Wireless Networking Projects

Ashok K. Agrawala

Udaya Shankar

University of Maryland

MIND

The MIND Lab

The Maryland Information and Network Dynamics Lab

Participants

- Ashok Agrawala
- Udaya Shankar
- Students
  - Moustafa
  - Jeowang
  - Arun
  - Andre
  - Bao
  - …

Activities

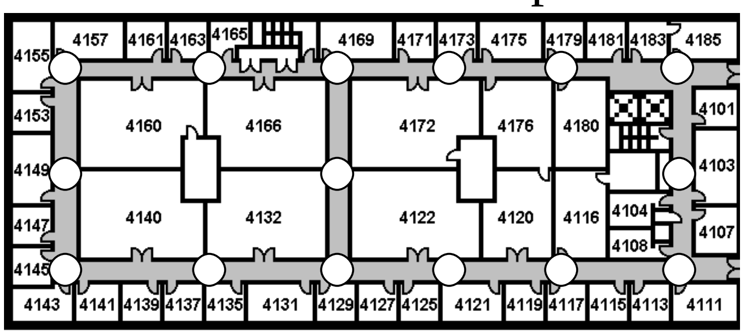
- WLAN Location Determination
- WLAN QoS Studies
- Characterization of User Behavior and Network Performance
- Z-Iteration for WLAN/WAN
- 3G Networks and Convergent Solutions

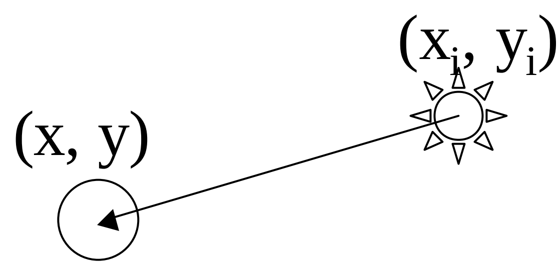
Location Determination

- Triangulate user location
  - Reference point: access point
  - Measure quantity: signal strength, time delay, …
- Signal strength= f(x, y, xi, yi)
  - Does not follow free space loss
  - Complex function of distance

Solution

- Use a lookup table
  - Radio map
  - Radio Map: f(x, y, xi, yi) for all i
    - at selected locations
- 2 phases
  - Offline phase
  - Location determination phase

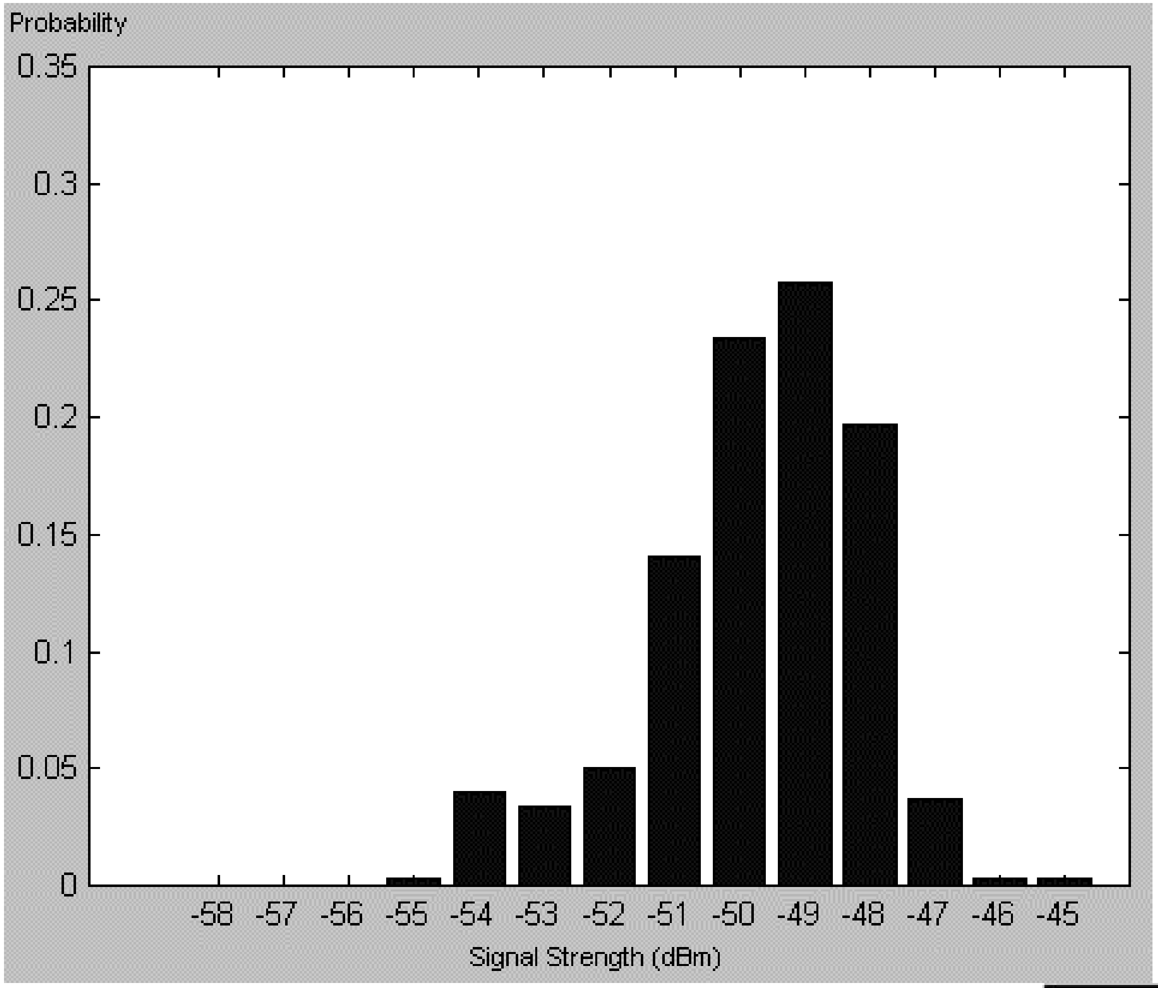




Signal Strength Characteristics

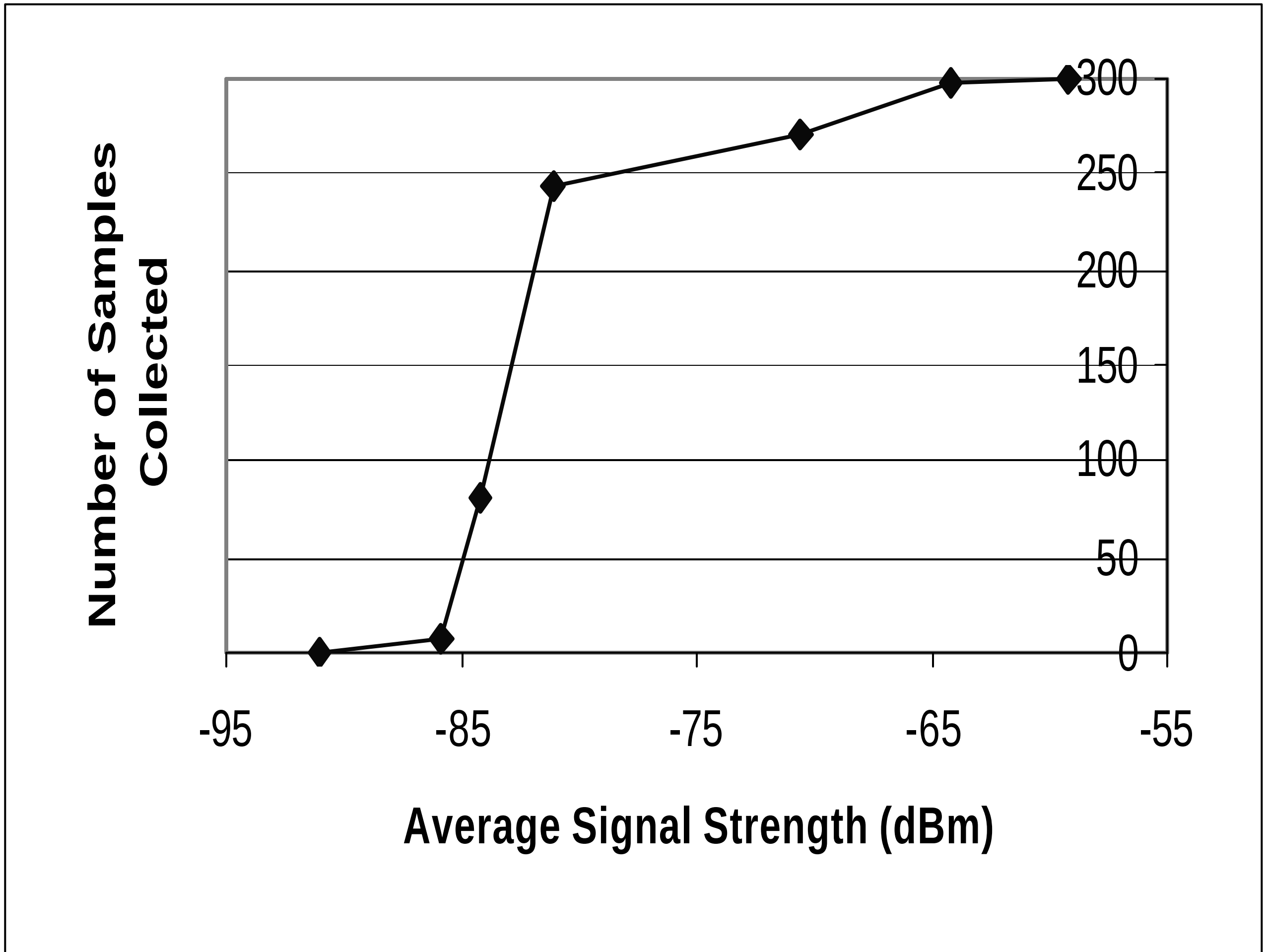
- Temporal variations
  - People movement, doors opening and closing, …
- Spatial variations
- Large scale
  - Signal attenuates with distance
  - Desired
- Small scale
  - Multi-path effect
  - Hard to capture by radio map (size/time)

Temporal Variations

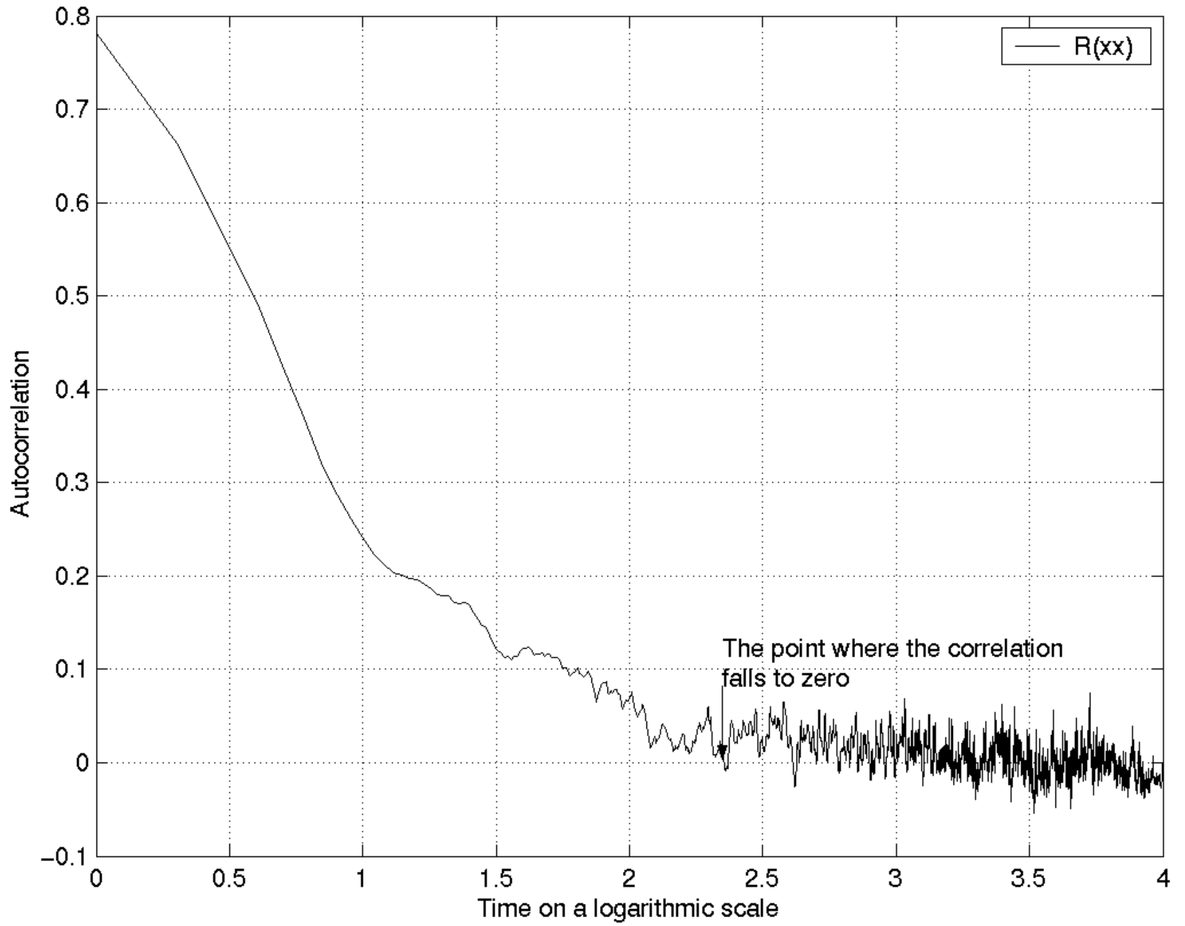


The Maryland Information and Network Dynamics Lab

Temporal Variations

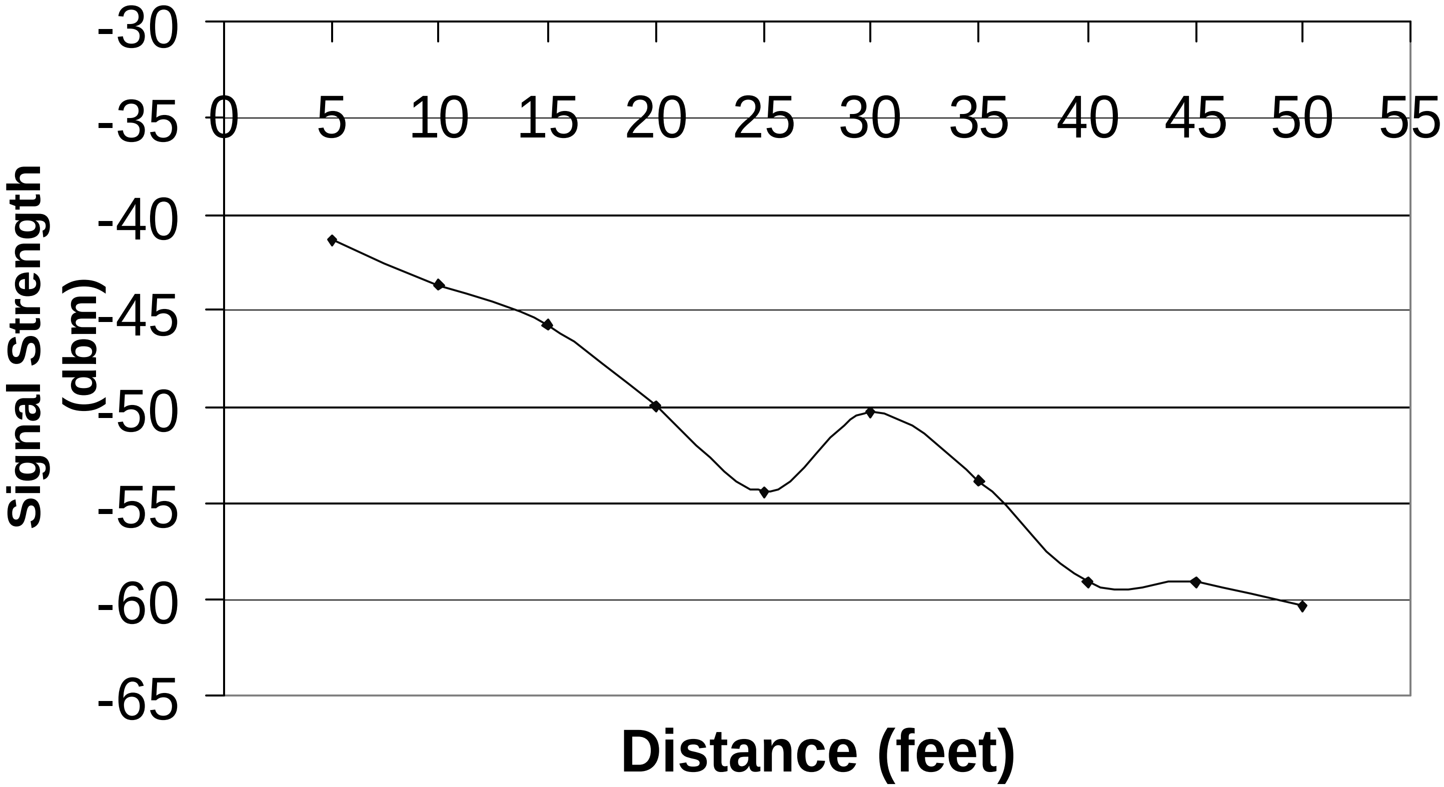


Temporal Variations: Correlation



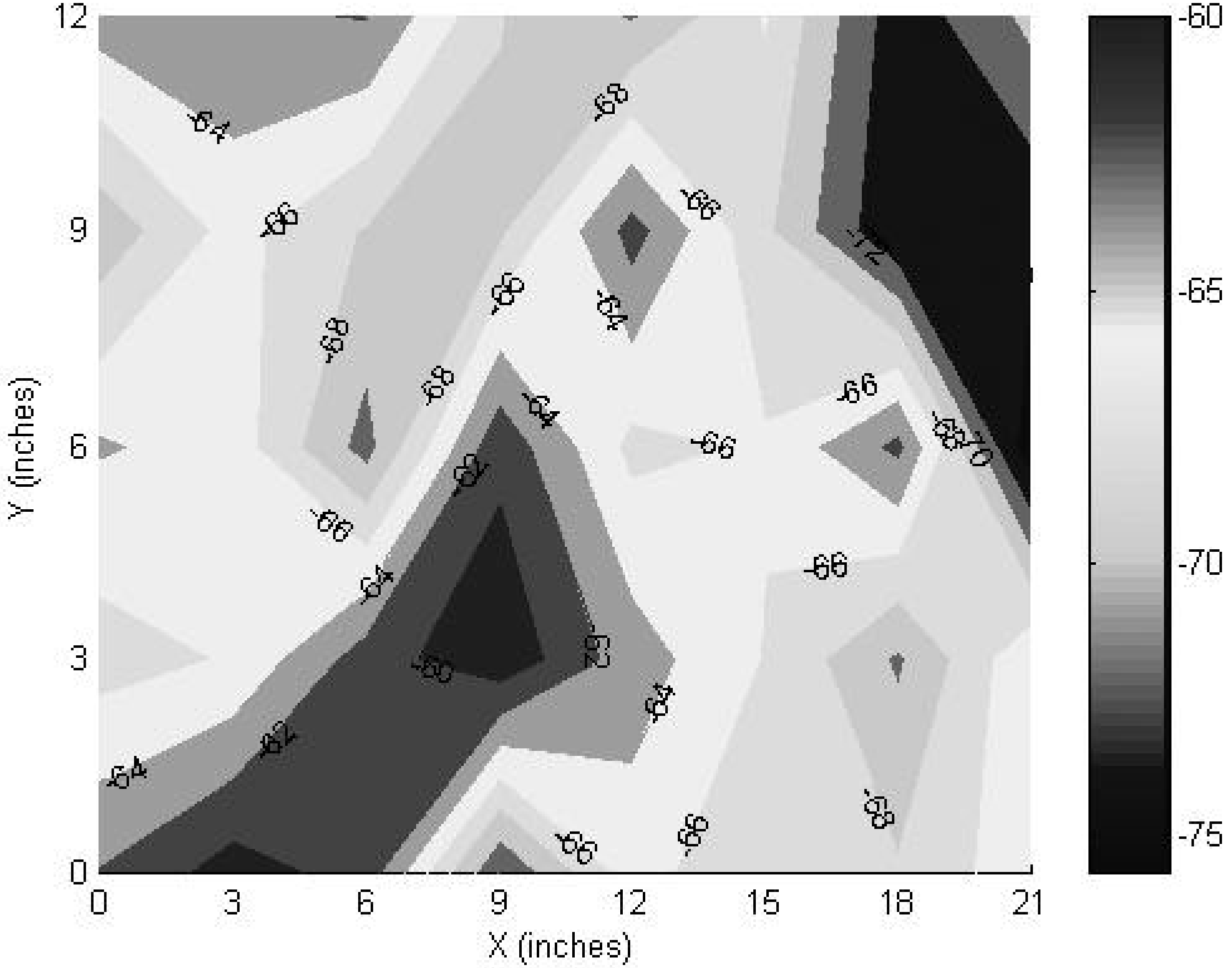
The Maryland Information and Network Dynamics Lab

Spatial Variations: Large-Scale



The Maryland Information and Network Dynamics Lab

Spatial Variations: Small-Scale



Approach

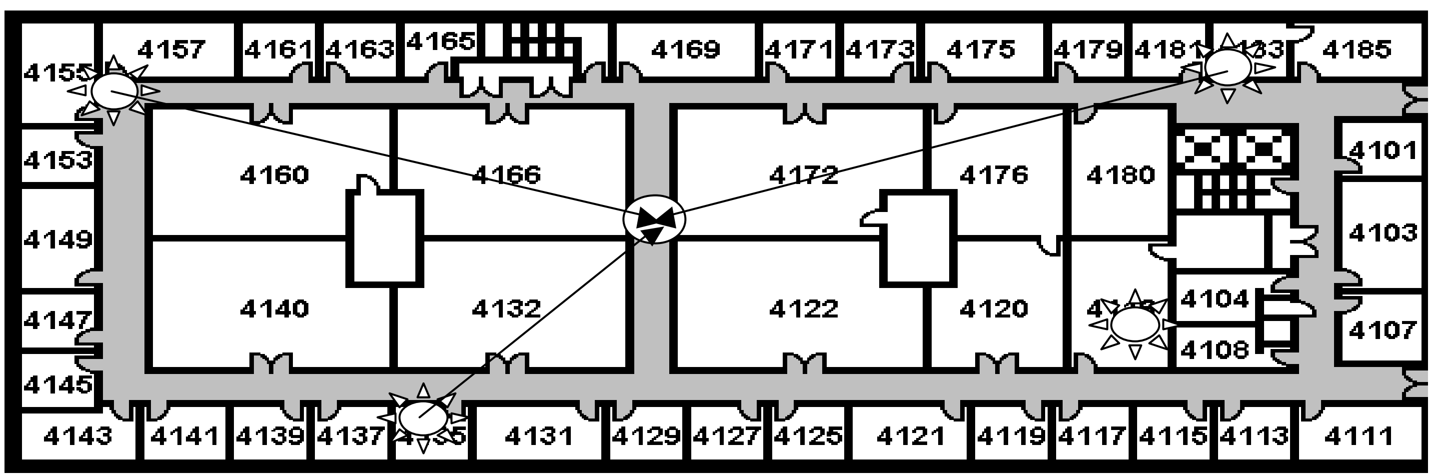
- To address noise characteristics
  - Radio map stores signal-strength distributions from K strongest access points

    (instead of scalar mean/maximum)
- To address scalability and cost of estimation
  - Clustering techniques for radio map locations
    - incremental clustering
    - joint clustering

Sampling Process

- Active scanning
  - Send a probe request
  - Receive a probe response
- Sample:

$$s = (s_1, s_2, \ldots)$$



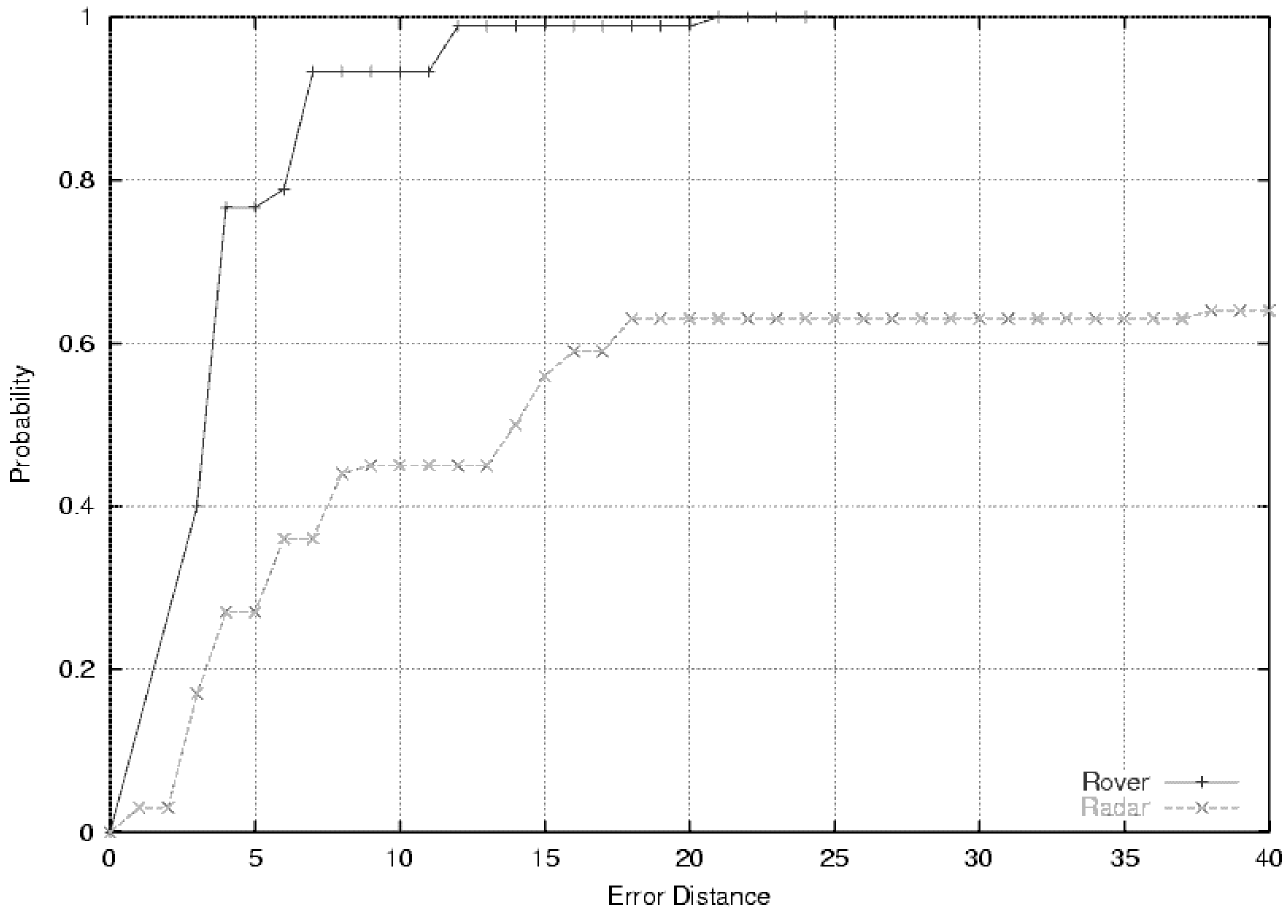
Mathematical Formulation

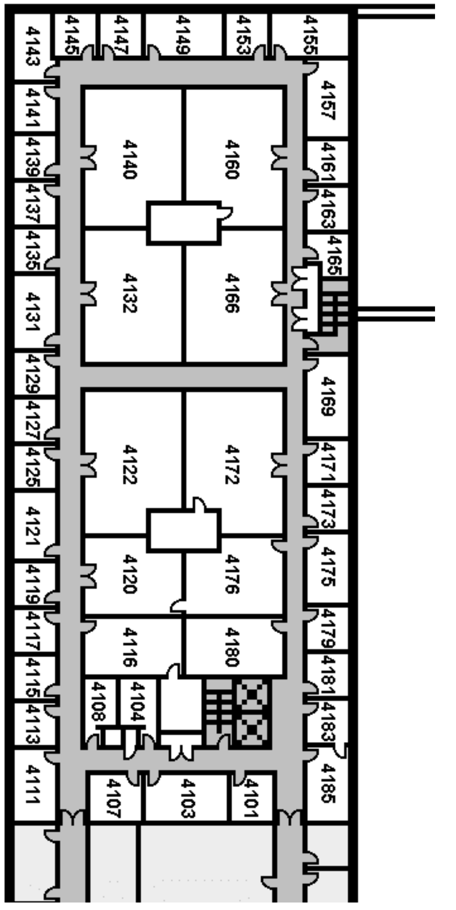
- x: Position vector
- s: Signal strength vector
  - One entry for each access point
- s(x) is a stochastic process
- P[s(x), t]: probability of receiving s at x at time t
- s(x) is a stationary process
  - P[s(x)] is the histogram of signal strength at x

Estimating Location

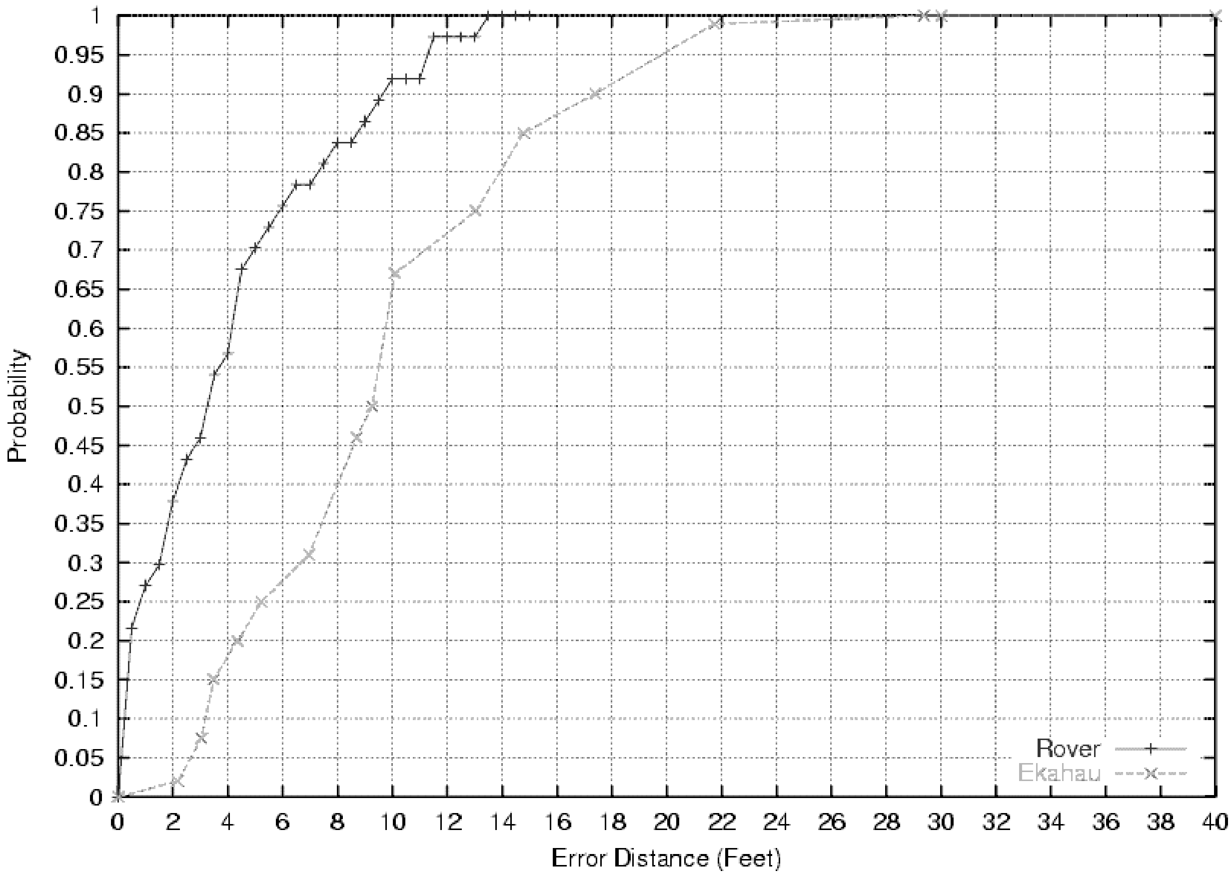
- $\text{Argmax}_x[P(x/s)]$
- Using Bayesian inversion
  - $\text{Argmax}_x[P(s/x).P(x)/P(s)]$
  - $\text{Argmax}_x[P(s/x).P(x)]$
- P(x): User history

Comparison With Other Systems: RADAR





Comparison With Other Systems: Ekahau



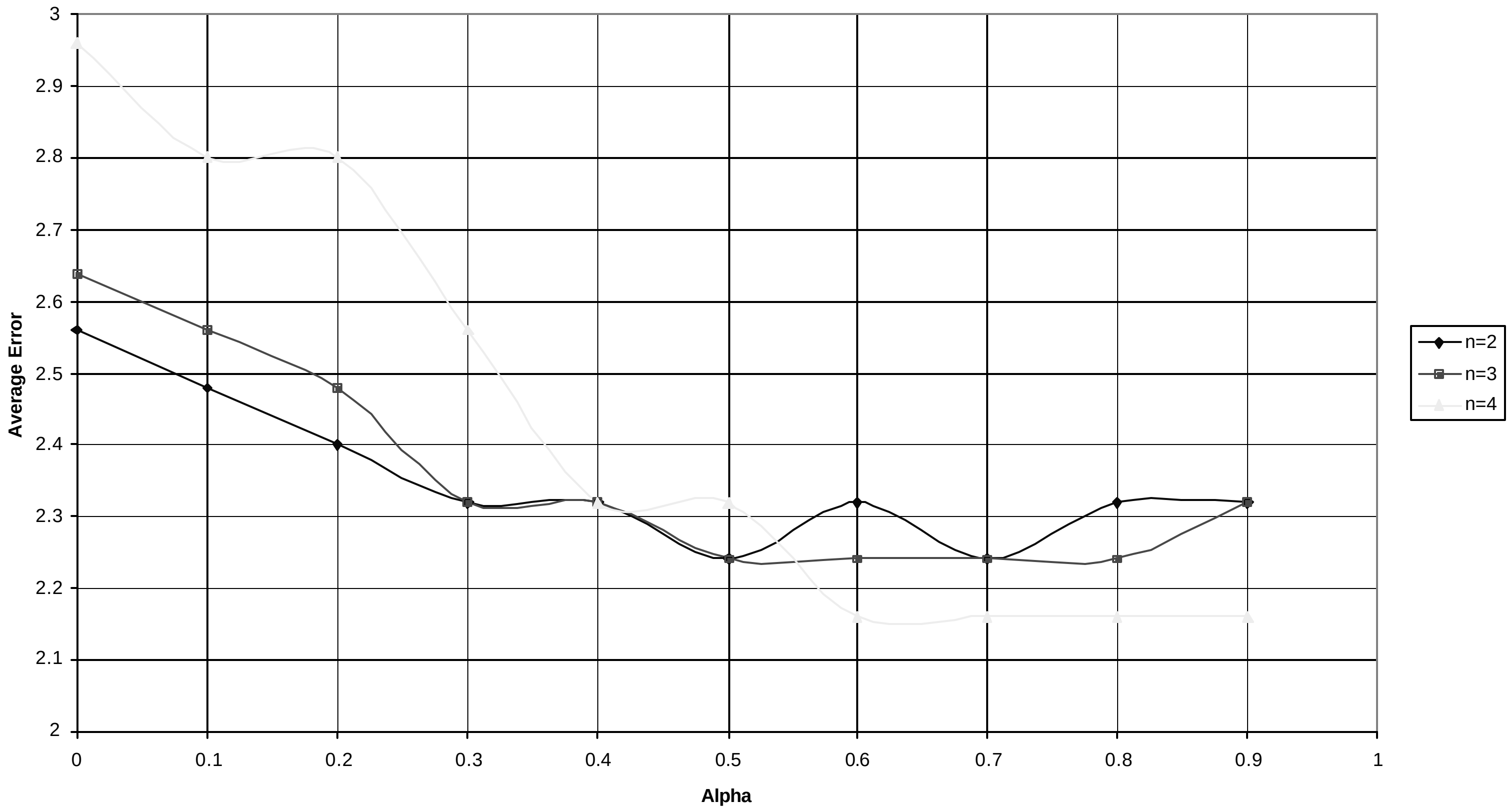
March 2002

The Maryland Information and Network Dynamics Lab

Handling Correlation: Averaging

- $s(t+1)=a.s(t)+(1-a).v(t)$
- $s \sim N(0, m)$
- $v \sim N(0, r)$
- $Y=1/n\ (s_1+s_2+...+s_n)$
- $E[Y(t)]=E[s(t)]=0$
- $Var[Y(t)]= m^2/n^2 \{ [(1-a^n)/(1-a)]^2 + n+ 1- a^2 * (1-a^{2(n-1)})/(1-a^2) \}$

Handling Correlation



The Maryland Information and Network Dynamics Lab

Characterization of Wireless Traffic

- Wireless traffic can not be characterized by monitoring the wired network only
  - Client to Client traffic
  - Control traffic
  - …
- Monitor the Wireless Medium
  - Use Sniffer(s)
  - Multiple Sniffers are required to assure full capture
  - Merge the traces from multiple sniffers
    - How??
      - Sequence numbers?
      - Time Stamp?

Synchronization of Multiple Sniffers by Least Square Method

- Timestamp of one sniffer can be approximated as a linear function of **reference time**.
- Reference time can be
  - Timestamp of another sniffer
  - Timestamp of beacon frames (from AP) that all sniffers commonly receive.
- LSM tool used
  - robustfit() in Matlab

Experimental Setup

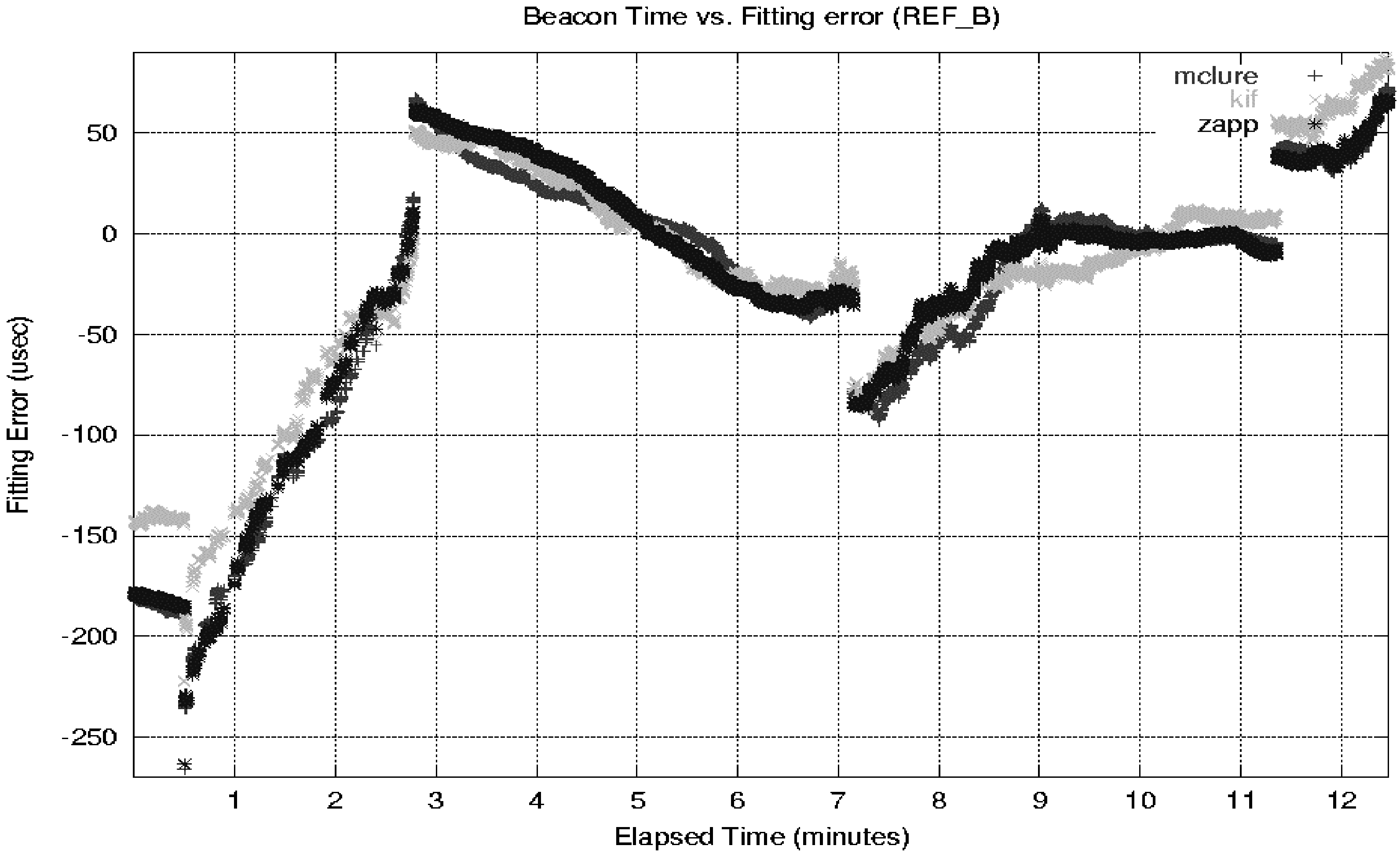
- Linux 2.4.19
- Orinoco_cs driver version 0.11b
- Libcap library version 0.7
- Ethereal network analyzer version 0.9.6
- Access Points monitored: 29 Cisco APs, 12 Lucent APs, 17 Prism2-based APs.
- Three sniffers: mclure (with Linksys card), kif (with NoName) and zapp (with NoName).

Synchronization: Using Beacon Time as Reference

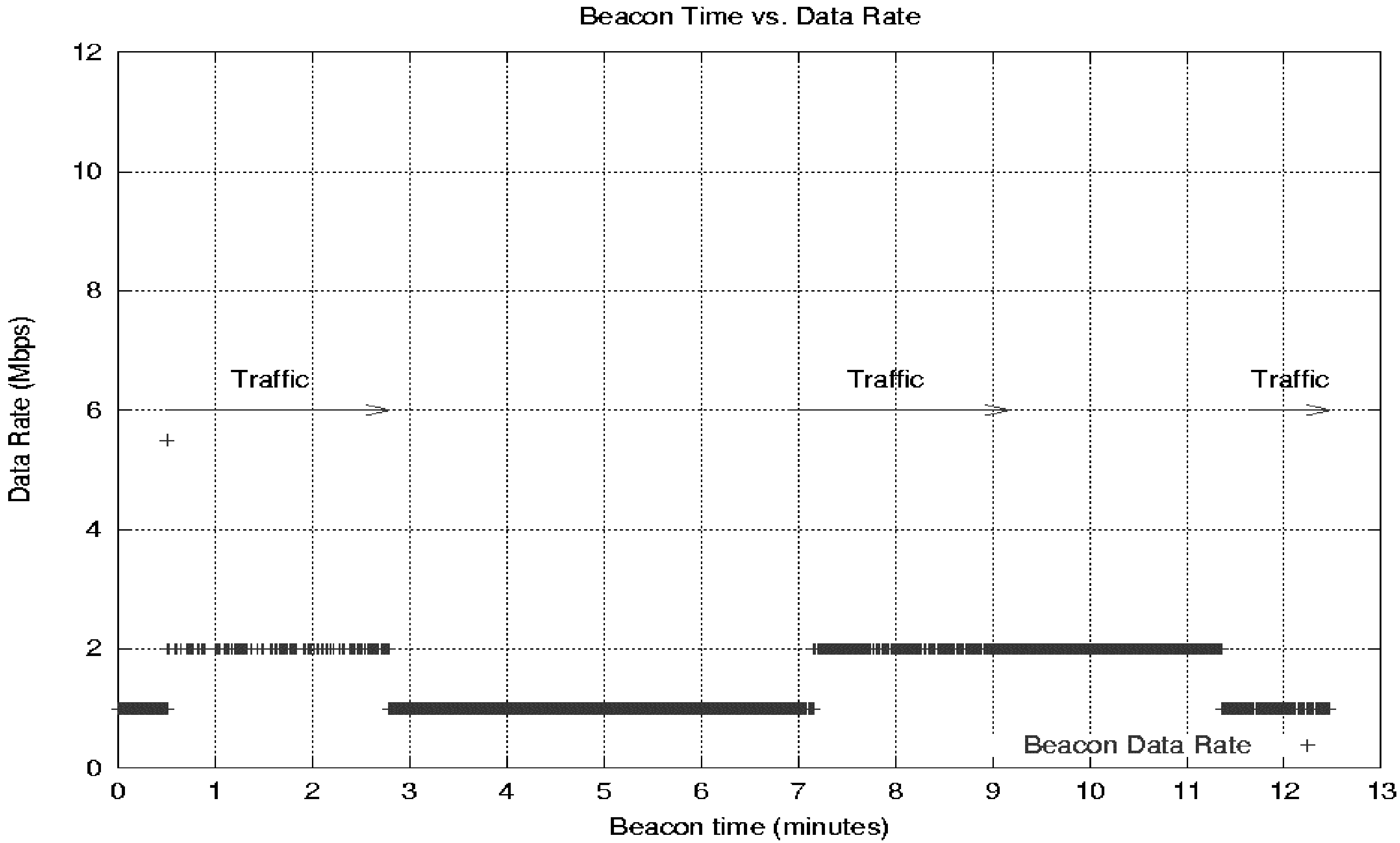
- Beacon timestamps are
  - more reliable than sniffer timestamps.
  - available to all sniffers.
- Simple linear regression [REF_B method]

  $\tau_{beacon} = \beta \, T_{sniffer} + \alpha$, where
  - Residue (error) = $\tau_{beacon} - T_{beacon} = (\beta \, T_{sniffer} + \alpha) - T_{beacon}$
- With our experimental data, REF_B method incurs many discontinuities in τ_{beacon}.
  - No transit delay for beacon frame is considered in REF_B.

Synchronization with Beacon Timestamps (REF_B)



Effect of Change in Data Rate and Traffics



Synchronization: Adjustment by Beacon Transit Delay

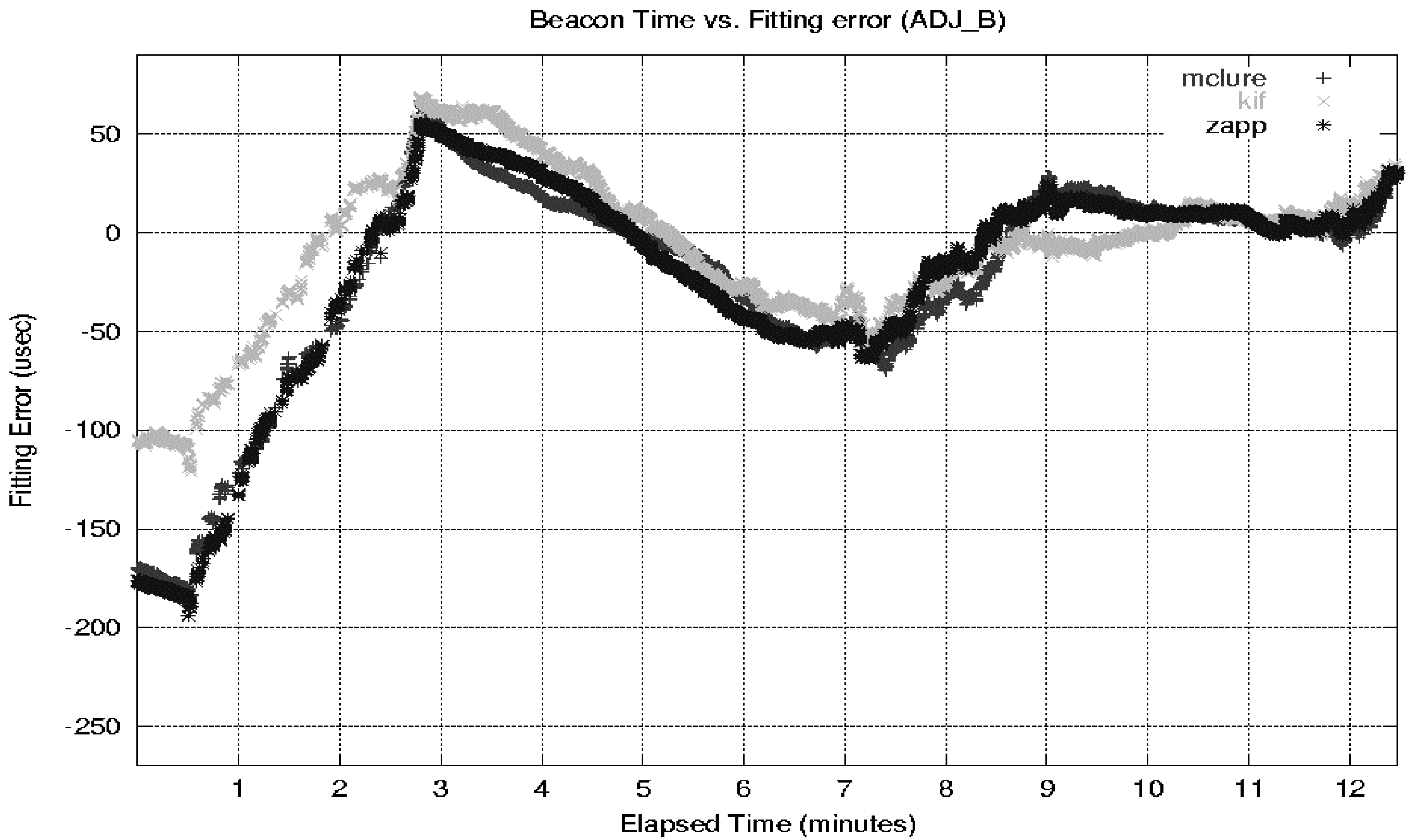
- Adjustment by transit delay [ADJ_B method]

  $\tau_{beacon} = \beta\,(T_{sniffer} - T_{delay}) + \alpha$ (1)

  $\tau_{beacon} - T_{delay} = \beta\,T_{sniffer} + \alpha$ (2)

- Which is correct, (1) or (2)?
  - Depends on the exact timing when T_{beacon} and $T_{sniffer}$ are generated.
- If sniffer's timestamp is generated after the **last** bit of a frame being received *and* the beacon timestamp exactly reflects the time when it was generated, then (1) is correct.
- If sniffer's timestamp is generated as soon as it received **the first bit** of the beacon frame *and* the beacon timestamp equals to the current time **plus the transit delay**, then (2) is correct.
- Experimental result: (2) is correct in our setup.

Synchronization with Beacon Timestamps (ADJ_B)



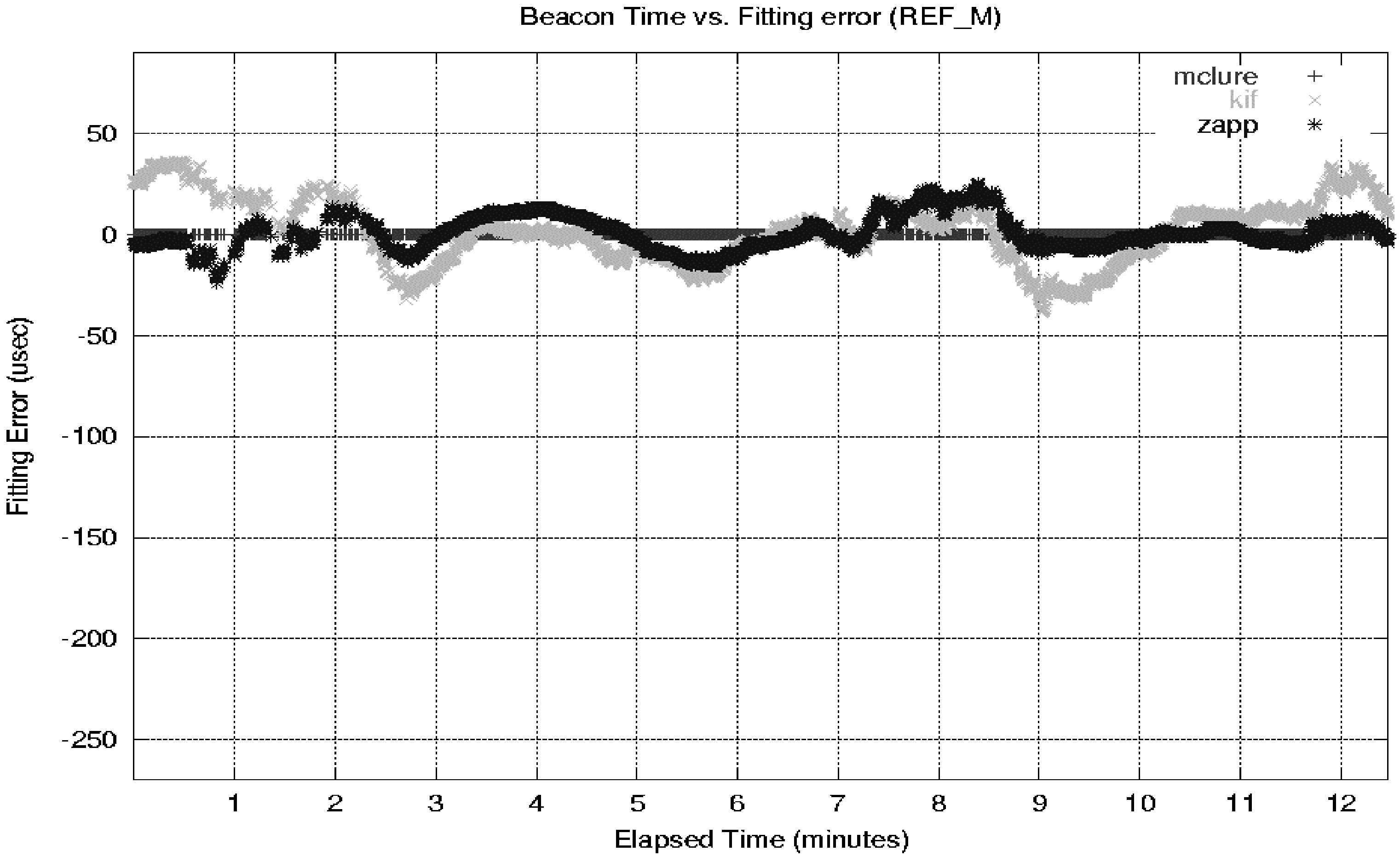
Synchronization: Using Sniffer Time as Reference

- Simple linear regression [REF_*sniffer_r* method]

  $\tau_{sniffer\_r} = \beta\, T_{sniffer} + \alpha$, where

  - $\tau_{sniffer\_r}$: Predicted reference timestamp
  - $T_{sniffer}$: Timestamp of target sniffer
  - Residue (error) = $\tau_{sniffer\_r} - T_{sniffer\_r} = (\beta\, T_{sniffer} + \alpha) - T_{sniffer\_r}$
- Synchronization performance depends on
  - clock difference between *sniffer* and *sniffer_r*.
  - Reliability of $T_{sniffer\_r}$ (e.g. what if $T_{sniffer\_r}$ is corrupted)
- Our setup: three sniffers (mclure, kif and zapp)

Synchronization with Sniffer Timestamps (REF_*mclure*)



Synchronization Performance Comparison

- Synchronization methods
  - REF_B: reference beacon timestamps
  - ADJ_B: reference ($T_{beacon} - T_{delay}$)
  - REF_*sniffer*: reference *sniffer*'s timestamps (*sniffer* can be m=mclure, k=kif, z=zapp)
- Performance metrics
  - Fitting performance by residue (= predicted – T_{beacon})
  - Pairwise performance - difference between two sniffer timestamps (e.g. $|T_{kif\_predicted} - T_{zapp\_predicted}|$)

Fitting Performance for Big Dataset (size = 5658, one set)

	Residue on mclure		Residue on kif		Residue on zapp	
	Min	Max	Min	Max	Min	Max
REF_B	-266	72	-222	88	-264	67
ADJ_B	-189	67	-121	69	-194	56
REF_M	0	0	-39	36	-24	25
REF_K	-36	39	0	0	-59	33
REF_Z	-25	24	-33	59	0	0
	-266	72	-222	88	-264	67

Pairwise Performance for Big Dataset (size = 5680, one set)

	Max Difference bet'n two sniffer timestamps		
	mclure-kif	kif-zapp	zapp-mclure
REF_B	48	49	26
ADJ_B	74	82	26
REF_M	39	44	25
REF_K	39	59	38
REF_Z	52	59	25
Total	74	82	38

Fitting Performance for Small Dataset (size = 200, 28 sets)

	Residue on mclure		Residue on kif		Residue on zapp	
	Min	Max	Min	Max	Min	Max
REF_B	-79	61	-77	61	-76	54
ADJ_B	-22	18	-22	19	-16	13
REF_M	0	0	-19	13	-13	28
REF_K	-10	19	0	0	-17	20
REF_Z	-28	13	-20	17	0	0
Total	-79	61	-77	61	-76	54

Pairwise Performance for Small Dataset (size = 196~202, 28 sets)

	Max Difference bet’n two sniffer timestamps		
	mclure-kif	kif-zapp	zapp-mclure
REF_B	25	20	43
ADJ_B	17	17	15
REF_M	19	23	26
REF_K	19	20	23
REF_Z	15	20	26
Total	25	23	43

Conclusion

- In fitting performance, ADJ_B and REF_*sniffer* perform better than REF_B.
- In matching performance, REF_*sniffer* performs better than REF_B and ADJ_B.
- Referencing beacon timestamps is more reliable than reference of sniffer timestamp.
- For small data size (e.g. 200), matching error is smaller than 50 μs, which is equal to DIFS (Distributed Inter-Frame Space) therefore, small enough to distinguish duplicates.

WLAN QoS Studies

The Impact of Physical-Layer Capture on Higher-Layer Performance in 802.11b WLANs

Throughput fairness in 802.11b WLANs

- Throughput fairness in 802.11 depends on
  - TCP/Application congestion control
  - MAC access mechanism
  - Physical-layer characteristics
- Most studies downplay physical-layer effect and focus on the MAC CSMA/CA/BEB and on the TCP/Application control
- We discovered that physical-layer capture is the dominant factor in throughput fairness

Physical-layer capture effect

- Physical-layer capture effect:
  - When two frames collide at a receiver, the receiver can extract the stronger frame
- Capture occurs consistently for even a few dBm difference in frame signal strengths
- Capture occurs frequently in WLANs (due to multipath and fading).

How do we decide collisions?

- A sniffer X' "tracks" each source X
  - Max strength signal at X' is from X
- In a collision involving a frame of X, sniffer X' records the frame of X
  - Because of capture at X'

Inferring Collisions (contd.)

- Construct global timeline
  - Using reception firmware time stamps at sniffers
  - Synchronizing using beacons
  - Accuracy of 5 microseconds
- Two events on timeline are collisions if transmission time intervals overlap

UDP/Ad-hoc Mode Experiments

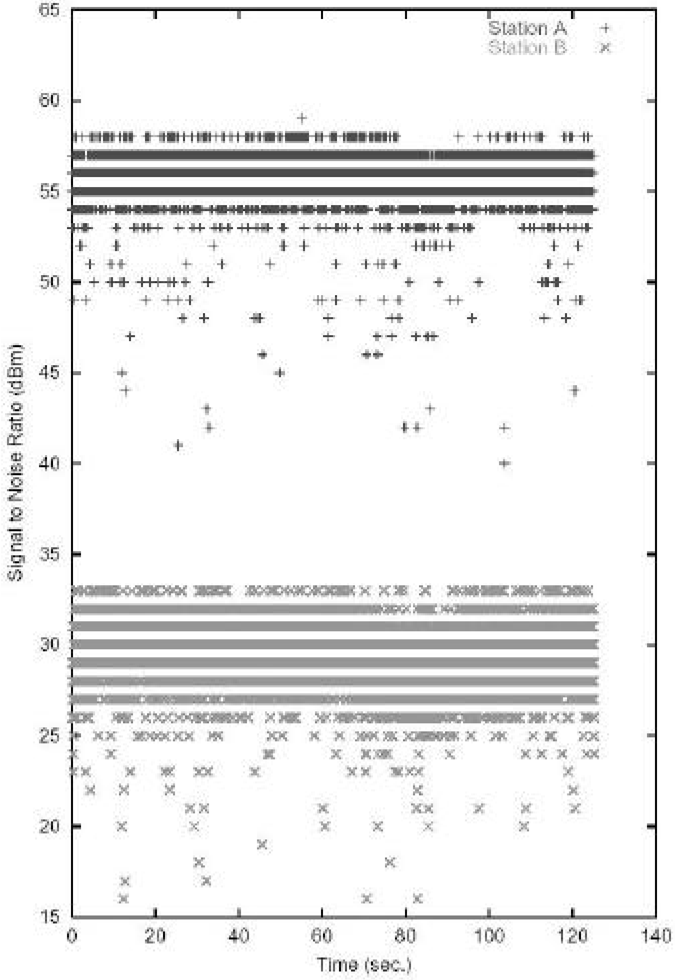
source 1 source 2 sniffer (sink)

sniffer 1 sniffer 2

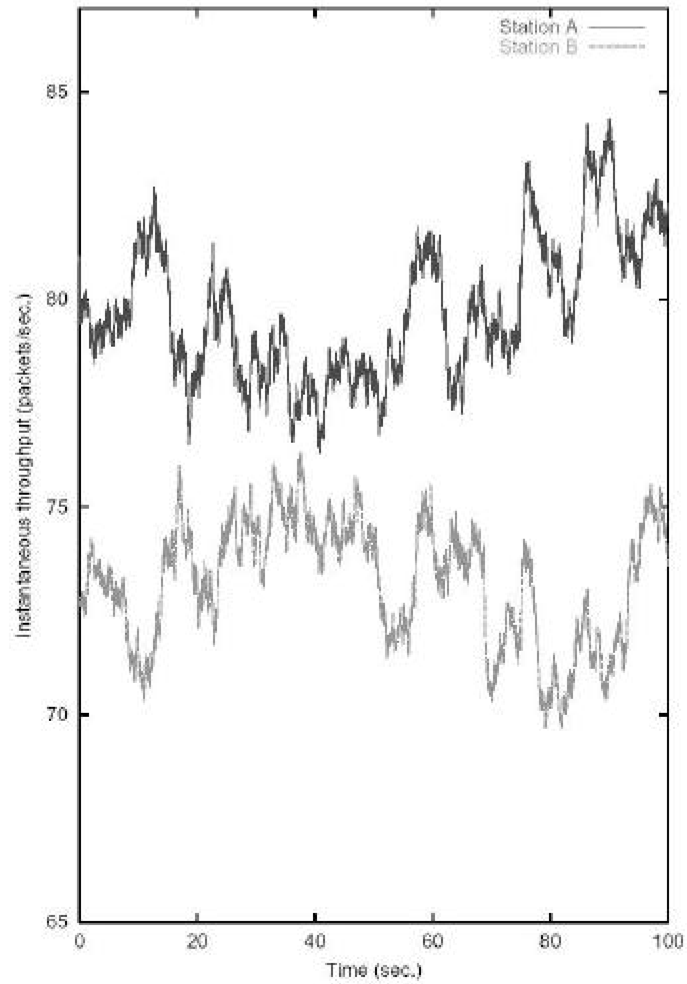
- Sources broadcasting in ad-hoc mode
  - no beacons, ACKs, and retransmissions
  - MAC-layer effect minimized
  - UDP workload, so no TCP/application congestion control
- Results
  - 8% of frames collided
  - 90% of collisions had capture
  - 8% higher throughput for stronger station

UDP/Ad-hoc Mode Experiments

Signal strengths



Throughputs



The MIND Lab
The Maryland Information and Network Dynamics Lab

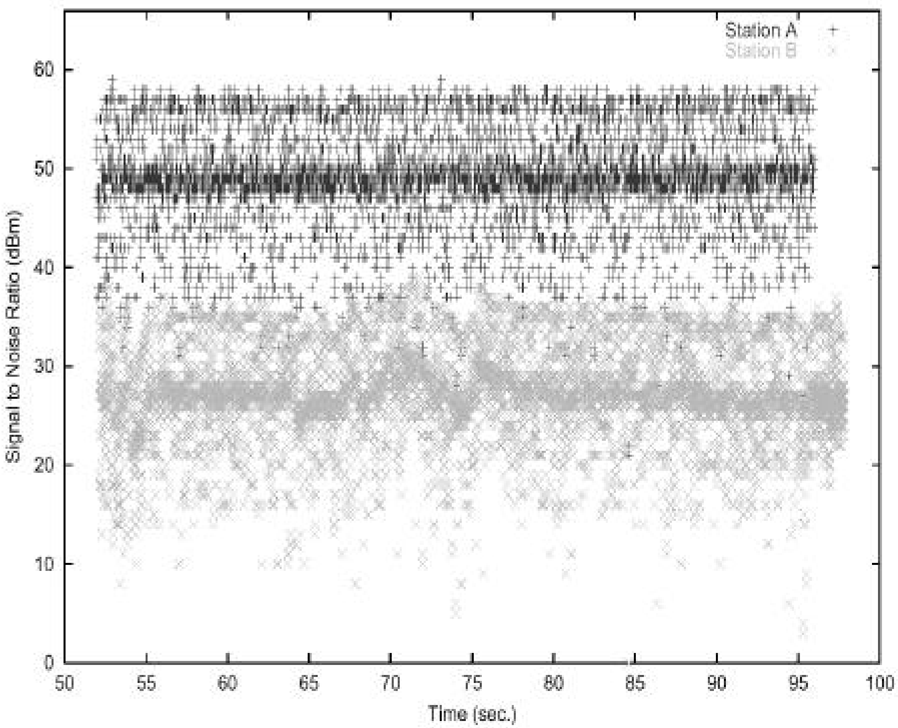
UDP/Infrastructure Mode without RTS/CTS

source 1 sniffer | source 2 sniffer | AP sink

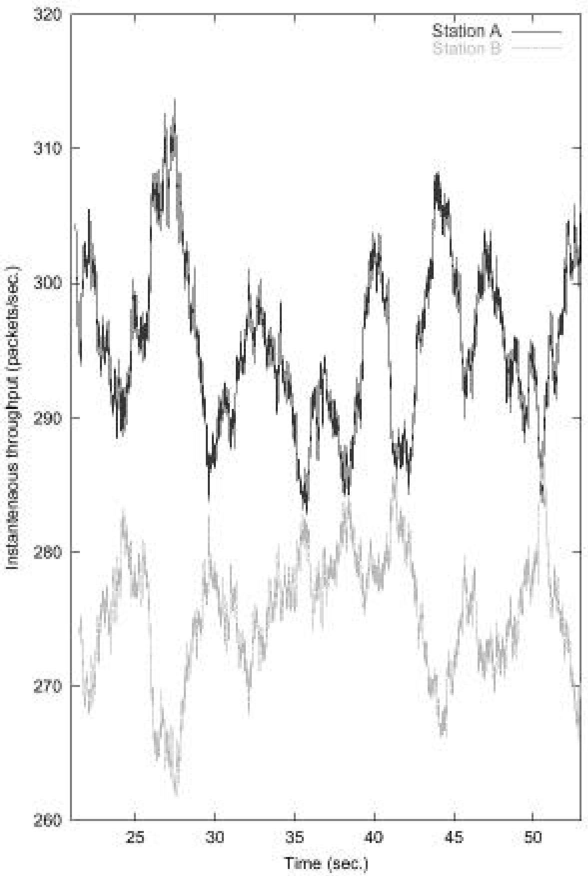
- Results
  - Weaker station retransmitted 5% of frames
  - Stronger station retransmitted 0.5% of frames
  - Stronger station had 8% higher throughput

UDP/Infrastructure Mode without RTS/CTS

Signal strengths

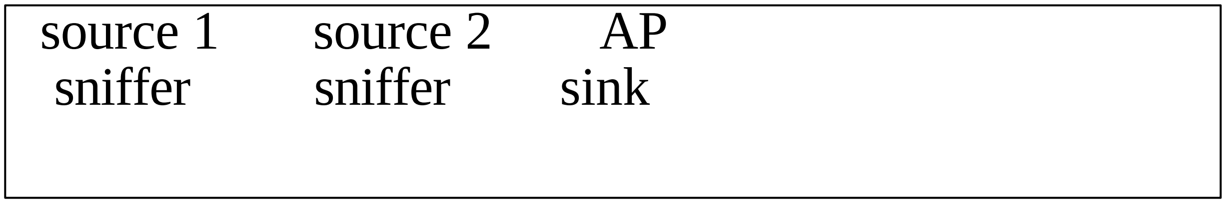


Throughputs



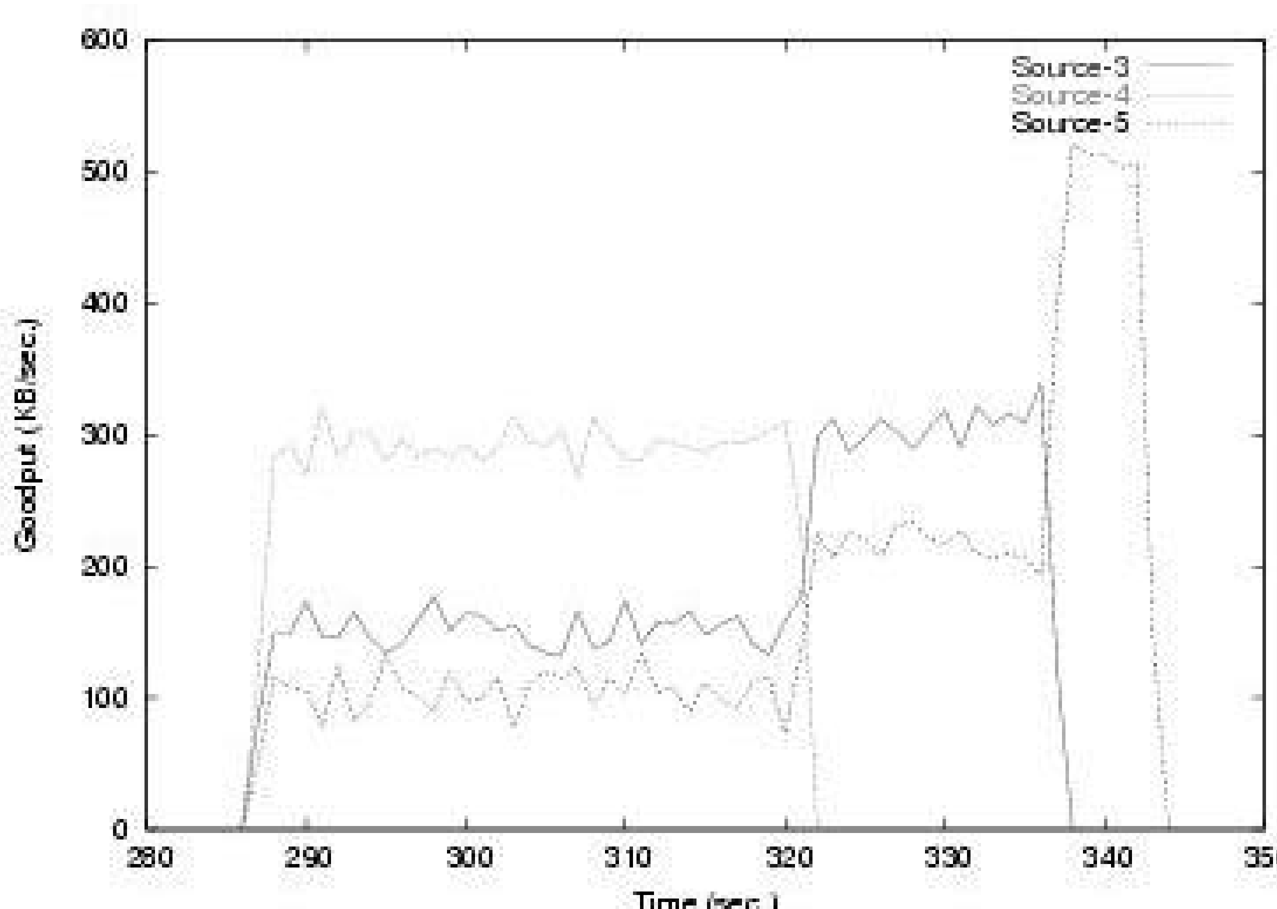
The Maryland Information and Network Dynamics Lab

UDP/Infrastructure Mode with RTS/CTS



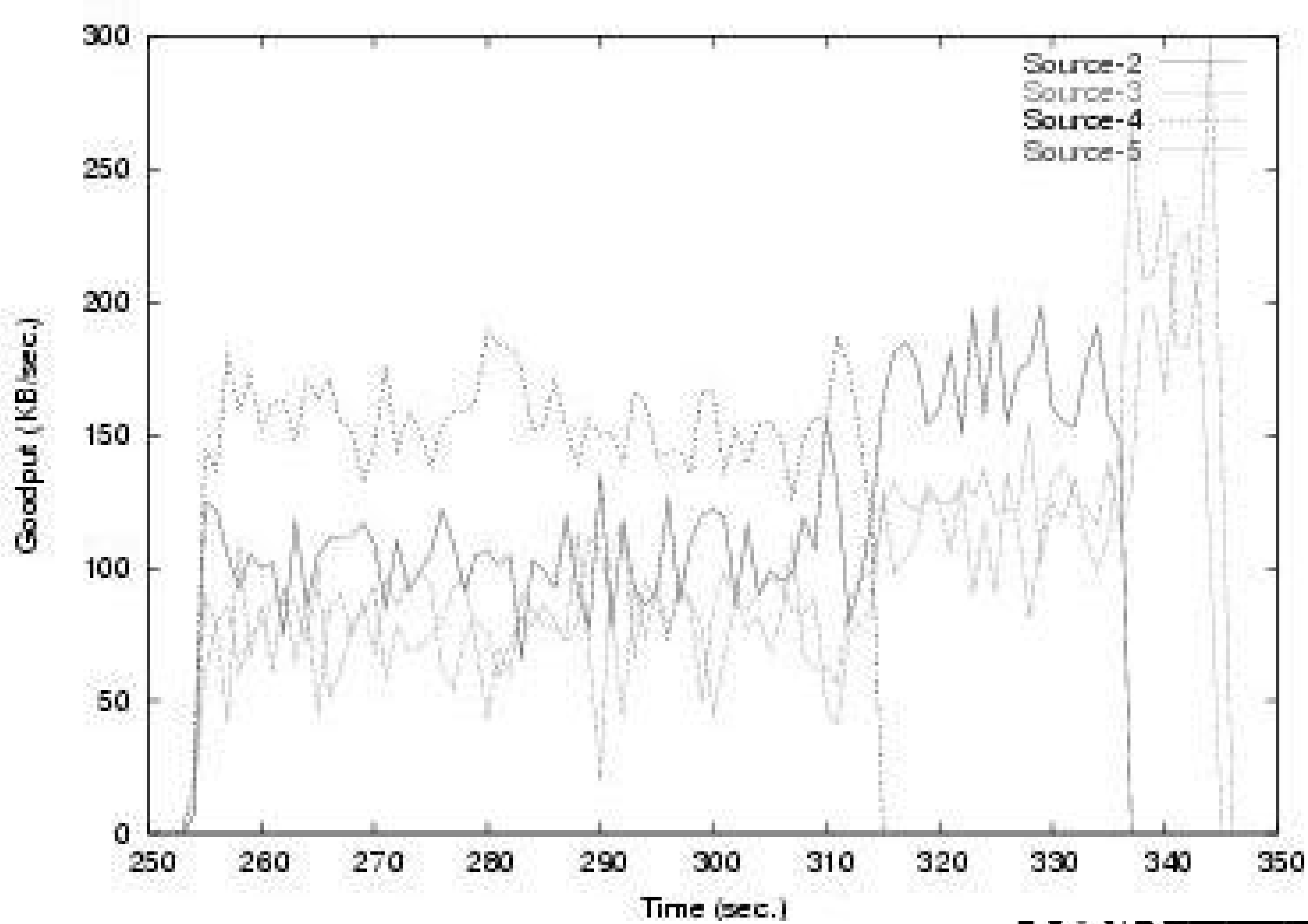
- Results
  - Each station retransmitted under 0.1% data frames
  - Weaker station retransmitted 5% of RTS frames
  - Stronger station retransmitted 0.1% of RTS frames
  - Stronger station had 12% higher throughput

Multiple UDP Sources: Infrastructure mode without RTS/CTS



The Maryland Information and Network Dynamics Lab

Multiple UDP Sources Throughput: Infrastructure mode with RTS/CTS



TCP/Infrastructure Mode

- Two sources, one AP, one sink
- Used netperf
  - Both sources were started at same time using a broadcast UDP signal
- Results
  - Throughput difference as high as 50%
  - Throughput depends on Signal Strength

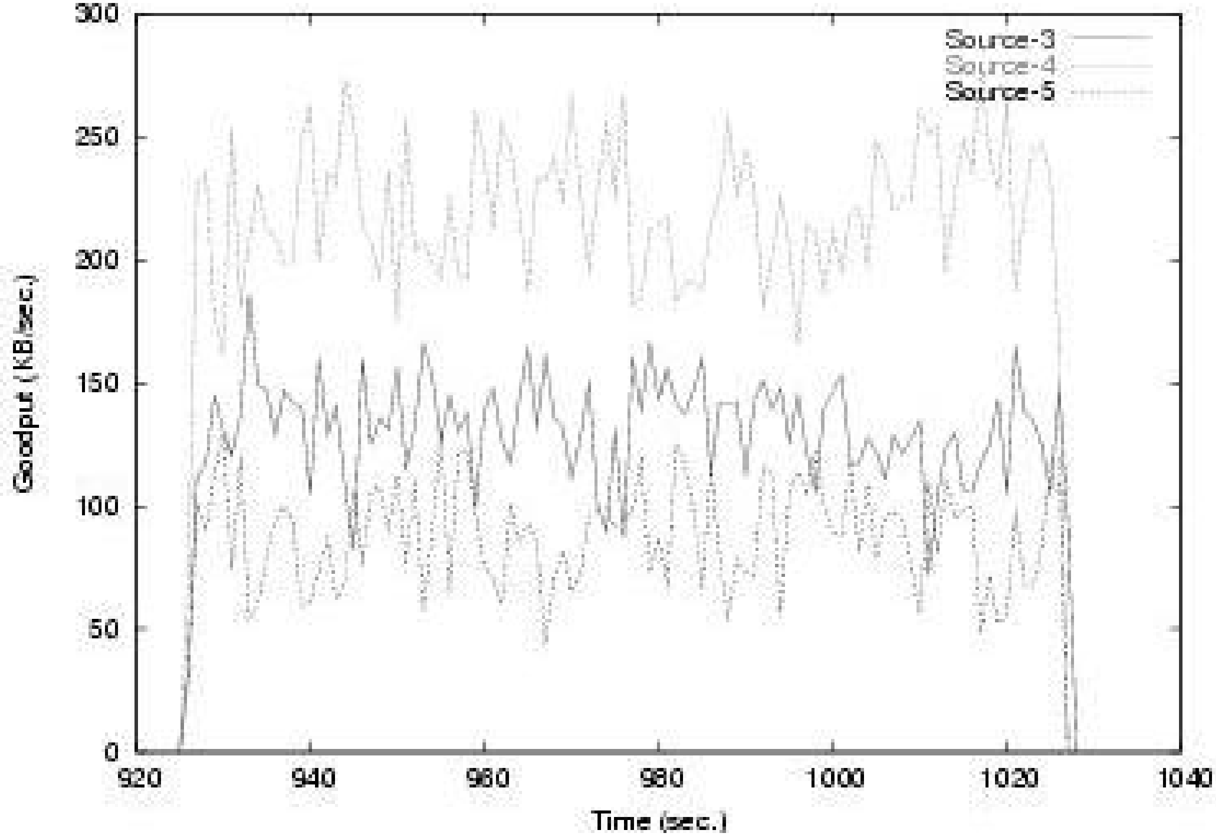
TCP/Infrastructure Mode: Typical Performance

Signal Strength	Throughput	Distance from AP
Signal: -55 dBm Noise: -88 dBm	2.92 Mbps	4 feet
Signal: -67 dBm Noise: -87 dBm	2.1 Mbps	7 feet

TCP/Infrastructure Mode: Typical Performance (contd.)

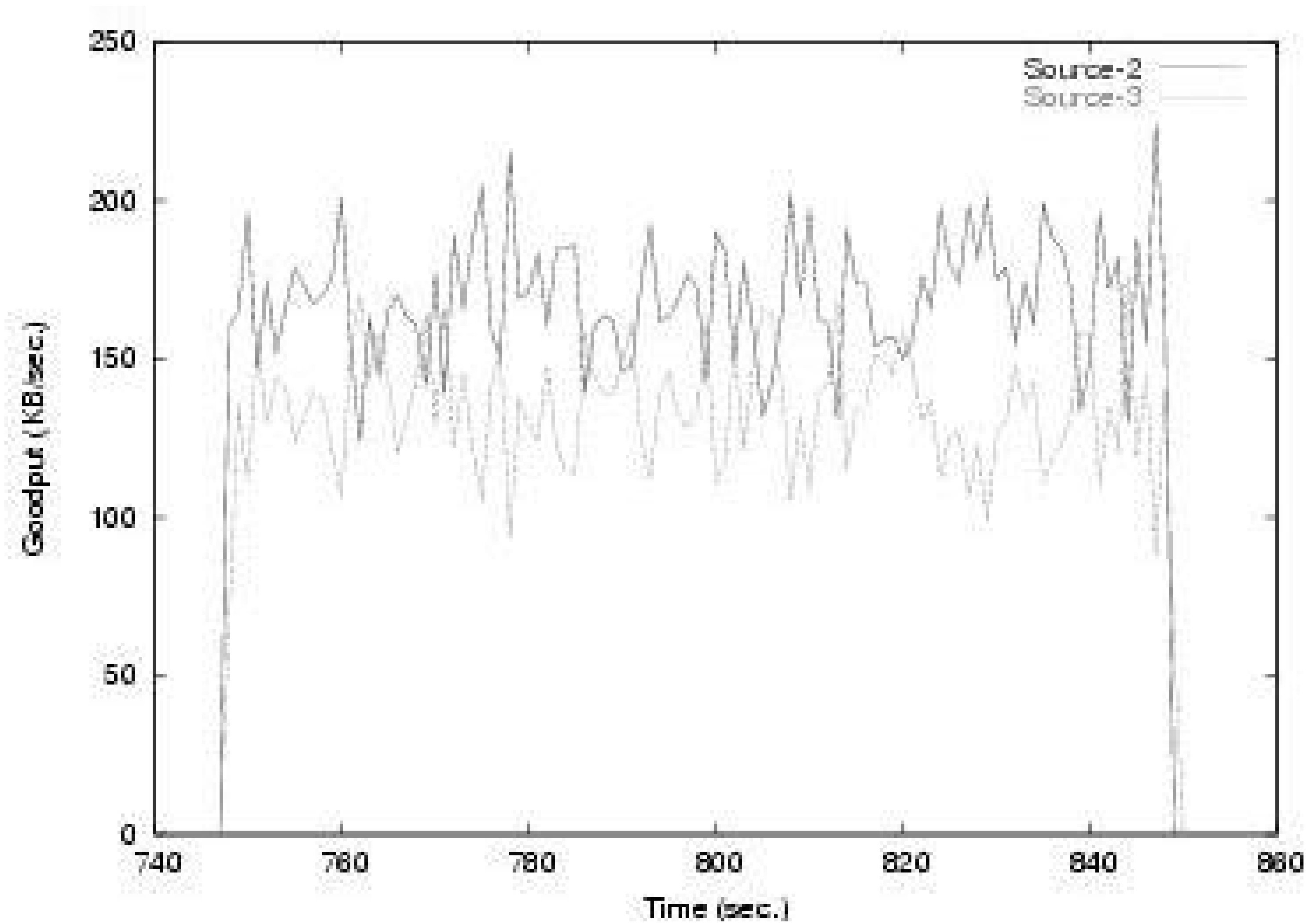
- TCP Tput = function(loss, RTT)
- Typical zero TCP level loss for two stations
  - Because of link-level ARQ in 802.11
- RTT varies significantly between stations
  - Related to signal strength
  - In presence of collision, retransmissions occur for one station
  - Other station’s frame is captured at AP
- Therefore, unfairness in TCP tput for station with weaker signal strength

Multiple TCP Sources Throughputs: Infrastructure without RTS/CTS



The Maryland Information and Network Dynamics Lab

Multiple TCP Sources Throughputs: Infrastructure with RTS/CTS



QoS: MAC layer conclusions

- Physical-layer capture is a major cause of MAC throughput unfairness.
- Resulting unfairness as high as 12% in favor of station with stronger signal (50% with TCP).
- Any QoS scheme must account for differing signal strengths of sources.

Link Layer Control for QoS MAC

- Random MAC (DCF) good at low load
  - Degrades at high load
- Scheduled MAC (PCF) good at high load
  - Not available yet
- Our Approach
  - Best of two worlds
  - Have Random MAC as base
  - Do *Link Layer Control* for improved performance at high load

Link Layer Control: The Big Picture

- Time is roughly divided into cycles
- Clients periodically inform AP of estimated load for next cycle
- AP computes fair shares of each client and broadcasts it
- Clients shape their outgoing traffic for next cycle at link layer

Link Layer Control: Specifics

- 802.11 allows 2304 bytes MTU
  - Our measurements show only 1500 bytes used
  - Because WLAN drivers emulate Ethernet interface to the kernel
- So piggyback load information at end of frame
  - Load information = size of firmware queue
  - DD write extra bytes to firmware buffer at EOFrame
    - Doesn't affect FCS
  - The receiving driver (at AP) strips it off and uses it for computation
    - Doesn't affect IP checksum

Link Layer Control: Specifics (contd.)

- Policing at client
  - Window based rate control at link layer
  - Use the Interface Queue (IFQ) as window
    - IFQ = Layer between device driver and kernel networking stack
- At AP
  - Collect estimated load
  - Compute fair share
  - Broadcast information

Link Layer Control: Implementation

- Linux OS Client with orinoco_cs driver
  - New queuing discipline (crmac) to implement our policy in IFQ as a kernel module
  - Patched the tc (transmission control) program to tell kernel to use crmac for an interface.
- Linux OS AP with hostap_cs driver
  - Added ability to strip off load information and compute fair share
- Current Work
  - Testing of different policies at AP and clients