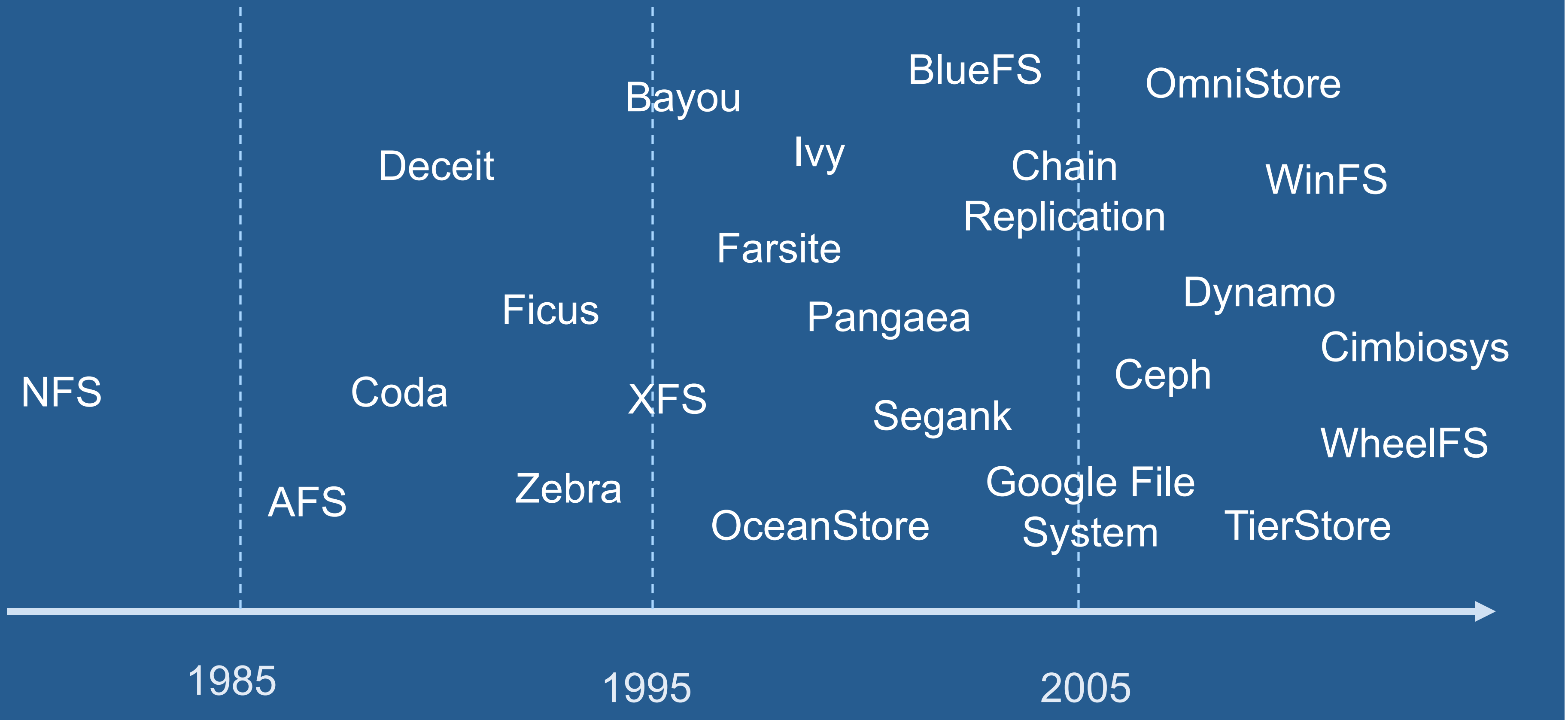
# PADS: Policy Architecture for Distributed Storage Systems

Nalini Belaramani, Jiandan Zheng, Amol Nayate,
Robert Soulé, Mike Dahlin and Robert Grimm.

University of Texas at Austin, Amazon Inc.,
IBM T.J. Watson, New York University

# Lots of data storage systems



NFS

AFS

Deceit

Coda

Ficus

Zebra

Bayou

XFS

Ivy

Farsite

Pangaea

Segank

OceanStore

BlueFS

Chain Replication

Google File System

OmniStore

WinFS

Dynamo

Ceph

Cimbiosys

WheelFS

TierStore

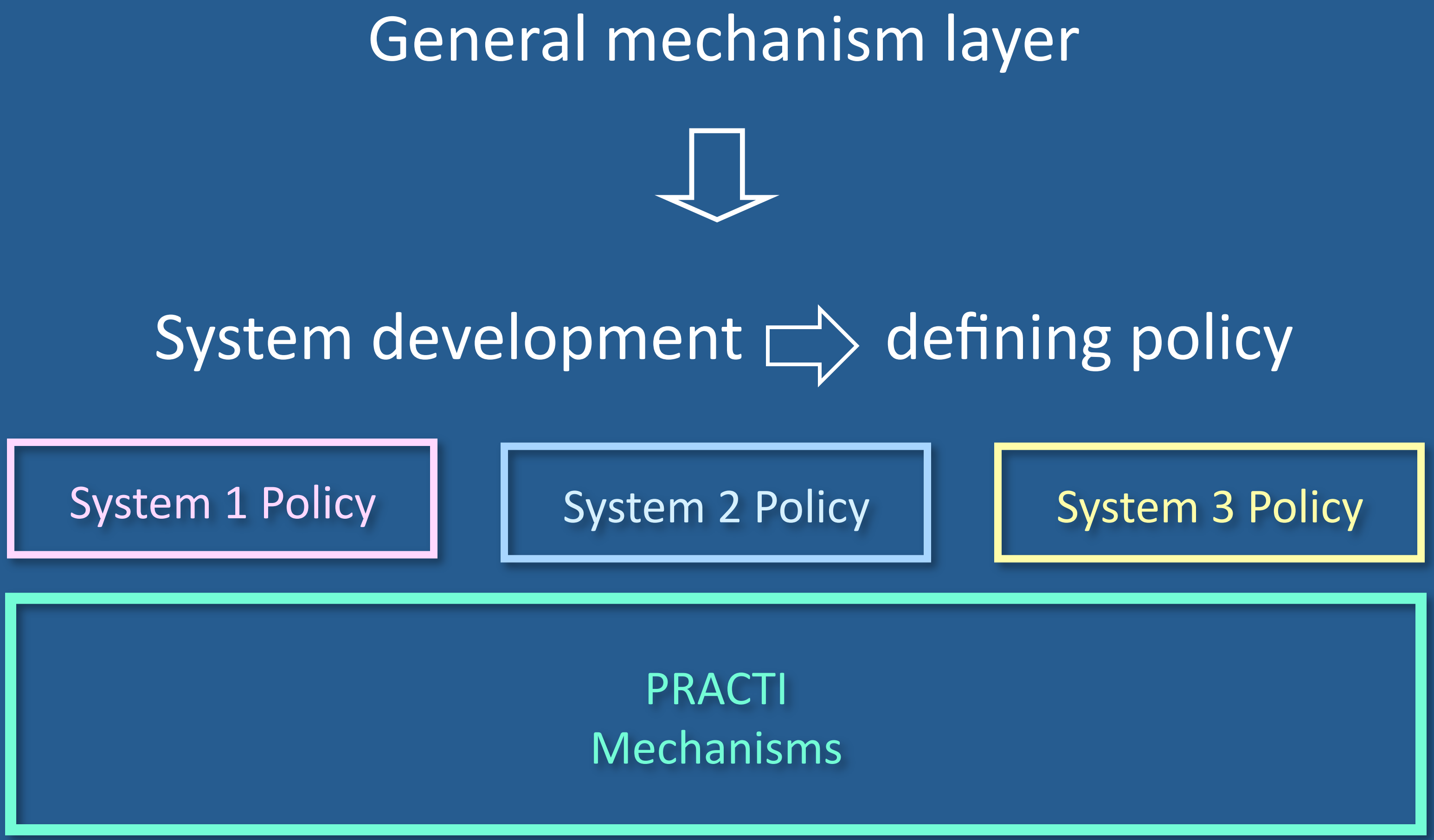
1985

1995

2005

Is there a better way to build distributed storage systems?

# Microkernel approach



General mechanism layer

System development ⇨ defining policy

System 1 Policy

System 2 Policy

System 3 Policy

PRACTI
Mechanisms

[*] "PRACTI Replication", Nalini Belaramani, Mike Dahlin, Lei Gao, Amol Nayate, Arun Venkataramani, Praveen Yalagandula, and Jiandan Zheng. NSDI 2006.

# Is it really a better way?

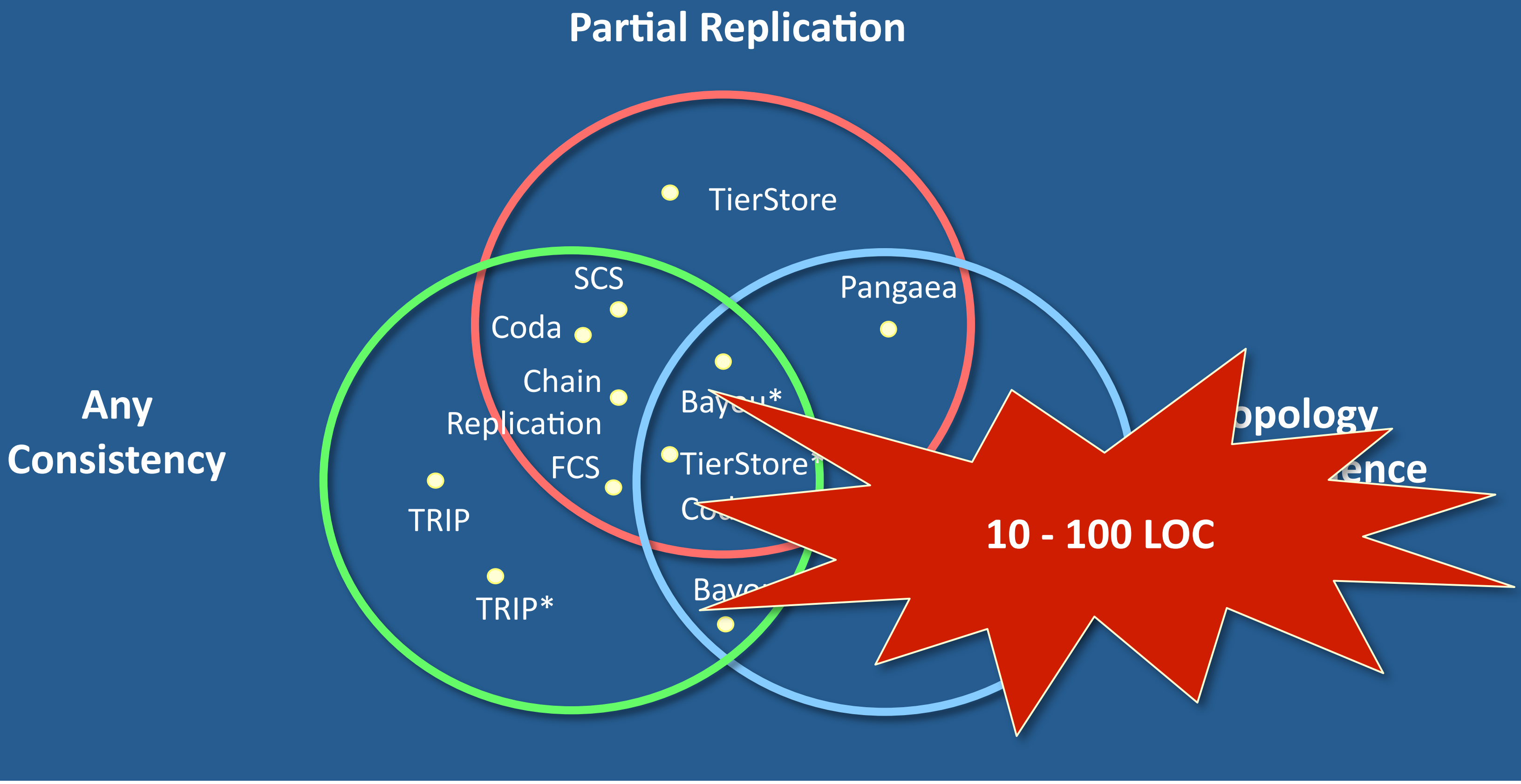
Challenge: 10 systems, 1K lines each before you graduate

*Gulp* How about 3?

# Yes it is!

With PADS:

2 grad students + 4 months = 12 diverse systems



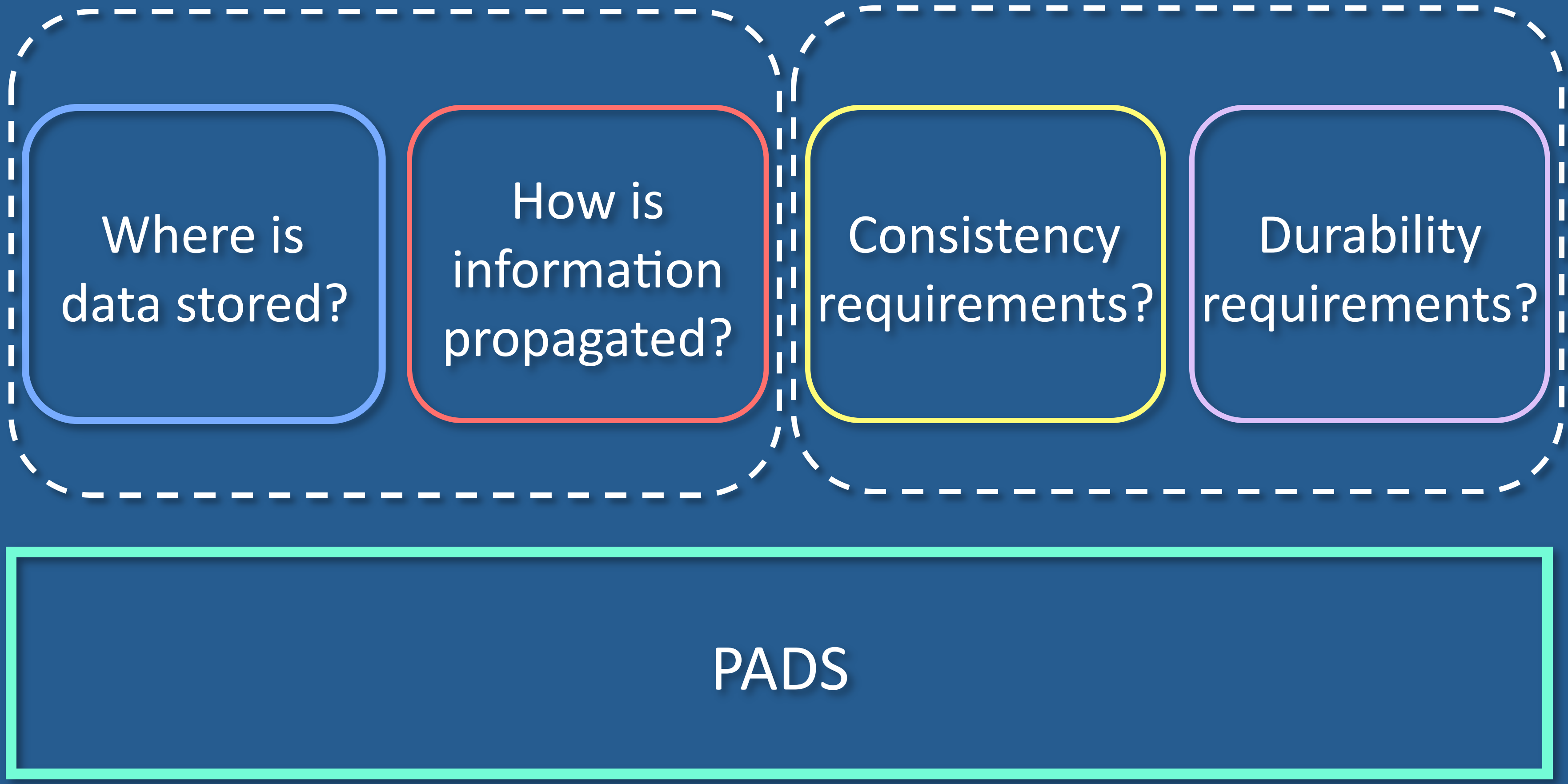
10 - 100 LOC

# Outline

- PADS  approach

- Policy
  - Routing
  - Blocking

- Evaluation



Routing

Blocking

Where is data stored?

How is information propagated?

Consistency requirements?

Durability requirements?

PADS

# Outline

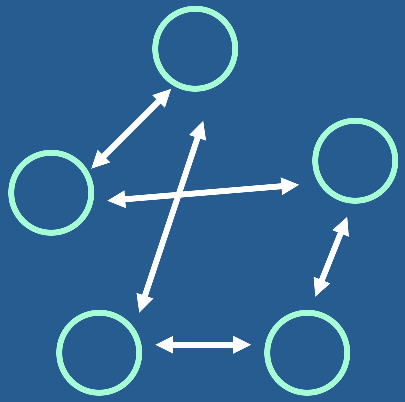
- PADS  approach
- Policy
  - Routing
  - Blocking
- Evaluation

# Routing

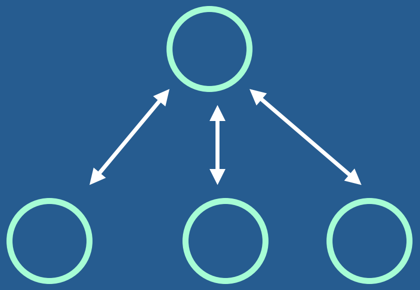
## Data flows among nodes

When and where to send an update?

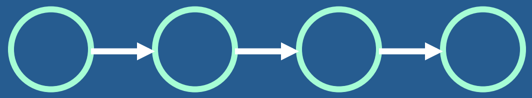
Who to contact on a local read miss?



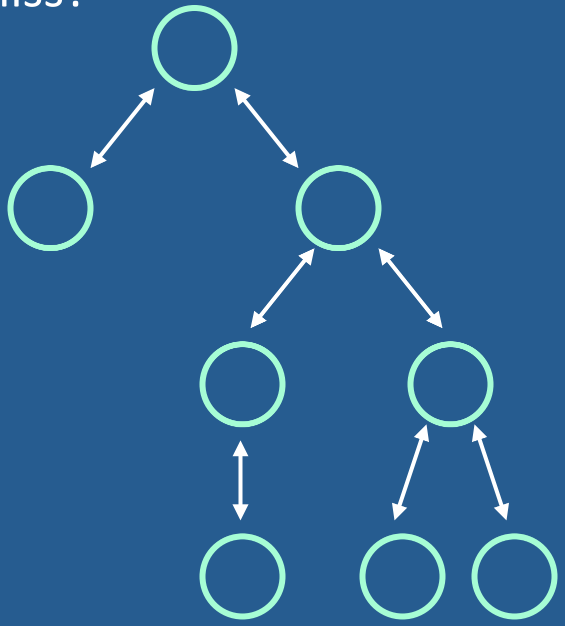
Bayou



Coda



Chain Replication



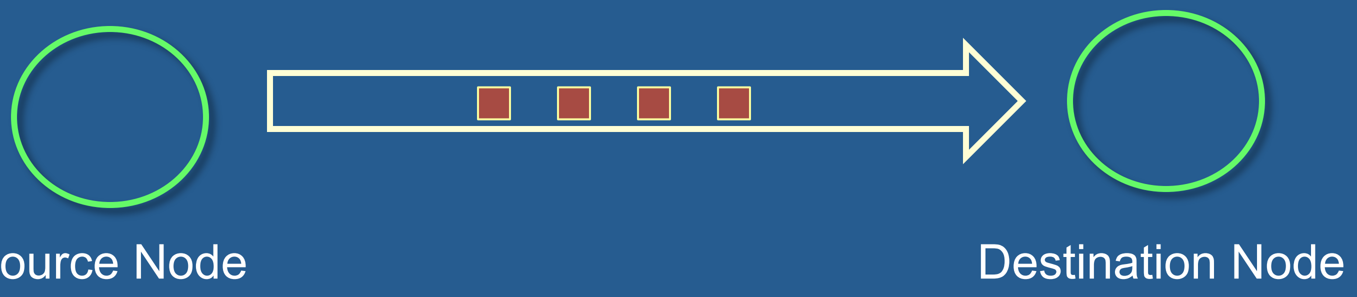
TierStore

# Subscription

Primitive for update flow

Options:

- Data set of interest (e.g. /vol1/*)
- Notifications (invalidations) in causal order or updates (bodies)
- Logical start time



# Event-driven API

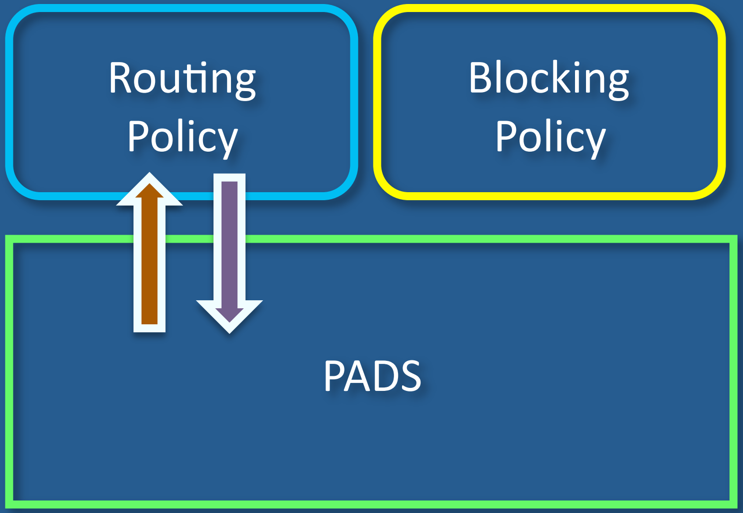
## To set up routing

### Events

Operation block
Write
Delete

Inval arrived
Send body succ
Send body failed

Subscription start
Subscription caught-up
Subscription end



Routing Policy

Blocking Policy

PADS

### Actions

Add inval sub
Add body sub

Remove inval sub
Remove body sub

Send body
Assign seq

B_action

# Domain-specific language

To specify routing

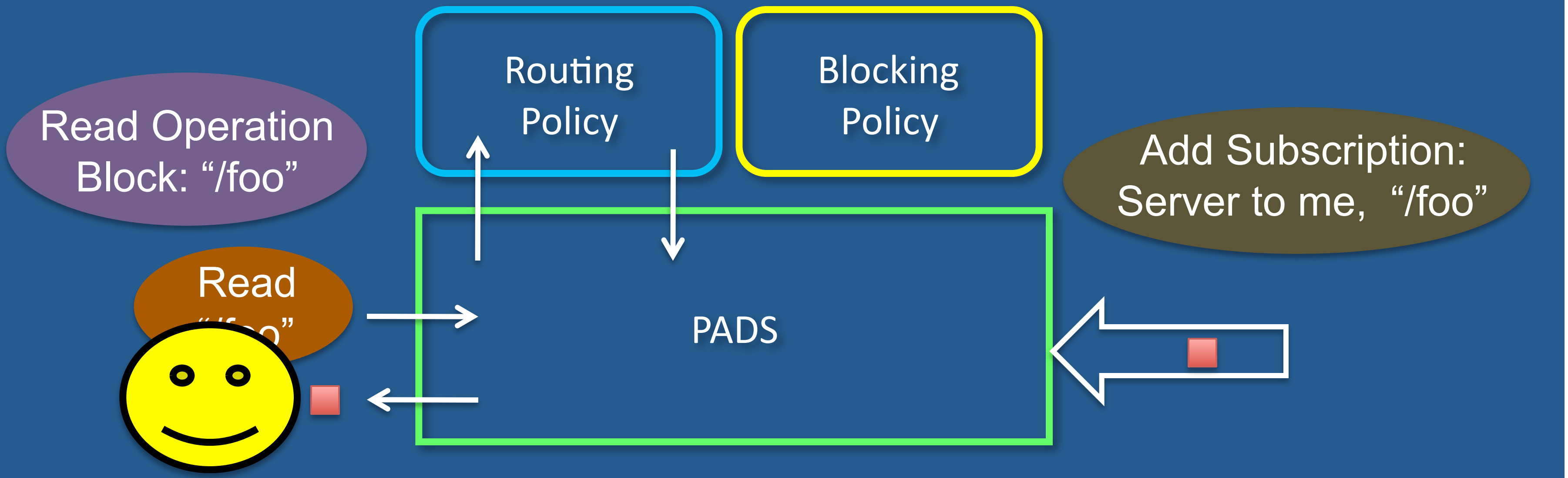
- R/Overlog
  - Routing language based on Overlog[*]
  - declarative rules fired by events

- Policy written as rules
  - invoke actions when events received

[*] "Implementing Declarative Overlays". Boon Thau Loo, Tyson Condie, Joseph M. Hellerstein, Petros Maniatis, Timothy Roscoe, Ion Stoica. SOSP 2005.

# Simple example

On read operation block,
establish subscription to server



Read Operation
Block: “/foo”

Routing
Policy

Blocking
Policy

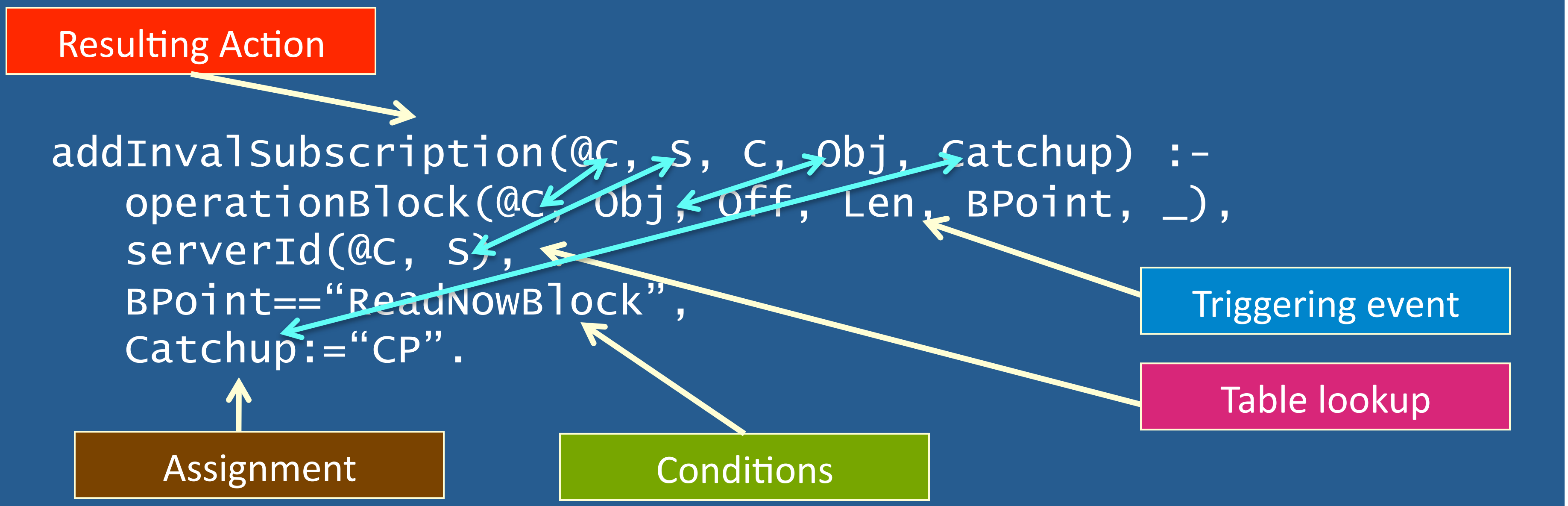
Add Subscription:
Server to me, “/foo”

Read
“/foo”

PADS

# Simple example

On read operation block,
establish subscription to server



Resulting Action

```
addInvalSubscription(@C, S, C, Obj, Catchup) :-
   operationBlock(@C, Obj, Off, Len, BPoint, _),
   serverId(@C, S),
   BPoint==“ReadNowBlock”,
   Catchup:=“CP”.
```

Triggering event

Table lookup

Assignment

Conditions

# P-TierStore Routing

Parent Config

```
in0 TRIG readEvent(@X, ObjId) :-
EVT initialize(@X), ObjId := "/.parent“.

pp0 TBL parent(@X, P) :-
RCV parent(@X, P).
```

Publications Config

```
pp1 TRIG readAndWatchEvent(@X, ObjId) :-
RCV initialize(@X), ObjId := "/.subList".

pSb0 TBL subscription(@X, SS) :-
RCV subscription(@X, SS).
```

Subscriptions from parent

```
pSb1 ACT addInvalSub(@X, P, X, SS, CTP) :-
RCV subscription(@X, SS), TBL parent(@X, P),
CTP=="LOG".

pSb2 ACT addBodySub(@X, P, X, SS) :-
RCV subscription(@X, SS), TBL parent(@X, P).

f1 ACT addInvalSub(@X, P, X, SS, CTP) :-
TRIG subEnd(@X, P, X, SS, , Type),
TBL parent(@X, P), Type=="Inval", CTP:="LOG".

f2 ACT addBodySub(@X, P, X, SS) :-
TRIG subEnd(@X, P, X, SS, , Type),
TBL parent(@X, P), TYPE=="Body", CTP:="LOG".
```

Subscriptions from child

```
cSb1 ACT addInvalSub(@X, C, X, SS, CTP) :-
TRIG subStart(@X, X, C, , Type), C 6= P,
Type == "Inval", SS := "/*", CTP := "LOG".

cSb2 ACT addBodySub(@X, C, X, SS, CTP) :-
TRIG subStart(@X, X, C, , Type), C 6= P,
Type == "Body", SS := "/*".
```

DTN support

```
dtn1 ACT addInvalSub(@X, R, X, SS, CTP) :-
EVT relayNodeArrives(@X, R),
TBL subscription(@X, SS), CTP=="LOG".

dtn2 ACT addBodySub(@X, R, X, SS) :-
EVT relayNodeArrives(@X, R),
TBL subscription(@X, SS), CTP=="LOG".

dtn3 ACT addInvalSub(@X, X, R, SS, CTP) :-
EVT relayNodeArrives(@X, R),
SS:="/*", CTP=="LOG".

dtn4 ACT addBodySub(@X, X, R, SS) :-
EVT relayNodeArrives(@X, R),
SS:="/*", CTP=="LOG".
```

[*] “TierStore: A Distributed Storage System for Challenged Networks”. M. Demmer, B. Du, and E. Brewer. FAST 2008.

# Outline

- PADS  approach
- Policy
  - Routing
  - Blocking
- Evaluation

# Blocking policy

Is it safe to access local data?

Consistency

Durability

What version of data can be accessed?

Whether updates have propagated to safe locations?

Block until semantics guaranteed

# How to specify blocking policy?

Where to block?

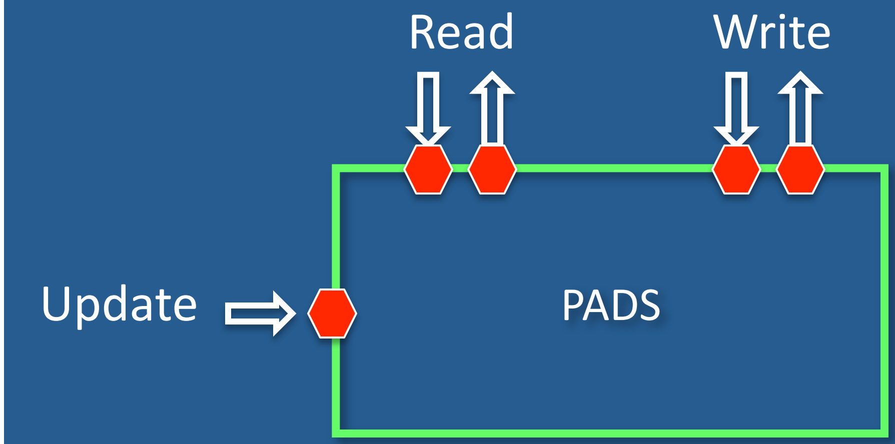
- At data access points

What to specify?

- List of conditions

PADS provides

- 4 built-in conditions (local bookkeeping )
- 1 extensible condition



Is valid
Is causal
Is sequenced
Max staleness
R_Msg

# Blocking policy examples

Consistency:

- Read only causal data

```
Read at block: Is_causal
```

Durability:

- Block write until update reaches server

```
Write after block : R_Msg (ackFromServer)
```

# Outline

- PADS  approach
- Policy
  - Routing
  - Blocking
- Evaluation

# Is PADS a better way to build distributed storage systems?

- General enough?
- Easy to use?
- Easy to adapt
- Overheads?

# General enough?

| | SCS | FCS | Coda | TRIP | Tier Store | Chain Repl | Bayou | Pangaea |
|---|---|---|---|---|---|---|---|---|
| Topology | Client/ Server | Client/ Server | Client/ Server | Client/ Server | Tree | Chains | Ad-Hoc | Ad-Hoc |
| Replication | Partial | Partial | Partial | Full | Partial | Full | Full | Partial |
| Demand caching | ✓ | ✓ | ✓ | ✓ | | | | |
| Cooperative caching | | ✓ | | | | | | |
| Prefetching | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Consistency | Sequential | Sequential | Open/ Close | Sequential | Mono-Reads | Linearizable | Causal | Mono-Reads |
| Callbacks | ✓ | ✓ | ✓ | | | | | |
| Leases | ✓ | ✓ | ✓ | | | | | |
| Disconnected operation | | | ✓ | ✓ | ✓ | ✓ | | ✓ |
| Inval v. update progagation | Inval | Inval | Inval | Inval | Update | Update | Update | Update |

# Easy to use?

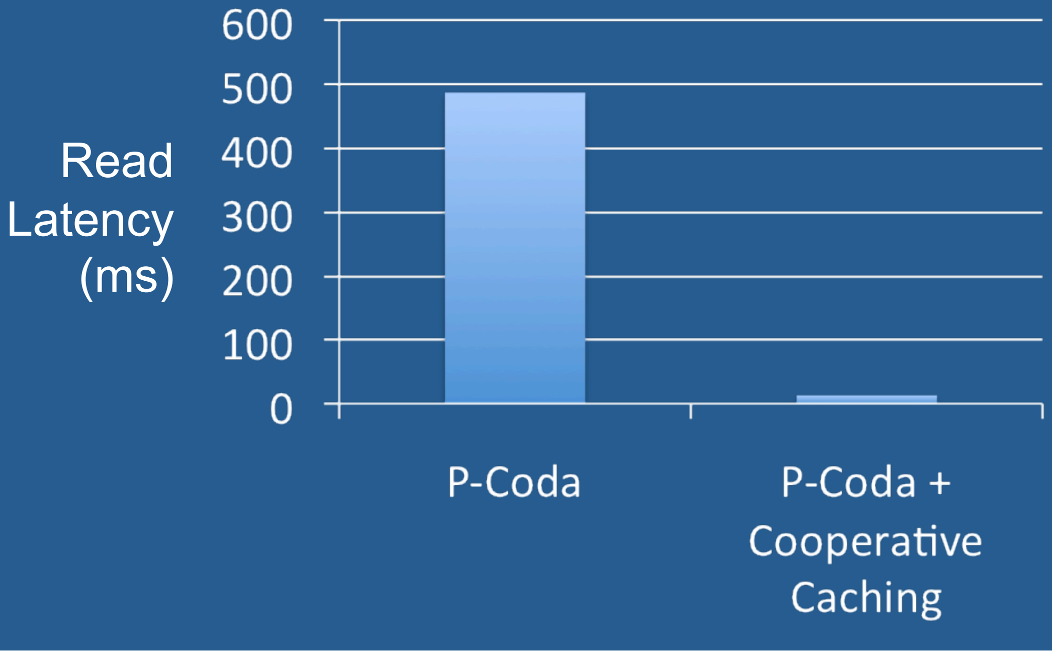
| System | Routing Rules | Blocking Conditions |
|---|---|---|
| P-Bayou | 9 | 3 |
| P-Bayou* | 9 | 3 |
| P-Chain Rep | 75 | 5 |
| P-Coda | 31 | 5 |
| P-Coda* | 44 | 5 |
| P-FCS | 43 | 6 |
| P-Pangaea | 75 | 1 |
| P-TierStore | 14 | 1 |
| P-TierStore* | 29 | 1 |
| P-TRIP | 6 | 3 |
| P-TRIP* | 6 | 3 |

# Easy to adapt?

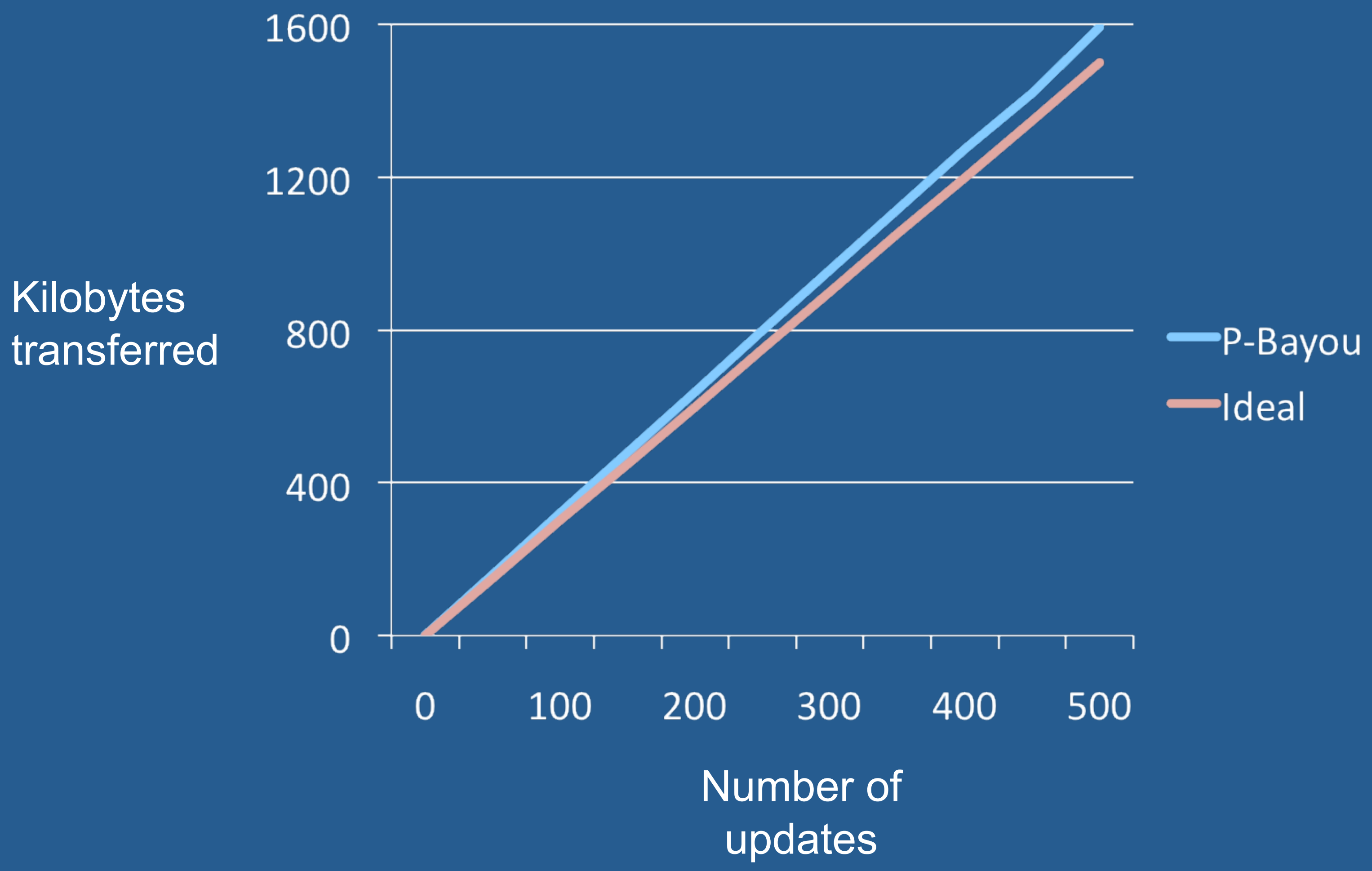
Coda

- Restricts communication to client-server only
- Cannot take advantage of nearby peers

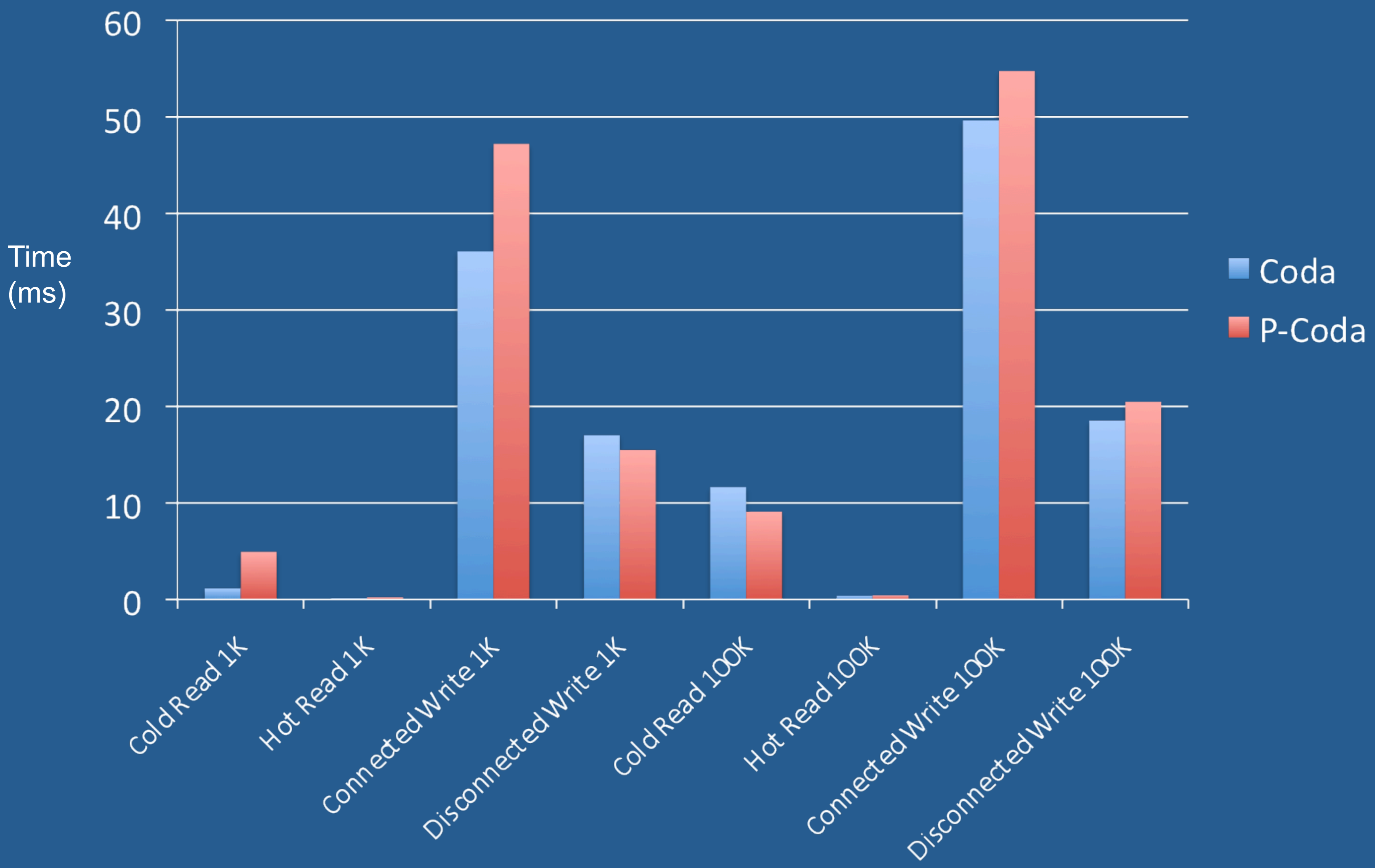
Added co-operative caching in 13 rules



# Overheads?

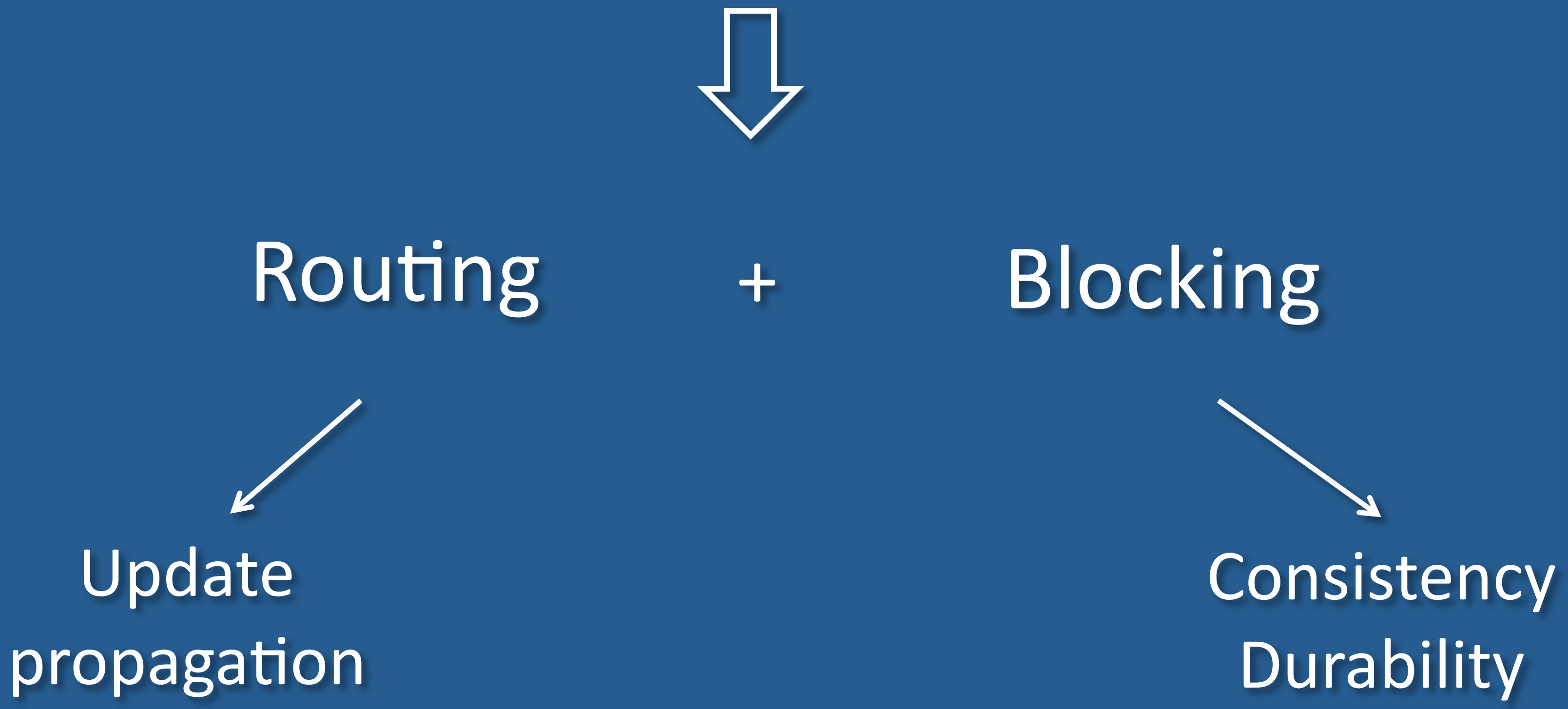

Kilobytes transferred
1600
1200
800
400
0
0
100
200
300
400
500
Number of updates
P-Bayou
Ideal

# Read/Write performance



# Take away lesson

# Distributed data storage systems



Routing + Blocking

Update propagation

Consistency

Durability

Thank you

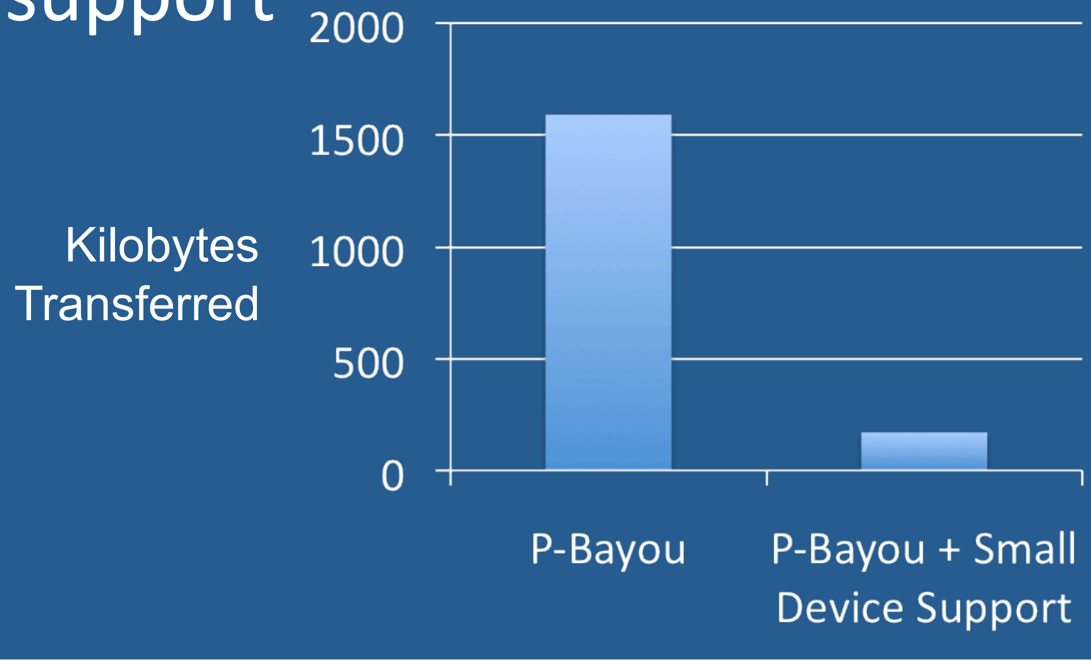
# Easy to adapt?

Bayou

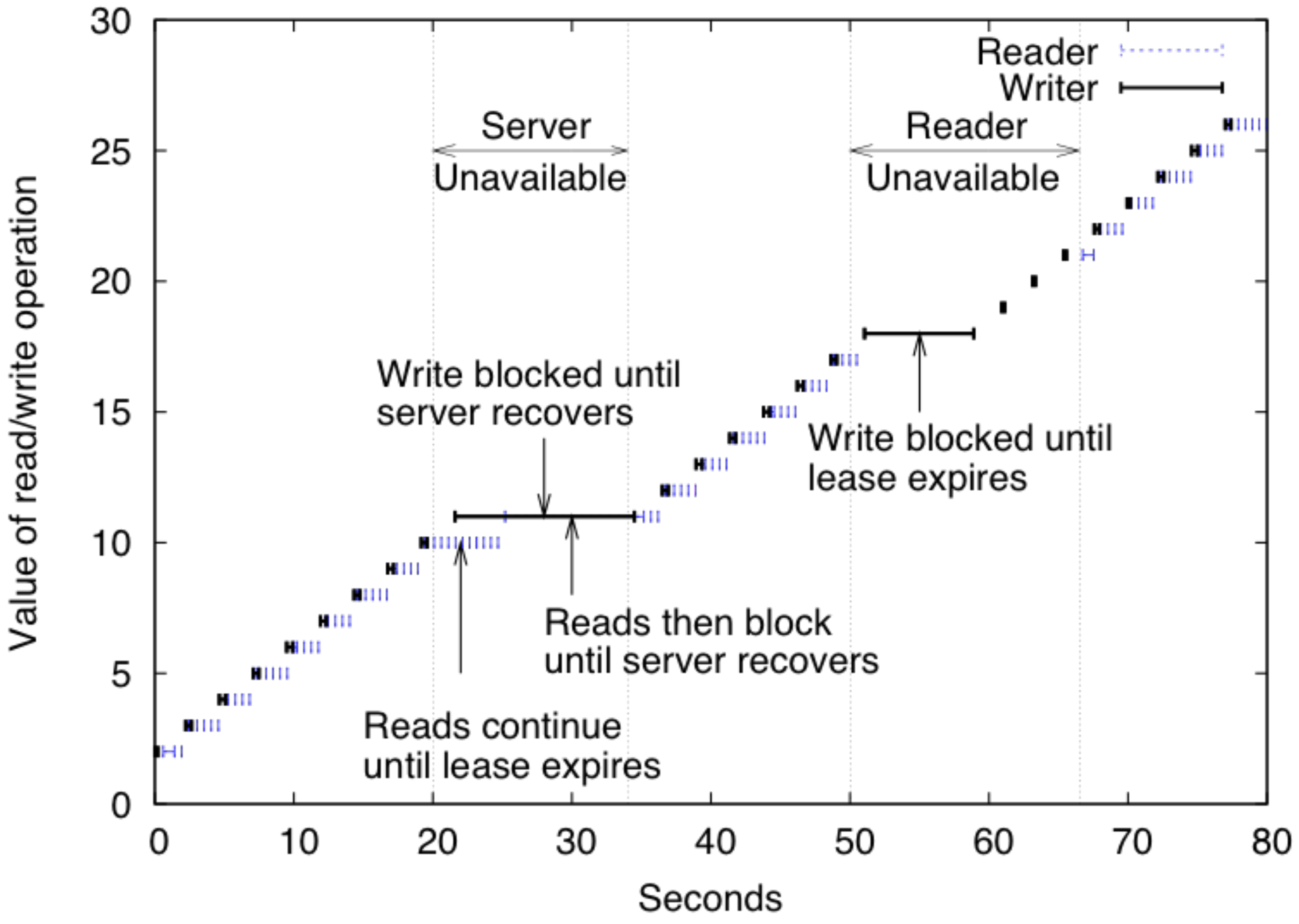
- Mechanisms only support full replication

Add small device support

- Change 4 rules



# Real enough?



Reader
Writer
Server
Unavailable
Reader
Unavailable
Write blocked until
server recovers
Write blocked until
lease expires
Reads then block
until server recovers
Reads continue
until lease expires
Value of read/write operation
Seconds
0
5
10
15
20
25
30
0
10
20
30
40
50
60
70
80

# TierStore

- Data storage for developing environments

- Publish-subscribe system
  - Every node subscribes to publications

- Hierarchical topology
  - Updates flood down the tree
  - Child updates go up the tree to the root

[*] “TierStore: A Distributed Storage System for Challenged Networks”. M. Demmer, B. Du, and E. Brewer. FAST 2008.