PeerMon: A Peer-to-Peer Network Monitoring System

Tia Newhall, Janis Libeks, Ross Greenwood, Jeff Knerr

Computer Science Department

Swarthmore College

Swarthmore, PA USA

newhall@cs.swarthmore.edu

Target: General Purpose NWs

Usually single LAN systems

Each machine's resources controlled by local OS

- NFS, but little other system-wide resource sharing

No central scheduler of NW-wide resources

- Users tend to statically pick node(s) to use
  (ex) write MPI hostfile once, use every time
- Users may not have a choice
  (ex) ssh cs.swarthmore.edu: target is chosen from static set
- Often large imbalances in NW-wide resource usage

Imbalances Cause Poor Performance

- Swapping on some while lots of free RAM on others
- Large variations in CPU loads
- Variations in contention for NIC, disk, other devices

- Parallel applications (ex. MPI)
  - Usually performance determined by slowest node
  - Picking one overloaded node can result in big performance hit

- Sequential applications
  - Low response rate for interactive jobs
  - Longer execution times for batch jobs

Want to do better load balancing

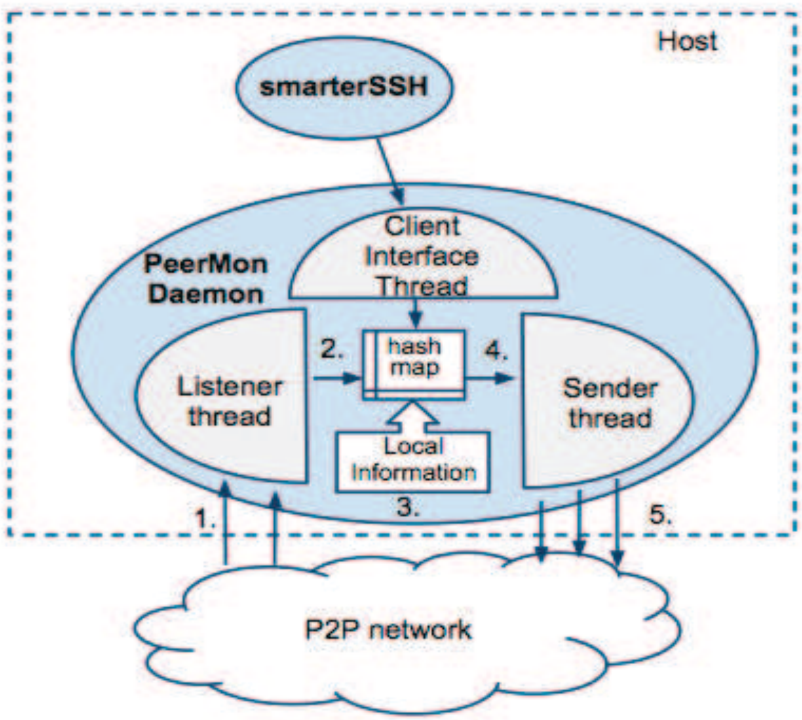
- Tool to easily and quickly discover "good" nodes
  - low CPU load, enough free RAM, fewest number of processes, total # CPUs, …
  - use to make better job/process placement
  - get better load balancing
  - avoid problems with load imballances
- But has to fit with constraints of target system
  - Still General Purpose system where each OS manages it local node's resources
  - Not implementing a global resource scheduler

PeerMon

- P2P Resource Monitoring System
  - Scalable, fault tolerant, low overhead system
    - No central authority, so no single bottleneck nor single point of failure
  - Each node runs equal peer that provides system-wide resource usage data to local users on its node
    - Fast local access to system-wide resource usage data
- Layered Architecture:
  - PeerMon does the system-wide data collection part
  - Higher-level services use PeerMon data to do load balancing, job placement, …

PeerMon Architecture

Peer-to-Peer Resource Monitoring System

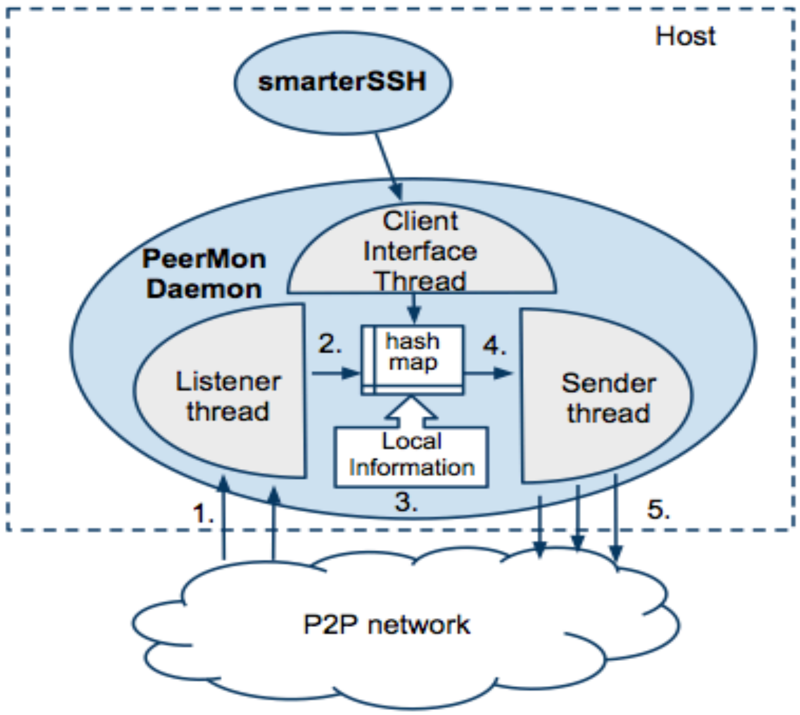


Every node runs equal peer that collects system-wide resource usage data

Sender and Listener Threads: communicate over P2P NW

Client Interface Thread: exports PeerMon data to higher-level services that use it (communicate with local peermon daemon only!)

Listener and Sender Threads



Listener Thread:

- receives resource usage data from other peers
- updates its system-wide resource usage data (stored in hashMap)

Sender Thread:

periodically wakes up & sends its data about whole system to 3 peers

Both use UDP/IP

- Fast, don't need reliable delivery
- Single UDP socket vs. one per connection w/TCP

Resource Usage Data

Each PeerMon peer:

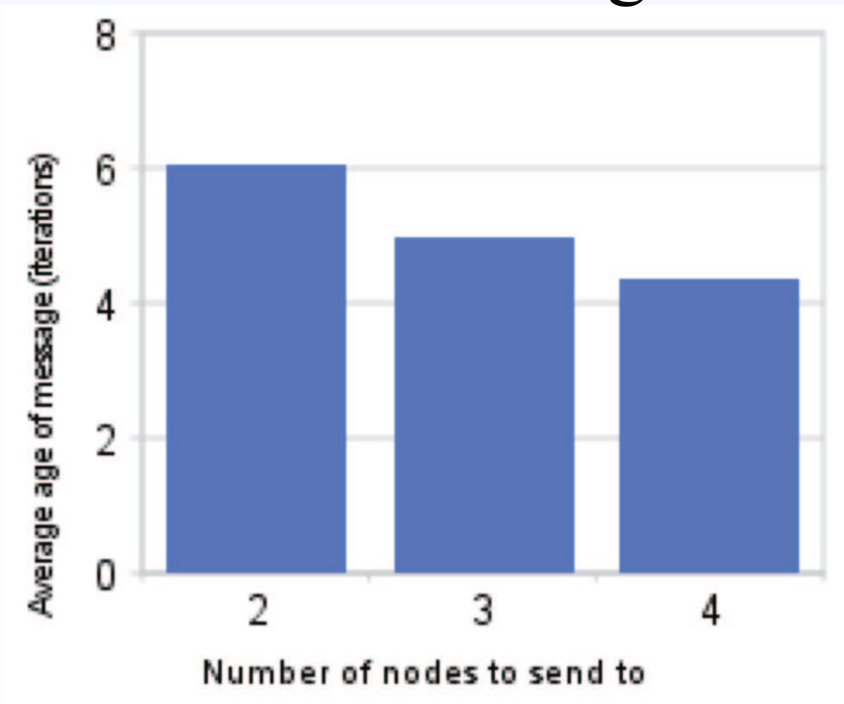
- Collects info about its own node
- Sends its full hashMap data to 3 peers
  - Cycle through different heuristics to choose 3 to ensure full conectivity & that new nodes get quickly integrated
- Receives info about other nodes from some of its peers

Constraints on PeerMon Peer's Data:

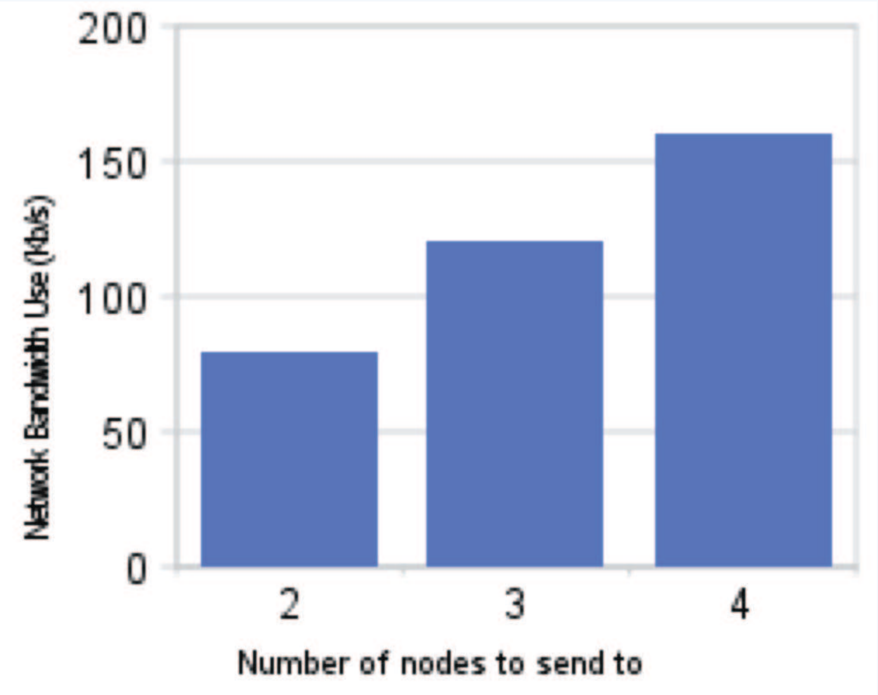
- Doesn't need to be consistent across peers
  - With good messaging heuristics it is close to consistent
- If higher-level service requires an absolute authority, then it can choose 1 PeerMon node to be that authority
  - No different from centralized SNMP systems

Why send to 3 peers?

Ave. Data Age



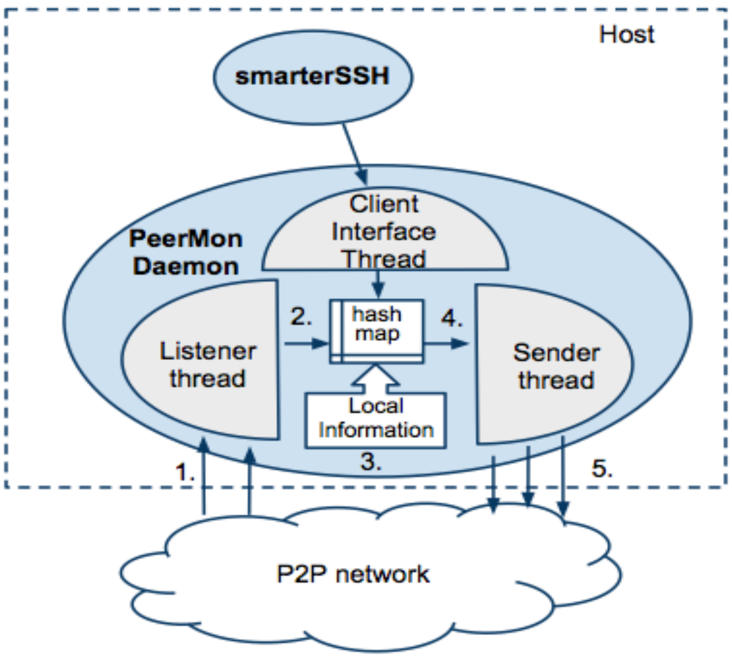
NW Bandwidth



Results for a 500 node system

Sending to 3 peers is good trade-off in Data Age vs. NW overheads

Client Thread



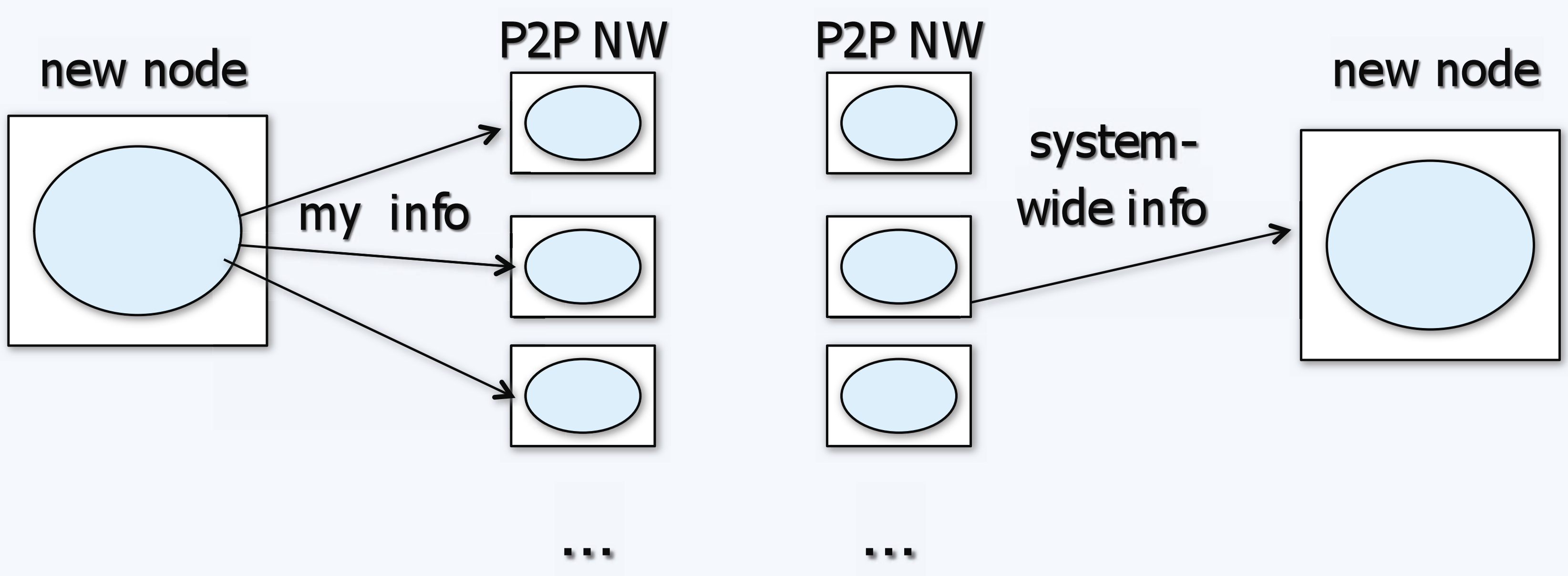
- Local PeerMon daemon provides all system-wide data to local users
- currently TCP interface
- If a higher-level service requires an absolute authority, then it can interact with exactly one PeerMon daemon or implement distributed consensus w/more than one
- For services that don't need absolute agreement, interact with local PeerMon daemon
  => purely distributed interaction

System start-up

New peermon process gets 3 peer IPs config file

Sender thread sends data to 3 peers to connect to P2P NW

If at least 1 of 3 eventually runs peermon, new node will enter PeerMon P2P NW



Fault Tolerance and Recovery

When a node fails or becomes unreachable, its data ages out of the system

- Users of PeerMon data at other nodes will not choose failed node as one of the "good" nodes

<u>Recovery:</u>

- No different from start-up
- No global state that needs to be reconstructed, new peerMon deamon will enter P2P NW and begin receiving system-wide resource usage data

Example Uses of PeerMon

- SmarterSSH:
  - Uses PeerMon data to pick best ssh target
- autoMPIgen
  - Generates MPI hostfile, choosing best nodes based on PeerMon data
- Dynamic DNS mapping
  - Dynamically binds name to one of current set of best nodes
  - Uses RR in BIND 9 to rotate through set of top N machines periodically updated by PeerMon

SmarterSSH and autoMPIgen

- Simple Python Programs, use PeerMon client TCP interface
- Can order “best” nodes based on CPU load, amount free RAM, or combination of both
- Uses a delta value in ordering nodes so small diffs in load are not significant to ordering
- smarterSSH randomizes the order of “equally” good nodes so subsequent quick invocations distribute ssh load over set of “best” nodes

Example smarterSSH commands

list top 5 by CPU&RAM

```
molasses,[~],11:29am%
molasses,[~],11:29am% smarterssh -v -n 5
IP              CPU load        free memory       CPU cores
------------------------------------------------------------
sesame          0.050              7303248              4
turmeric        0.000              6401308              4
molasses        0.000              6302120              4
lime            0.120              6771928              4
myrtle          0.730              9301760              8
molasses,[~],11:29am%
molasses,[~],11:29am% smarterssh -v -n 5 -c
IP              CPU load        free memory       CPU cores
------------------------------------------------------------
molasses        0.000              6302120              4
turmeric        0.000              6401308              4
lettuce         0.010              2207396              4
bacon           0.040              2376356              4
sesame          0.050              7303248              4
molasses,[~],11:29am%
molasses,[~],11:29am% smarterssh -c
sshing into  turmeric
Enter passphrase for key '/home/newhall/.ssh/id_rsa':
```

top 5 by CPU load

ssh into best by CPU load

How much does PeerMon help?

- Three benchmark programs:
  1. Memory Intensive sequential program
  2. CPU intensive OpenMP program (single node)
  3. RAM&CPU intensive parallel MPI program (ran on 8 of 50 nodes)
- Experiments comparing:
  - Runs on randomly selected node(s): no PeerMon
  - Nodes chosen using PeerMon data with:
    - Ordered by CPU only
    - Ordered by available RAM only
    - Ordered using both CPU load and available RAM

Speed-up of PeerMon vs Random

Node Ranking	Sequential (RAM Intensive)	OpenMP (CPU Intensive)	8 node MPI (Both)
CPU only	0.87	1.63	1.27
RAM only	4.62	2.19	1.78
CPU & RAM	4.62	2.29	1.83

+ Using PeerMon significantly improves performance

  random only does better when PeerMon ordering criterion is bad match for application

+ Combination of CPU&RAM best ordering criterion

Scalability of PeerMon

- Tested PeerMon NWs of 2-2,200 nodes
- Collected traces of MRTG data for CPU, RAM, NW bandwidth

Results:

- Per node CPU and RAM Usage remains constant
- Per node NW bandwidth grows slightly with size of P2P NW, but still very small
  - Up to .16 Mbit/s for 2,200 node system
  - Each node sends information about every node in NW, so as PeerMon NW grows, so does amt data

Conclusions

- PeerMon: P2P, low overhead, scalable, fault-tolerant resource monitoring system for general purpose LANs
- It provides system-wide resource usage data and an interface to export data to higher-level tools and services
- Our example tools that use PeerMon data provide some load balancing in general purpose NW systems and result in significant improvements in performance

Future Work

- Release beta version under GPL
  we hope before end of summer

  `www.cs.swarthmore.edu/~newhall/peermon`

- Further investigate security & scalability issues
  - PeerMon that spans multiple LANs?
- Implement easier to use client interface
- Add extensibility interface to change set of system resource monitored and how
- Implement more tools that use PeerMon