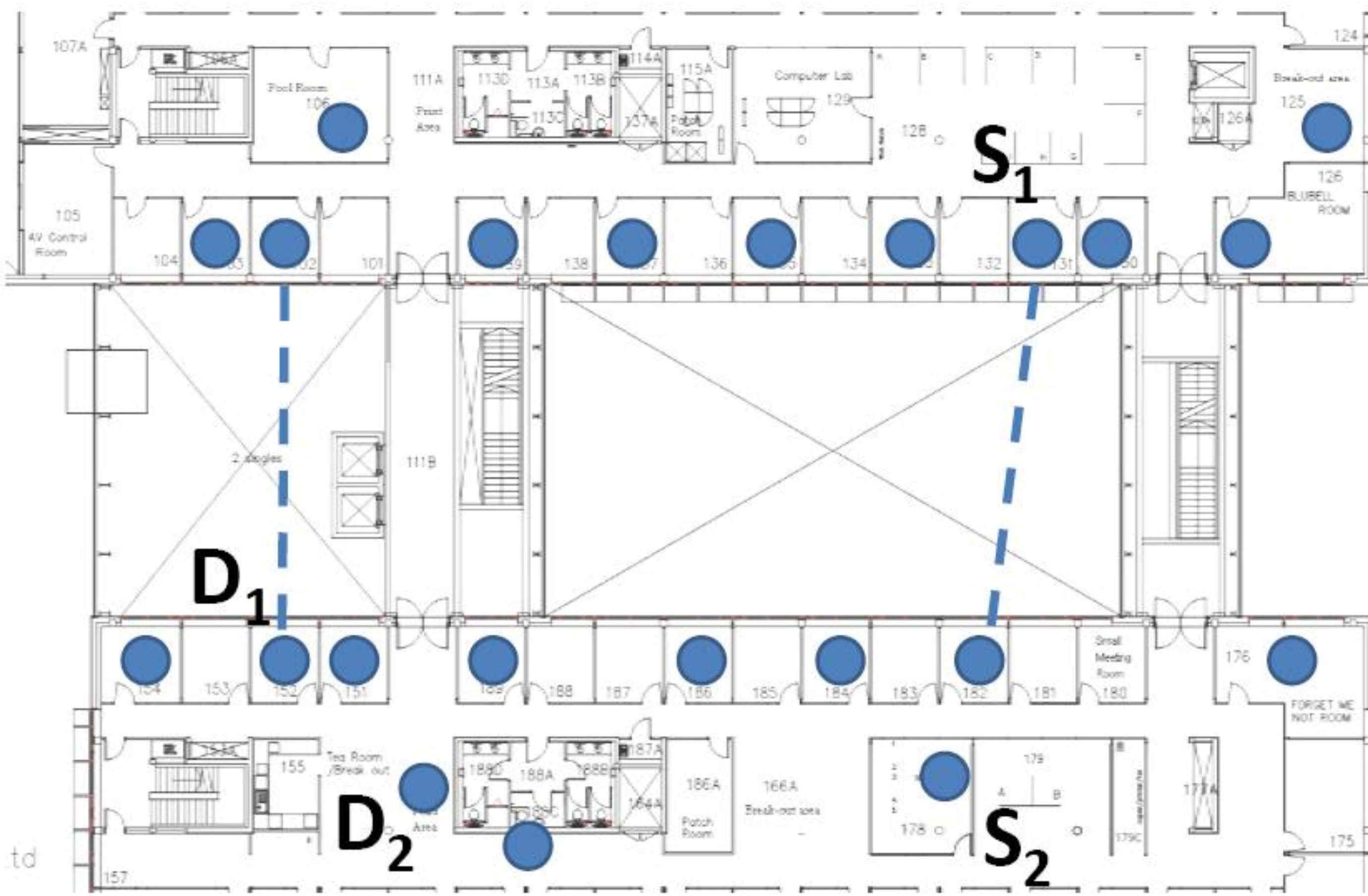
Horizon: Balancing **TCP** over **multiple** paths in wireless **mesh** networks

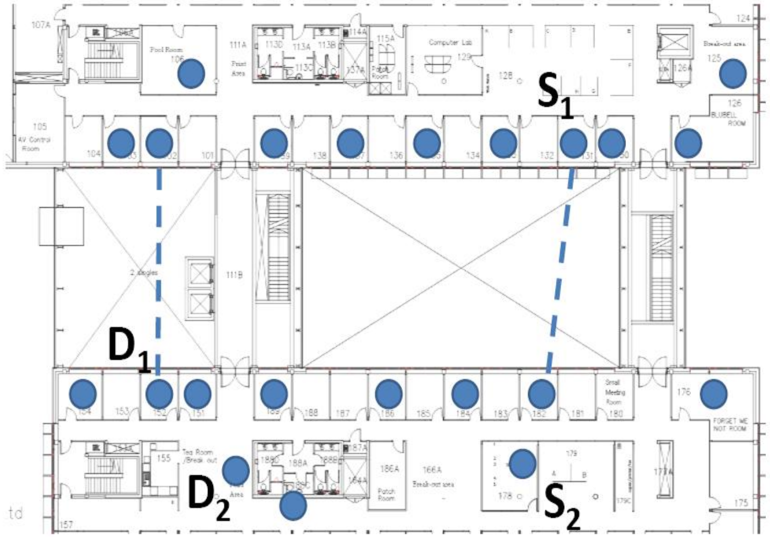
Bozidar Radunovic, Christos Gkantsidis,
Dinan Gunawardena, Peter Key

Microsoft Research
Cambridge, UK

Wireless Mesh Networks



Goals

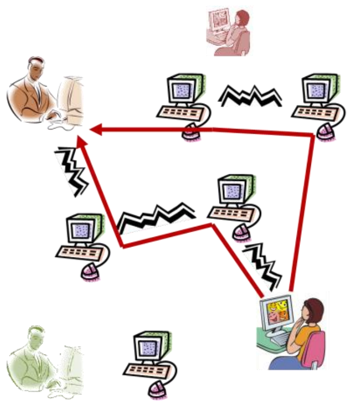


1. **Efficient use of resources**
   - Use multiple paths
2. **"Fair" allocation of resources**
   - Many users, long-distance vs. short-distance flows
3. **Good application performance**
   - TCP in particular
4. **Deployable on existing network**
   - Minor modifications of the existing 802.11 stack.

Outline

- Controlling the network: Backpressure
- Dealing with TCP
- Experimental results

Controlling the network: Backpressure

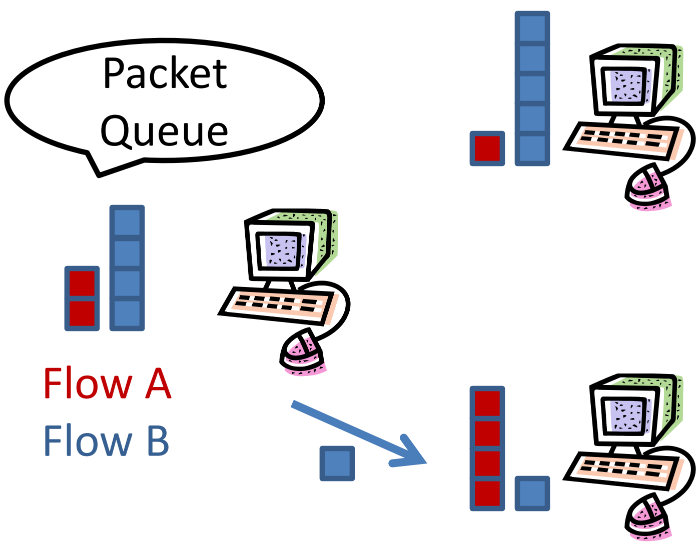


Algorithms to achieve utilization, fairness, good experience?

Protocol:

A. Which node transmits?
B. Which user/flow takes priority?
C. Where to forward the packets?

Our answer: Backpressure-based algorithms

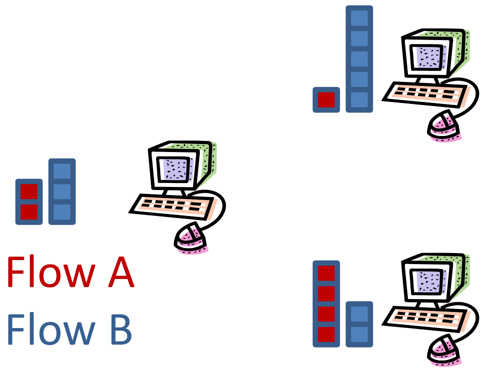


Essence of backpressure: give priority to

1. Nodes that have smaller queues
   - Difficult to implement
2. Flows that have smaller number of packets

✓ Provably optimal
✓ Very long theoretical support

Backpressure challenges

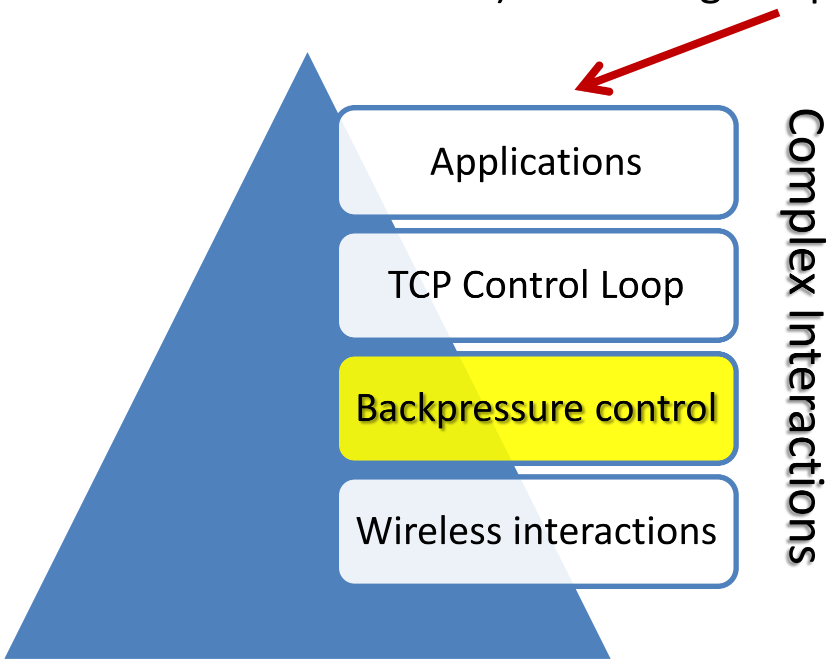


Our goal:

A) First to implement backpressure for wireless meshes
   - Lots of theory, no practice so far
   - Lots of challenges arise from practical constraints

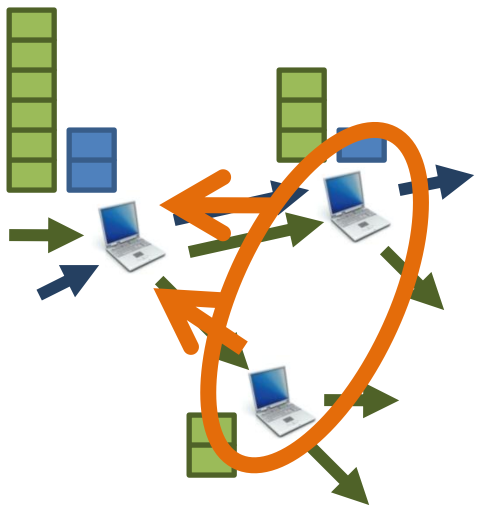
B) Use multiple paths

C) Provide good performance for current applications



Examples:

A. Queuing by pressure triggers TCP congestion control
B. Back-pressure increases delay
C. Limited TCP window size penalizes path estimation
D. Out-of-order packet arrivals
E. ... And many others



Our Multi-Path Routing

- **Input:**
  - *queue$_i$(f)*: packet from flow *f* queued at node *i*
- **Output:**
  - *C$_i$(f)*: cost from node *i* to destination of flow *f*
  - *bestFlow$_i$*: the flow to select for transmission at *i*
  - *bNH$_i$(f)*: the best next hop at *i* for flow *f*

Algorithm at node *i*:

1. **Select where to transmit a packet:**

   $bNH(f)_i = bestNextHop_i(f)$

   $= \text{argmin}_j \left(queue_i(f) / rate(i,j) + C_j(f)\right)$

2. **Select the flow from which to transmit a packet:**

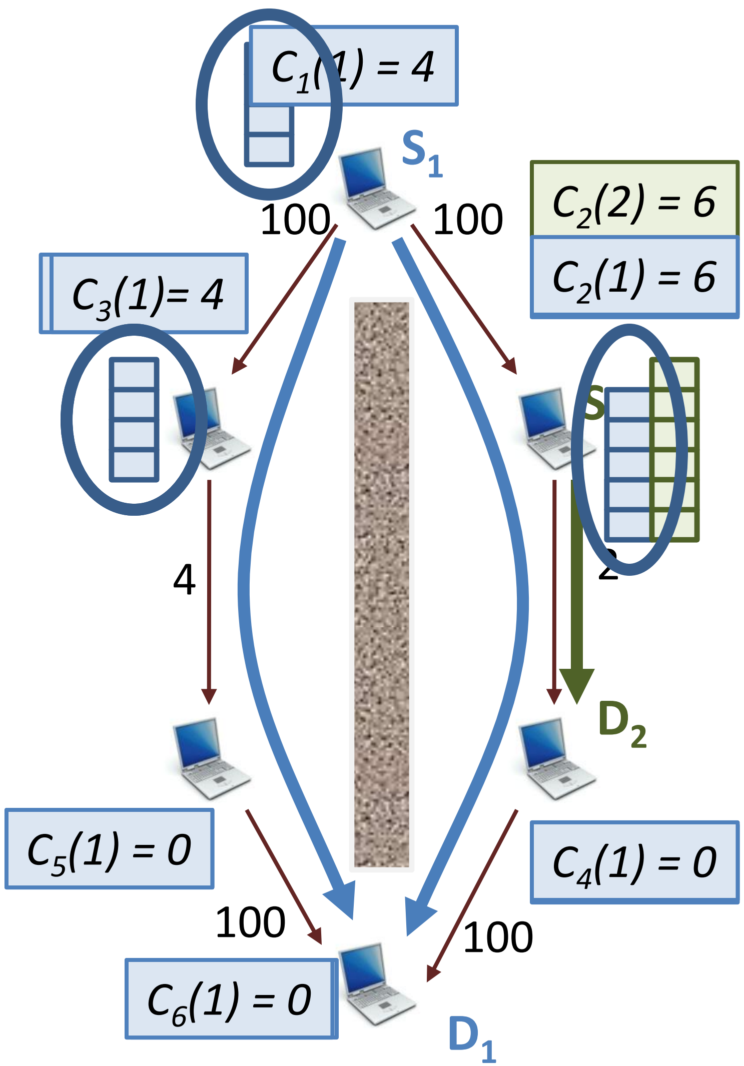
   $bestFlow_i = \text{argmax}_f \left(queue_i(f) / rate(i,bNH_i(f))\right)$

3. **Update costs:**

   $C_i(f) = \max_f\left(queue_i(f) / rate(i,bNH_i(f))\right) + C_{bNH_i}(f)$

4. **Propagate costs**

Simple Example



Comparison:

Our scheme : **4** packets

Back-pressure: **13** packets

Main advantages:

1. Minimal queuing: queue sizes do not grow with network
2. Estimates path quality with realistic TCP window size
3. Fast convergence

Outline

- Controlling the network: Backpressure

- **Dealing with TCP**

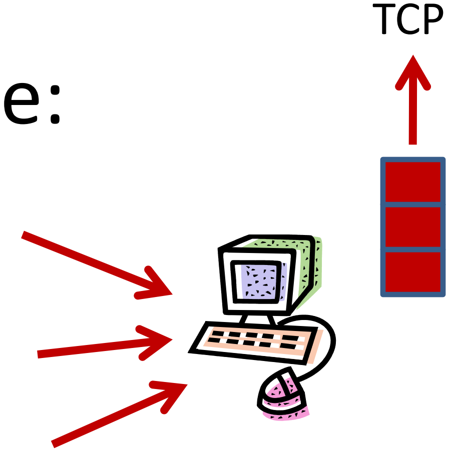
- Experimental results

Challenges dealing with TCP (1)

- Our system performs congestion control ...
- ... so does TCP
- ... need to make sure that they are compatible
- **Idea**: signal congestion to TCP
  - ECN-like approach
  - in some cases we communicate congestion by generating **duplicate ACKs**

Challenges dealing with TCP (2)

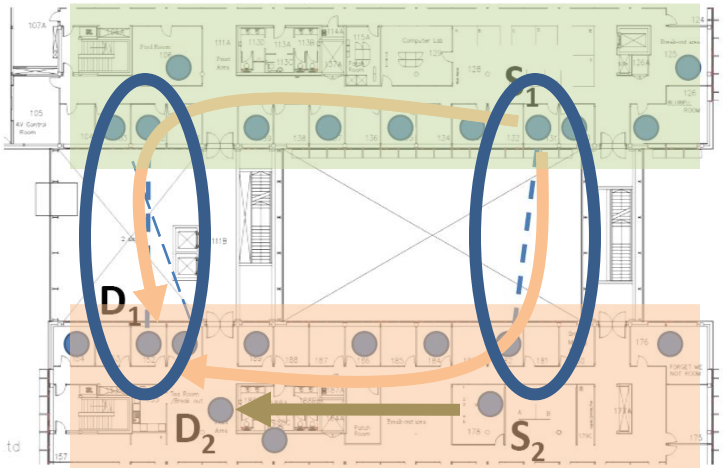
- Recall: We use multiple paths ...
- ... TCP gets confused (path delay estimation, out of order delivery, etc.)
- **Solution:** Use reassemble queue:
  - Minimize packet reordering
  - Avoid time-outs at all costs



Outline

- Controlling the network: Backpressure
- Dealing with TCP
- **Experimental results**

Load balancing across two flows

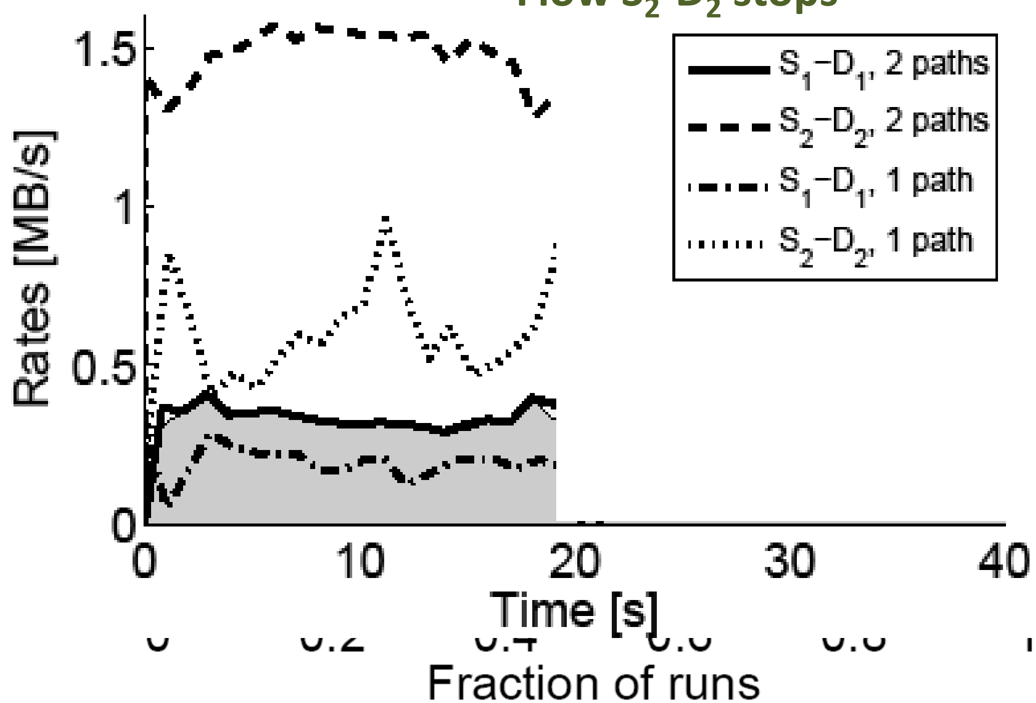


- 2 disjoint areas
- Only 6 nodes are dual-homed

Performance decreased: need more channels or better MAC

Both total rate and fairness improved

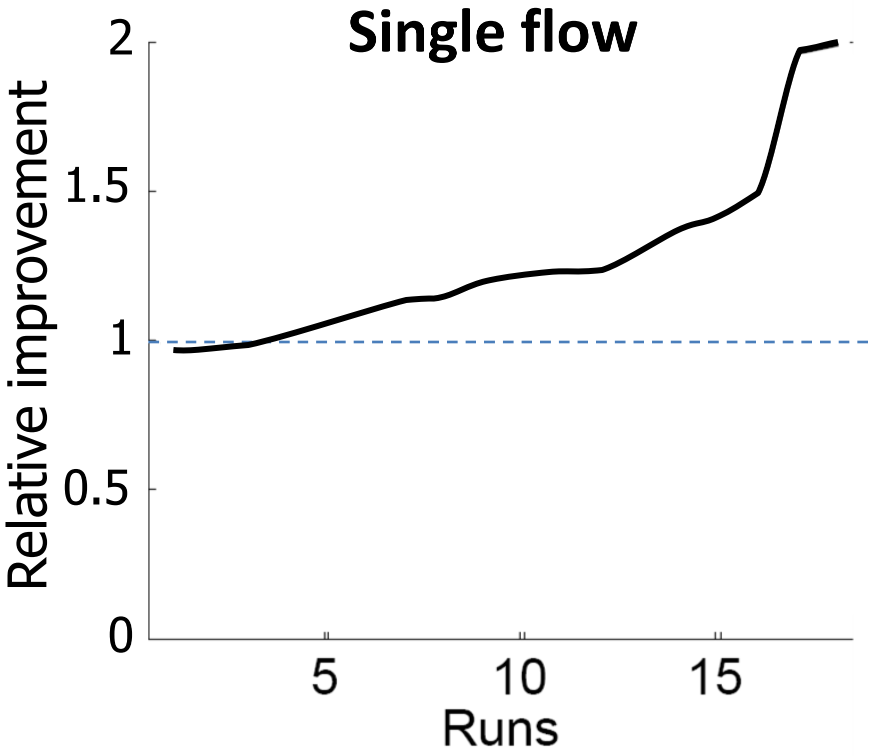
Flow S_2-D_2 stops



~ double the rate for both flows

Multi-homed networks

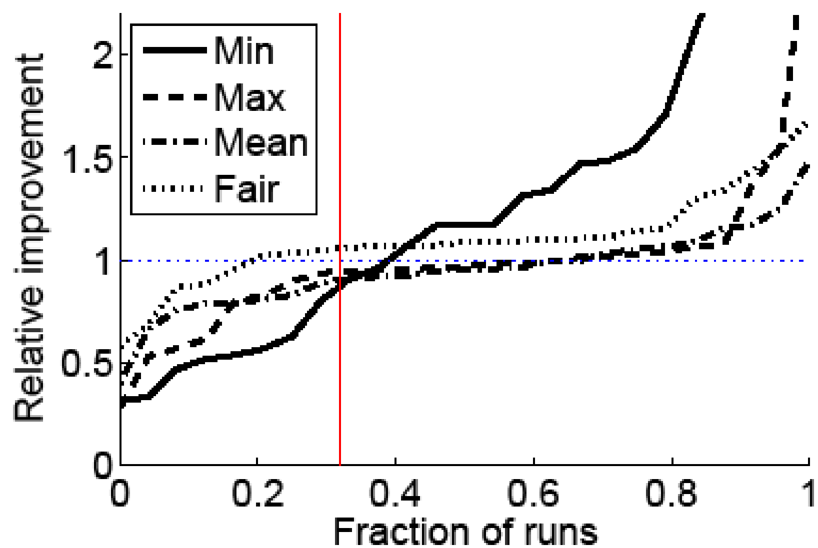
- Different access points/base stations
- Same or different radios



More flows

- More flows → all resources used → cannot increase total rate
- Instead, we improve fairness (e.g. smallest rate)

8 flows



Goals

1. **Efficient use of resources**
   – Use multiple paths
2. **"Fair" allocation of resources**
   – Many users, long-distance vs. short-distance flows
3. **Good application performance**
   – TCP in particular
4. **Deployable on existing network**
   – No modifications of the existing 802.11 stack.

How to select paths?

Other types of traffic?
How to deal with UDP?

What can we do with small changes?

THANK YOU