Footloose: A Case for Physical Eventual Consistency and Selective Conflict Resolution

Justin Mazzola Paluska
David Saff, Tom Yeh, Kathryn Chen

MIT CSAIL

Today's Situation

- Data is scattered throughout devices:
  - All of my phone numbers on my cell phone
  - Some other contact information on my PDA
  - Still more on my laptop
- But no way to manage data

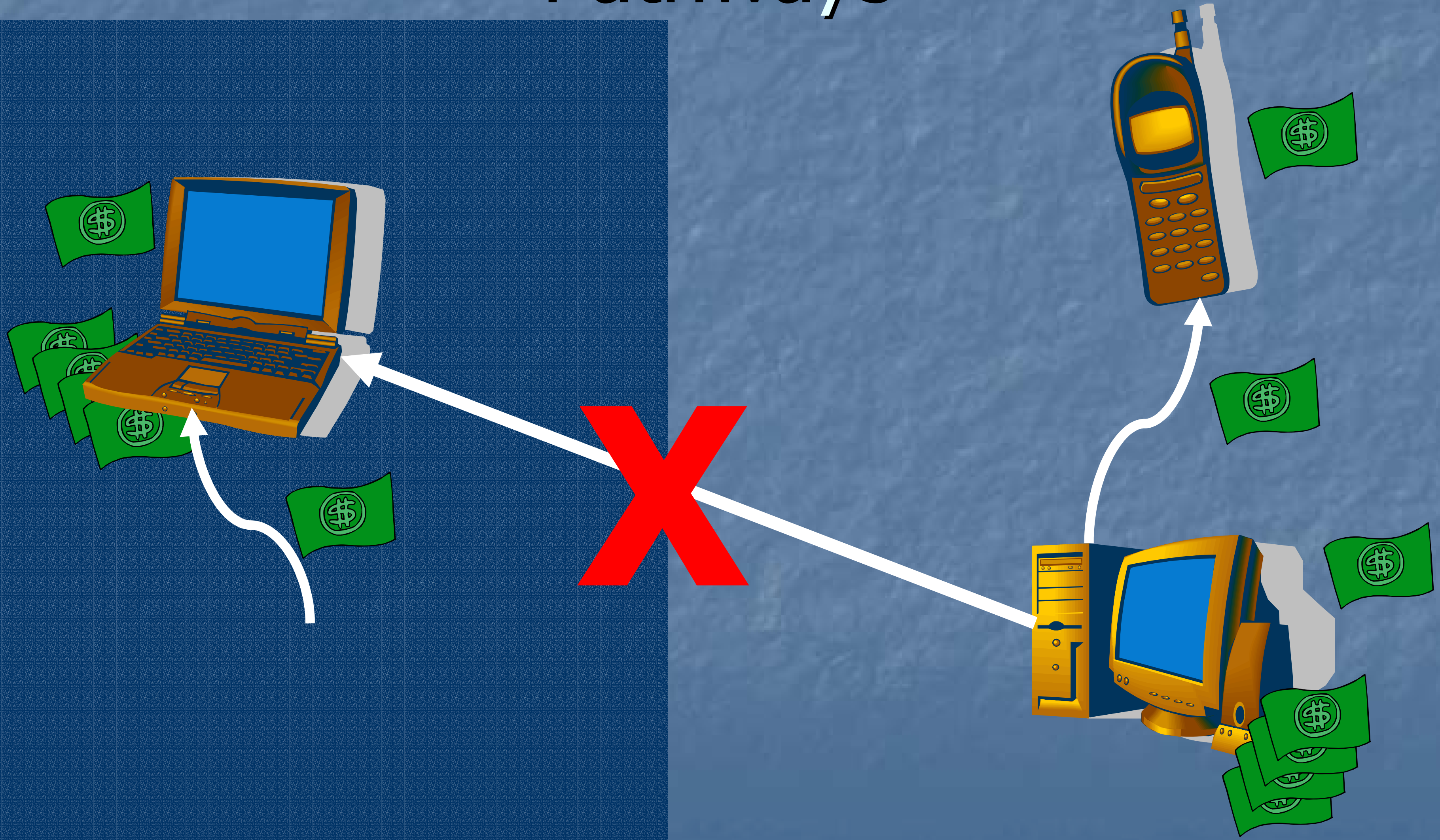
Device Characteristics

- A single primary user
- Some memory
- A wireless communications medium
- Shared contexts

- **What can we do with these resources?**

Automatic Management

Effective Use of Communication Pathways



Formal Requirements

- Distribution of heterogeneous data for increased availability
- Optimistic writes on all devices with application-level conflict resolution
- Automatic management of replicas
- On whatever network is available

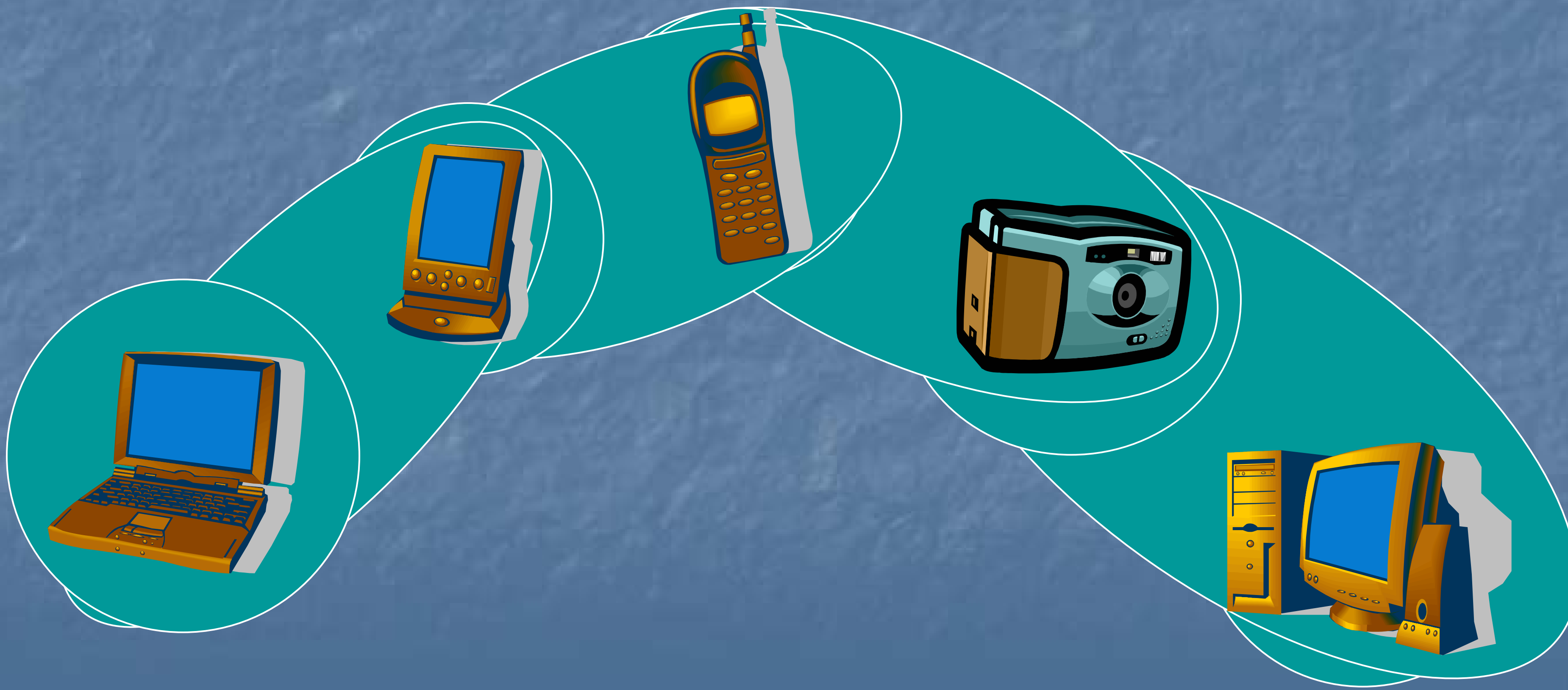
New Assumptions

- Mostly disconnected operation on non-Internet networks
- Applications may run on devices that may never directly talk
- Devices only understand and can resolve conflicts for a few data types
- Devices have finite storage capabilities

New Solutions

- Physical Eventual Consistency
  - Use a pervasive device’s location to enhance consistency
- Selective Conflict Resolution

Physical Eventual Consistency



Physical Eventual Consistency

- Weak eventual consistency
- "Sneaker net" approach to data transfer
- The device with the most updates should be closest to the user.

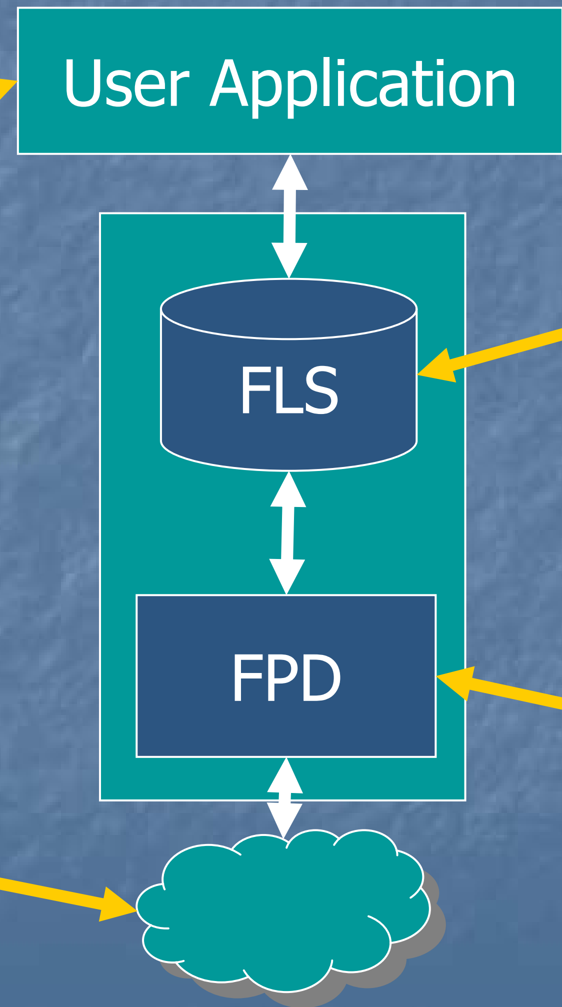
Selective Conflict Resolution

- Two classifications:
  - Smart – can resolve conflicts
  - Dumb – cannot resolve conflicts
- All devices can move all data
- **Separate conflict transfer from conflict resolution**

Footloose

- Shared data store for pervasive applications
  - Guarantees "no lost updates"
  - Automatic management and routing of shared data.
  - Application-level device-distributed conflict resolution

Footloose Architecture



User Application

Register types that will be stored and shared.

Footloose Store

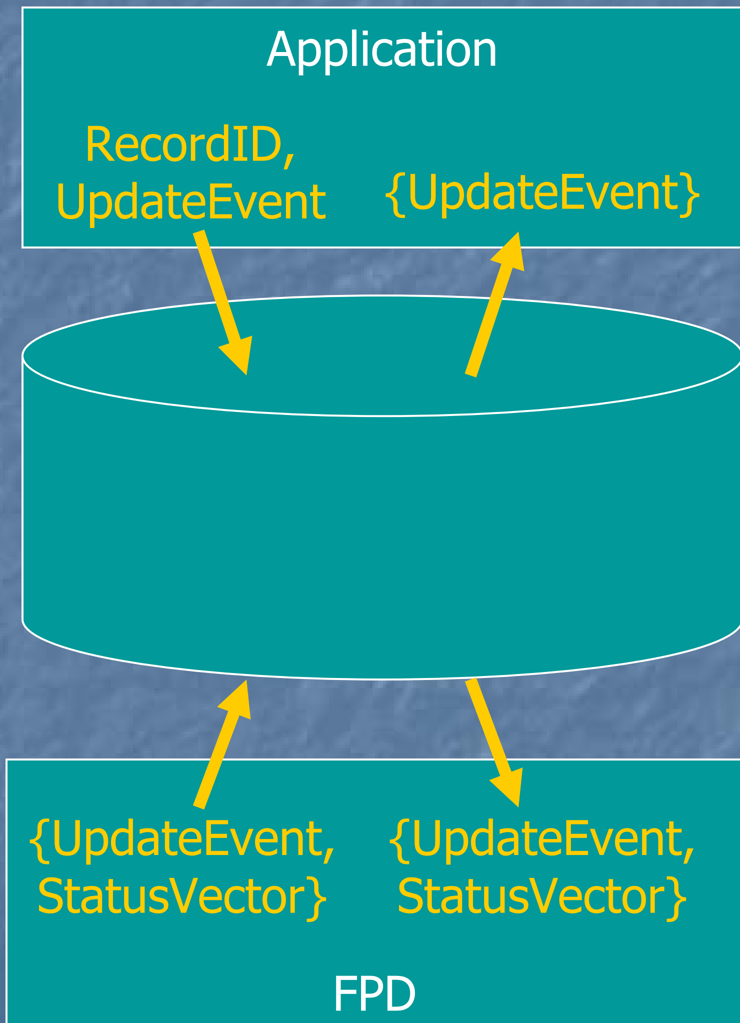
Application interface and storage facility.

Footloose Protocol Daemon

Device interface and consistency maintainer.

Any possible network connection

The Footloose Store

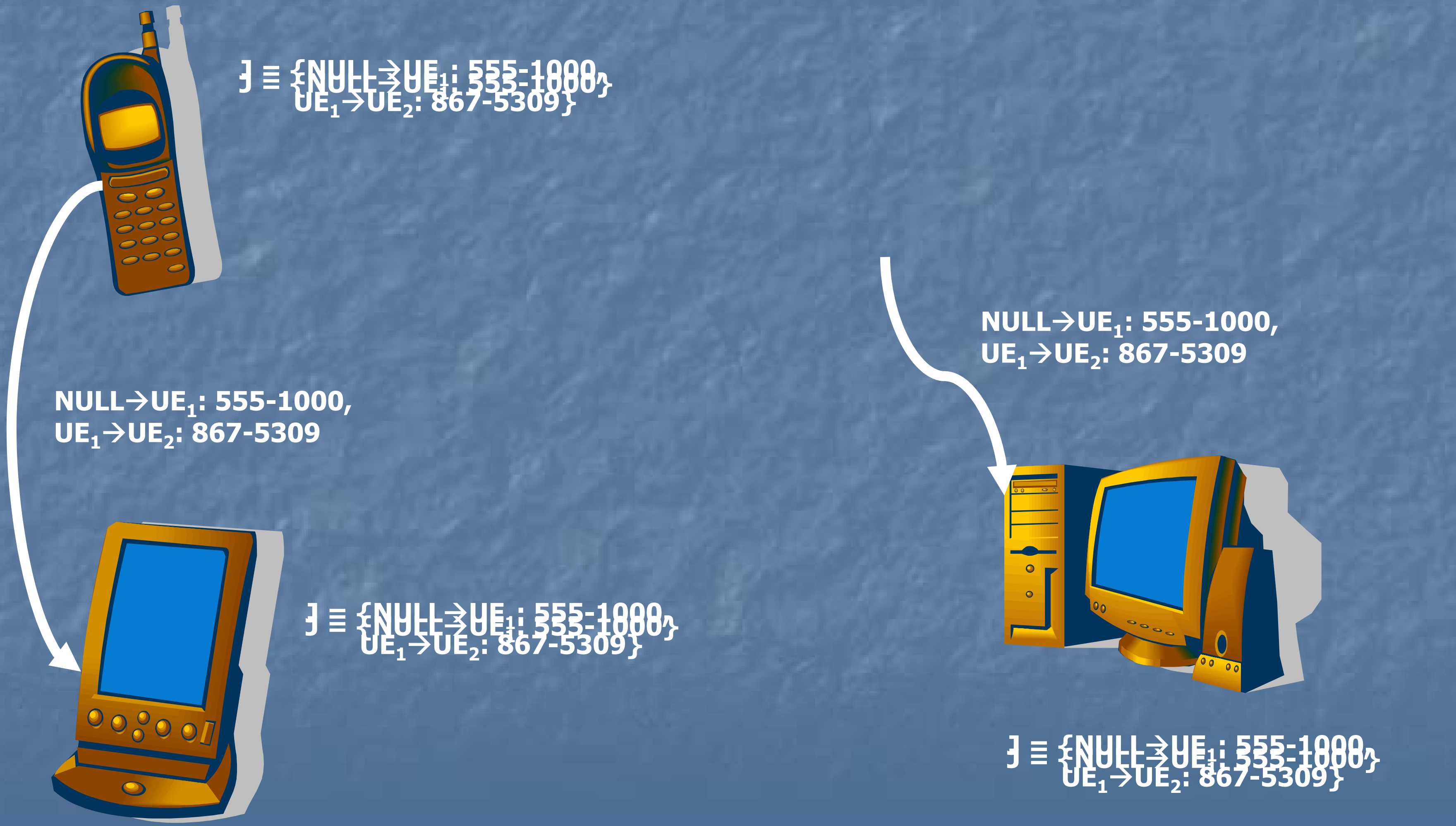


- Mutab… for ap…
  - … R… …ps to a se… …Events
  - Only … UpdateEvents get returned to …ps.
- Applications … make updates …
  - Th…
- List… …ts for the F…

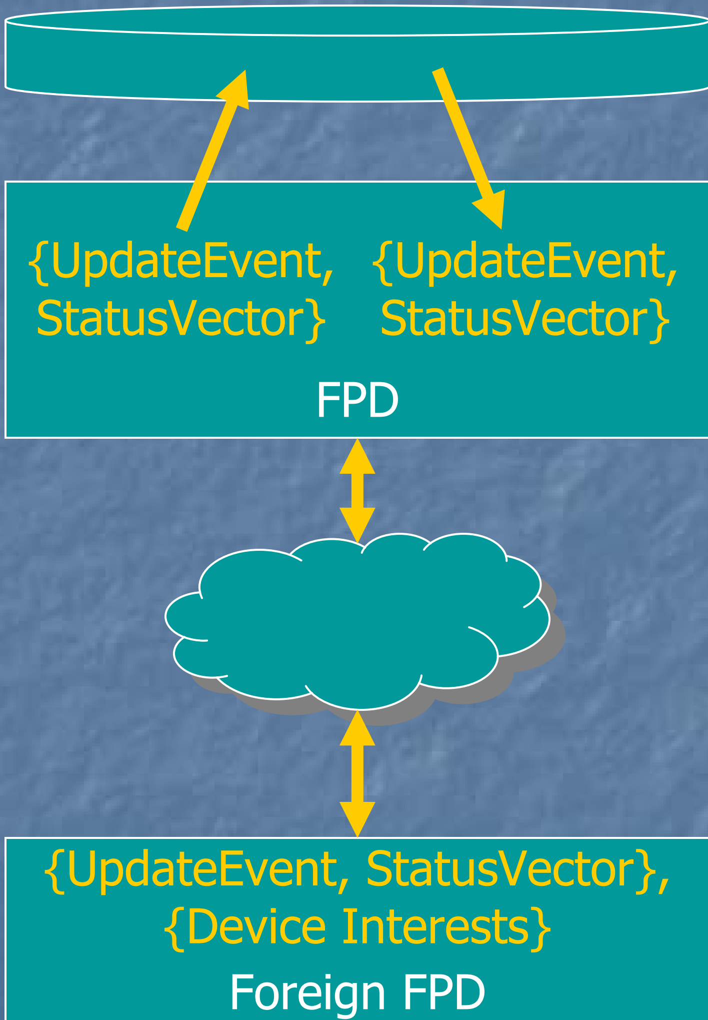
Easy conflict detection!

Transparent Transport of Conflicts!

Enabling Automatic Management



The Footloose Protocol Daemon

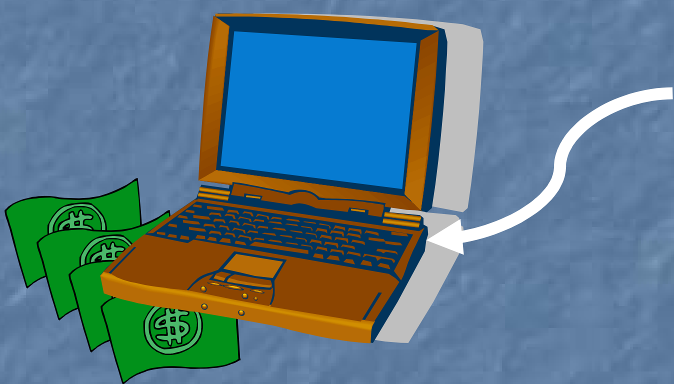


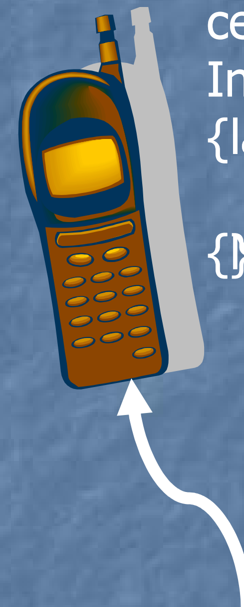
- Maintains "s… ed knowle…
  - … ating … e about a… on all dev…
  - Devi… interests
- Manages D… Comm…
  - Da…

Garbage Collection & Purging

Physical Routing

Effective Use of Communication Pathways





cell:
Interest in =
{laptop, work-comp}

{NULL→UE$_1$: }

laptop:
Interest in =
{laptop, work-comp}

{NULL→UE$_1$: ,
…}

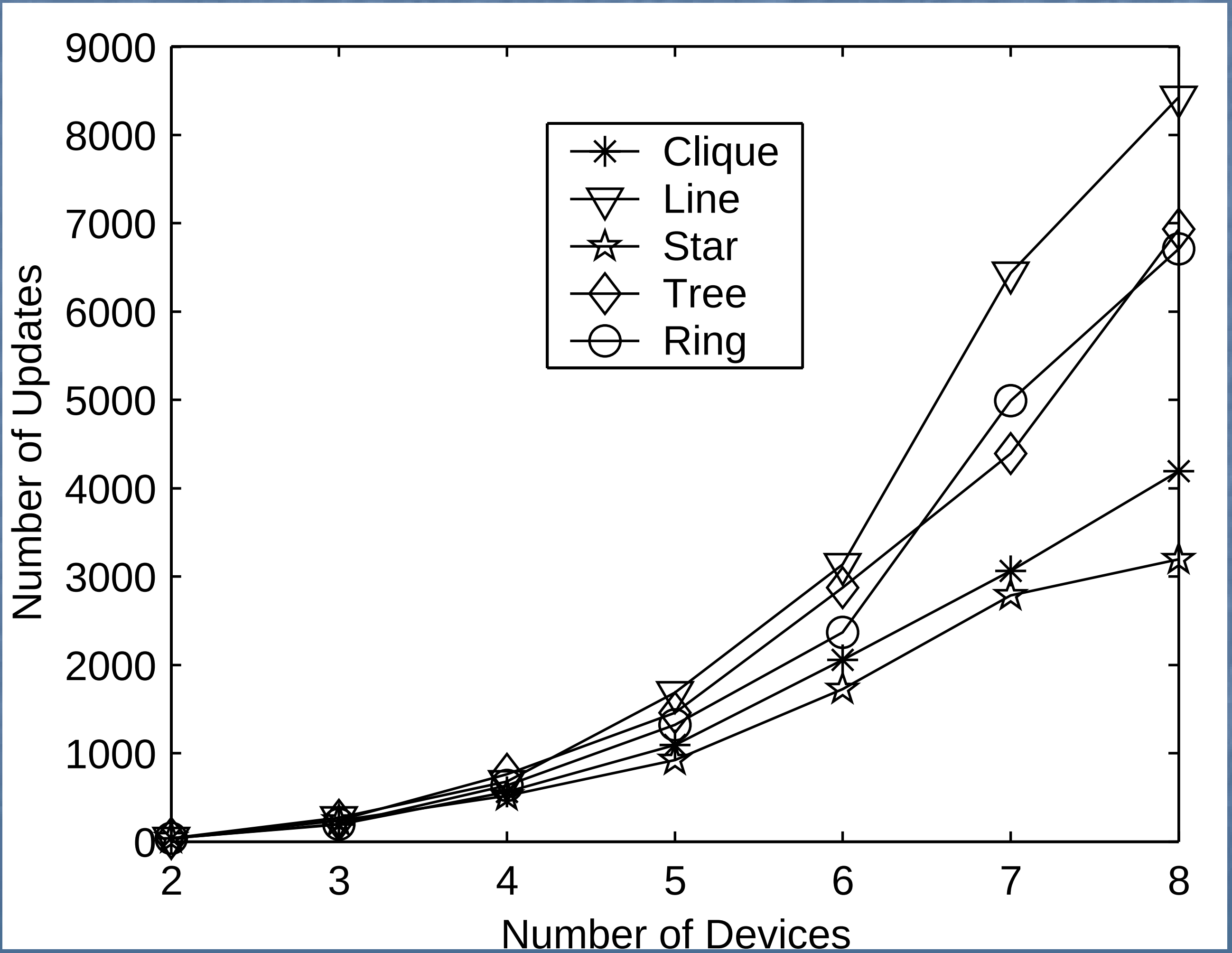
work-comp:
Interest in =
{laptop, work-comp}

{NULL→UE1: ,
…}

Implementation

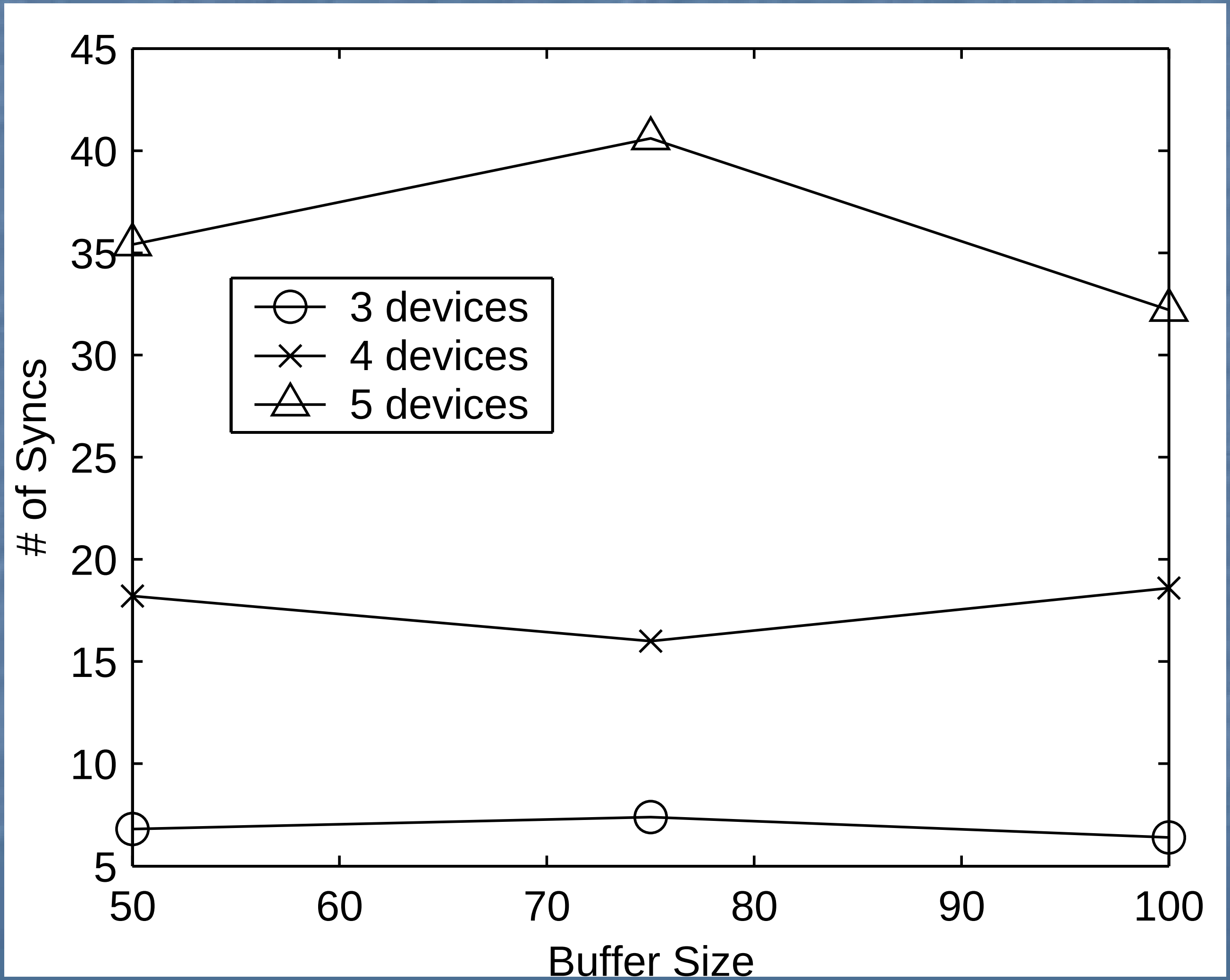
- FLS and FPD built in Java
- Two applications:
  - Wishlist Application
  - Phone Number Database
    - 25 lines of Footloose-dependent code
- Simulation Framework

Evaluation



Clique
Line
Star
Tree
Ring
Number of Updates
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
2
3
4
5
6
7
8
Number of Devices

Evaluation



45
40
35
30
25
20
15
10
5
# of Syncs
3 devices
4 devices
5 devices
50
60
70
80
90
100
Buffer Size

Design Evaluation and Future Work

- Need better support for complex "directory-like" types
- Support for large updates
- Variable Number of Devices
- Security Framework
- User Study

Further Information

Justin Mazzola Paluska
jmp@mit.edu
http://web.mit.edu/jmp/www/