

Structured Admission Control in Heterogeneous Wireless Networks with Mesh Underlay

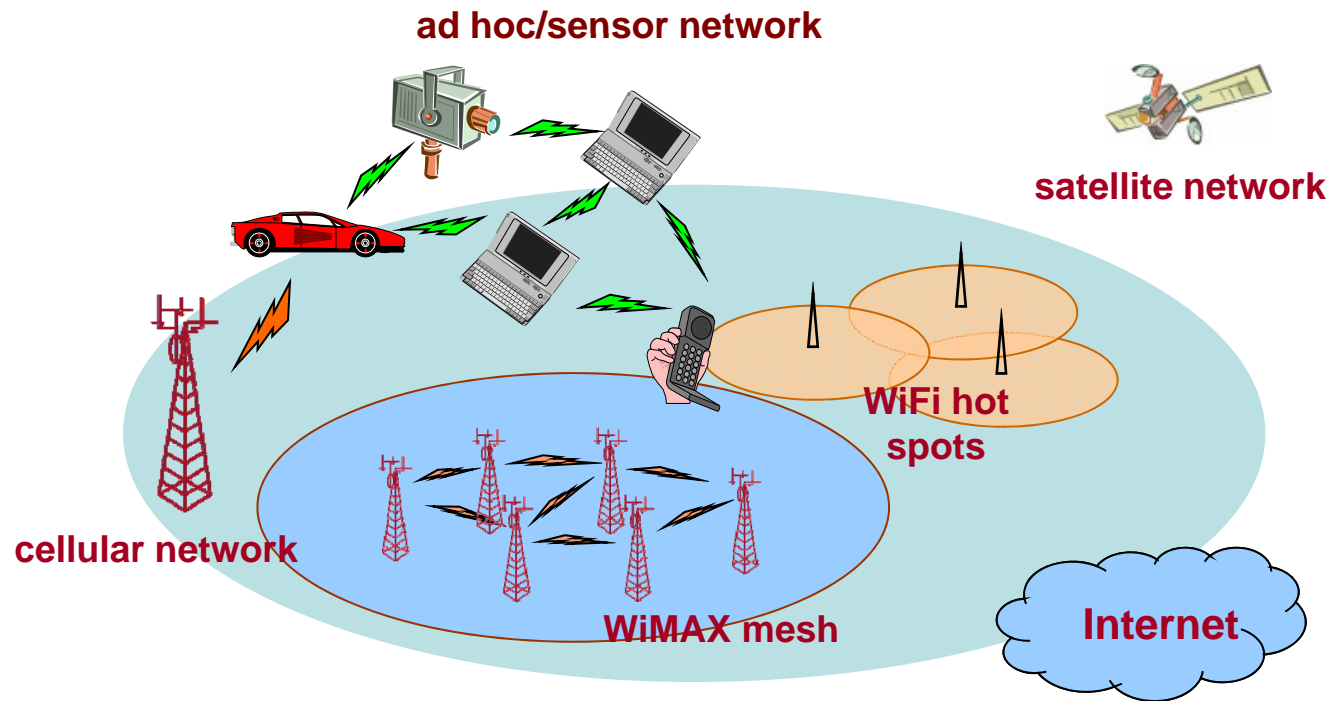
Amin Farbod* and Ben Liang
{afarbod, liang}@comm.utoronto.ca
Dept. Electrical and Computer Engineering
University of Toronto



Research supported in part by Bell Canada through the Bell University Labs program

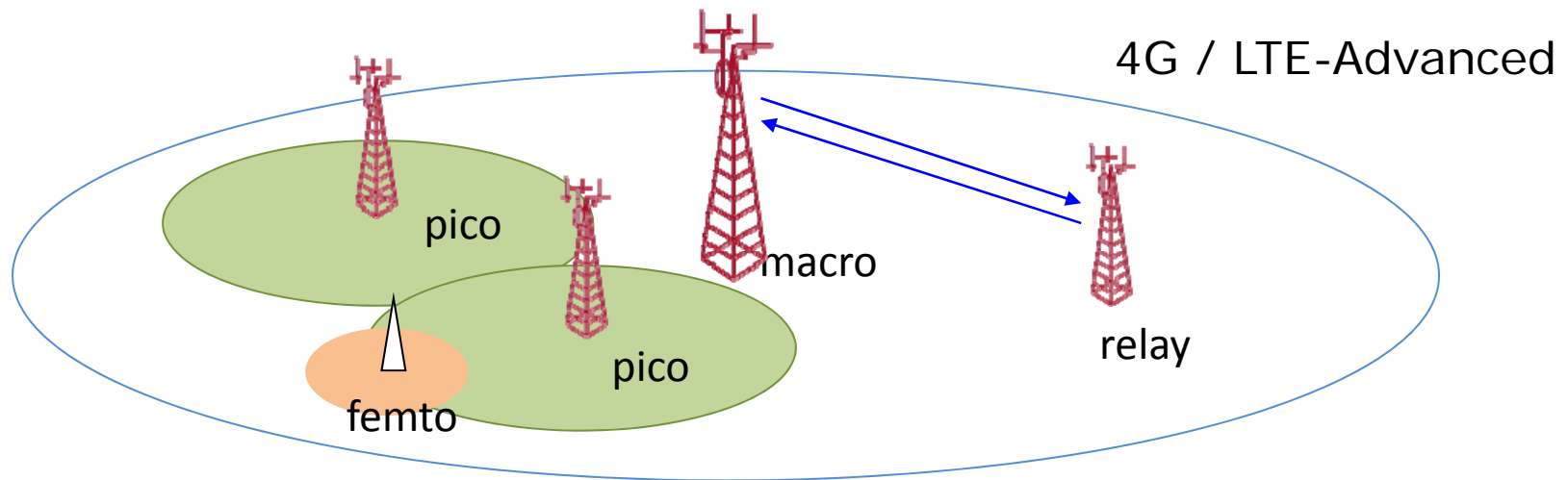
* Available for research job/internship opportunities!

Heterogeneous Wireless Networking



- Multiple co-existing wireless access technologies

Heterogeneous Wireless Networking



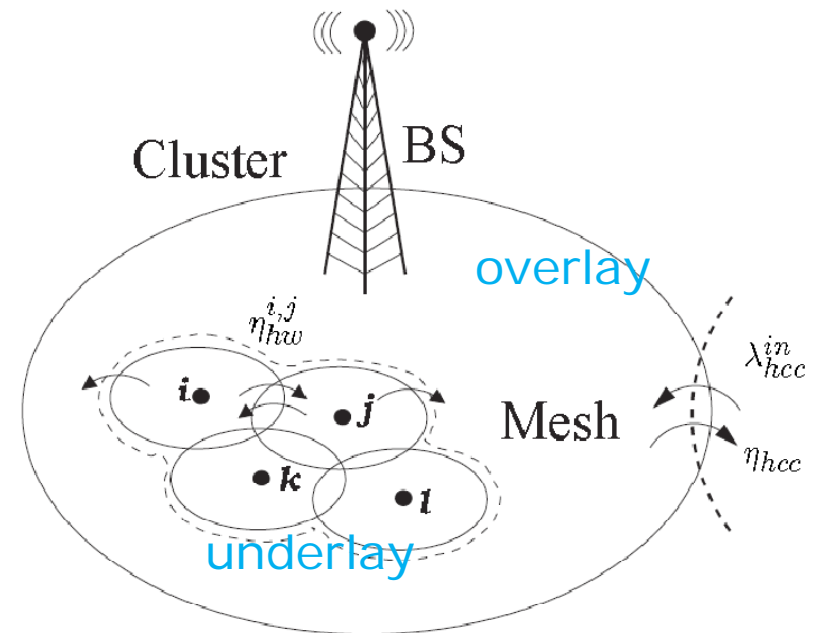
- Pro:
 - universal coverage AND high-bandwidth access where available
 - offset traffic load in more expensive network
- Con: higher design complexity

Design Challenges

- Integration protocols
 - Tight coupling vs. loose coupling
 - Interference avoidance
- Mobility management
 - Mobility modeling
 - Dependency between sub-networks
 - Vertical handoff – between different technologies
 - Smaller cells → more frequent handoffs
- Resource allocation
 - Spectrum and content management
 - Call admission control (CAC)

Call Admission Control in HWN

- CAC maintains QoS for streaming sessions
- Difficult in HWNs
 - Complexity
 - Up-going handoffs
 - Require special accommodation
 - Resources more limited at higher layer
 - Dimension of state space



Call Admission Control in HWN

- General approaches
 - Heuristics
 - Decision theoretic control
 - Systematic
 - Optimal performance
 - Curse of dimensionality
 - Markovian assumptions
- This work:
 - Based on dynamic programming
 - Complexity reduction through observation of special structure in optimal CAC policy

Related Work

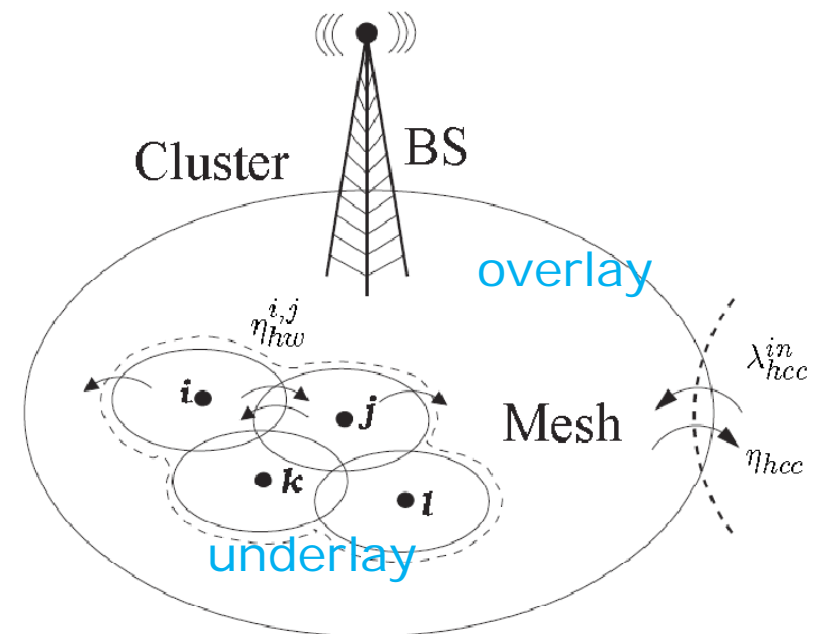
- Homogeneous systems:
 - [Ramjee et al, WINET 1997] Showed optimal CAC for a single cellular base station has threshold structure
 - [Altman et al, TComm 2001] Dynamic programming to formulate CAC of multiple classes in resource-sharing systems
 - [Ni et al, Tcomm 2007] Fast reservation and threshold policies for resource-sharing systems

Related Work

- Heterogeneous wireless networks:
 - [Song et al, ICBN'05] Admission region for voice and data services using heuristics
 - [Navarro et al, Globecom'07] Linear programming formulation for guard channel optimization
 - [Farbod et al, MONET 2007] Two-dimensional threshold structure results for optimal CAC with a single underlay access point

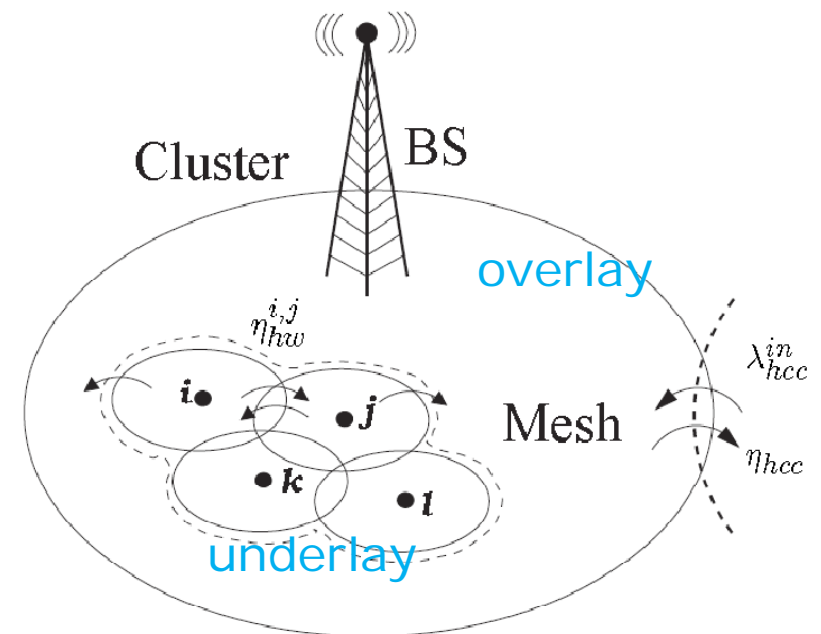
Network Model

- Admission decision made independently in a *cluster*
 - one 3G cell + M mesh APs
- Mesh APs have different traffic characteristics
 - cannot consider separately
- Memoryless
 - external arrivals
 - service times
 - mobility transitions



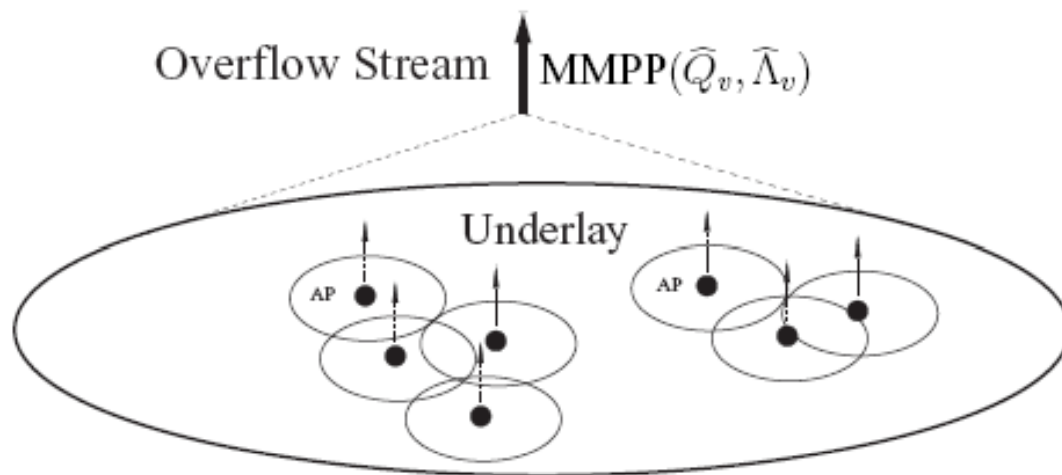
Network Model

- Data services best-effort (ignored)
- Each multimedia stream requires one Basic Bandwidth Unit (BBU)
 - Overlay capacity: C_c BBUs
 - Underlay capacity: C_w BBUs each AP
- When overlay cell is at full capacity:
 - New calls blocked
 - Handoff calls dropped



Network Model

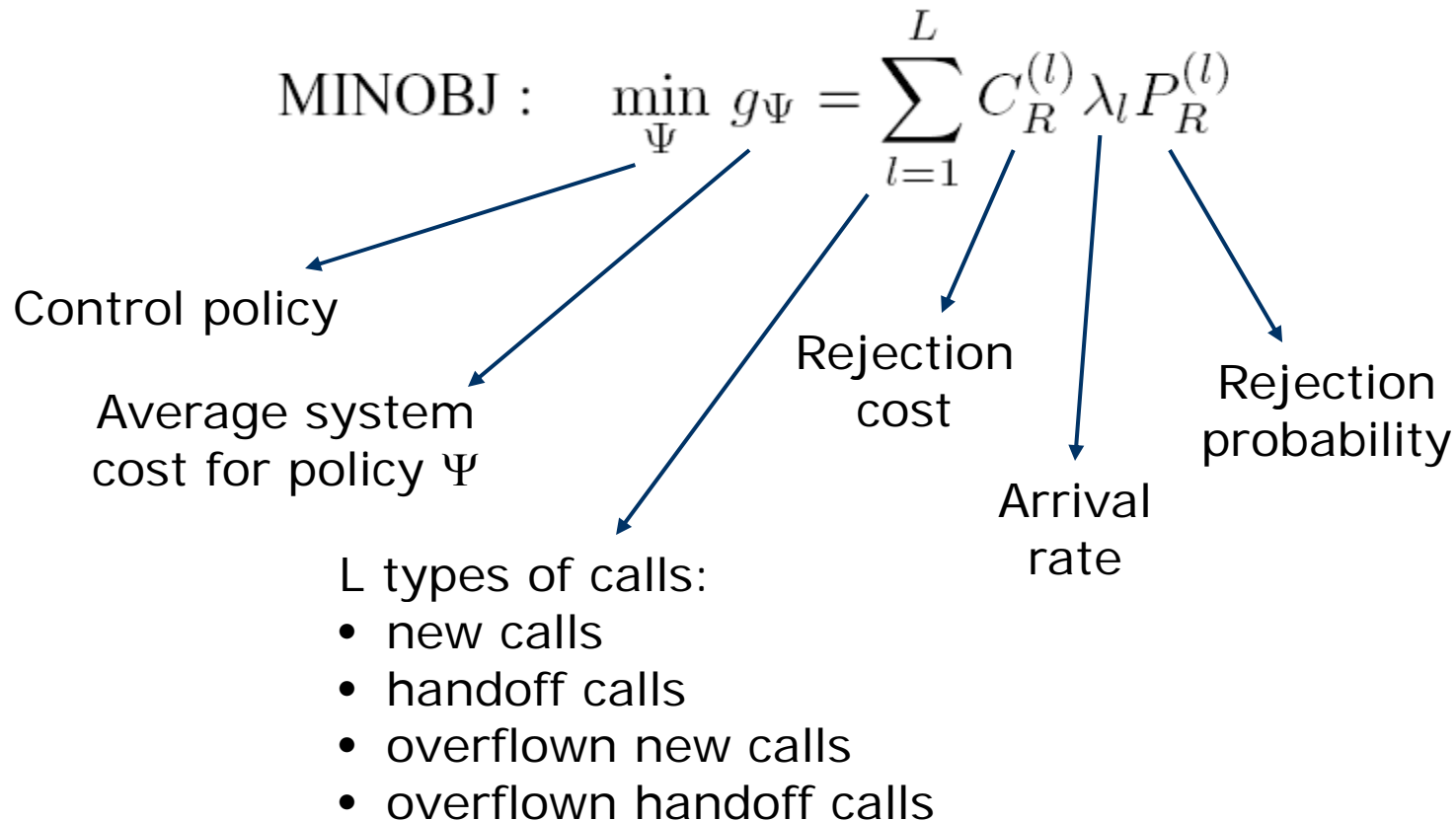
- Three events trigger the overflow of a call
 - an active call associated with an AP leaves the double-coverage area
 - a new call arrives to a blocking AP and
 - a handover call comes to an AP that drops it



Only the first type of overflow events are Poisson

Optimization Objective

- Minimize average cost of call rejections **at overlay**



Markov Decision Process (MDP)

- Markov chain with set of states: $s \in S$
- Set of possible actions: $a \in A$
- Cost function for action a in state s : $c_s(a)$
- Probability of state transition from s to t given action a : $P_{st}(a)$
- Policy $\pi: S \rightarrow A$

- Goal: Optimize π to minimize cost
- Bellman equation

$$V_n(s) = \min_{a \in A(s)} \left\{ c_s(a) + \sum_{t \in S} P_{st}(a) V_{n-1}(t) \right\}.$$

Solution Approach

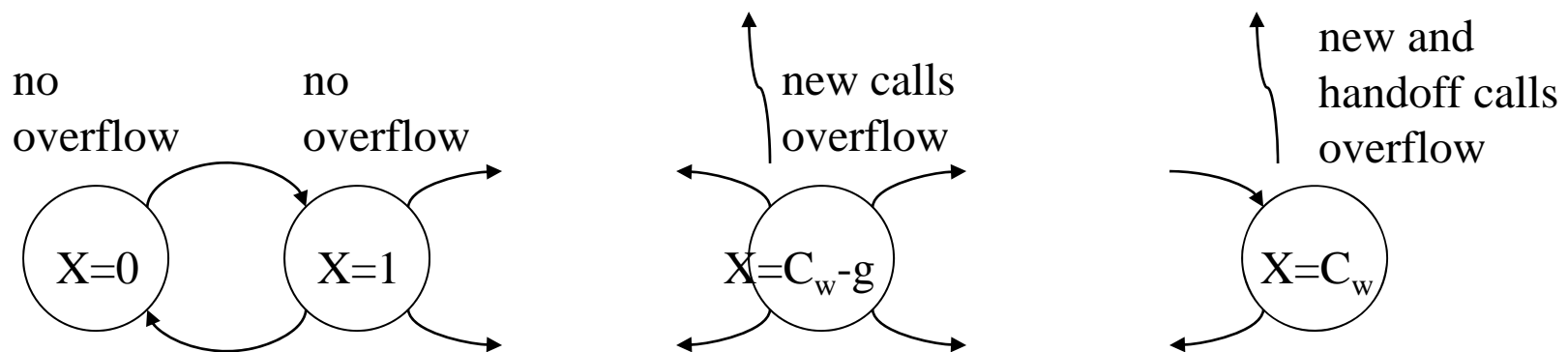
- Characterize overflow process
- Establish structural results
- Design efficient computation algorithm

Solution Approach

- Characterize overflow process
- Establish structural results
- Design efficient computation algorithm

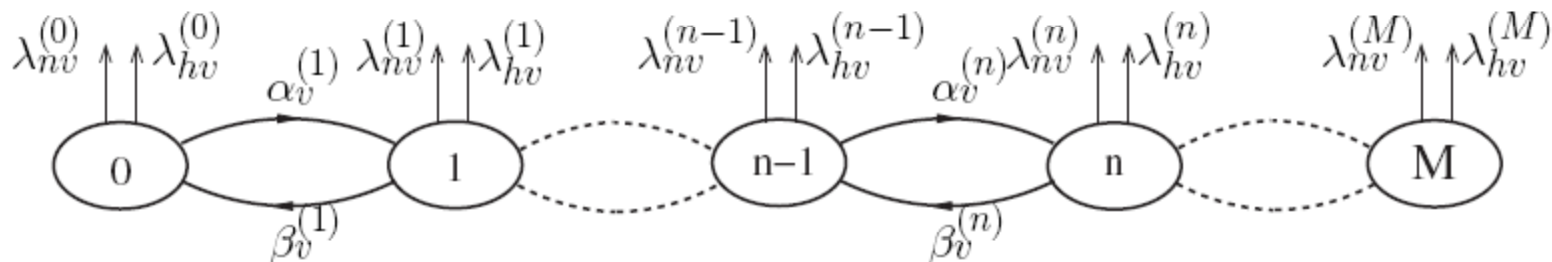
Overflow Modeling

- Each AP's channel occupancy modeled as a Markov process
 - Assume absence of loops in mobility pattern
- Then overflow traffic from each AP is a Markov Modulated Poisson Process (MMPP)
 - Time-varying rate determined by CAC policy at AP
 - E.g., guard-channel policy



Overflow Modeling

- Challenge: M APs $\rightarrow C_w^M$ possible states to keep track
 - Unacceptable communication overhead in practice
 - Intractable state space in analysis
- Solution: consider total overflow from mesh underlay as a PO-MMPP traffic source into overlay cell
 - Only observable underlay status is # of busy APs
 - $(M + 1)$ -state MMPP source:



Dynamic Programming Formulation

- System state $s = (i, n)$
 - i : number of calls at overlay
 - n : number of busy APs at underlay
- Four controllable events
 - new call arrivals to overlay
 - new call overflows from underlay
 - handover call arrivals to overlay from neighbor clusters
 - handover call overflows from underlay to overlay
- Control actions: **admit** or **reject**

Dynamic Programming Formulation

- State operators:

$$\mathcal{A}s : s = (i, n) \rightarrow s' = (i + 1, n)$$

$$\mathcal{D}s : s = (i, n) \rightarrow s' = (i - 1, n)$$

$$\mathcal{Q}s : s = (i, n) \rightarrow s' = (i, n + 1)$$

$$\mathcal{R}s : s = (i, n) \rightarrow s' = (i, n - 1)$$

- Bellman equation:

rate of event cost of operator cost to reject

$$V_{k+1}(s) = \frac{1}{v_{max}} \left\{ \lambda_{hcc}^{in} \min[\Delta V_k(\mathcal{A}s), C_{DCC}] \right. \\
 + \lambda_{hv}^{(n)} \min[\Delta V_k(\mathcal{A}s), C_{DHO}] \\
 + \lambda_{nv}^{(n)} \min[\Delta V_k(\mathcal{A}s), C_{DNO}] \\
 + \lambda_{nc} \min[\Delta V_k(\mathcal{A}s), C_{BNC}] \\
 + \alpha_v^{(n)} \Delta V_k(\mathcal{Q}s) + \beta_v^{(n)} \Delta V_k(\mathcal{R}s) \\
 \left. + i(\mu_c + \eta_{hcc}) \Delta V_k(\mathcal{D}s) + v_{max} V_k(s) \right\}$$

uniformization

control decisions

underlay status changes

uniformization

departures

Solution Approach

- Characterize overflow process
- Establish structural results
- Design efficient computation algorithm

Structure of Optimal Policy

- Properties of $V_k(s)$:

$$A) \quad \Delta V_k(\mathcal{A}s) \geq 0$$

$$B) \quad \Delta V_k(\mathcal{A}^2s) \geq \Delta V_k(\mathcal{A}s)$$

$$C) \quad \Delta V_k(\mathcal{A}Qs) \geq \Delta V_k(\mathcal{A}s)$$

A) Admitting a call increases cost

B) $V_k(s)$ is convex in the number of calls admitted

C) $V_k(i+1, n+1) - V_k(i, n+1) \geq V_k(i+1, n) - V_k(i, n)$

- Proof: divide-and-conquer over Bellman equation

Optimality of Threshold Policy

- Observation #1: The optimal control policy is threshold-based.

Threshold policy: admit a call of class l_1 if and only if system state (# of calls admitted) is less than threshold for class l_1

$$i \leq \vec{T}[l_1]$$

- Proof: convexity of $V_k(s)$

Ordered Monotonic Threshold Policy

- Observation #2: For a given class of calls, the optimal threshold is decreasing in the number of busy APs.

$$\text{Monotonic : } \vec{T}(l, n) \leq \vec{T}(l, n - 1)$$

- Observation #3: For a given number of busy APs, the optimal threshold is increasing in the rejection cost of call classes.

(Sort call classes by increasing rejection cost)

$$\text{Ordered : } \vec{T}(l, n) \leq \vec{T}(l + 1, n)$$

- The optimal CAC policy is an **Ordered Monotonic Threshold Policy (OMTP)**

Solution Approach

- Characterize overflow process
- Establish structural results
- Design efficient computation algorithm

Structured Coordinate Search Algorithm (SCSA)

- Standard iterative solutions to dynamic programming:
 - Policy Iteration (PI)
 - Value Iteration (VI)
 - Main idea: iteratively improve policy for each state assuming previous policy is applied in subsequent states
 - Both guarantee convergence to optimal solution
 - Neither can handle large problems
- Observations 1 – 3 suggest structure of optimal policy for efficient solution
 - SCSA: maintains OMTP in every iteration

Algorithm 1 Structured Coordinate Search Algorithm (SCSA)

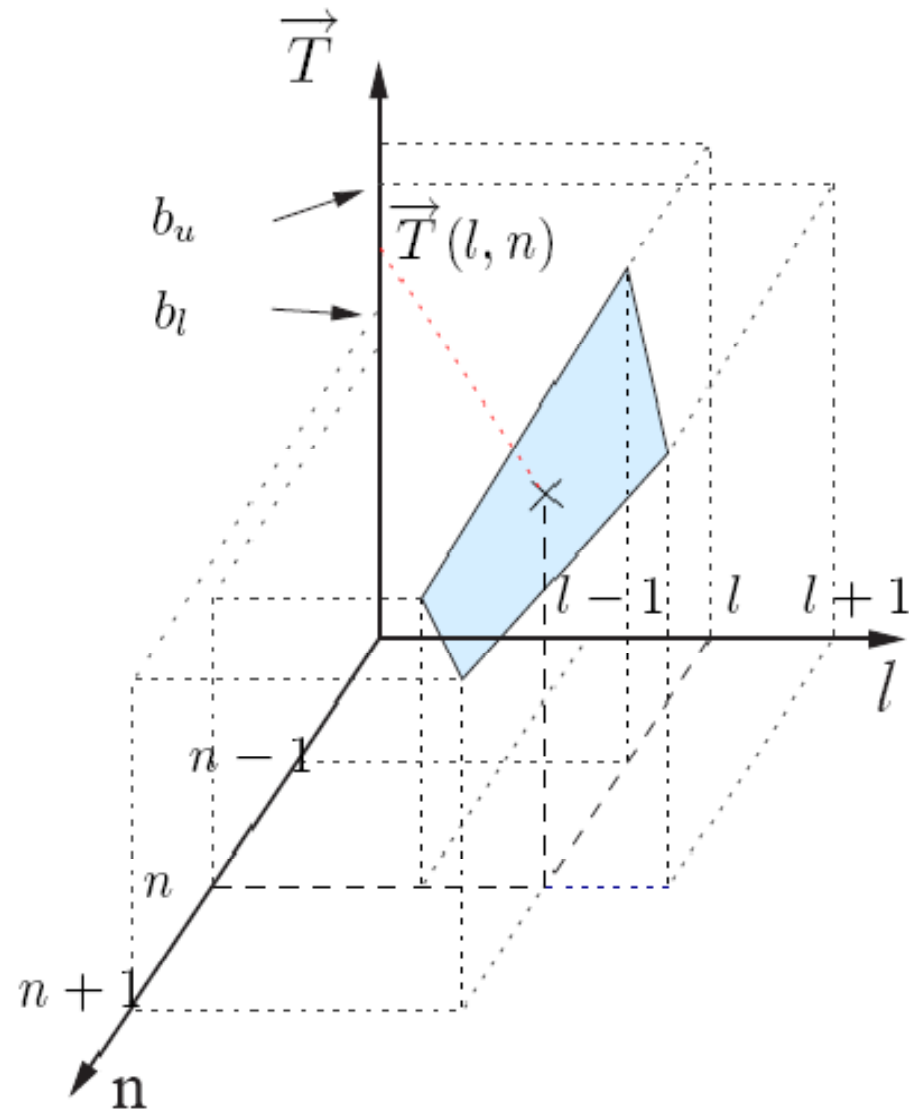
1: **Initialize** Ψ_0 such that: $\forall(l, n) \quad \vec{T}_{\Psi_0}(l, n) = C_c$
2: $r := 0$
3: $r = r + 1, \Psi_r = \Psi_{r-1}$
4: **for** $l := 1, \dots, L$
5: **for** $n := 0, \dots, M$
6: $b_l = \max [\vec{T}_{\Psi_r}(l-1, n), \vec{T}_{\Psi_r}(l, n+1)]$
7: $b_u = \min [\vec{T}_{\Psi_r}(l+1, n), \vec{T}_{\Psi_r}(l, n-1)]$
8: $\vec{T}_{\Psi_r}(l, n) = \underset{b_l \leq t \leq b_u}{\operatorname{argmin}} g_{\Psi_r}[\vec{T}(l, n) = t]$
9: **if** $g_{\Psi_r} = g_{\Psi_{r-1}}$ **then**
10: Return Policy Ψ_r
11: **else**
12: Go to step 3
13: **end if**

Initialize with Complete Sharing policy

Search within OMTP boundaries

Terminate if policy cannot be improved

Structured Coordinate Search Algorithm (SCSA)



SCSA Convergence

- Fact: SCSA converges to an OMTP.
 - SCSA maintains OMTP property in every iteration
 - SCSA improves policy in every iteration and terminates when policy cannot be improved
 - There is a finite number of policies

SCSA Optimality

- Does SCSA produce an optimal policy?
- SCSA searches for an optimal policy using
 - not the structure of cost function $g_{\Psi_r}(\vec{T})$
 - but the structure of the input that give minimum cost
- We know $V_k(s)$ is convex, but no such observation on $g_{\Psi_r}(\vec{T})$
- Extensive simulation results suggest that SCSA is optimal
 - Conjecture: $g_{\Psi_r}(\vec{T})$ has no local minimum

What If Guaranteed Optimality is Desired?

- SCSA-OPT: feed the result of SCSA to a policy improvement step similar to Value Iteration

Algorithm 2 Policy Improvement (PLI)

- 1: **Input:** OMTP Policy Ψ_r .
 - 2: Calculate relative values $v(s)$ for Ψ_r . 1st half of Policy Iteration
 - 3: Assign $V_k(s) = k g_{\Psi_r} + v(s)$ (for an arbitrary k).
 - 4: Use (22) to find Ψ_{r+1} . Value Iteration via Bellman Equation
 - 5: Return Ψ_{r+1} .
-

- OMTP input \rightarrow convex $v(s)$
 \rightarrow convex $V_k(s) \rightarrow$ OMTP output

SCSA-OPT

Algorithm 3 Optimal Structured Coordinate Search Algorithm (SCSA-OPT)

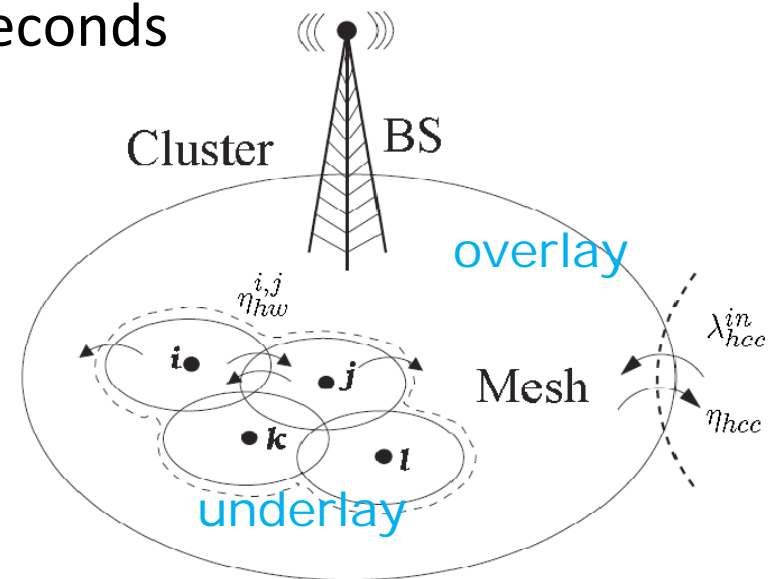
```
1: Initialize  $\Psi_0$  such that:  $\forall(l, n) \quad \vec{T}_{\Psi_0}(l, n) = C_c$ 
2:  $r := 0$ 
3:  $r = r + 1, \Psi_r = \Psi_{r-1}$ 
4: for  $l := 1, \dots, L$ 
5:   for  $n := 0, \dots, M$ 
6:      $b_l = \max [\vec{T}_{\Psi_r}(l-1, n), \vec{T}_{\Psi_r}(l, n+1)]$ 
7:      $b_u = \min [\vec{T}_{\Psi_r}(l+1, n), \vec{T}_{\Psi_r}(l, n-1)]$ 
8:      $\vec{T}_{\Psi_r}(l, n) = \operatorname{argmin}_{b_l \leq t \leq b_u} g_{\Psi_r}[\vec{T}(l, n) = t]$ 
9:   if  $g_{\Psi_r} = g_{\Psi_{r-1}}$  then
10:     [if  $\Psi_r = \text{PLI}(\Psi_r)$  then]
11:       Return Policy  $\Psi_r$ 
12:     else  $\Psi_r := \text{PLI}(\Psi_r)$ 
13:   end if
14: Go to step 3
```

- SCSA-OPT terminates when the result of SCSA cannot be improved by PLI step
- Same termination condition as VI
- Hence SCSA-OPT converges to an optimal policy

Numerical and Simulaton Setup

- Two stages:
 1. Compute policy in Matlab
 2. Apply policy in discrete event simulation
- Simulated system time: 200,000 seconds

Parameter	Value	Parameter	Value
C_c	50 calls	μ_c	0.01
C_w	20 calls	μ_w	0.01
λ_c	0.4 calls/sec	C_{BNC}	5
λ_w	0.2 calls/sec	C_{BNO}	10
η_{hec}	5×10^{-3}	C_{DCC}	30
η_h^i	0.01	C_{DHO}	40



Convergence Time

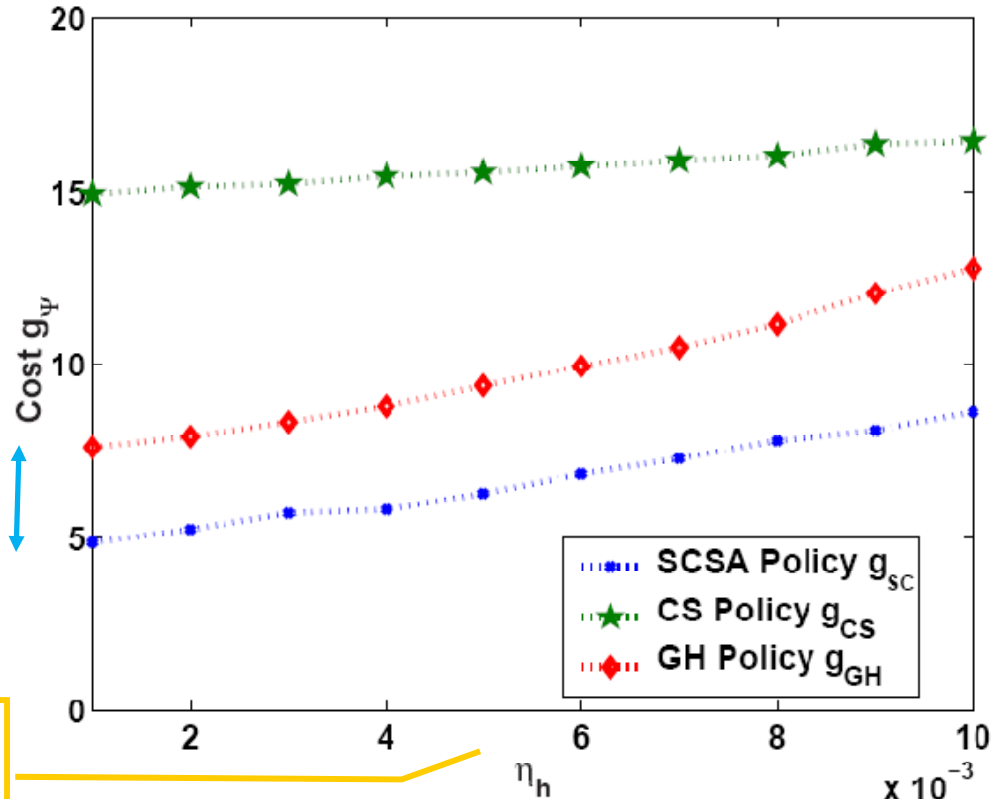
	SCSA	SCSA-OPT	VI	ES
M	Time (sec)	Time (sec)	Time (sec)	Order
4	12	12	483	2^{200}
8	59	59	5685	2^{400}
16	307	309	$\gg 5685$	2^{800}

- Matlab CPU time
- Extra time in SCSA-OPT is only on confirming optimality

Cost Comparison

Performance gain when partial observation of mesh status is integrated into the control algorithm.

Mobility: 100 ~ 1000 s residence time in WLAN



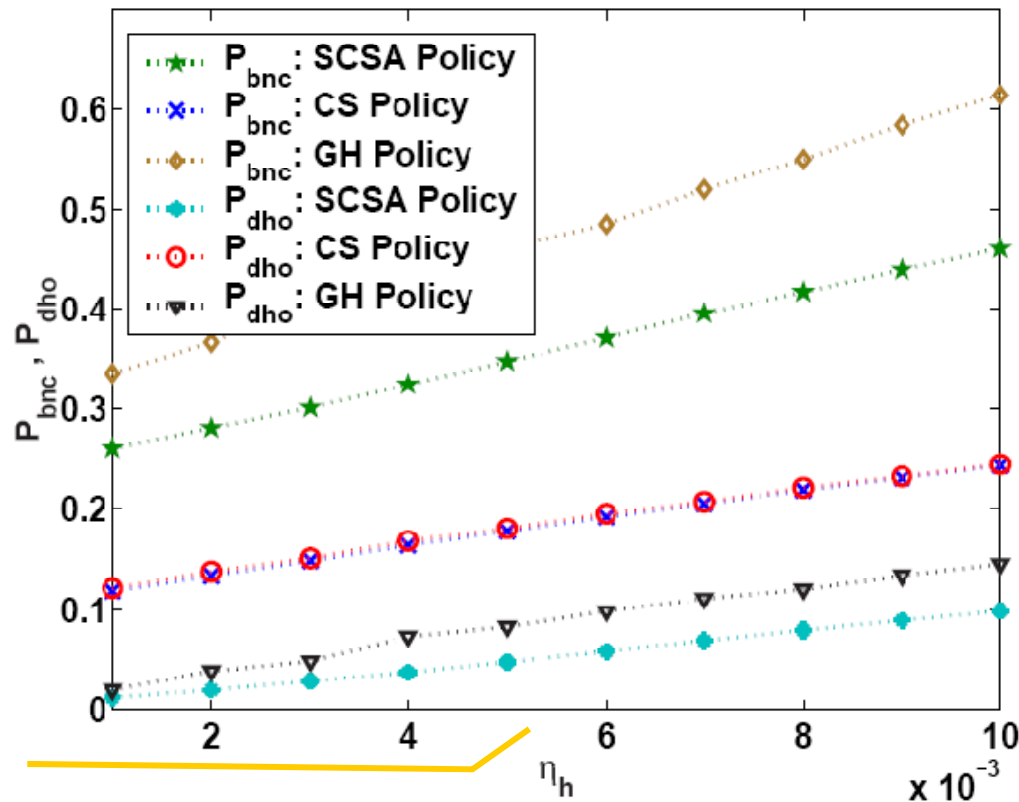
CS = Complete-Sharing Policy, GH= Optimal Guard-Channel Policy
 GH = simple Poisson modeling

Dropping and Blocking Rates

SCSA reduces both dropping and blocking rates over GH

Mobility: 100 ~ 1000 s residence time in WLAN

CS = Complete-Sharing Policy, GH= Optimal Guard-Channel Policy
 GH = simple Poisson modeling



Conclusions

- The PO-MMPP model captures
 - burstiness of overflow traffic
 - imperfect observability of mesh underlay states
- DP based on this model is structured
- SCSA/SCSA-OPT
 - Efficient vs. Value Iteration
 - Effective vs. Complete Sharing and Guard Channel
- On-going work:
 - Is SCSA optimal?
 - Non-Markovian mobility/arrival models?