lambda_{i,j}, \quad \forall (i,j) \in \mathcal{L}$$

- Scheduling policy

$$\hat{c}_{i,j}^* = \operatorname{argmax}_{\hat{c}_{i,j} \in \Lambda} \sum_{(i,j) \in \mathcal{L}} \lambda_{i,j} \hat{c}_{i,j}$$

Distributed Cross-Layer Optimization Algorithm



- Subgradient descent method to update the dual variables

$$\lambda_{i,j}^{(n+1)} = \left\{ \lambda_{i,j}^{(n)} - \gamma \left(\hat{c}_{i,j}^{(n)} - \sum_{s \in \mathcal{S}} \sum_{k: (i,j) \in p_{sk}} \left(t_{sk}^{(i,j)} x_{sk}^{(n)} \right) - \sigma_{i,j}^{(n)} \right) \right\}^+$$

$$\mu_{sk}^{(n+1)} = \left\{ \mu_{sk}^{(n)} - \gamma \left(\mathcal{D}_s - \sum_{(i,j) \in \mathcal{L}} \phi(\sigma_{i,j}^{(n)}) (R_s)_{k(i,j)} \right) \right\}^+$$

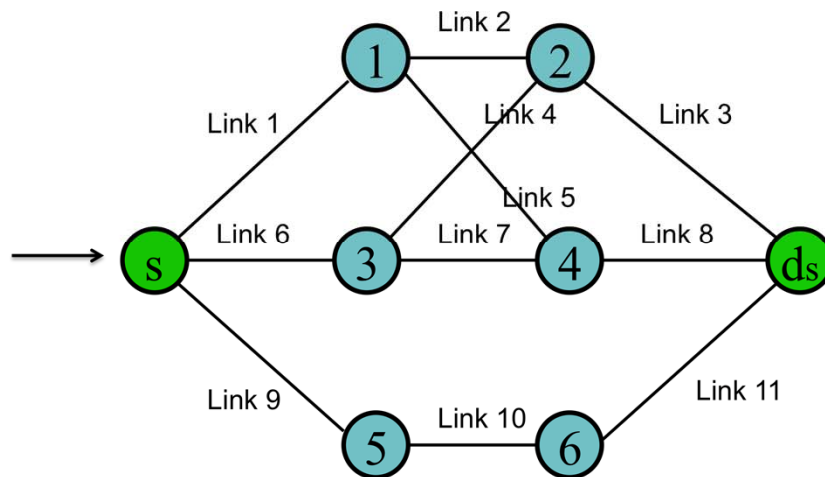
Algorithm 1 Distributed Cross-Layer Optimization

- 1: **INITIALIZE** primal and dual variables
 - 2: **while** $\mathbf{1}^T |\mathbf{x}_s^{(n)} - \mathbf{x}_s^{(n-1)}| \leq \epsilon$ **do**
 - 3: *Dual Variables Update*
 - 4: Each link (i, j) updates its dual variable $\lambda_{i,j}$
 - 5: Each source s updates the dual variables μ_{sk}
 - 6: *Sources exchange dual variables*
 - 7: Each source s evaluates $\lambda^{s,(n)}$.
 - 8: Each source s computes its traffic rate vector $\mathbf{x}_s^{(n)}$
 - 9: Each link (i, j) evaluates $\mu^{(i,j),(n)}$.
 - 10: Each link (i, j) computes its $\sigma_{i,j}^{(n)}$
 - 11: Each node performs scheduling via a max-weight algorithm
 - 12: **end while**
-

Network Scenario



- Network scenario with 8 nodes and 11 wireless links
- 5 alternative paths



Trust Estimates in four update periods

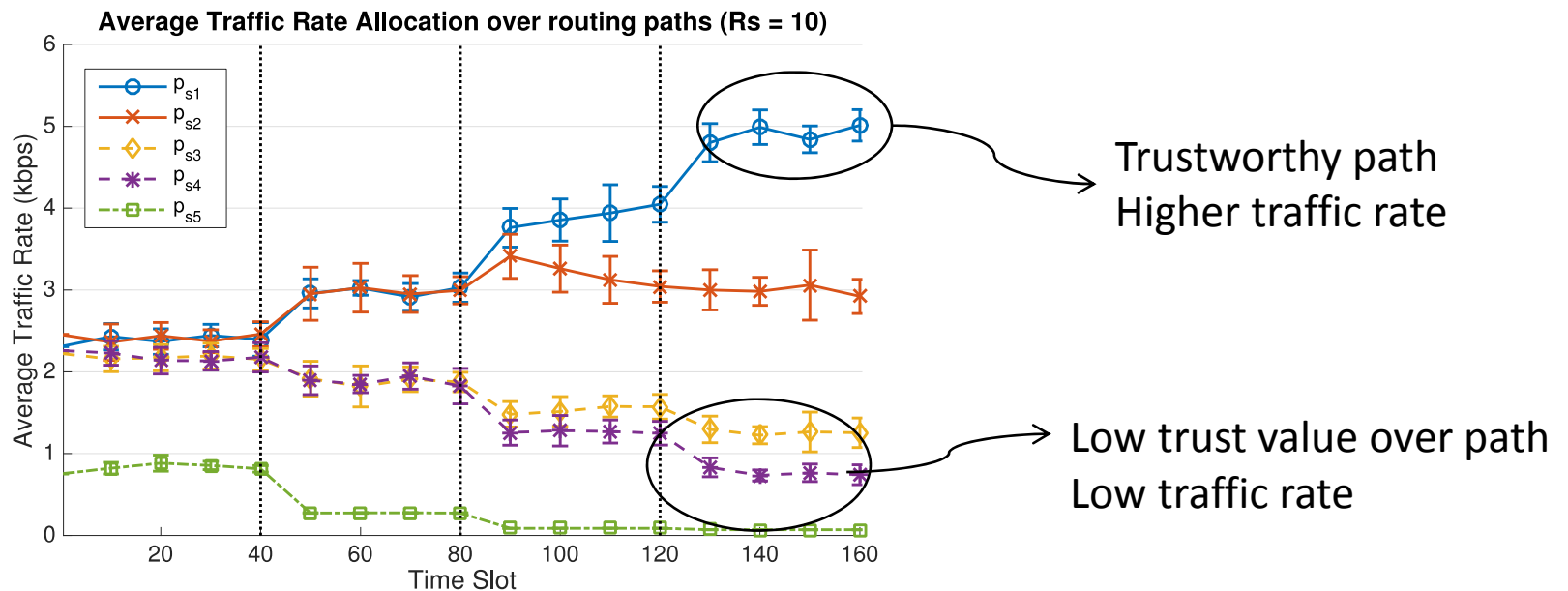
$$\nu = \begin{matrix} & s & 1 & 2 & 3 & 4 & 5 & 6 & d_s \\ \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} & 1 & 1 & 1 & 0.7 & 1 & 0.7 & 0.5 & 1 \\ & 1 & 0.9 & 0.9 & 0.5 & 0.9 & 0.2 & 0.2 & 1 \\ & 1 & 0.9 & 0.9 & 0.3 & 0.7 & 0.1 & 0.1 & 1 \\ & 1 & 0.9 & 0.9 & 0.2 & 0.5 & 0.1 & 0.1 & 1 \end{pmatrix}$$

Allocate more traffic Allocate less traffic

Performance Evaluation



- Average allocated traffic rate for $R_s = 10$ kbps

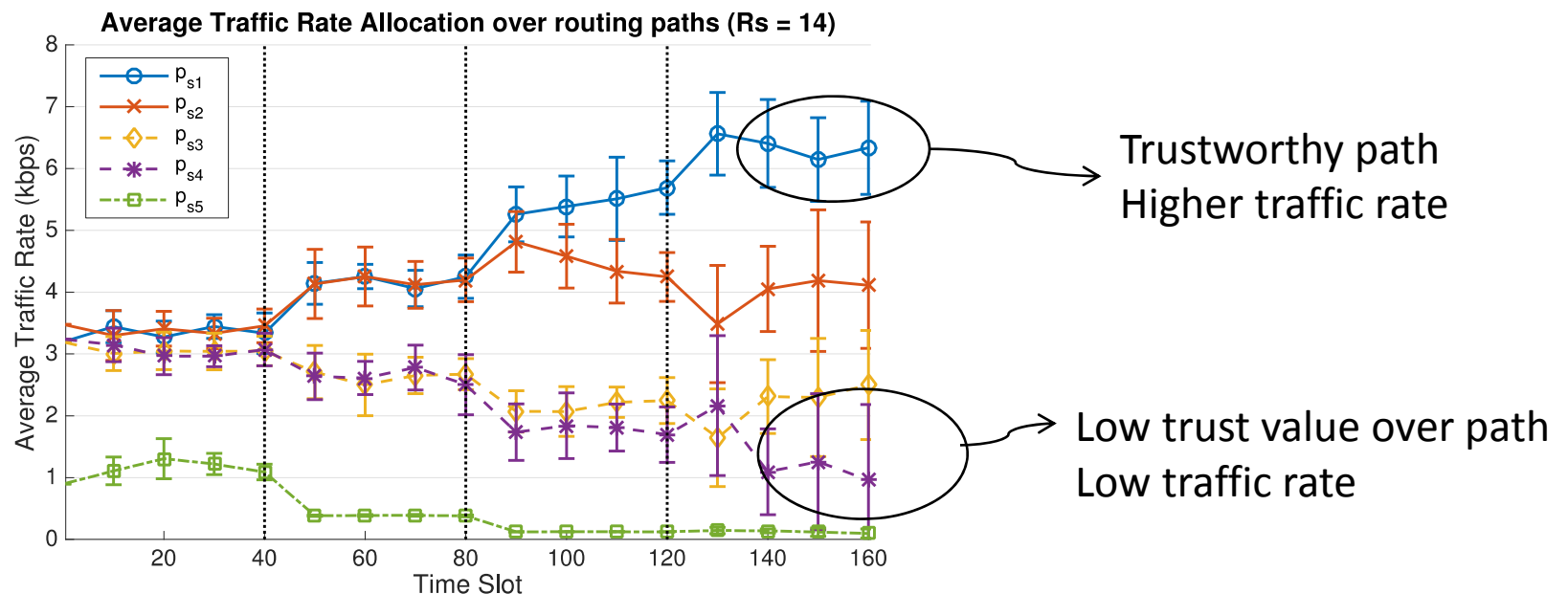


- More traffic allocated to trustworthy paths as trust evolves over time

Performance Evaluation



- Average allocated traffic rate for $R_s = 14$ kbps

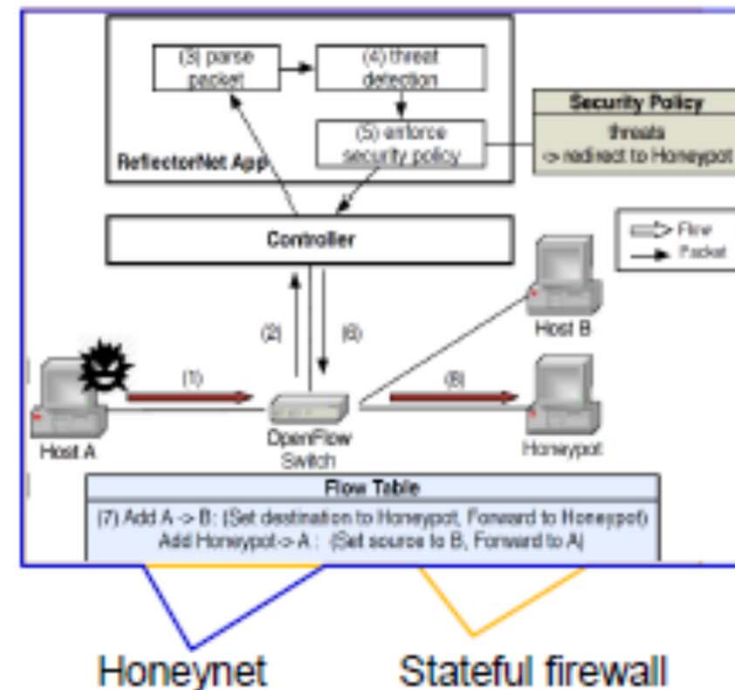
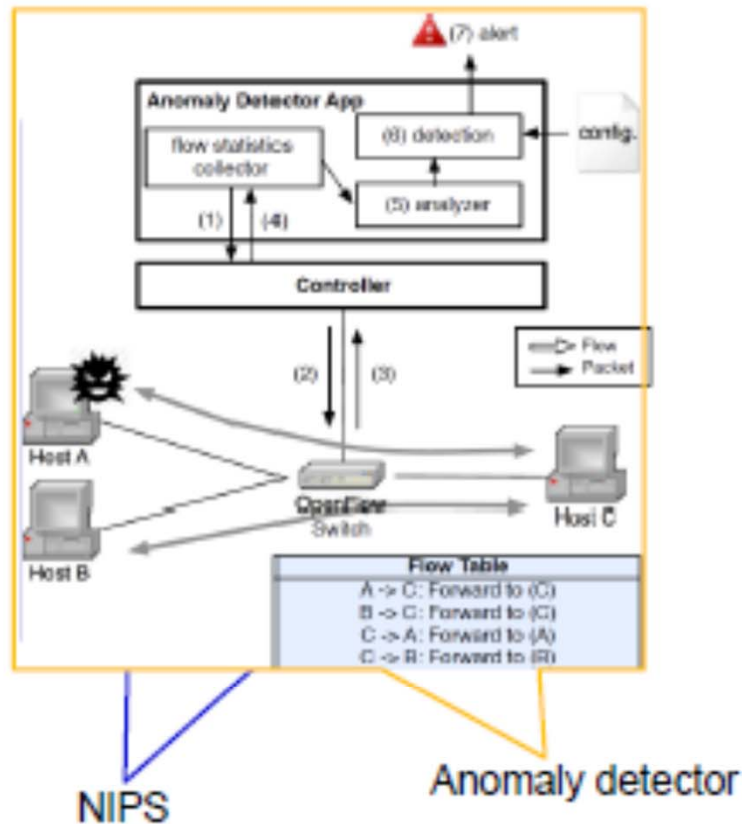


- More traffic allocated to trustworthy paths as trust evolves over time
- Notice that due to higher traffic demand, higher traffic is allocated to untrusted paths

Future: SD Security

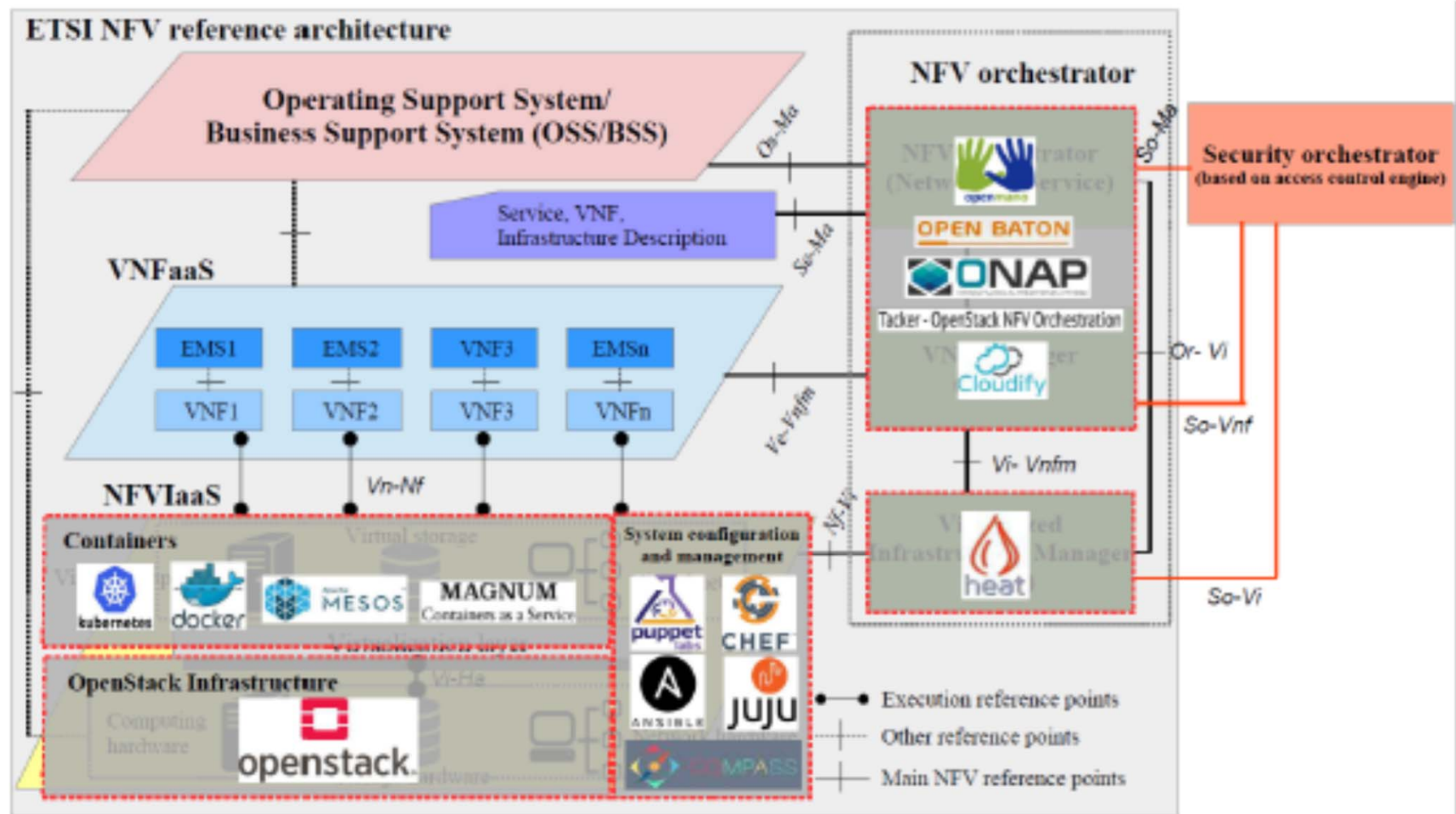
- Increasingly, we see Security as Code taking shape
- Security process automation needs to move at DevOps speed
- Less ownership, visibility, control
- Distributed “Workloads” – virtual, abstract, transient
- Large, flat, shared networks
- More and faster changes, with less control

SDN Based Security Functions

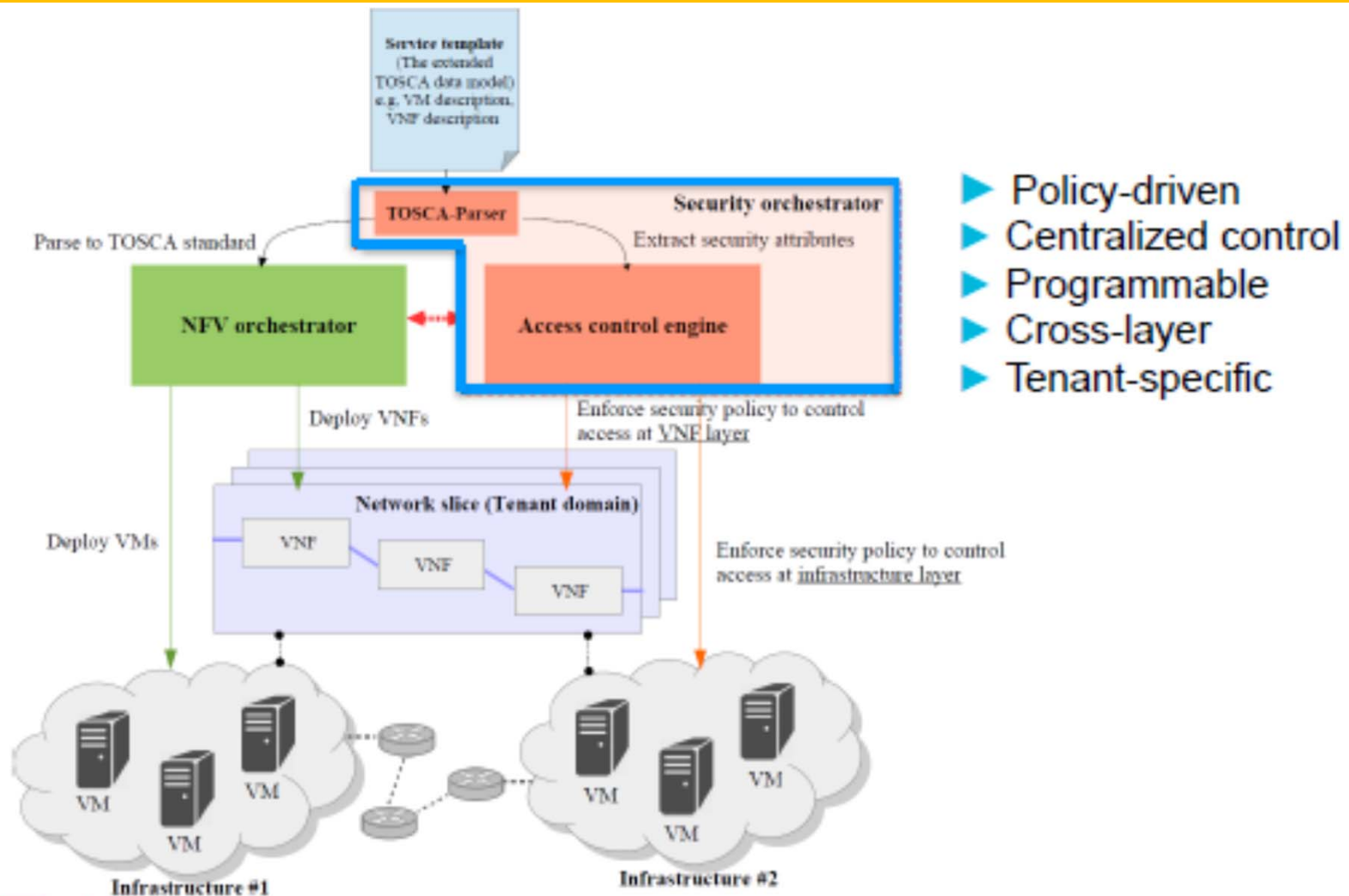


- ▶ Hardware matters (e.g., multi-port-forwarding, picket header modification)
- ▶ Performance bottleneck is due to control messages (packet-in)
- ▶ Data plane has a rich set of network status information (SDN as database)

NFV Based Security Management and Orchestration

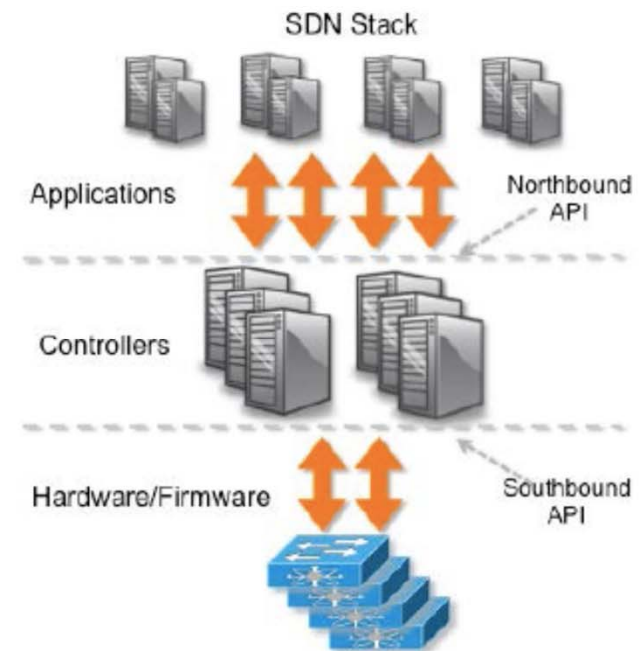


Security Orchestrator



SDN, Controller Security

- **Controllers are the “brains” of SDN**
 - Centralized
 - Programmable
 - Attackable
- **Focus on: patching and basic service security (HTTPS, SSH)**
- **Focus on: role-based access and authentication/authorization**



- **Control of and interaction with automation platforms can be very risky**
 - Poor development, scripting, resource design and instantiation
 - System availability issues or resource hijack/compromise
 - Malicious insiders or lack of “least privilege”
 - Vendor lock-in (architecture, language, etc.)
 - Poor authentication/credential management
 - Weak or non-existent integration with security products
- **Configuration management and access control are critical**

- Analysis via formal methods; composable security
- Testing and validation
- Taxonomy of trust-based vs trustless security in future wireless networks
- Integrated security, reliability and safety of Net-CPS
- Integrated security, reliability and safety of Net-HCPS
- Human behavior effects and mitigation
- Role of hardware in security and trust of future wireless networks
- Standardizing architectures and methods
- Formal modeling and performance analysis of networks as a service

Part II: Hardware Software Co- design for Automotive CPS using Architecture Analysis and Design Language

Challenges in Automotive CPS



- Cars are complex cyber physical systems.
 - Embedded chips buses, and physical components of the cars are the physical parts.
 - Algorithms within are the cyber parts.
- Embedded systems for engine control and active safety are complex and integrated CPS.
- Challenges arises from integrating different parts developed by multiple groups
 - In particular, control algorithms developed in abstract level and software implementation on hardware platform are performed separately.
 - Architecture design that bring together hardware design and software design is crucial to integrate the design.

Challenges in Automotive CPS



- A major challenge in architecture design is to balance multiple constraints and objectives including
 - Schedulability and timing
 - Performance
 - Cost
- Current practice for architecture design in automobile industry involves very abstract structure models that have limited analysis power
 - Detail and quantitative properties are either not captured or captured in other tool environment
 - Analysis becomes very tool dependent and limits the engineering process
 - It becomes cost inefficient to transfer information and maintain designs

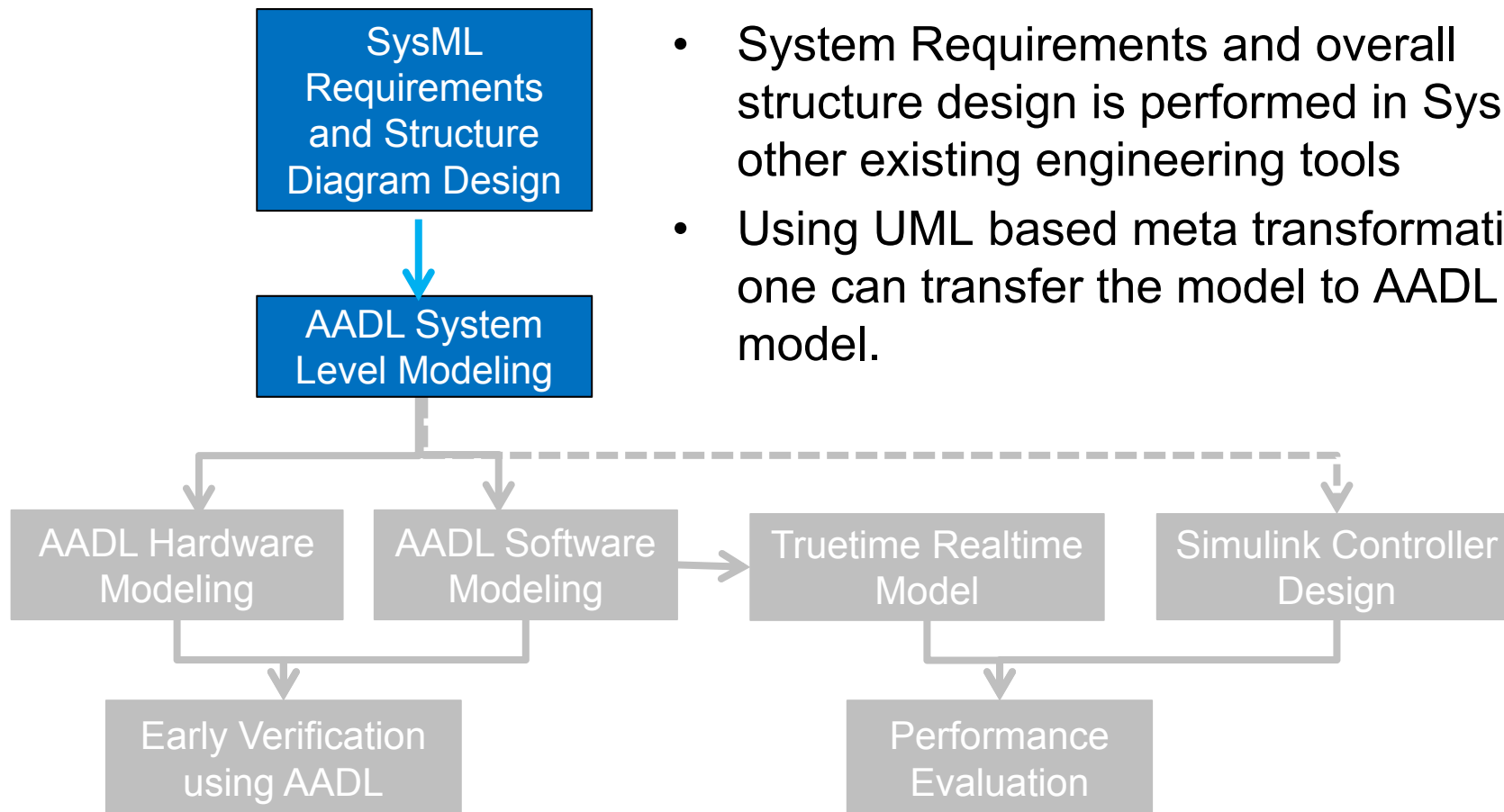
Architecture Design Language



- To address hardware software co-design, one need an architecture description language that captures both software and hardware
 - Architecture Analysis and Design Language (AADL) can capture this model in high level hardware model, software partition and low level implementation.
 - Enable analysis in different level of abstraction
 - As development of hardware and software proceed, system level property can be validated with more details.
- Our method involves the AADL toolset OSATE for architecture design and Cheddar for temporal analysis. Later on in the development process, performance analysis is evaluated using Truetime, CarSim and Simulink.

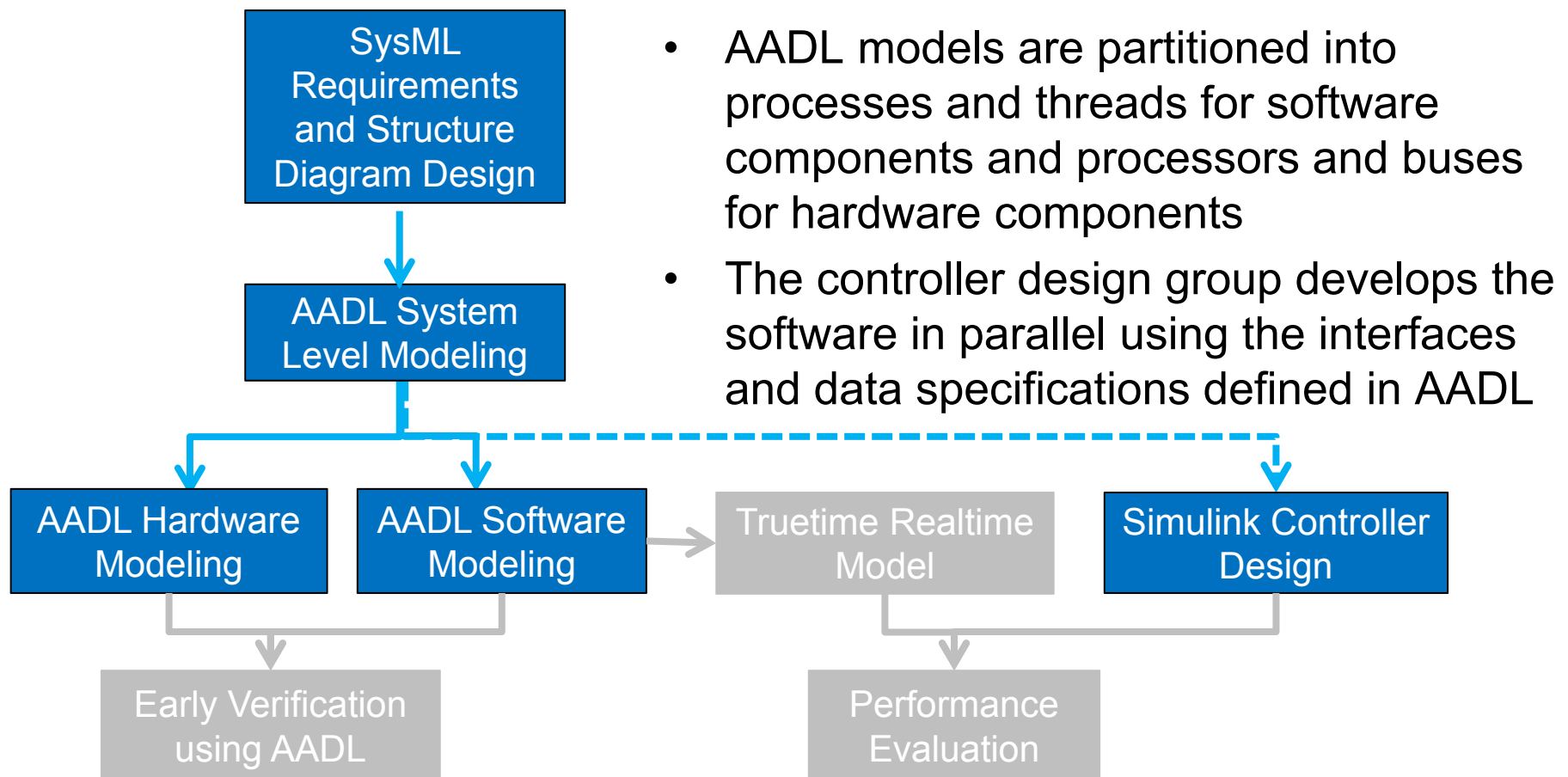
- Automobile CPS and hardware software co-design problem
- Architecture Design Language and Toolsets
- Our Approach
 - Systematic Design Flow
 - Hardware and Software Architecture Modeling
 - AADL and Cheddar Analysis
 - Simulink and Truetime Integration
 - Result

Design Flow

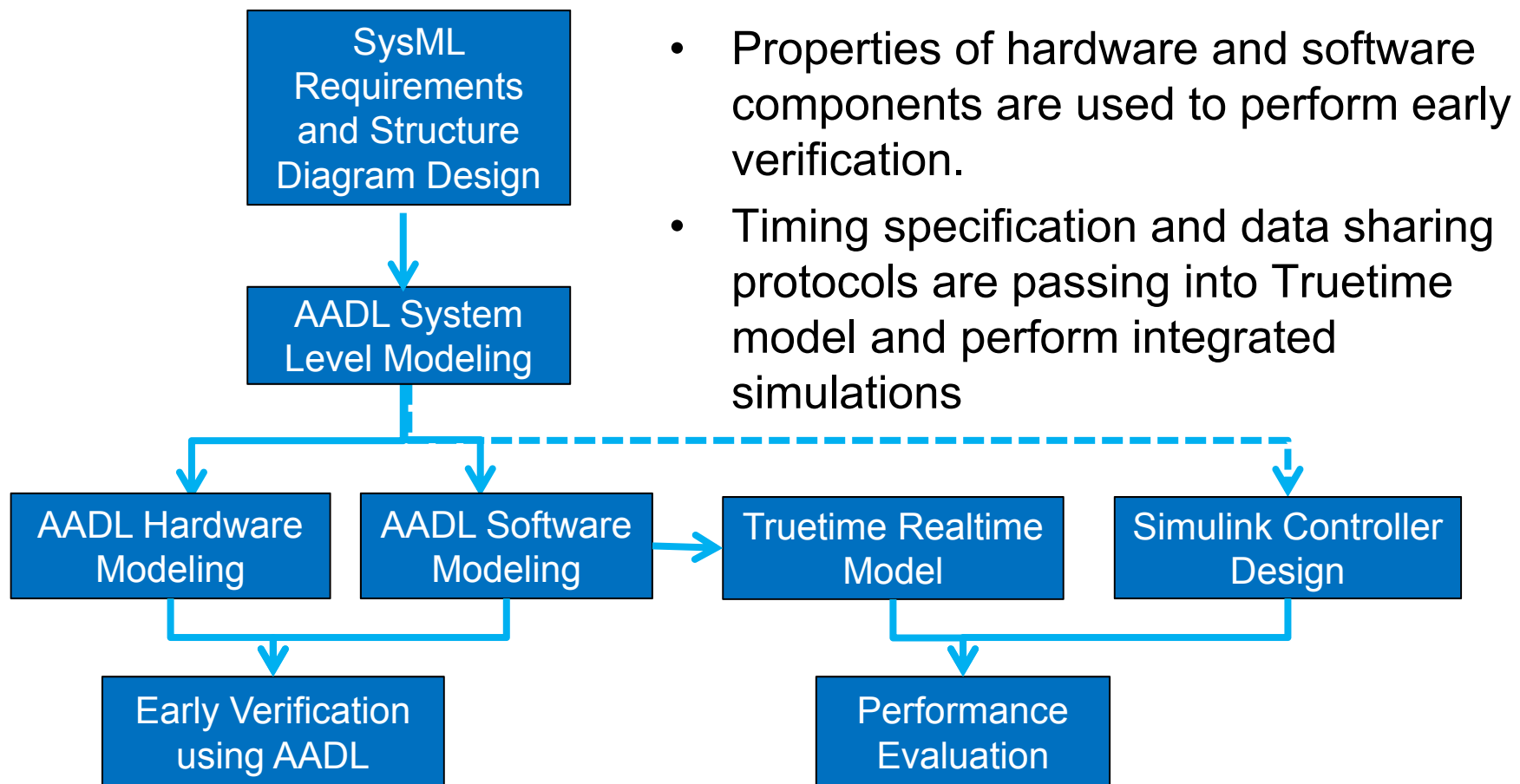


- System Requirements and overall structure design is performed in SysML or other existing engineering tools
- Using UML based meta transformation, one can transfer the model to AADL model.

Design Flow



Design Flow

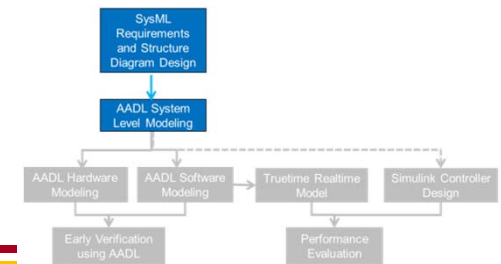


Case Study: Adaptive Cruise Control

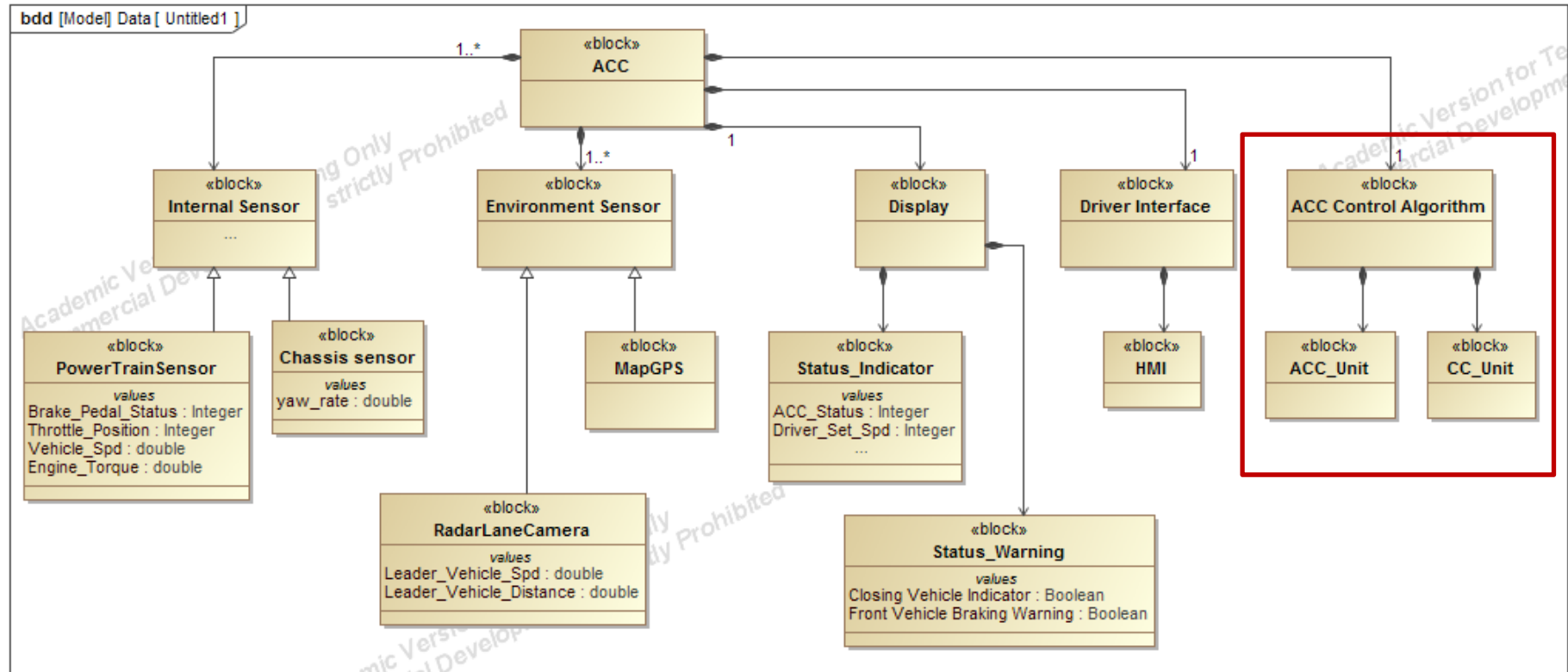


- The Adaptive Cruise Control (ACC) is an automobile algorithm that helps the vehicle to maintain distance to front vehicle in high way scenario.
 - When the radar and front camera sense the speed and distance of the front vehicle, the desired speed for the host vehicle is computed by the algorithm.
 - It also works as a cruise control unit, when there is no leading vehicle in the range.
- Hardware choices in the case study involves a dual core and a single core implementation.
- The main performance requirement is to maintain the distance between vehicles in a safe.
- We focus on timing analysis and real time performance in this study.

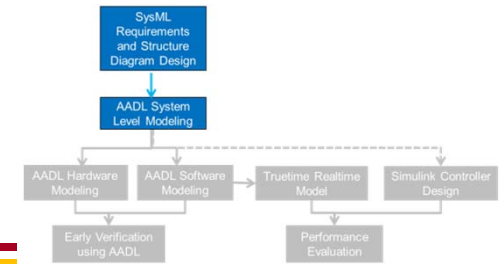
Systematic Design



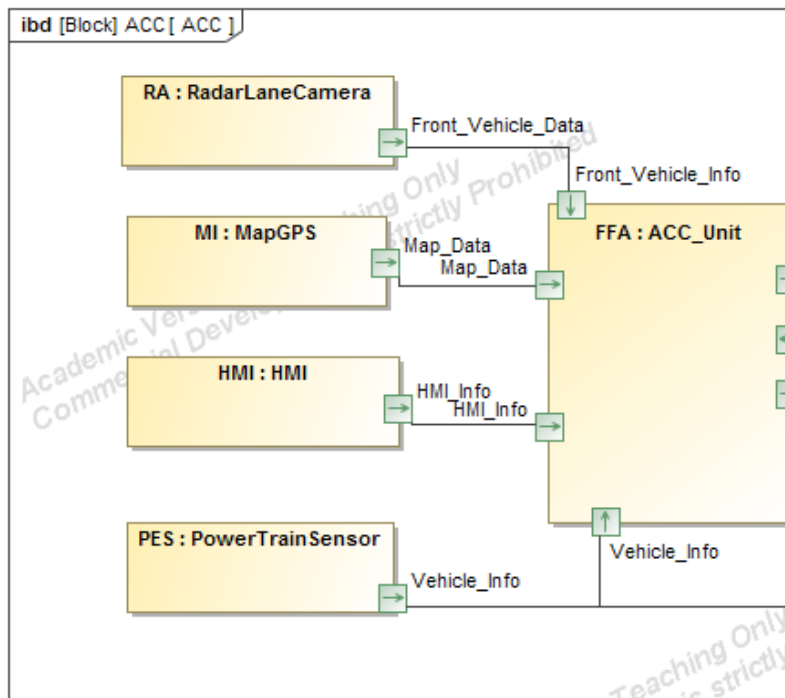
- Model based system design approach for an embedded controller unit for Adaptive Cruise Control (ACC)
 - This study emphasizes on controller real time performance



Model Transformation



- Model Transformation from Internal Block Diagram to AADL
 - Rockwell Collins

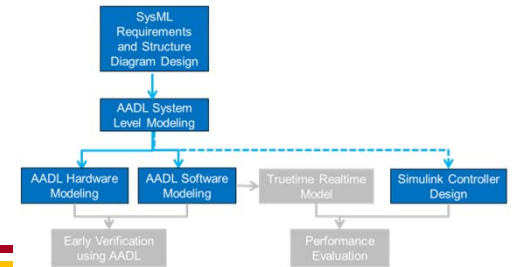


```

system ACC
features
  Map_info: feature group ACC_Map_socket;
  ACC_state: out data port;
end ACC;

system implementation ACC.impl
subcomponents
  FFA: system ACCUnit;
  PFA: system CCUnit;
  FLC: system LKI;
  RA: device RadarLaneCamera;
  MI: device MapGPS;
  PES: device PowerTrainSensor;
  HMI: device HMI;
connections
  C1: port FFA.ACC_state -> ACC_state;
  C2: port FFA.CC_Engage_Flag -> PFA.CC_Engage_Flag;
  C3: port PFA.CC_state -> FFA.CC_state;
  C4: feature group MI.Map_info -> Map_info;
  ...
  C20: feature group PES.VehicleInfo -> FLC.Vehicle_info;
end ACC.impl;
    
```

Hardware and Software Modeling



- Hardware Model
 - Dual core case

```

processor implementation dualCore.core1
properties
  Clock_period => 5 ms;
  SEI::MIPSCapacity => 30.0 MIPS;
  Scheduling_Protocol => RATE_MONOTONIC_PROTOCOL;
  Cheddar_Properties::Preemptive_Scheduler => true;
end dualCore.core1;

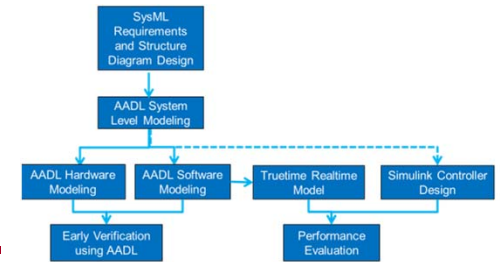
system implementation ACC_RD.impl
subcomponents
  ECU1: processor dualCore.core1;
  ECU2: processor dualCore.core2;
  FFRP: process ACC_RD_process.impl;
properties
  Actual_Processor_Binding => reference ecu1
applies to FFRP.Input;
...
  Actual_Processor_Binding => reference ecu2
applies to FFRP.Output;
end ACC_RD.impl;
  
```

- Software Model
 - Process are partitioned to threads and data sharing protocols

```

process implementation ACC_RD_process.impl
subcomponents
  Input: thread ACC_RD_Input.impl;
  SetResume: thread ACC_RD_HMISetResume.impl;
  ...
  Output: thread ACC_RD_Output.impl;
  ACCInput2PCS_I: data ACCInput2PCS.impl;
  ACCInput2ACCMode_I: data ACCInput2ACCMode.impl;
  ...
  SpdCtrl2Output_I: data SpdCtrl2Output.impl;
connections
  data access ACCInput2PCS_I -> ACCInput.ToPCS;
  data access ACCInput2PCS_I -> PCS.FromACCInput;
  ...
  data access SpdCtrl2Output_I ->
Output.FromSpdCtrl;
end ACC_RD_process.impl;
  
```

Schedulability Analysis



- Scheduling

- Cheddar tool can import AADL model and perform schedulability and timing analysis for early verification



```

thread ACC_RD_Input
  features

```

ToPCS: **requires data access**

```
ACCInput2PCS.impl;
```

ToACCState: **requires data access**

```
ACCInput2ACCMODE.impl;
```

...

ToCrzState: **requires data access**

```
ACCInput2CrzState.impl;
```

```
end FLAAD_FSRACC_RD_Input;
```

```

thread implementation ACC_RD_Input.impl
  properties

```

Dispatch_Protocol => Periodic;

Compute_Execution_Time => 4 Ms .. 5 Ms;

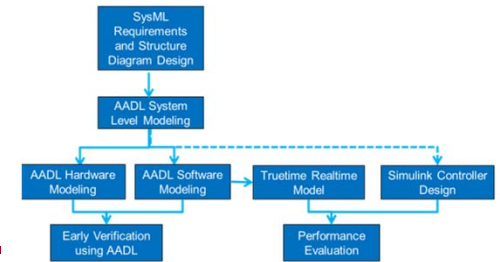
Deadline => 20 Ms;

Period => 20 Ms;

Cheddar_Properties::Fixed_Priority => 6;

```
end ACC_RD_Input.impl;
```

Truetime Model and AADL



- Truetime is a real-time toolbox for Simulink environment.
 - It is a wrapper outside Simulink blocks to specifies real-time specifications. AADL properties are translated into Truetime script manually in this study.

```

thread ACC_RD_Input
  features
    ToPCS: requires data access ACCInput2PCS.impl;
    ToACCState: requires data access
    ACCInput2ACCMODE.impl;
    ...
    ToCrzState: requires data access
    ACCInput2CrzState.impl;
  end FLAAD_FSRACC_RD_Input;

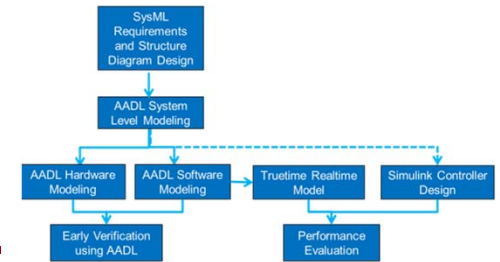
thread implementation ACC_RD_Input.impl
  properties
    Dispatch_Protocol => Periodic;
    Compute_Execution_Time => 4 Ms .. 5 Ms;
    Deadline => 20 Ms;
    Period => 20 Ms;
    Cheddar_Properties::Fixed_Priority => 6;
  end ACC_RD_Input.impl;
  
```

```

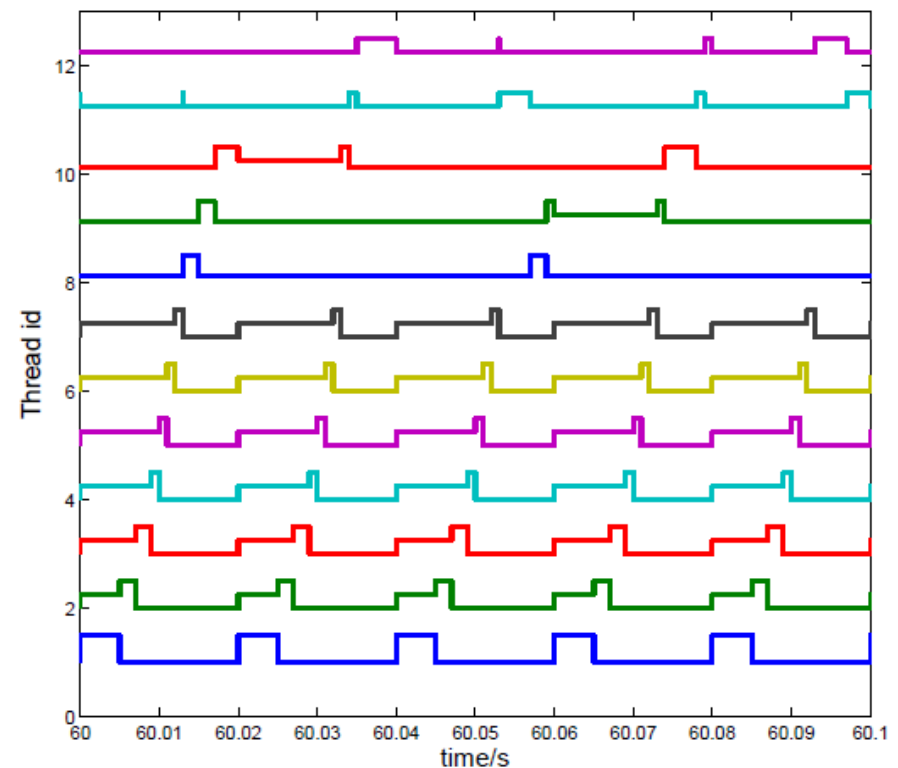
function [exectime,data] = ACC_Read_Inputs
(segment,data)
switch segment
case 1
  %Get Inputs
  data.ToPCS=ttAnalogIn(1);
  data.ToACCState=ttAnalogIn(2);
  ...
  data.ToCrzState=ttAnalogIn(12);

  %Post read data inputs
  ttPost('RdACCInput', data);
  exectime = 0.005;
case 2
  exectime = -1;
end
  
```

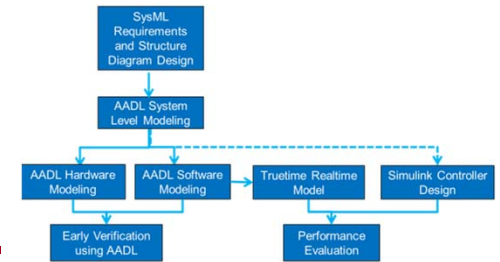

Truetime Scheduling and Performance Simulation



- Truetime
 - Realtime simulation
 - Capture real time delays between threads and communication loops
- CarSim
 - Physical simulator
- Simulink
 - Controller design and overall system simulator
- Testing Scenario
 - Two vehicles are driving in the same lane of a straight road.
 - Initial separation is 200 m
- Single core with 40ms sampling is shown on the right



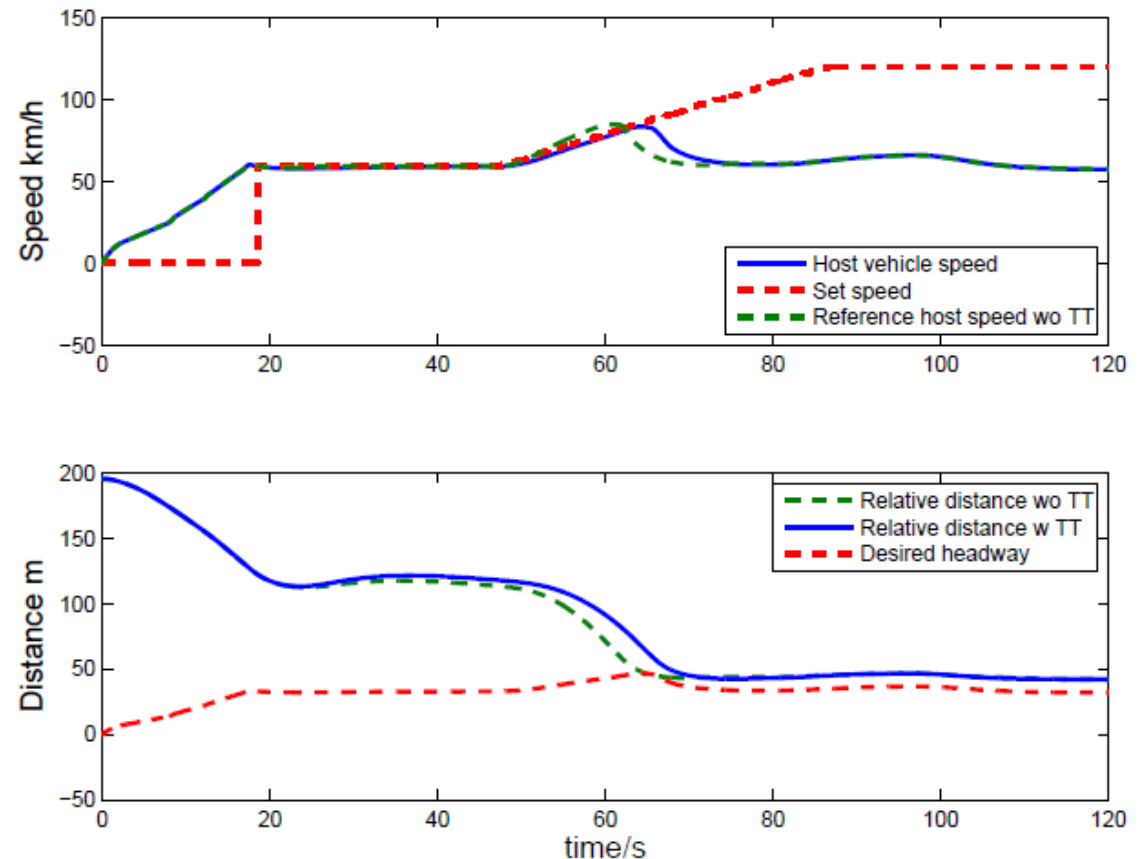
Performance Evaluation



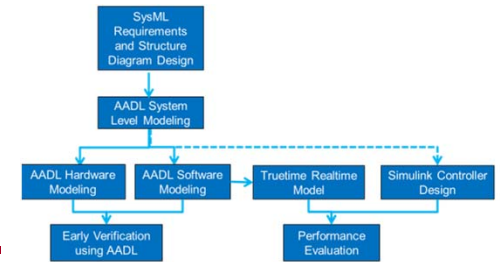
Testing Scenario

- The leading vehicle is driving manually accelerate and maintain speed around 60km/h
- The following host vehicle is manually accelerated to around 60km/h and ACC is switched on afterwards.
- Around 1 mins later, the following driver set speed to 120km/h
- Due to the decrease in relative distance, second car decelerated and followed the leading car.

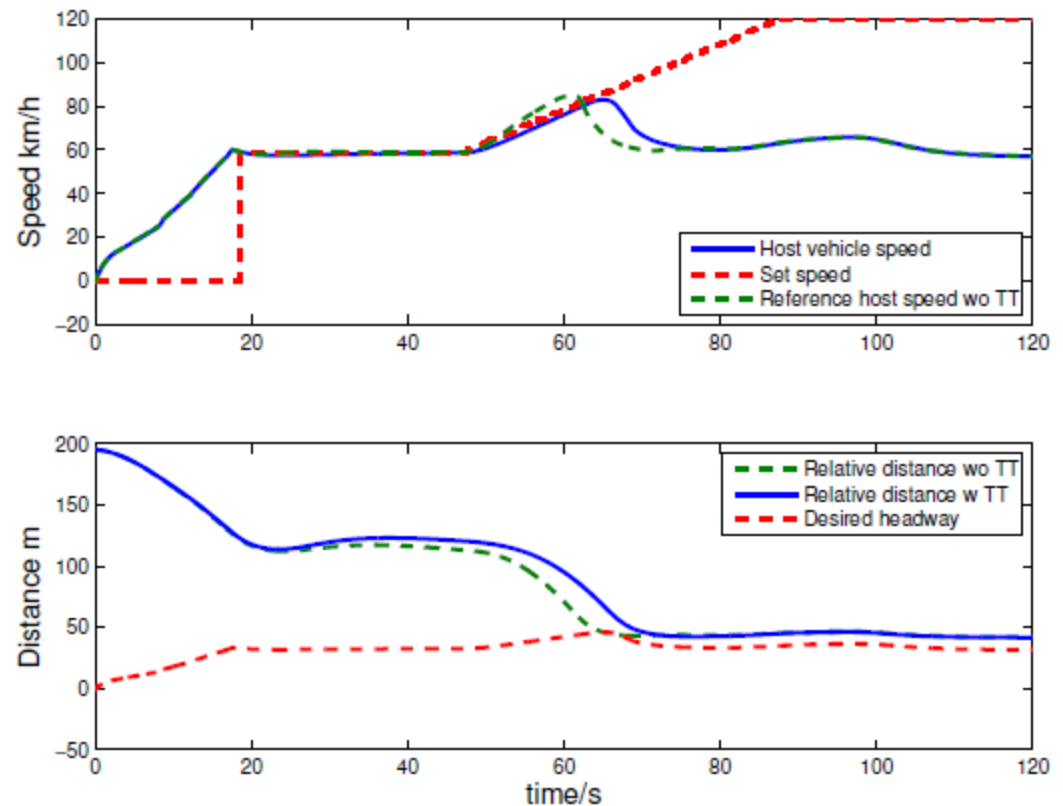
Overall system is still safe with temporal delay of real time execution and scheduling



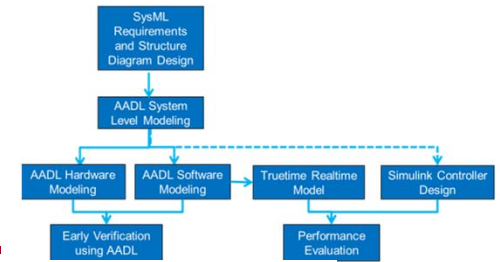
Test Scenario for Different Sampling Rate



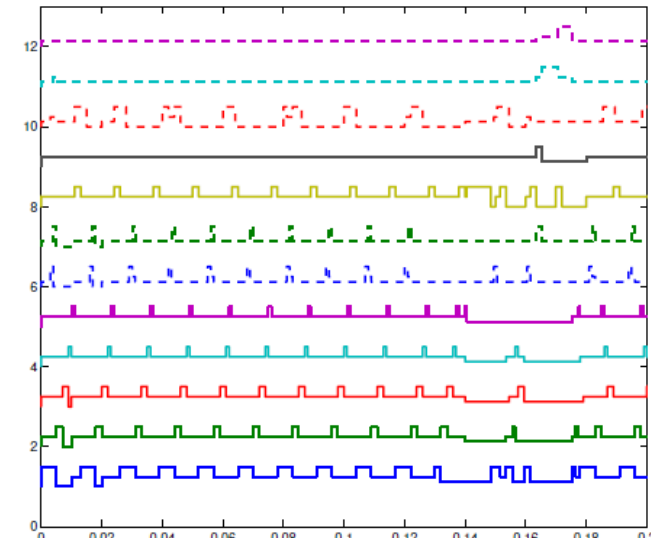
- Simulation for different sampling rate is also tested for 20ms and 80ms
- Performance wise, the result is similar, the algorithm is very robust to delays
- 40ms sampling case has few times when threads misses their deadline, and best cost wise for single core.



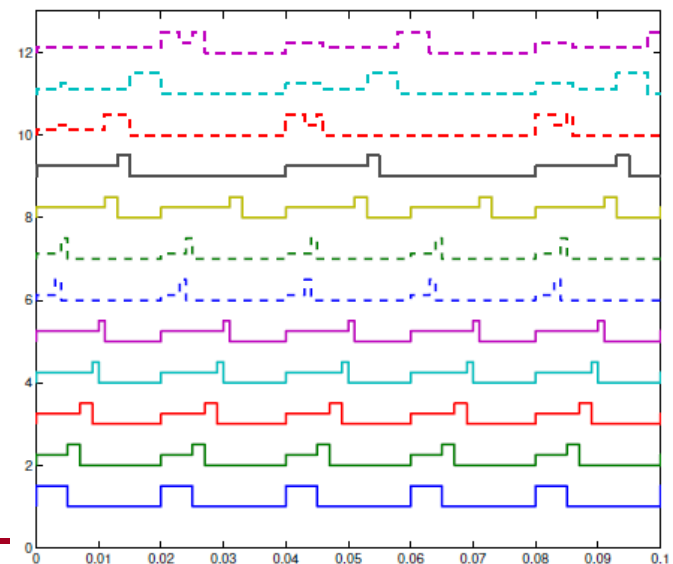
Performance for Different Hardware Configuration



- Dual-core case has similar performance comparing to the single core case.
- Scheduling table for 20ms and 40ms case is shown in the right. Clearly the 40ms case has far fewer misses on deadline.
- The single core 40ms configuration is chosen due to limited performance degradation, cheaper price and easier implementation.



20ms



40ms

- AADL acts as a backbone in the design process
- AADL is used in both architecture design and simulation guidance which helps the overall hardware software co-design process
- Behavior modeling and performance evaluation performed using realtime simulation help to verify and finalize the design.
- Future works include performing power and error analysis for the AADL model, as well as formal transformation of AADL model to Truetime simulation model.

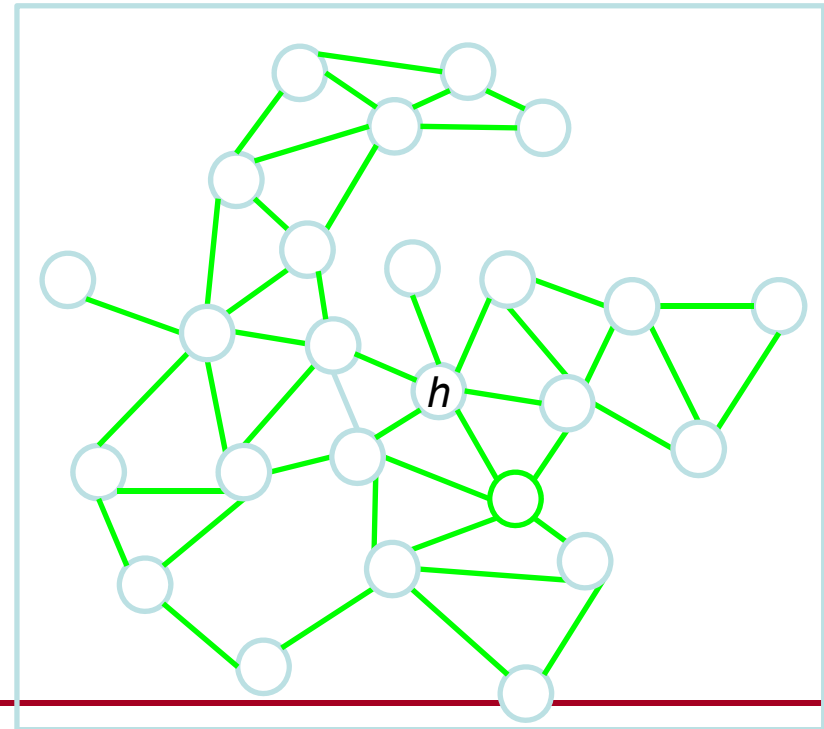
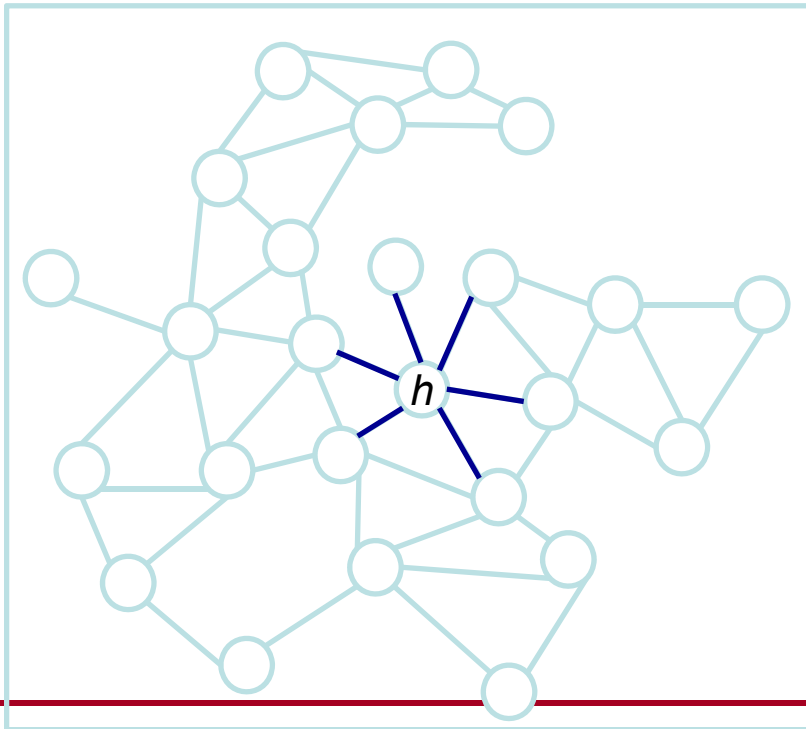
Part III: Distributed Topology Control for Stable Path Routing in Multi-Hop Wireless Networks

- Among the routing mechanisms for MANETs, **link state routing protocols** have shown good performance.
 - **Better convergence** for dynamic metrics and topology.
- However, link state algorithms suffer from **broadcast storm problems**
 - In MANETs, the link state is highly dynamic due to mobility and erasures.
 - There is significant overhead due to link-state broadcast.
- Several **local pruning approaches** have been proposed to **reduce the broadcast storm problem**.
 - Pruning local neighborhood information to reduce the link-state to be broadcast.

Traditional Link-State Routing

Neighbor Discovery
Component (NDC)

Topology
Dissemination
Component (TDC)

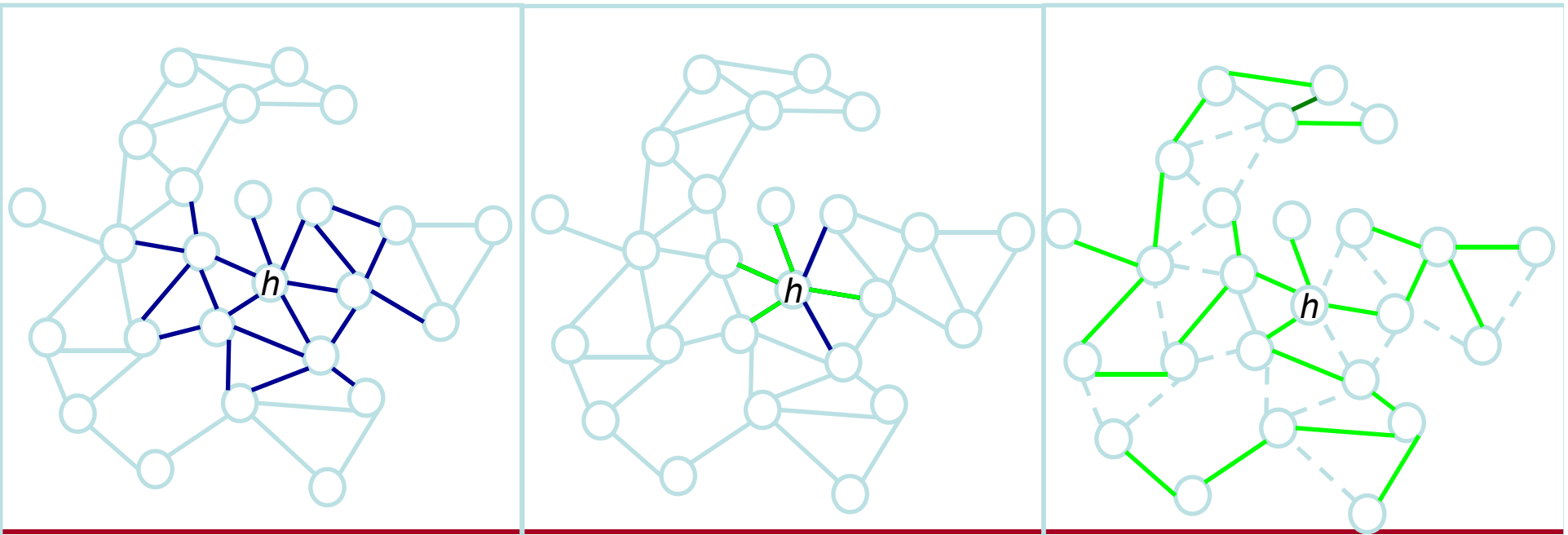


Compressed Link-State Routing – Topology Control

Neighbor Discovery
Component (NDC)

Selector of Topology
Information to
Disseminate (STIDC)

Topology
Dissemination
Component (TDC)



Graphs and Neighborhoods

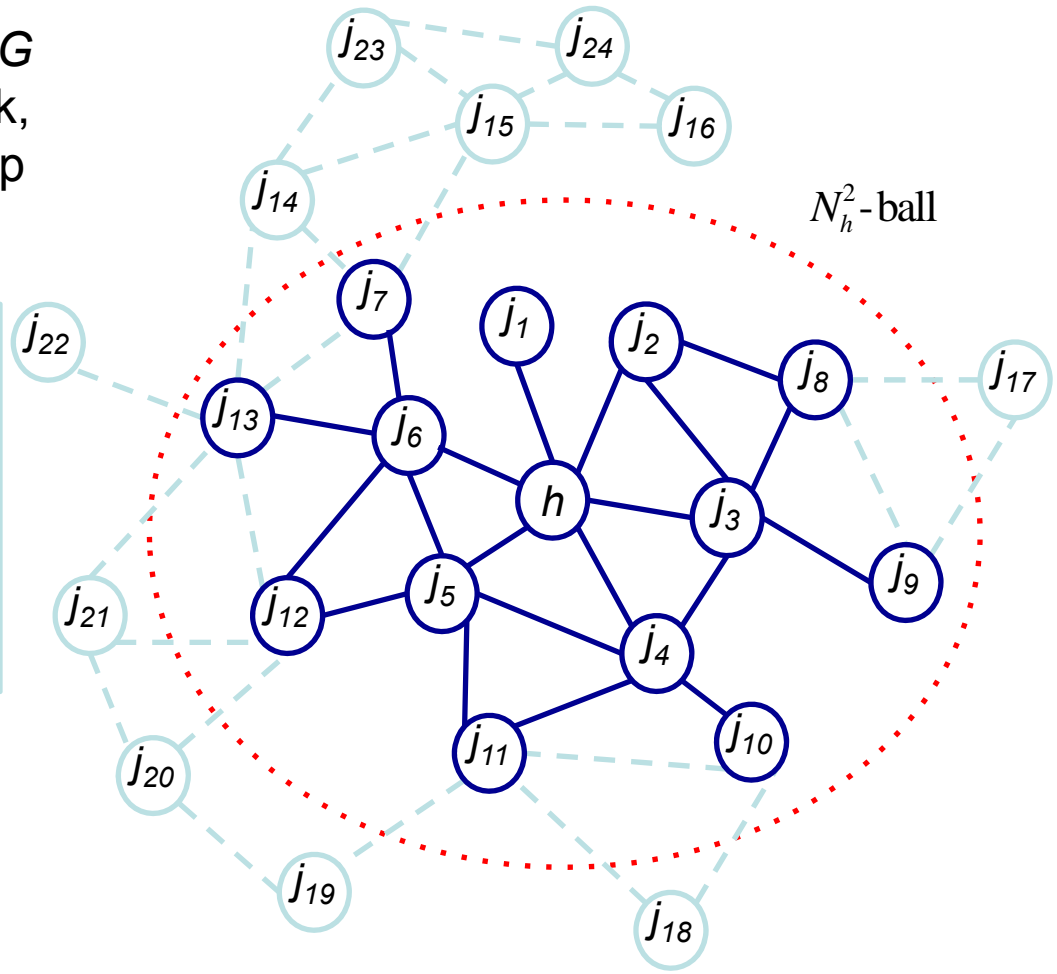
$G(V, E)$	Undirected graph G , with vertex set V and edge set E
$a_{uv}, (u, v) \in E$	Edge label
$G' \subseteq G$	Subgraph
$P_{ij}^{G'}$	Set of paths from i to j in G' .
Ω_i^G	Set of edges incident to i in G .
$d_{hc}(i, j)$	Minimum hop-count between from i to j in G .
$N_h^k = \{j \in V : d_{hc}(h, j) \leq k\}$	k -hop neighborhood of h
$\partial N_h^k = N_h^k \setminus N_h^{k-1}$	k -hop neighbors

G_h^{local} - **local view** is a subgraph of G induced by the k -hop neighbors of h , excluding the arcs of the strict k -hop neighbors.

Vertex set: N_h^k

Edge set:

$\{(u, v) : u, v \in N_h^k, \{u, v\} \not\subset \partial N_h^k\}$



G_h^{local} - **local view** is a subgraph of G induced by the k -hop neighbors of h , excluding the arcs of the strict k -hop neighbors.

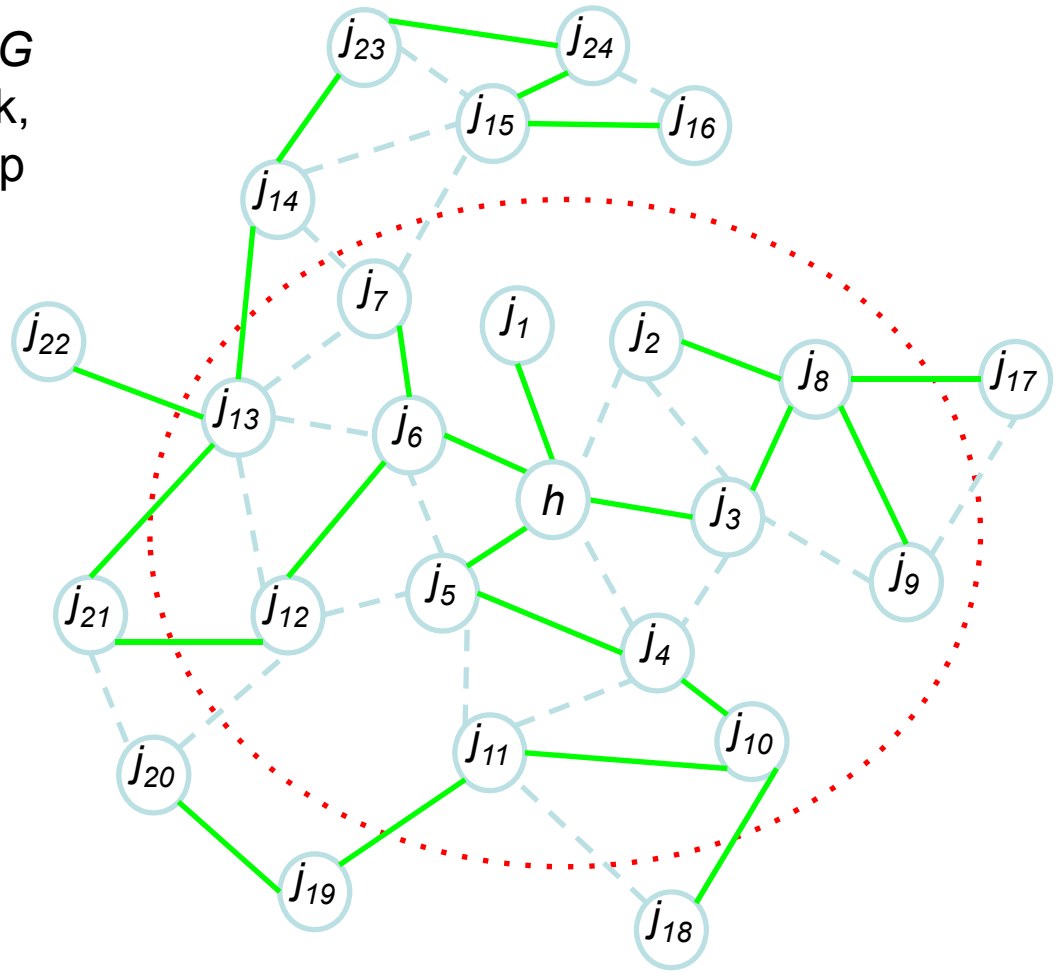
Every host vertex h broadcasts a selective subset of the out-arcs. This forms a **broadcast graph** $G^{broadcast}$

Every host h selects a

$$\Omega_h^{pruned} \subseteq \Omega_h^G$$

$$E^{broadcast} = \bigcup_{h \in V} \Omega_h^{pruned}$$

$E^{broadcast}$ induces $G^{broadcast}$

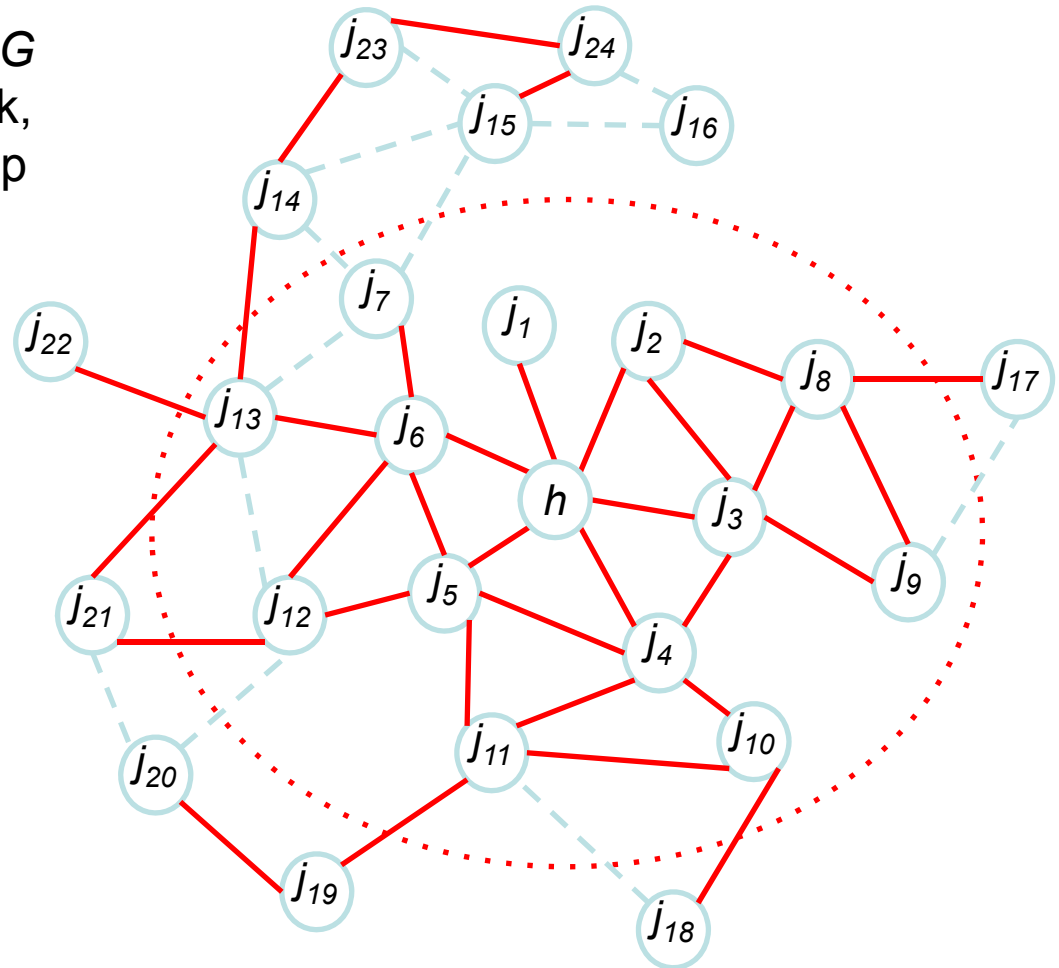


G_h^{local} - **local view** is a subgraph of G induced by the k -hop neighbors of h , excluding the arcs of the strict k -hop neighbors.

Every host vertex h broadcasts a selective subset of the out-arcs. This forms a **broadcast graph** $G^{broadcast}$.

Global view

$$G_h^{global} = G_h^{local} \cup G^{broadcast}$$



Topology Control for QoS Rule Based Routing

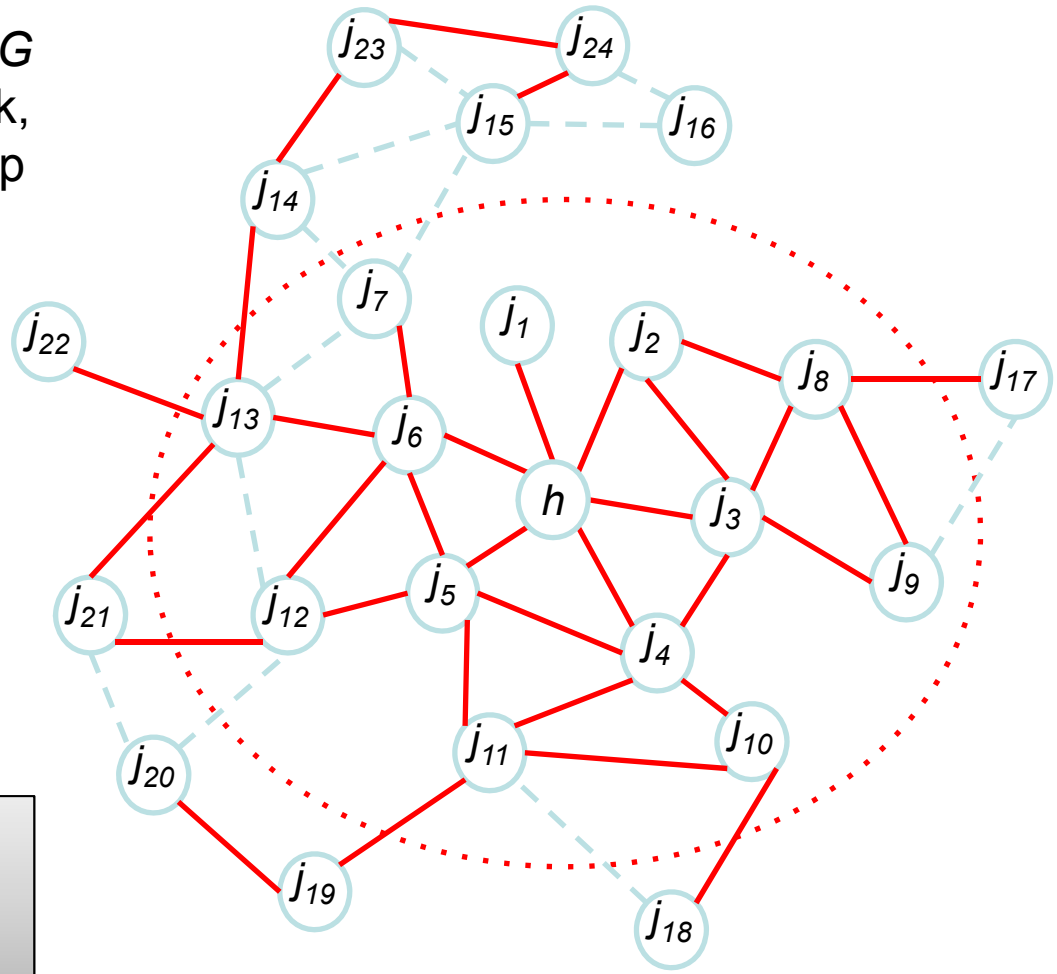
G_h^{local} - **local view** is a subgraph of G induced by the k -hop neighbors of h , excluding the arcs of the strict k -hop neighbors.

Every host vertex h broadcasts a selective subset of the out-arcs. This forms a **broadcast graph** $G^{broadcast}$.

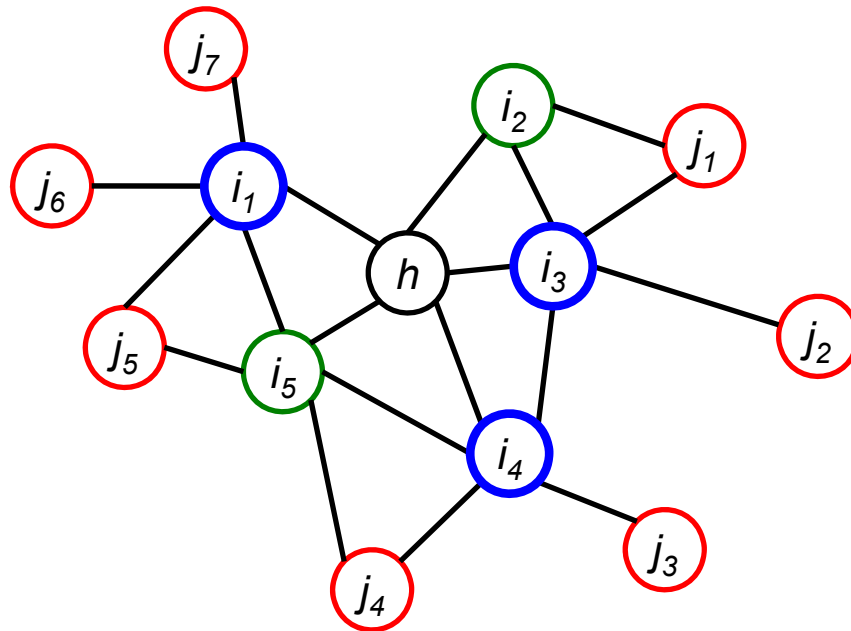
Global view

$$G_h^{global} = G_h^{local} \cup G^{broadcast}$$

Does G_h^{global} preserve the QoS optimal paths for routing from h to every destination?



Optimized Link State Routing Algorithm (OLSR)



Host h 's one-hop neighbors i_1 , i_3 and i_4 cover all two hop neighbors, i.e., there exists links from these one-hop neighbors all two-hop neighbors

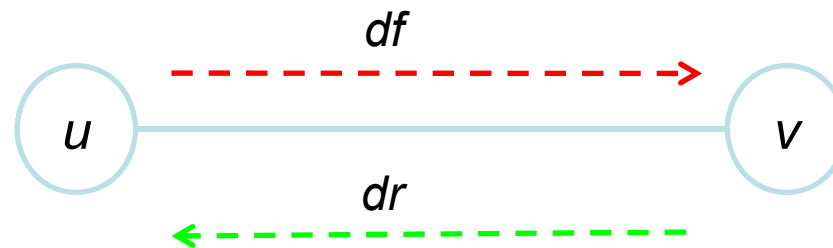
$$\Omega_h^{pruned} = \{(h, i_1), (h, i_3), (h, i_4)\}$$

Each vertex h ,

- chooses a **minimal subset of one-hop links**,
- such that the chosen one-hop neighbors **cover all two hop neighbors**

This is a set-cover problem – computationally hard.

ETX Link Stability Metric



df – forward delivery ratio

dr – reverse delivery ratio

$$ETX(u, v) = \frac{1}{df \times dr}$$

- OLSR-ETX uses the ETX metric to select the pruned edge set, Ω_h^{pruned} .

The best ETX metric for a two hop neighbor j

$$\min_{i \in \partial N_h^1} ETX(h, i) + ETX(i, j)$$

Note: best ETX path is of the form (h,i,j), i is a one-hop neighbor!

OLSR-ETX

- chooses a minimal subset of one-hop neighbors
- such that all two-hop neighbors are reachable by their best ETX metric path

QoS (Path Stability) Preserving Topology Control

- Path stability metric of a path p in G is the additive composition of its link stability metrics, $a_{uv} = ETX(u, v)$:

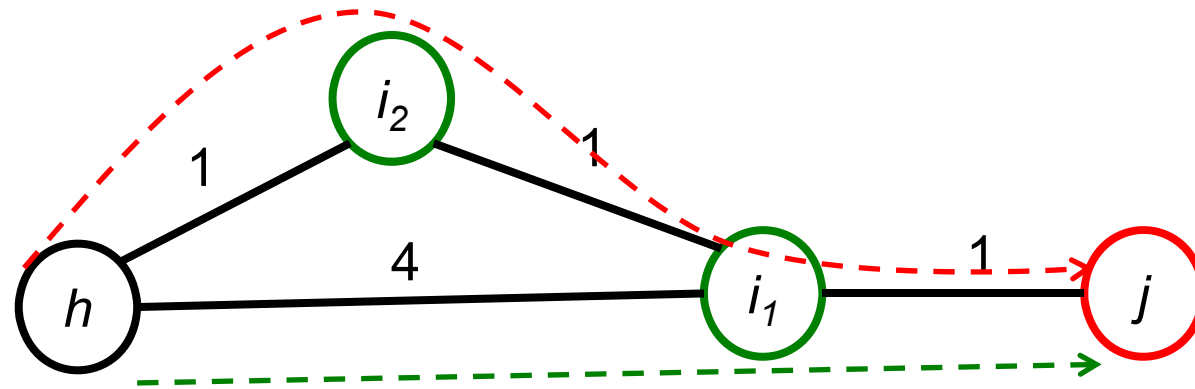
$$w_p = \sum_{(u,v) \in p} a_{uv}$$

- The optimal path stability is

$$x_{ij}^G = \min_{p \in P_{ij}^G} w_p$$

- Does OLSR-ETX pruning preserve the optimally stable path?

Limitation of Existing Pruning Methods – Fork Network Example



- Consider OLSR pruning at host h . The only **one-hop neighbor of h that covers all two-hop neighbors** is i_1 .
- However, there is an **alternative better path**:
($h \rightarrow i_2 \rightarrow i_1 \rightarrow j$)
- Set-cover based pruning methods, like OLSR, **guarantee only connectivity** of the pruned graph.

Stable/Shortest Path Preserving Topology Control

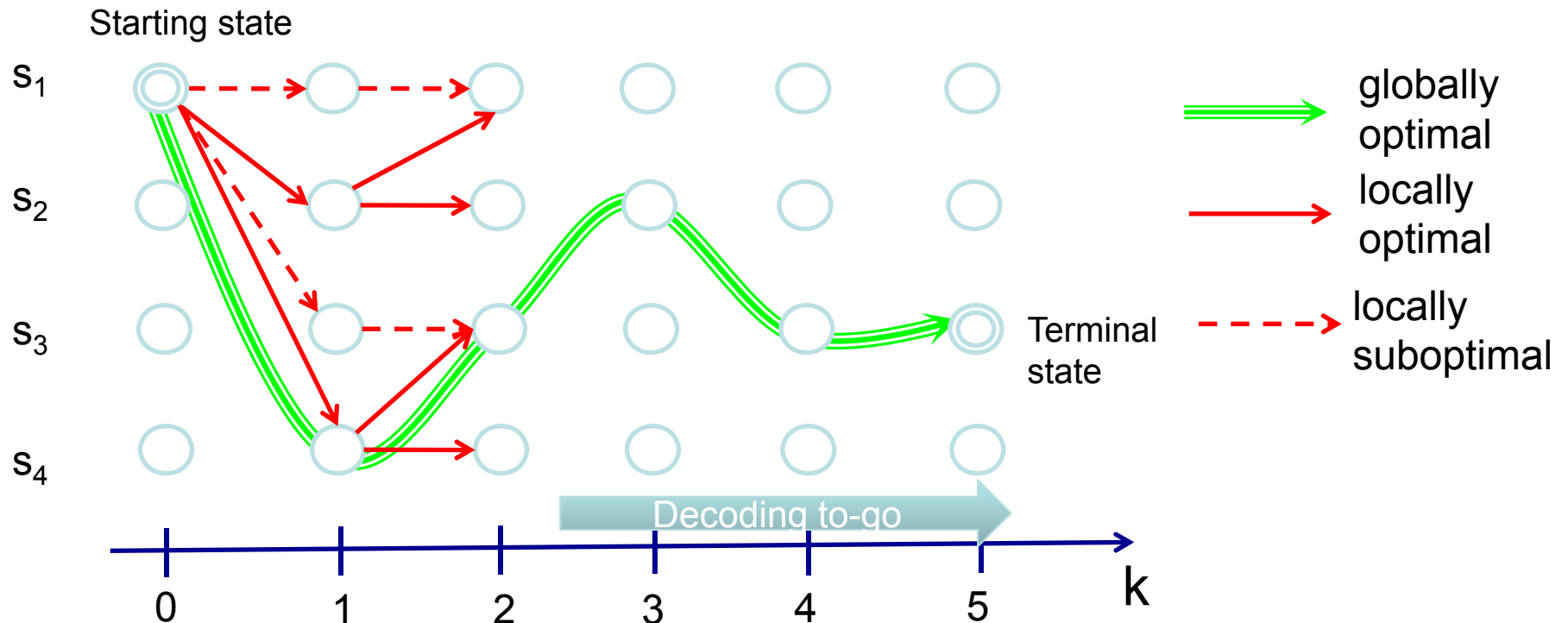
New TC mechanism that preserves QoS optimal paths (shortest paths)

Intuition:

- The **stable path** computation is a *(min,+)* computation.
- **Bellman principle** suggests that **preserving a local solution preserves the global solution.**

Algorithm Idea:

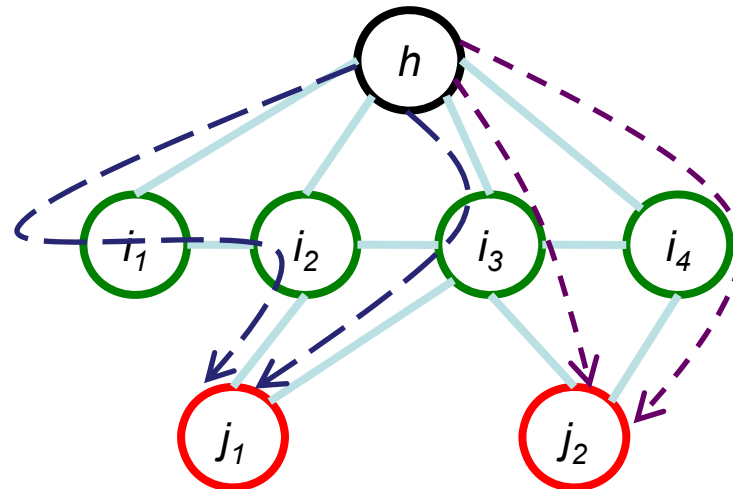
- Choose a **local solutions** such that **the amount of topology information disseminated is minimal.**



- At time $k=2$, you can prune away the dominated paths to keep track of only the non-dominated paths.

- e.g., $\text{likelihood}(s_1 \rightarrow s_2 \rightarrow s_1) > \text{likelihood}(s_1 \rightarrow s_1 \rightarrow s_1)$
 $\text{likelihood}(s_1 \rightarrow s_4 \rightarrow s_3) > \text{likelihood}(s_1 \rightarrow s_3 \rightarrow s_3)$

Weak Pruning



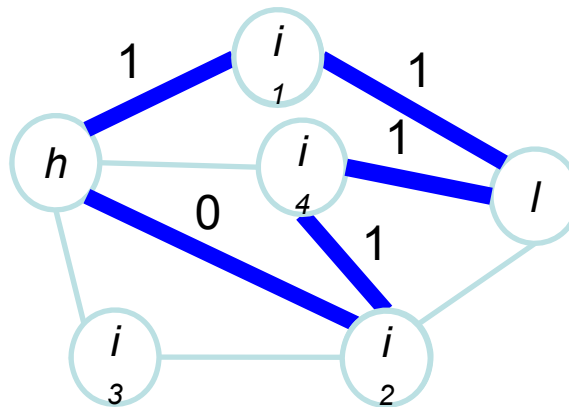
All one-hop links other than (h, i_2) are chosen for broadcast, i.e., significant links:
 $\{(h, i_1), (h, i_3), (h, i_4)\}$.

Theorem:

Choosing **all one-hop links**, (h, i_k) , which lie **on locally optimal path to every k-hop neighbor**, $j \in \partial N_h^k$, in the local view, G_h^{local} , is **sufficient** to preserve globally optimal paths in the global view, G_h^{global}

Strong Pruning and Loop-Freedom

- **Strong pruning**: Rather than preserving all optimal paths, preserving **one optimal path to every k-hop neighbor**.



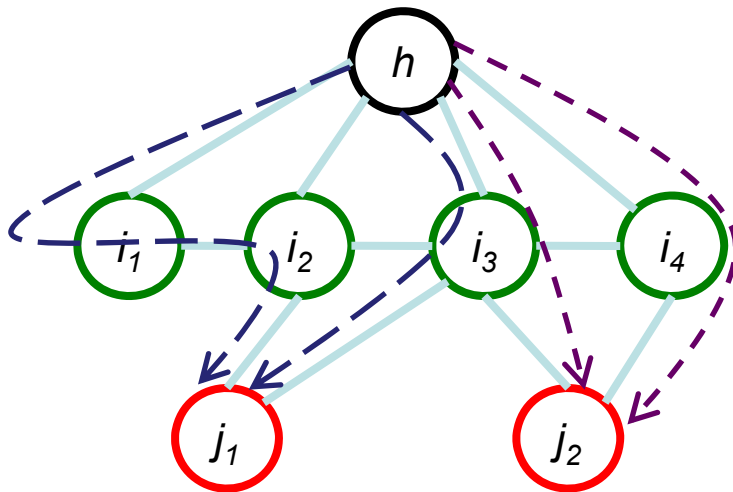
In the example, the optimal paths to l are $(h \rightarrow i_1 \rightarrow l)$ and $(h \rightarrow i_2 \rightarrow i_4 \rightarrow l)$.

- The non-triviality is in **ensuring loop-freedom** in distributed pruning.

Sufficient Conditions for Local Pruning

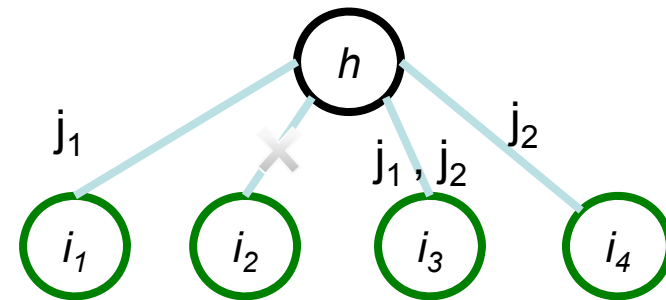
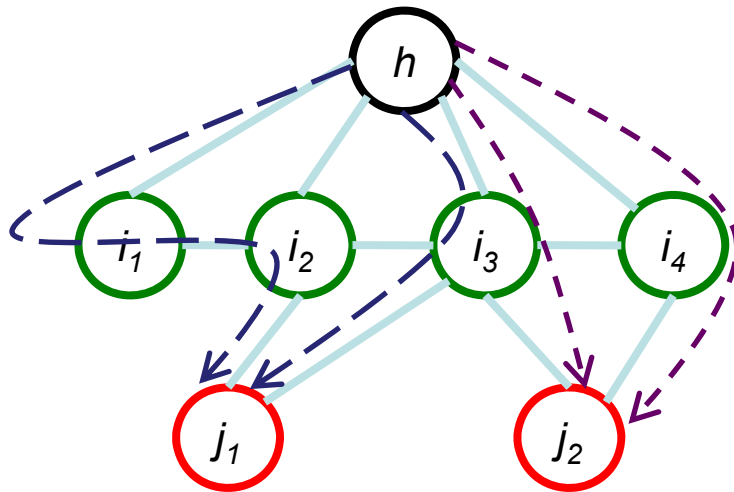
Theorem:

If the link weights are strictly positive, $a_{uv} > 0$, then preserving any one-hop link (h, i_k) on at-least one locally optimal path to every k -hop neighbor, $j \in \partial N_h^k$, in the local view, G_h^{local} , is sufficient to preserve the globally optimal paths in the global view, G_h^{global} .



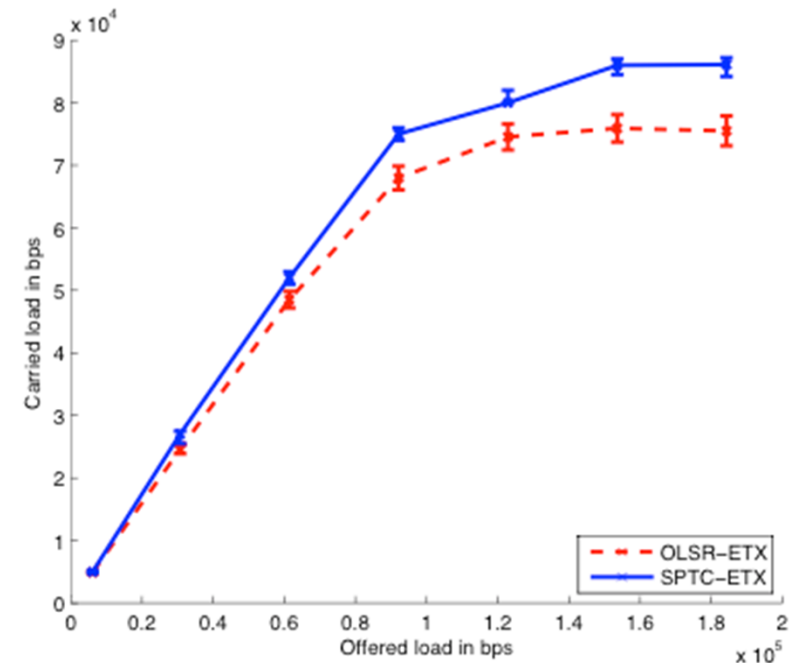
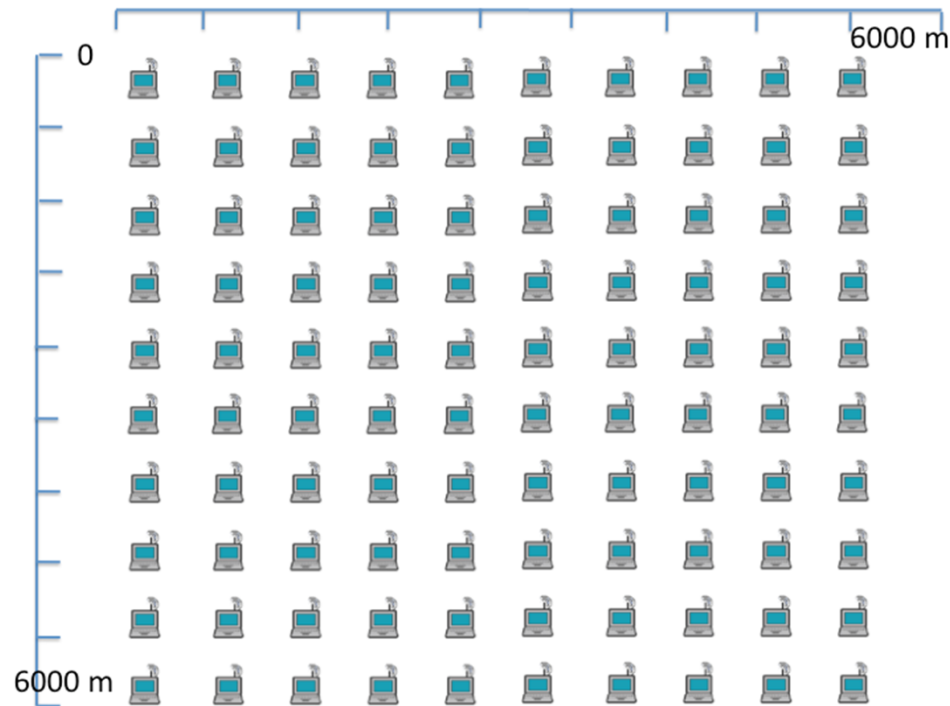
- For two-hop neighbor j_1 , h has to choose either (h, i_1) or (h, i_2) .
- For two-hop neighbor j_2 , h has to choose either (h, i_3) or (h, i_4) .

Algorithm: Stable Path Topology Control (SPTC)



- Remove the links not in any optimal paths: (h, i_2) .
- Label each link incident to the host with the two-hop neighbors whose optimal paths intersect with that link.
- Set-cover problem:
 - Choose a minimal subset of labeled links such that the union of the labels is all two-hop neighbors: **solution:** (h, i_3)
 - Computationally hard – greedy approximation algorithm.

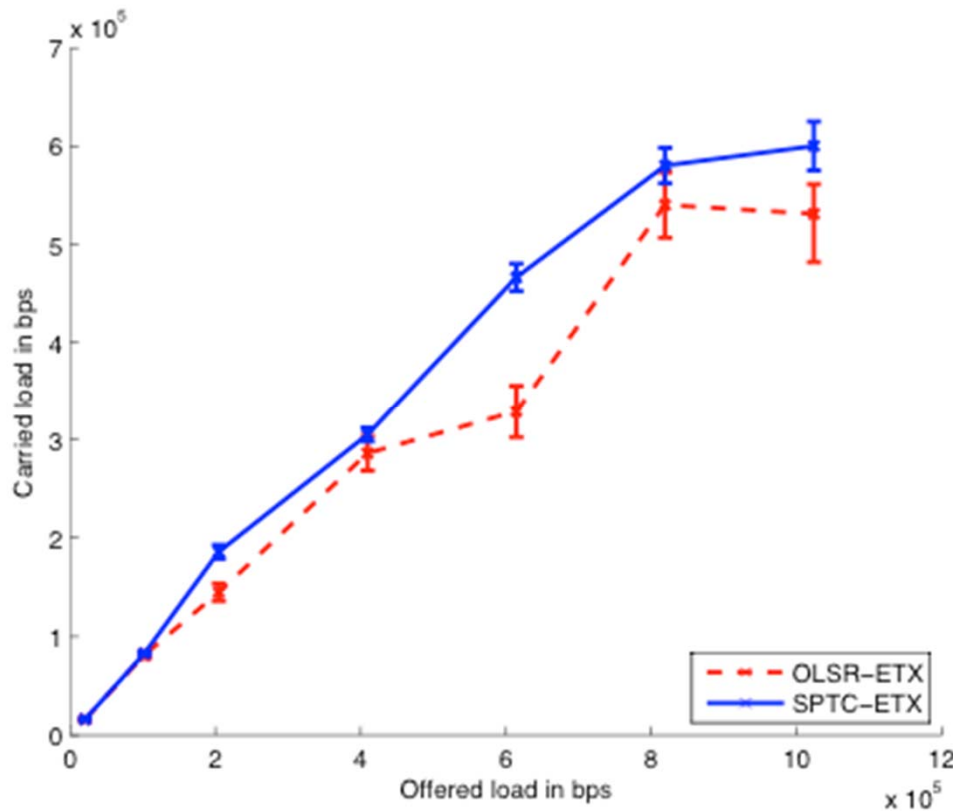
Static Grid Topology



- 5 UDP connections between random source pair destinations.

	OLSR-ETX	SPTC-ETX
Saturation CL	75 kbps	86 kbps
TC message rate	930 bps	681 bps

Random Waypoint Mobility Scenario

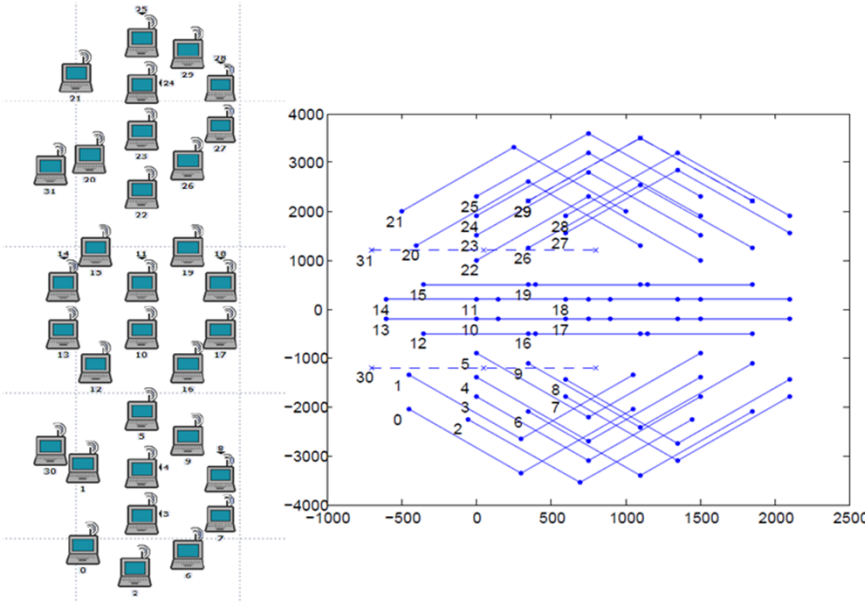


Parameter	Value
No. of stations	25
Simulation Area	3000m x 3000m
Speed	(5, 20] m/s
Pause time	0 s

	OLSR-ETX	SPTC-ETX
Saturation CL	50 kbps	61 kbps
TC message rate	8.3 kbps	6 kbps

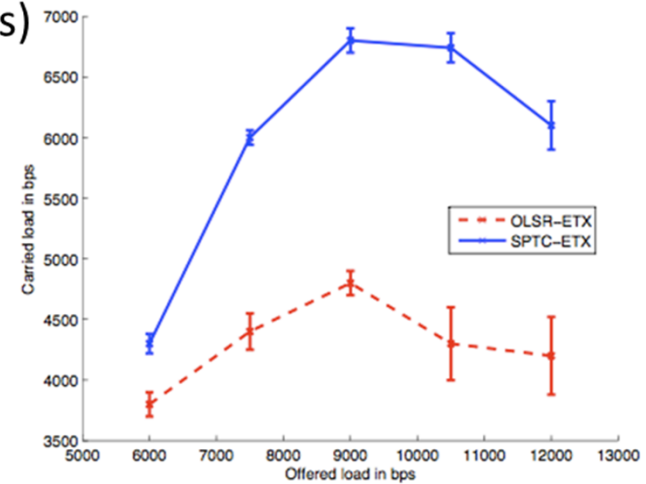
UDP sources between two different random source-pair destinations.

3 Platoon Mobility Scenario



	OLSR-ETX	SPTC-ETX
Saturation CL	~ 2 Mbps	~ 2 Mbps
TC message rate	923 kbps	890 kbps

Long connection from 20 to 0 (platoon heads)



Type	Connection	Offered-load
Intra-platoon	(1,3),(2,9),(4,6),(7,5),(20, 29), (14,17),(16,11),(17,18),(19,12), (21,22),(23,27),(23,28)	12 kbps
Inter-platoon	(1,18) (20,11),(20,0) (10,1),(21,10)	2.4 kbps 6 kbps 12 kbps

Summary

- A local pruning mechanism that ensures that globally optimal routing paths are preserved in the pruned graph.
- SPTC-ETX, when compared to OLSR-ETX
 - **Carries more traffic** – stable paths are long-lived
→ long-lived sessions
 - **Fewer topology changes** – stable links are long-lived
→ stable routing graph

Part IV: Composable and Assured Autonomy

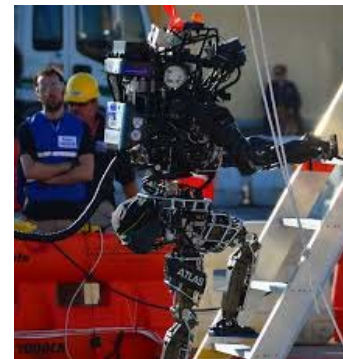
- Multi-sensory autonomous platforms
- Learning manipulation tasks
- Models for collaborative autonomy
- MBSE extensions
- Spatial and temporal specifications
- Port-Hamiltonian systems
- Contract-based design
- Learning to collaborate



The
Institute for
Systems
Research

UMIACS

Collaborative Autonomy and Trust



Learning Tasks, Changing Environments



- Teach through demonstrations
 - Easy training, hard to generalize to new constraints
- Program planning techniques
 - Generalize to constraints, manually design objectives

Example Task: Transferring

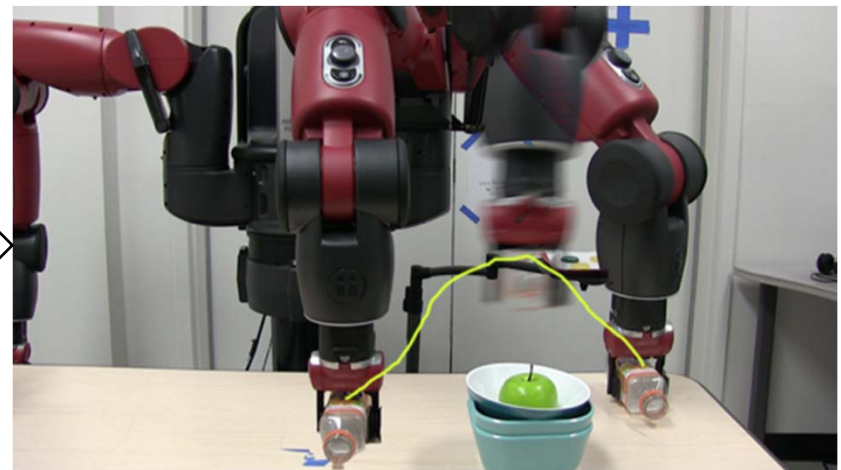


■ New environments

- Plan with new constraints: e.g. obstacles, safety margin
- Still tend to follow the imitation trajectory
- Generalize to new environments

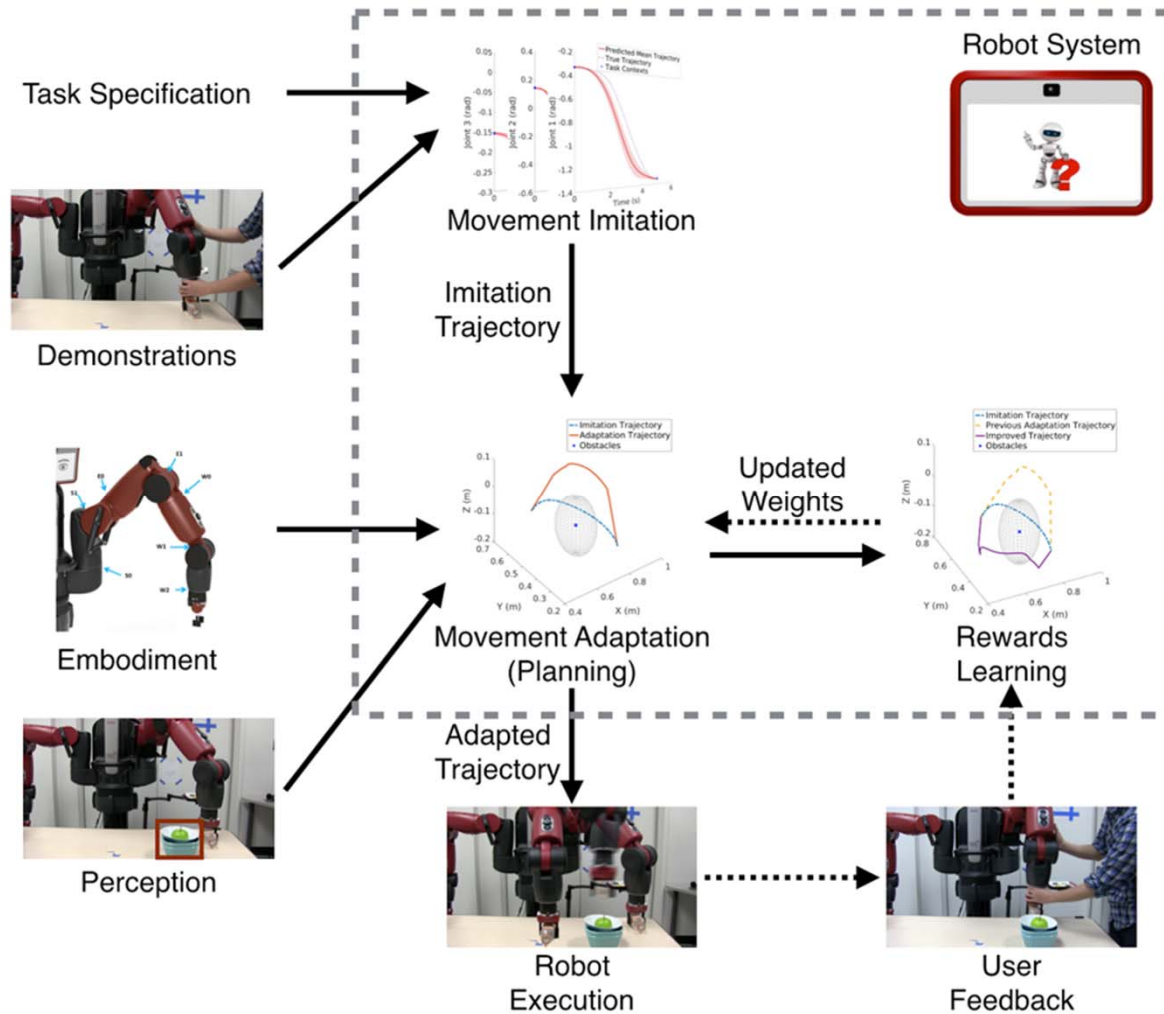


Movement Imitation



Movement Adaptation

Our System



Current Research Project

- Intelligent and Learning Autonomous Systems: Composability and Correctness
- Part of Science of Autonomy
- **Self-managed safety and correctness essential for autonomy** (single or collaborative agents):
 - Self-monitoring, Self-learning, Self-adjustment
- **Composability essential to control complexity**
 - Two fundamental ways to assure composability and correctness:
 - Intrinsically composable components
 - Hierarchical monitoring, adaptation and self-corrections via formal models

Current Research Project

Tasks:

(1) Extension of rigorous MBSE frameworks with internal world models and contracts.

The resulting framework will integrate results and allow systematic progress evaluation.

(2) Intrinsically composable system models with safe learning.

Develop port-Hamiltonian models enhanced with space-time safe learning.

(3) The cognitive dialogue between the cognition and perception/action executives.

Develop dynamic attention, logic based executives, to manage abstractions and executions.

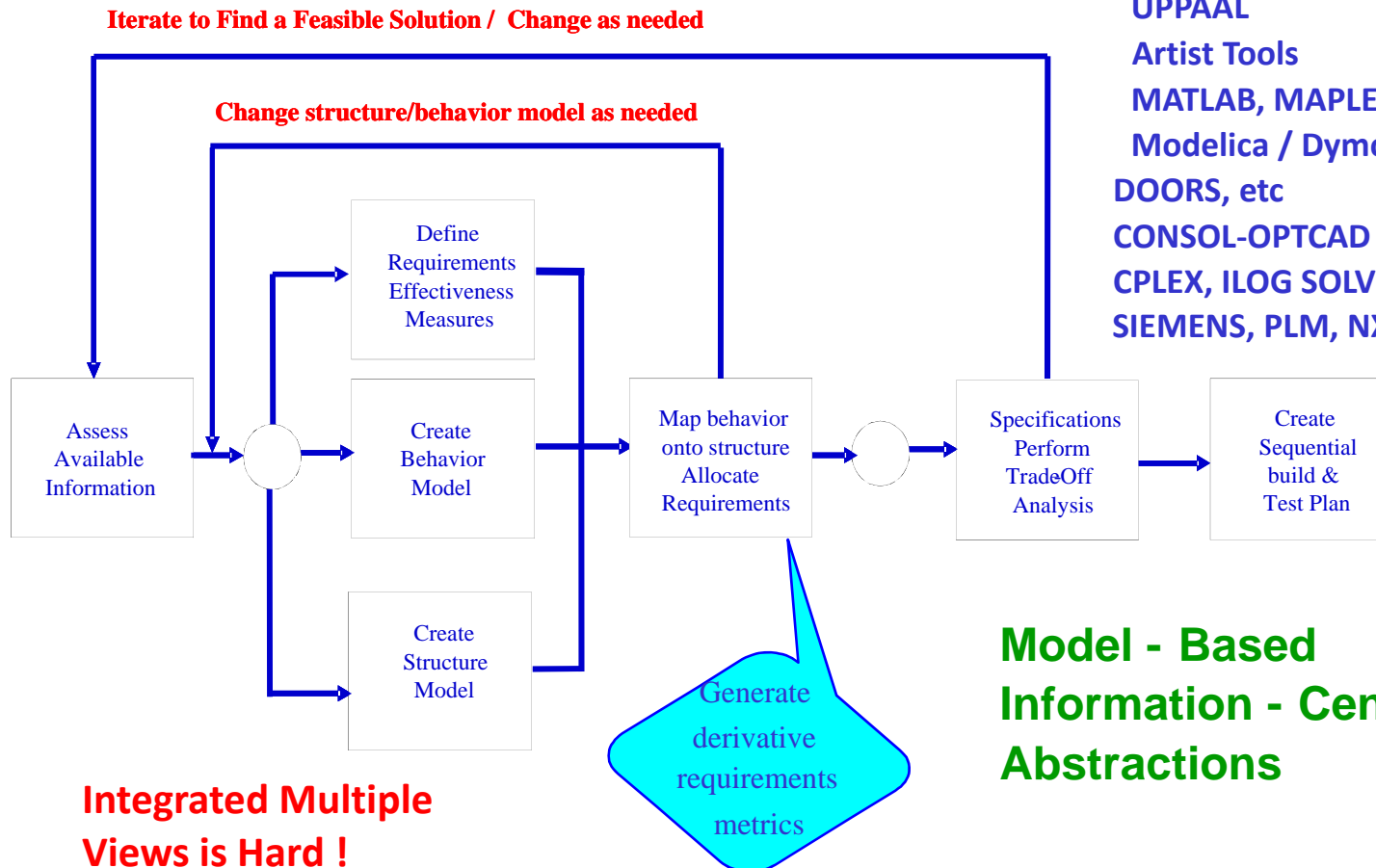
(4) Adaptable verification using runtime monitoring and learning.

Develop adaptive contracts, runtime monitoring, learning, for design changes verification.

MODEL-BASED SYSTEMS ENGINEERING COMPONENTS -- ARCHITECTURE

**Integrated System Synthesis Tools
& Environments missing**

Model- - based
UML - SysML - GME - eMFLON
Rapsody
UPPAAL
Artist Tools
MATLAB, MAPLE
Modelica / Dymola
DOORS, etc
CONSOL-OPTCAD
CPLEX, ILOG SOLVER,
SIEMENS, PLM, NX, TEAM CENTER

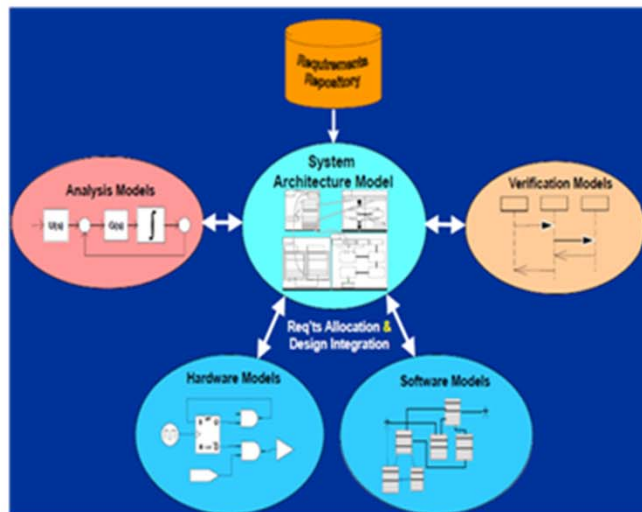


**Model - Based
Information - Centric
Abstractions**

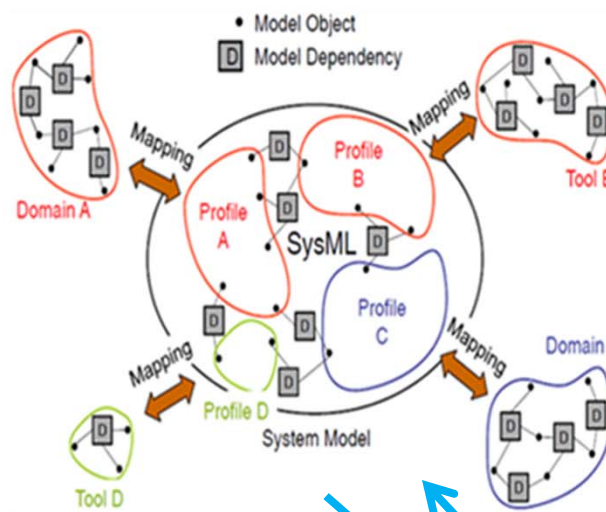
The Challenge & Need:

Develop scalable holistic methods, models and tools for enterprise level system engineering

Multi-domain Model Integration
via System Architecture Model (SysML)



System Modeling Transformations

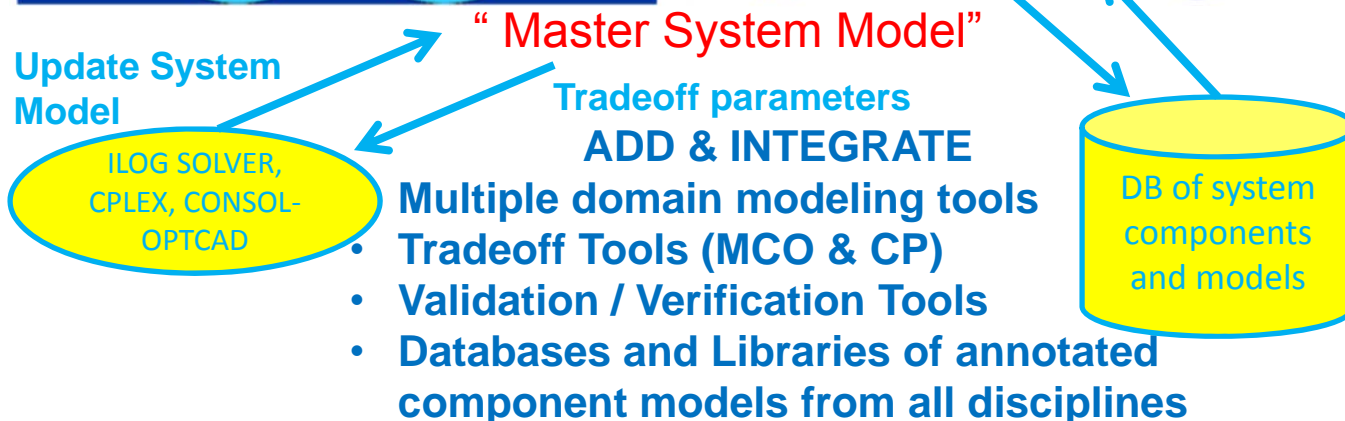


BENEFITS

- Broader Exploration of the design space
- Modularity, re-use
- Increased flexibility, adaptability, agility
- Engineering tools allowing conceptual design, leading to full product models and easy modifications
- Automated validation/verification

APPLICATIONS

- Avionics
- Automotive
- Robotics
- Smart Buildings
- Power Grid
- Health care
- Telecomm and WSN
- Smart PDAs
- Smart Manufacturing





Multi-sensory Autonomous Platform Architecture



- Integration is achieved through a dialogue between the vision and audio system, the motor system and the linguistic system, the visual executive (VE), the audio executive (AE), the control executive (CE) and the language executive (LE).
- VE, AE, CE provide answers and the LE poses questions
- To accomplish this for purposive intelligent systems we model the underlying actions – the task or the plan, is the driving force in this integration

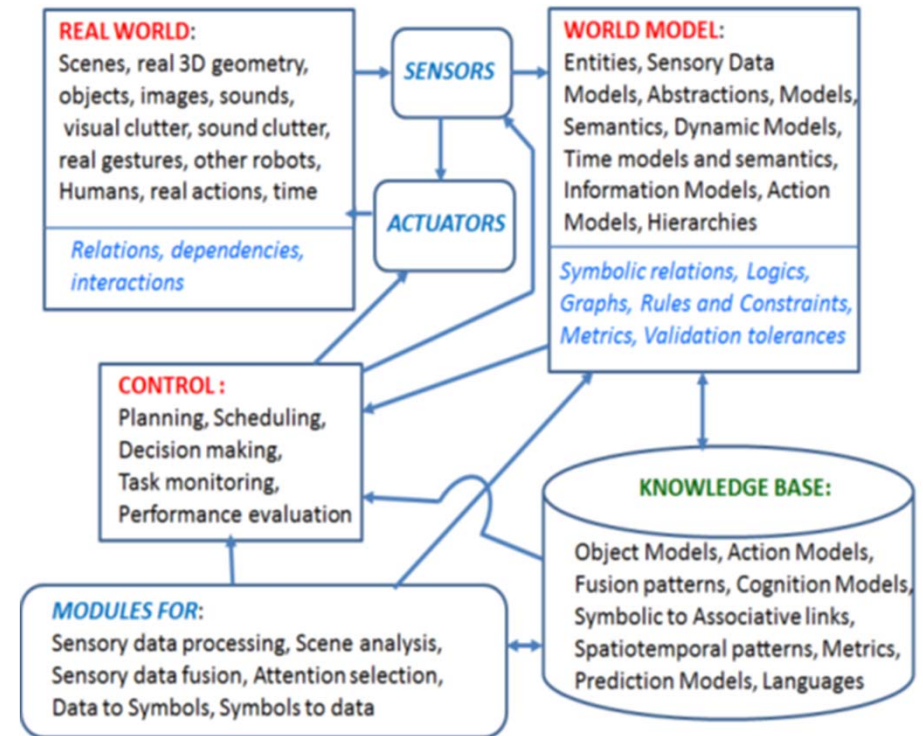


Fig. 1: Proposed MSEE platform system architecture

- **How to represent requirements?**
 - Automata, Timed-Automata, Timed Petri-Nets
 - Dependence-Influence graphs for traceability
 - Set-valued systems, reachability, ... for the continuous parts
 - Constraint – rule consistency across resolution levels
- **How to automatically allocate requirements to components?**
- **How to automatically check requirements?**
 - **Approach:** Integrate contract-based design, model-checking, automatic theorem proving
- **How to integrate automatic and experimental verification?**
- **How to do V&V at various granularities and progressively as the design proceeds – not at the end?**
- **The front-end challenge:** Make it easy to the broad engineering user?

- Traditional formal methods
 - Formulate specification, system
 - Prove that system satisfies spec
 - Model checking: proof search is automatic
 - Theorem proving: proof search requires human assistance
 - Developed for discrete systems
- For **compositionality**: contract-based specifications
 - Spec includes assumption A, guarantee G
 - Idea: system satisfies A/G if, whenever environment satisfies A, environment composed with system satisfies G
- What about
 - Hybrid systems?
 - Evolving environments?
 - Systems that learn?

Invariant Reconstruction

- Formal methods, **traditionally**
 - Given formal specification, system model
 - Determine if system satisfies specification
- Presupposes formal specification!
 - Non-trivial
 - Requires expertise, training in formal methods
- **Our work:** automated production of invariants from system models
 - Models simulated to produce behavioral data
 - Data mining used to propose invariants (= relationships among state / input / output variables that must be preserved)
 - Proposed invariants double-checked via “directed simulation”
- Uses
 - Proposed formal specifications
 - Model comprehension (“what does model do?”)
 - Double-checking of informal specifications (“did developers introduce new, undocumented assumptions?”)

Invariant Reconstruction (cont.)

- Results
 - Develop of invariant reconstruction tool for Simulink® models
 - Development of static-analysis techniques for improved invariant extraction
 - Evaluation on Simulink models for automotive, medical-device applications
 - Missing invariants detected
 - Omissions in requirements documentation uncovered

Safe Learning and Control

- Safety is (often) paramount for autonomous systems
- Learning (apparently) implies uncertainty
- How to ensure safety in face of uncertainty?
- **Earlier work:** dynamic collision-avoidance in the face of uncertainty using *control tubes*
 - Tubes = time-varying sets describing desired robustness
 - Idea: rather than devising controls based on desired trajectories, use (set-valued) tubes to ensure vehicles' tubes do not overlap
- Extend work on safety analysis and control synthesis, including **safe learning of space-time tolerances**
- **Set-valued semantics** for hybrid automata to support hierarchical analysis

Assurance: Spatial and Temporal Tolerances

- Reachable set based safety verification and control synthesis
 - Reachable set based verification
 - Control synthesis using optimization
- Motion planning for temporal logics with finite time intervals
 - Mixed integer optimization based method
 - Timed automata based method

- Passivity theory and compositionality
 - Passivity traditionally related to system energy
 - Key insight: storage functions can be decoupled from energy, used to compute Lyapunov, other functions compositionality
- Port-Hamiltonian system modeling
 - Ports: modeling construct for energy / information exchange between system components
 - Port-Hamiltonian approach used for modeling multi-physics systems
 - This project: explore port-Hamiltonian formalism as uniform basis for compositional modeling
 - Inclusion of timed automata, hybrid automata
 - Tooling based on Modelica, etc.
- Hierarchical timed automata and associated verification techniques for multi-scale timed systems

Port-Hamiltonian Models to the Rescue

Key ideas:

- Plant and controller – energy processing dynamical systems
- Exploit the interconnection – control as interconnection
- Shape energy
- Modify dissipation
- Work across multiple physics
- Work for many performance metrics not just stability
- Automatic composability -- scalable
- Underlying math models for Modelica!

Proof-Based Model Checking for Hybrid Systems



- Model-checking
 - Automated verification techniques, traditionally for discrete systems
 - More recently: systems with continuous + discrete dynamics (“hybrid systems”)
- State-of-the-art for hybrid systems
 - Approximate checking
 - Restricted classes of properties (“reachability”)
- **Our work:** a general proof-search strategy for hybrid systems, “modal mu-calculus”
 - Mu-calculus: expressive temporal logic
 - Our idea: combine general proof-search strategies with specific proof rules for different classes of systems

Contracts for Hybrid Systems

- How to specify A , G ?
- Idea: use hybrid automata
 - A : hybrid automaton describing “plant”
 - G : hybrid automaton describing “desired composite behavior
 - Composition operator(s) derived from e.g. hybrid process algebra (parallel composition, superposition, etc.)
- Theory, algorithms, synthesis approaches need development

Evolving Contracts

- Suppose system proven correct with respect A/G , and A is different at “run-time”?
 - Must adapt in the moment (e.g. Simplex architecture)
 - Must factor in change to contract
 - But how?
- Contract adaption
 - Theory of contract monitoring to detect deviations
 - Adaptation of A , G based on proofs of correctness
 - Use of on-the-fly model-checking techniques to compute, adapt proofs
- Contract synthesis
 - Use ideas from synthesis of temporal-logic specs from run-time data
 - Combine observations of environment, system to mine contracts from systems

Contracts -- Composability

Contract-based design and interface theories

- **Modal interfaces** as an algorithmic foundation of contract-based reasoning (residuation of modal specifications)
- **Quantitative interfaces**: Timed modal specifications, Timed I/O automata, probabilistic contracts, constraint Markov chains
- **Mica**: Modal interface compositional analysis Ocaml library

Supervisory control

- **Opacity**: optimal control of information flow in open systems
- Supervisory **control** of modal specifications of **services**
- Solution to the **quasi-static scheduling** problem
- Residuation of **tropical power series**

The Meta-theory: Components and Contracts

- ▶ Components: actual pieces of SW/HW/devices, open system
- ▶ Environment: context of use (a component), often unknown at design time
- ▶ Components cannot constrain their environment
- ▶ Contracts are intentionally **abstract**
- ▶ Pinpoint **responsibilities** of component vs. environment

$$\text{semantics}(\mathcal{C}) = \left(\underbrace{\mathcal{E}_c}_{\text{set of environments}}, \underbrace{\mathcal{M}_c}_{\text{set of components}} \right)$$

The Meta-theory

- ▶ We assume some **primitive concepts**:

Component	M
Composition	\times is partially defined, commutative and associative
Composability	$M_1 \times M_2$ being well-defined is a typing relation
Environment	E is an environment for M iff $E \times M$ is well-defined

- ▶ On top of these primitive concepts we define
 - ▶ generic concepts and operators
 - ▶ satisfying generic properties
- ▶ How concepts, operators, and properties, are made effective
 - ▶ depends on the specific framework

The Meta-theory

► Generic Relations and Operators:

Contract	$\text{sem}(\mathcal{C}) = (\mathcal{E}_c, \mathcal{M}_c)$ where $\mathcal{C} \in \mathbf{C}$: underlying class of contracts
Consistency	$\mathcal{M}_c \neq \emptyset$ say that $(\mathcal{C}_1, \mathcal{C}_2)$ is consistent iff $\mathcal{C}_1 \wedge \mathcal{C}_2$ is consistent
Compatibility	$\mathcal{E}_c \neq \emptyset$ say that $(\mathcal{C}_1, \mathcal{C}_2)$ is compatible iff $\mathcal{C}_1 \otimes \mathcal{C}_2$ is compatible
Implementation	$M \models^M \mathcal{C}$ iff $M \in \mathcal{M}_c$; $E \models^E \mathcal{C}$ iff $E \in \mathcal{E}_c$
Refinement	$\mathcal{C}' \preceq \mathcal{C}$ iff $\mathcal{E}_{c'} \supseteq \mathcal{E}_c$ and $\mathcal{M}_{c'} \subseteq \mathcal{M}_c$
Conjunction	$\mathcal{C}_1 \wedge \mathcal{C}_2 = \text{GLB for } \preceq \text{ within } \mathbf{C}$ $\mathcal{C}_1 \vee \mathcal{C}_2 = \text{LUB for } \preceq \text{ within } \mathbf{C}$
Composition	$\mathcal{C}_1 \otimes \mathcal{C}_2 = \min \left\{ \mathcal{C} \mid \left[\begin{array}{l} \forall M_1 \models^M \mathcal{C}_1 \\ \forall M_2 \models^M \mathcal{C}_2 \\ \forall E \models^E \mathcal{C} \end{array} \right] \Rightarrow \left[\begin{array}{l} M_1 \times M_2 \models^M \mathcal{C} \\ E \times M_2 \models^E \mathcal{C}_1 \\ E \times M_1 \models^E \mathcal{C}_2 \end{array} \right] \right\}$
Quotient	$\mathcal{C}_1 / \mathcal{C}_2 = \max \{ \mathcal{C} \in \mathbf{C} \mid \mathcal{C} \otimes \mathcal{C}_2 \preceq \mathcal{C}_1 \}$

Generic Properties:

Refinement	<p>substituability ↗ of sets of environments</p> <p>substituability ↘ of sets of implementations</p>
Composition	$\left. \begin{array}{l} (C_1, C_2) \text{ compatible} \\ C'_i \preceq C_i \end{array} \right\} \Rightarrow \left\{ \begin{array}{l} (C'_1, C'_2) \text{ compatible} \\ C'_1 \otimes C'_2 \preceq C_1 \otimes C_2 \end{array} \right.$ <p>independent implementability</p> $(C_1 \otimes C_2) \otimes C_3 = C_1 \otimes (C_2 \otimes C_3)$ <p>associativity</p> $[(C_{11} \wedge C_{21}) \otimes (C_{12} \wedge C_{22})] \preceq [(C_{11} \otimes C_{12}) \wedge (C_{21} \otimes C_{22})]$ <p>sub-distributivity: sets the freedom in design processes, fusing viewpoints before/after composing sub-systems</p>
Quotient	$C \preceq C_1 / C_2 \Leftrightarrow C \otimes C_2 \preceq C_1$

Abstracting and Testing

- ▶ Restrictions must hold for relations and operators on contracts to be analyzable
- ▶ Such restrictions may not hold for system models in practice
- ▶ Typical obstacles are infinite data types and functions operating on them

- ▶ Two complementary ways of overcoming this consist in
 - ▶ performing **abstractions**
 - ▶ performing **testing**

- ▶ The meta-theory offers generic means:
 - ▶ **abstraction** on top of **abstract interpretation** for components
 - ▶ **observers** for contract-compliant testing

- ▶ **Component:**
 - ▶ Kahn Process Network (KPN) or
 - ▶ Synchronous Transition System (STS)
- ▶ **Contract:** pair (**Assumption, Guarantee**) = (KPN,KPN) or (STS,STS)
 $\mathcal{C} = (A, G)$ defines a contract $(\mathcal{E}_c, \mathcal{M}_c)$ following the meta-theory:

$$\begin{aligned}\mathcal{E}_c &= \{E \mid E \subseteq A\} \\ \mathcal{M}_c &= \{M \mid A \times M \subseteq G\}\end{aligned}$$

- ▶ The following (existing) definitions specialize the meta-theory:

$$\begin{aligned}\mathcal{C}_1 \wedge \mathcal{C}_2 &\equiv (A_1 \cup A_2, G_1 \cap G_2) \\ \mathcal{C}_1 \otimes \mathcal{C}_2 &\equiv ((A_1 \cap A_2) \cup \neg(G_1 \cap G_2), G_1 \cap G_2)\end{aligned}$$

- ▶ No quotient exists

A/G Contracts: the Contracts

$\mathcal{C} = (A, G)$; A (the **assumptions**) and G (the **guarantees**) are assertions over Σ

Behaviors must be of the same kind for both components and contracts
(either both KPN or both synchronous)

$\mathcal{C} = (A, G)$ defines a contract $(\mathcal{E}_c, \mathcal{M}_c)$, where:

$$\begin{aligned}\mathcal{E}_c &= \{E \mid E \subseteq A\} \\ \mathcal{M}_c &= \{M \mid A \times M \subseteq G\}\end{aligned}$$

Contracts \mathcal{C} and \mathcal{C}' such that

$$A = A' \quad \text{and} \quad G \cup \neg A = G' \cup \neg A'$$

are **equivalent** as they yield identical sets of environments and components.

\mathcal{C} can always be **saturated** meaning that $G \supseteq \neg A$. This is assumed next.

A/G Contracts: the Contracts

Refinement (for $\mathcal{C}_1, \mathcal{C}_2$ saturated):

$$\mathcal{C}' \preceq \mathcal{C} \text{ holds iff } \begin{cases} A' \supseteq A \\ G' \subseteq G \end{cases}$$

Composition (for $\mathcal{C}_1, \mathcal{C}_2$ saturated):

$$G = G_1 \cap G_2$$

$$A = \max \left\{ A \mid \begin{array}{l} A \cap G_2 \subseteq A_1 \\ A \cap G_1 \subseteq A_2 \end{array} \right\} = (A_1 \cap A_2) \cup \neg(G_1 \cap G_2)$$

No quotient exists

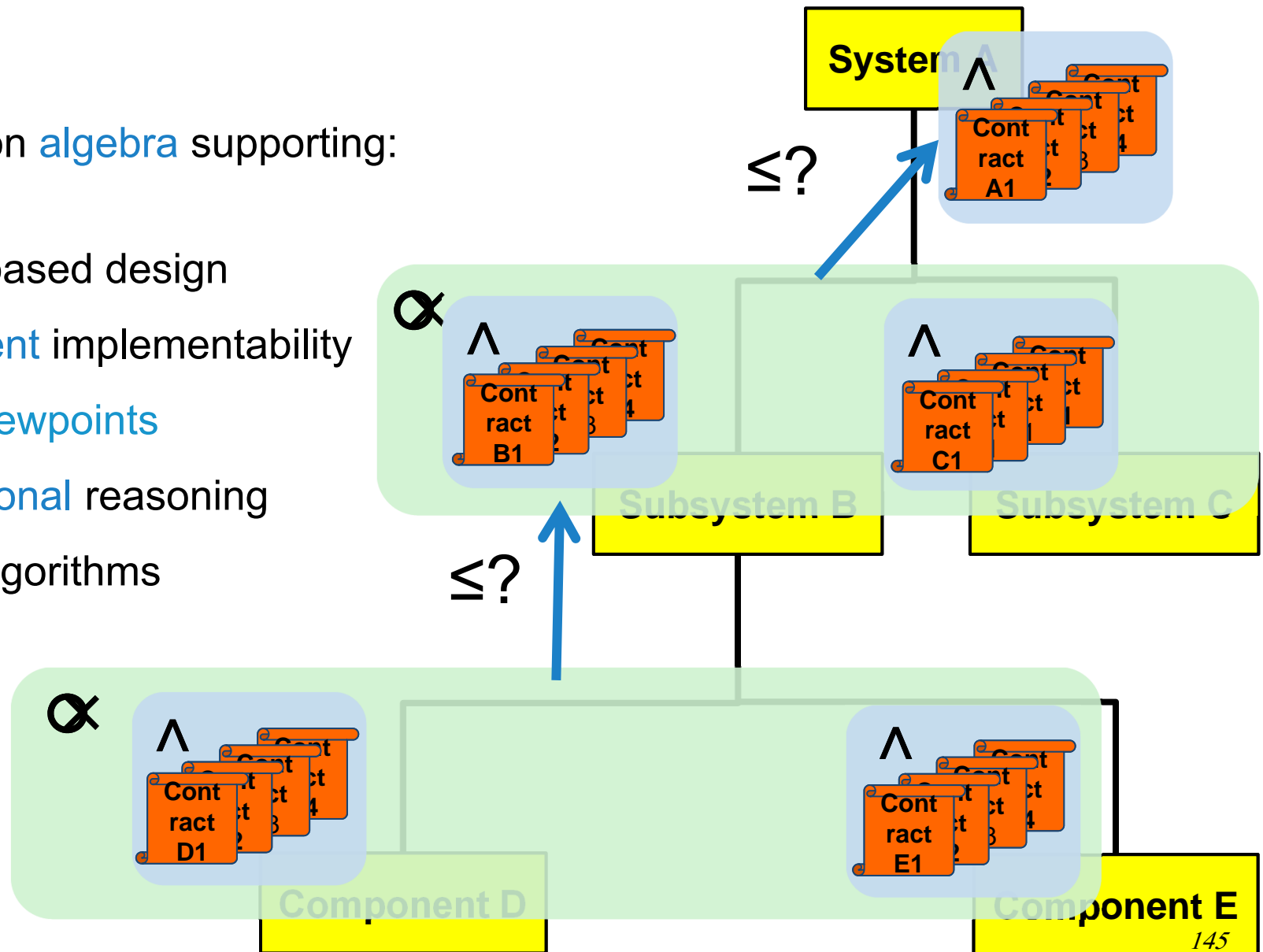
Problem: need to complement assertions. Use observers or abstractions?

Abstractions and observers can be defined

Modal Interfaces: Algorithmic Foundation of (A,G) Contracts

Specification **algebra** supporting:

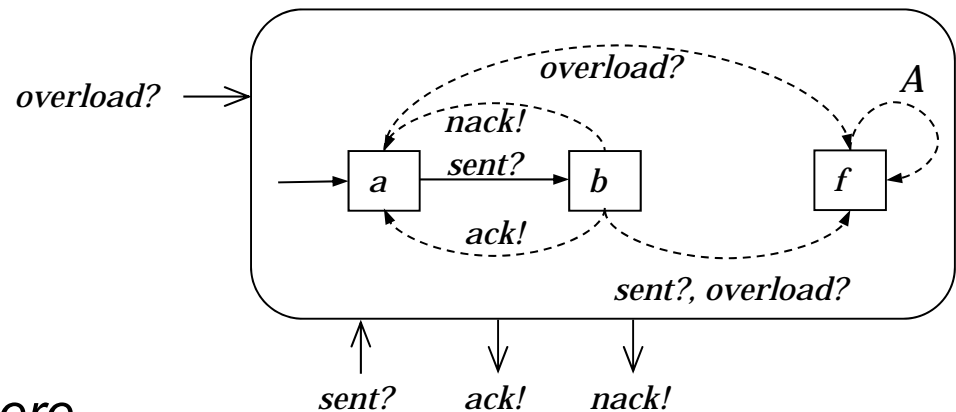
- **Contract**-based design
- **Independent** implementability
- Multiple **viewpoints**
- **Compositional** reasoning
- **Efficient** algorithms



Modal Interfaces

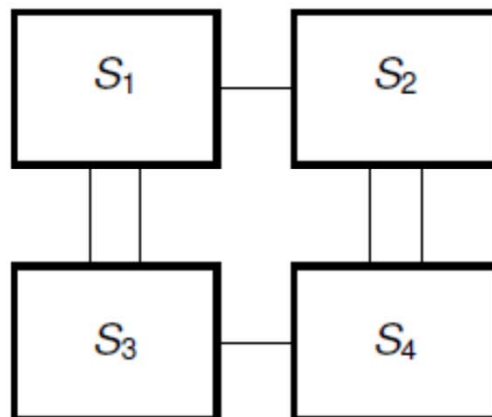
Deterministic I/O modal transition system: transitions are given a label *may* or/and *must*

- *may* transitions are dashed
- *must* transitions are solid
- *implementation: must* everywhere
- *refinement:* simulation rel. strengthening *must* and weakening *may*
- *extend* Interface Automata, ~ conjunctive fragment μ -calculus,
- *polynomial* complexity (unlike μ -calculus) (prototype tool *Mica*)

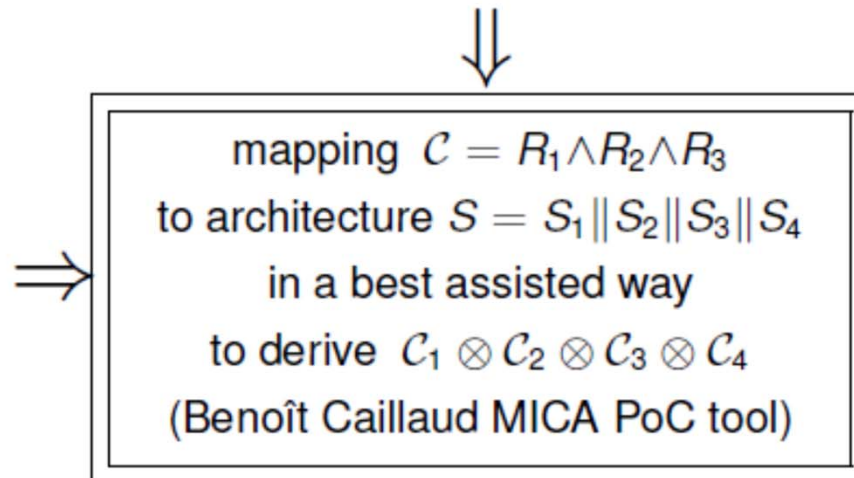


- Consistency and Compatibility
- Refinement and Conjunction
- Composition and Quotient
- Methods for Translating Assumptions and Guarantees into Modal Interfaces
- Use Observers for Behavior Monitoring
- Use Observers for Safety Analysis

Contract-Based Requirements Engineering



system architecture (SysML)

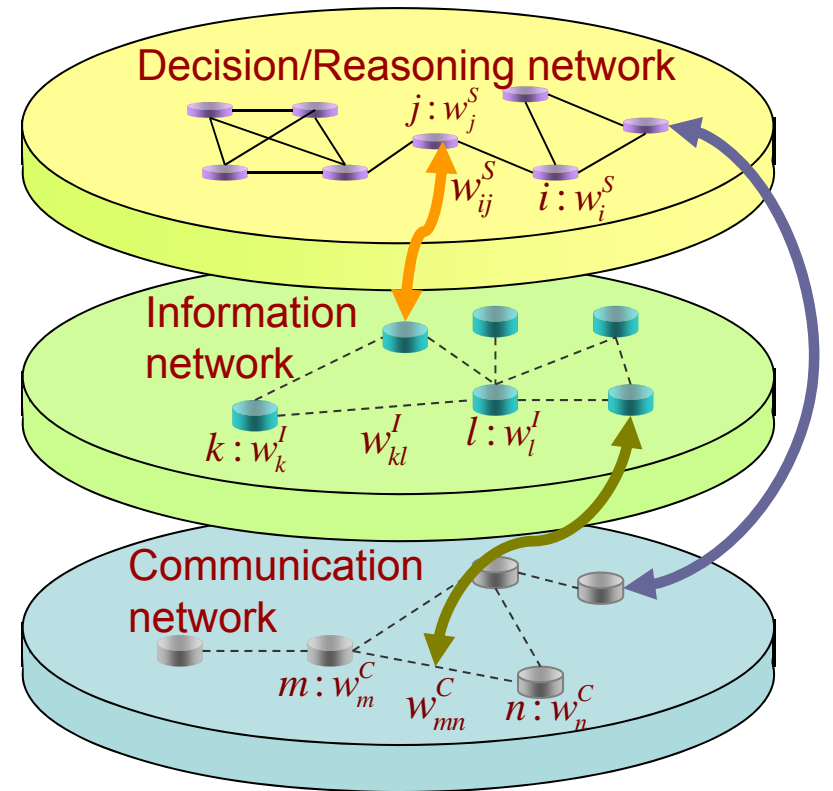


Modal Interfaces: Algebraic Properties

- Implementations M , parallel composition \times
- Specifications S , satisfaction relation $M \models S$
- **Refinement** $S \leq T$ iff $\forall M, M \models S \Rightarrow M \models T$
- **Product** $S \otimes T = \min\{X \mid M \models S \text{ and } N \models T \text{ implies } M \times N \models X\}$
 - Combine specifications related to distinct components
- **Conjunction** $M \models S \wedge T$ iff $M \models S$ and $M \models T$
 - Combine specifications related to the same component
- **Residuation** $S/T = \max\{X \mid X \otimes T \leq S\}$
 - Component reuse: $C' = C/D$
 - Assume/Guarantee Reasoning $C = (A, G) = (G \otimes A)/A$

Collaborating Agents: Multiple Coevolving Multigraphs

- Multiple Interacting Multigraphs
 - **Nodes**: agents, individuals, groups, organizations
 - Directed graphs
 - **Links**: ties, relationships
 - **Weights on links** : value (strength, significance) of tie
 - **Weights on nodes** : importance of node (agent)
- **Value directed graphs with weighted nodes**
- **Real-life problems: Dynamic, time varying graphs, relations, weights, policies**



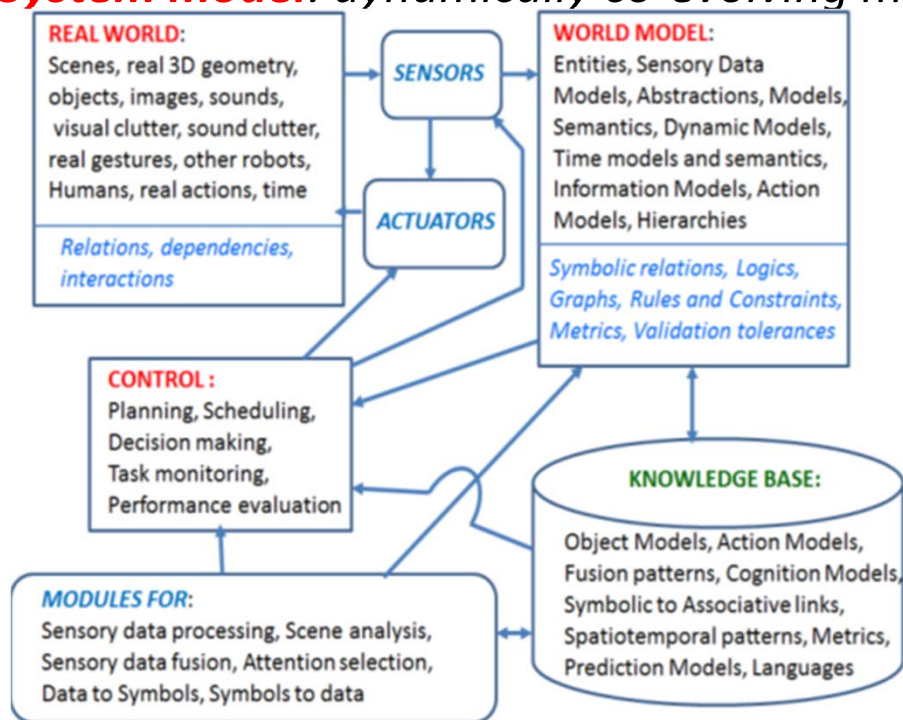
**Networked System
architecture & operation**



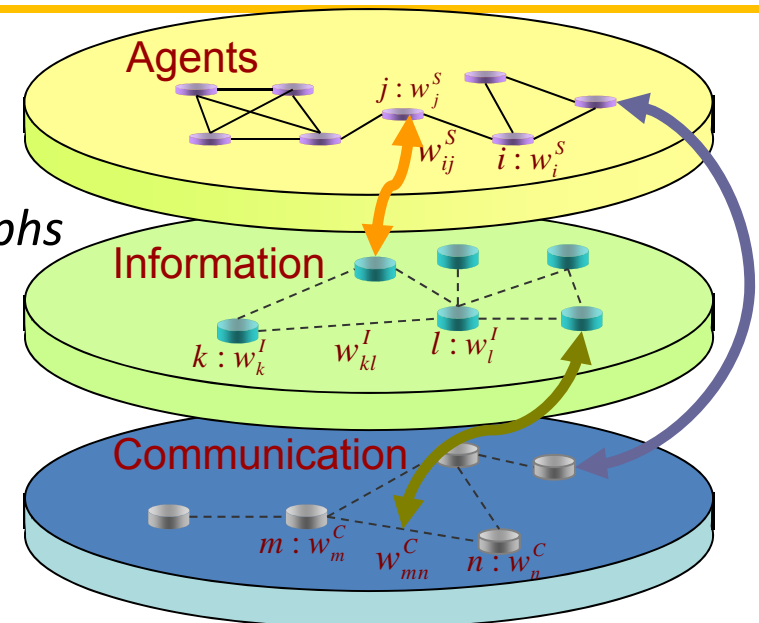
Model for Collaborative Autonomy

Collaborating agents architecture

- *Nodes and links annotated by weights* or rules
- Annotations are associated across layers,
- **System model**: dynamically co-evolving multigraphs



Each node carries this real vs world model framework



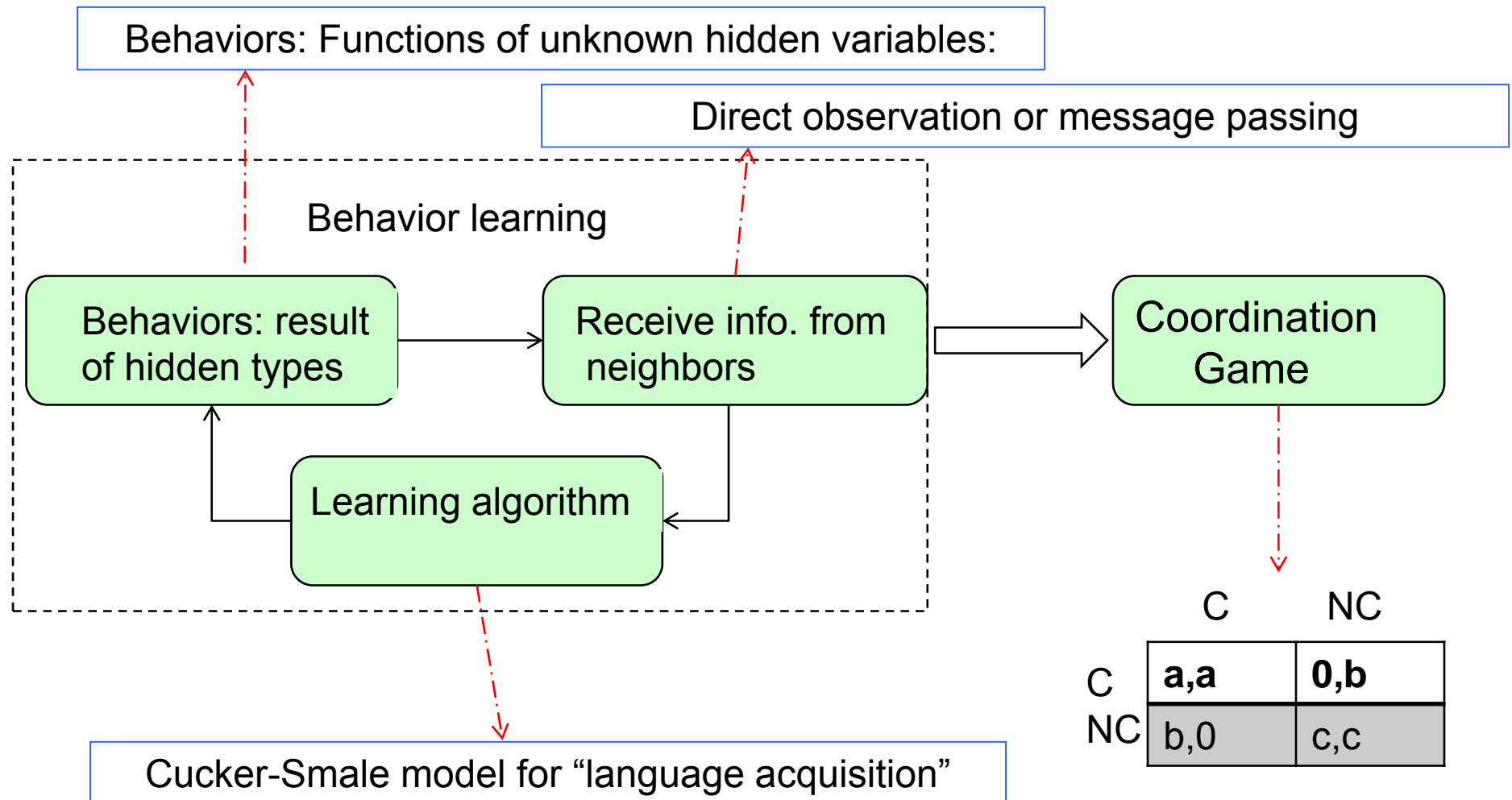
Task-driven integration of perception, control, language

- Cognitive dialogue
- Dynamic attention mechanism
- Manipulation grammar
- Three-layer architecture

Autonomous Decision Making: Learning to Collaborate

- When making a decision, an agent is influenced by its knowledge about the other agents' behavior
- **Problem:** *Modeling decision making on whether to cooperate in a group effort as a result of two person games on a network*
- Adaptation to neighbors' strategies as a **coordination mechanism**
- Classes of linear and bounded linear behavior functions; **Generalized consensus** determines strategy coordination
- *Emerging collaboration* graph: from agents' behavioral tendencies as well as the communication graph

Learning to Collaborate



$$a > b > c > 0.$$

- Developing a framework for synthesis of intelligent autonomous systems with self-managed safety
- Composability
- Self-monitoring, Correctness
- Initial demonstrations in UAV, robotic, autonomous ground vehicle examples
- Continue development and integration
- Looking for unmanned ship and underwater vehicle applications
- Incorporate advanced learning, safe learning, sensor fusion
- Link to advanced simulators and on-line data input

1. J.S. Baras and G. Theodorakopoulos, *Path Problems in Networks*, Synthesis Lectures on Communication Networks, Morgan & Claypool Publishers, February 2010.
2. G. Theodorakopoulos and J. S. Baras, “On Trust Models and Trust Evaluation Metrics for Ad-Hoc Networks”, *Journal of Selected Areas in Communications, Security in Wireless Ad-Hoc Networks*, Vol. 24, Number 2, pp. 318-328, February 2006. [2007, IEEE Communications Society Leonard G. Abraham Prize]
3. G. Theodorakopoulos and J. S. Baras, “Linear Iterations on Ordered Semirings for Trust Metric Computation and Attack Resiliency Evaluation”, *Proc. 17th International Symposium on Mathematical Networks and Systems*, pp. 509-514, Kyoto, Japan, July 24-28, 2006.
4. K.K. Somasundaram and J.S. Baras, “Performance Improvements in Distributed Estimation and Fusion Induced by a Trusted Core”, *Proceedings of the 12th International Conference on Information Fusion-Fusion 2009*, pp.1942-1949, Seattle, Washington, USA, July 6-9, 2009.

References

5. I. Matei, T. Jiang and J. S. Baras, A Trust Based Distributed Kalman Filtering Approach for Mode Estimation in Power Systems, *Proceeding of the First Workshop on Secure Control Systems (SCS) as part of CPSWeek 2010*, pp. 1-6, Stockholm, Sweden, April 12, 2010.
6. K. Somasundaram and J. S. Baras, “Solving Multi-metric Network Problems: An Interplay Between Idempotent Semiring Rules”, *LAA Special Issue on the occasion of 1st Montreal Workshop on Idempotent and Tropical Mathematics*, Volume 435, Issue 7, pp. 1494–1512, 1 October 2011.
7. K. K. Somasundaram and J. S. Baras, “Semiring Pruning for Information Dissemination in Mobile Ad Hoc Networks”, *Proceedings of The First International Conference on Networks & Communications (NetCoM - 2009)*, pp. 319 – 325, Chennai, India, December 27-29, 2009.
8. K. Somasundaram and J. S. Baras, “Path Optimization and Trusted Routing in MANET: An Interplay Between Ordered Semirings”, *Proceedings of The Second International Conference on Networks & Communications (NetCoM - 2010)*, pp. 88-98, Chennai, India, December 27-29, 2010.

References

9. K. Somasundaram, J. S. Baras, K. Jain and V. Tabatabaee , “Distributed Topology Control for Stable Path Routing in Multi-hop Wireless Networks”, *Proceedings 49th IEEE Conference on Decision and Control (CDC 2010)*, pp. 2342-2347, Atlanta, Georgia, December 15-17, 2010.
10. E. Paraskevas, T. Jiang, P. Purkayastha and J. S. Baras, "Trust-Aware Network Utility Optimization in Multihop Wireless Networks with Delay Constraints" , *Proceedings of the 24th Mediterranean Conference on Control and Automation*, pp. 593-598, Athens, Greece, June 21-24, 2016.
11. Y. Zhou, J. S. Baras, S. Wang, "Hardware Software Co-design for Automotive CPS using Architecture Analysis and Design Language", *Proceedings of the 5th Analytic Virtual Integration of Cyber-Physical Systems Workshop (AVICPS 2014)*, Rome, Italy, December 2, 2014.

Thank you!

baras@isr.umd.edu

301-405-6606

<http://dev-baras.pantheonsite.io/>

Questions?